A Continuous-Time Reinforcement Learning Framework for Fine-Tuning Discrete Diffusion Models

Zikun Zhang* Jiayuan Sheng* David D. Yao* Wenpin Tang*

Abstract

We formulate reinforcement learning (RL) in continuous time with discrete state spaces and possibly arbitrary action spaces via a stochastic control approach, where the state dynamics are modeled as a controlled continuous-time Markov chain (CTMC). We consider policy optimization problems and derive the corresponding policy gradient methods, leading to continuous-time variants of proximal policy optimization (PPO) and group relative policy optimization (GRPO). As a primary application, we develop a complete continuous-time RL framework for fine-tuning score-based discrete diffusion models. The proposed framework enables reward-driven optimization without requiring differentiability on the reward signals. In contrast to the existing GRPO-based approaches that only rely on terminal rewards, our formulation allows intermediate reward or advantage signals to be incorporated throughout the denoising trajectory. Importantly, when specialized to masked diffusion models (MDMs), our framework encompasses a rich class of policy parameterizations over the vocabulary simplex with analytically tractable probability ratios, providing a unified perspective on exploration and policy optimization in MDMs. For masked diffusion large language models (dLLMs), we further propose trajectory subsampling techniques to efficiently estimate computationally prohibitive trajectory likelihoods, reducing the computational cost of computing per-position probability ratios. We showcase the effectiveness of our methods on both low-dimensional entropy-regularized optimization problems and RL post-training of dLLMs on mathematical reasoning and coding tasks.

1 Introduction

Score-based diffusion models, which was originally designed for image generation in continuous space (Ho et al., 2020; Song et al., 2021), have recently emerged as a powerful paradigm for discrete data generation in the context of LLMs (Austin et al., 2021; Campbell et al., 2022; Sun et al., 2023; Lou et al., 2024). For discrete diffusion models with the forward process formulated by a continuous-time Markov chain (CTMC), one learns the *discrete score function* (Lou et al., 2024; Meng et al., 2022; Benton et al., 2024), i.e., the probability ratio of the forward marginals, to generate the reverse process. A notable example is masked diffusion models (Shi et al., 2024; Sahoo et al., 2024) that corrupt data into fully masked sequences and train a denoiser to predict the clean data from the masked tokens. As was shown by Shi et al. (2024), the score function reduces to the denoiser probability vector on the vocabulary space for masked positions, making the model formulation, training, and inference much simpler by directly predicting the clean data at each denoising step (x_0-prediction). This simplified structure gains dominant popularity, and further enhances the rapid development of mask-based diffusion large language models (Nie et al., 2026b;

*Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027. Emails: zz3367@columbia.edu, js6646@columbia.edu, yao@columbia.edu, wt2319@columbia.edu.

Ye et al., 2025; Khanna et al., 2025; Liu et al., 2025; Cheng et al., 2025; Zhu et al., 2026; Nie et al., 2026a). dLLMs mark a promising generative modeling paradigm for text generation because they enable more flexible parallel and any-order token decoding, in contrast to the traditional left-to-right sequence decoding in autoregressive large language models (ARMs).

Fine-tuning discrete diffusion models via reinforcement learning (RL) is challenging because sampling from the discrete categorical distribution is non-differentiable, so standard gradient methods are not applicable. Specific to language models, the ARM fine-tuning process is naturally constructed as a Markov Decision Process (MDP) thanks to the left-to-right decoding, in which the action is to predict the next token with the state being the current decoded sequence, leading to a factorized policy likelihood by the chain rule. However, it is difficult to scale and enhance the reasoning capabilities of dLLMs by online RL that has been well-developed in ARM post-training, because of the intractable likelihood of the generated sequences in any-order decoding. Recent methods, e.g., d1 (Zhao et al., 2025c), UniGRPO (Yang et al., 2026), wd1 (Tang et al., 2026), SPG (Wang et al., 2026b), and d2 (Wang et al., 2026c), focused on estimating the trajectory log-likelihood, either by heuristic estimators lacking a clear probabilistic interpretation or by tractable evidence lower and upper bounds. While these approaches have achieved promising empirical performance, they often suffer from coarse likelihood approximations and limited control over the estimation errors. More fundamentally, existing GRPO methods are not grounded in a unified MDP formulation, leaving the connection between iterative diffusion denoising for rollout generation and RL policy optimization largely heuristic and inconsistent. Moreover, most GRPO-based algorithms rely on reward signals only at the final completion, resulting in coarse credit assignment for the intermediate token-unmask decisions that shape the generation process. Although a recent work (Oba et al., 2026) introduced branching-based resampling at selected intermediate steps to provide intermediate feedback, the rewards are still being derived from the terminal evaluation, and are potentially computation-expensive due to branching.

The purpose of this work is to tackle the aforementioned issues by developing an RL framework for CTMC models, which enables us to propose a principled approach for fine-tuning *any* score-based discrete diffusion model via continuous-time reinforcement learning (CTRL). To see the motivation, Zhao et al. (2024, 2025a) and Gao et al. (2025) first proposed to fine-tune diffusion models in continuous spaces via CTRL. In that setting, the score function is the only unknown component in the generative process, so one can treat it as a control/action/policy, and hence, the reverse process is a state-controlled stochastic differential equation (SDE). Then a score-related stochastic policy is designed for exploration, and the corresponding policy gradient and estimation can be obtained by the well-developed stochastic control and RL theory in continuous time and space (Wang et al., 2020; Jia and Zhou, 2022a,b, 2023; Zhao et al., 2023, 2026; Tang and Zhou, 2024). Since the state dynamics of diffusion models are inherently formulated as continuous-time processes (either SDEs or CTMCs), fine-tuning them via CRTL is a natural choice, which not only exhibits robustness to time discretization (Zhao et al., 2025a), but also allows us to carry out all clean theoretical derivations in continuous time, while retain great flexibility for discretizing the continuous-time processes at the final algorithmic stage.

However, RL in continuous time and discrete spaces has yet been developed, and it remains unclear whether controlled-based RL fine-tuning carries over to discrete diffusion models. In this work, we address these problems affirmatively by formulating a CTRL framework for discrete state spaces with potentially arbitrary action spaces. On one hand, this framework opens a new gate for research at the intersection of control theory and RL in discrete spaces, where policy evaluation, policy gradient theory, and practical algorithms remain underexplored. On the other hand, we demonstrate its

application to fine-tuning discrete diffusion models by treating the denoiser probability vector as the action, which offers flexibility for policy parameterization design. This perspective naturally induces a controlled CTMC that explores the output distribution of the denoiser, and yields a clean analytical expression for the policy log-likelihood. Consequently, it bridges the gap between autoregressive-style policy optimization objectives and diffusion-based generative modeling, providing a principled RL framework for fine-tuning discrete diffusion models.

1.1 Contributions

1. *RL theory and algorithms.* We first formally propose a CTRL framework in discrete spaces, which to our best knowledge, is novel. Analogous to classical stochastic control and RL in continuous time and space (Wang et al., 2020; Jia and Zhou, 2023; Zhao et al., 2023), we define controlled CTMC state dynamics and the (optimal) value function, and derive the corresponding analytical forms of Hamilton-Jacobi-Bellman (HJB) equation, optimal control, instantaneous advantage rate (q-function), and policy gradient. By utilizing these, we propose PPO and GRPO algorithms for general CTRL in discrete spaces.

2. *Fine-tuning discrete diffusion models.* Inspired by the works (Zhao et al., 2025a; Gao et al., 2025), we view the sampling process of discrete diffusion models as a stochastic control problem, where the discrete score function is viewed as the control/action/policy. So the problem of fine-tuning discrete diffusion models naturally fits into the CTRL in discrete spaces, thereby solving it via the aforementioned RL algorithms. To this end, we propose several policy parameterization methods (Dirichlet policy, temperature softmax policy, and logistic normal policy), and derive the corresponding policy probability ratios. Our framework is flexible with any predefined process reward models that assign credits to intermediate partially denoised states. We also adopt trajectory subsampling to reduce the number of forward passes during training.

3. *Experiments.* We conduct experiments on both low-dimensional synthetic data (So et al., 2026), and LLaDA (Nie et al., 2026b), an open-sourced 8B dLLM, for reasoning and coding tasks. For the low-dimensional synthetic example, we illustrate that the PPO algorithm outperforms its alternatives (d1 (Zhao et al., 2025c) and DAM (So et al., 2026)) in both generation performance and convergence. For LLaDA experiments, we train our CTRL on the GRPO algorithms, demonstrating strong effectiveness in both mathematical reasoning tasks (Sudoku, GSM8K and MATH500), and coding tasks (HumanEval and MBPP) compared to the state of the arts algorithms – d1 (Zhao et al., 2025c), d2 (Wang et al., 2026c) and SPG (Wang et al., 2026b). For instance, we achieve 88.2% accuracy on Sudoku with 0-shot prompting training.

1.2 Related Works

Continuous-Time RL for Fine-Tuning Diffusion Models. Zhao et al. (2024, 2025a) introduced a continuous-time RL pipeline to fine-tune continuous diffusion models, where the action is parameterized by Gaussian with mean as the score network, and proposed an actor-critic type PPO algorithm by leveraging q-function. Gao et al. (2025) used a similar idea but enabled pretraining a diffusion model without knowing any prior of score function or learning the score. They theoretically proved that the optimal control is Gaussian distribution, and developed an actor–critic type q-learning algorithm to solve the continuous-time RL problem.

Fine-Tuning Discrete Diffusion Models via RL. In the early stage, policy gradient methods for fine-tuning score-based discrete diffusion models were proposed to tackle the non-differentiability of categorical distribution inherent in discrete diffusion model sampling or non-differentiability of

reward models when maximizing the entropy-regularized reward objective. DRAKES (Wang et al., 2025) made the originally non-differentiable trajectories differentiable using the Gumbel-Softmax trick for the general CTMC-based discrete diffusion models, but the reward function is required to be differentiable. Similar to our work, they also derived the optimal value function and HJB equation specified to CTMC-based diffusion model dynamics, while our derivation is for the more general controlled CTMC diffusion processes. Score Entropy Policy Optimization (Zekri and Boullé, 2026) fine-tuned general discrete diffusion models over non-differentiable rewards by deriving importance sampling gradient. Later, the rising of mask-based dLLMs (Nie et al., 2026b; Ye et al., 2025) drived a line of work focusing on enhancing their reasoning abilities via RL post-training. Due to fundamental modeling differences between the auto-regressive and diffusion generative paradigms for LLMs, dLLMs enable parallel or any-order decoding at the cost of intractable or computationally prohibitive trajectory likelihood. D1 (Zhao et al., 2025c) was the first to scale the reasoning ability of dLLMs with GRPO by using mean-field approximation to estimate the intractable policy log probability in a single forward pass. SPG (Wang et al., 2026b) utilized the advantage-weighted GRPO framework and estimated the log likelihood by sandwiching its evidence lower bound (ELBO) and deriving an evidence upper bound (EUBO), where the evidence bounds are estimated via Monte Carlo sampling so multiple forward passes are needed. WD1 (Tang et al., 2026) adopted the probability ratio estimation from d1 but integrated the estimation of the old and reference policy log likelihoods into an exponential-weighted objective. UniGRPO (Yang et al., 2026) estimated the log likelihoods via either a one-step unmasking from d1 or Monte Carlo estimation using the ELBO. DIFFPO (Zhao et al., 2025b) proposed a two-times mean-field approximation by conditioning on one additional latent at a randomly sampled timestep at each optimization step, yielding a better surrogate policy and a loss contained two importance sampling probability ratios. D2 (Wang et al., 2026c) approximated trajectory likelihood by products of tractable factors over blocks, reducing the number of forward passes from the number of denoising steps to the number of blocks. ESPO (Ou et al., 2026) modeled entire-sequence generation as a single action and uses the ELBO as a tractable approximation to the sequence-level likelihood. LFPO (Wei et al., 2026) overcame the likelihood intractability by directly optimizing denoising logits via contrastive positive/negative trajectories with significantly faster inference. In order to exploit the information of intermediate sequence, DISPO (Oba et al., 2026) proposed to branch from an intermediate sequence by resampling the currently masked positions from rollout-cached logits, scores the resulting completions by the reward model, and updates only the newly filled tokens. TraceRL (Wang et al., 2026d) introduced a diffusion-based value model that enhances training stability and accommodates the process rewards, providing stronger reward supervision. DTRPO (Zhang et al., 2026) adapted direct policy optimization (DPO) (Rafailov et al., 2023) to dLLM policy optimization and proposed to estimate the trajectory log likelihood by leveraging subsampling denoising time steps and block-attention mask. Although this estimator can be implemented in a single forward pass, dTRPO is an offline RL method via constructed preference set; estimating the trajectory likelihood in online RL methods like GRPO in a single forward pass is still challenging. LLaDOU (Huang et al., 2026) defined action at each step to be first determine the set of tokens to unmask and second predict the values of these tokens to obtain the next-time sequence, where some score is predicted to rank masked tokens under the current state, which can be viewed as a specific action policy in our framework. Although JustGRPO (Ni et al., 2026) indicated that any-order generation during RL training may limit the reasoning capabilities of dLLMs, we still focus on maintaining consistency between rollout generation and policy optimization to align with the continuous-time RL dynamics, as our fine-tuning framework is designed to be general rather than specific to dLLMs.

Organization of the paper: The remainder of the paper is organized as follows. In Section 2, we

formulate a general RL framework in continuous time and discrete spaces. Section 3 focuses on the application to fine-tuning discrete diffusion models, with the special case of masked diffusion models discussed in Section 4. We report the experimental results in Section 5. Concluding remarks are provided in Section 6.

2 An RL Framework in Continuous Time and Discrete Spaces

2.1 Model Formulation

We formulate CTRL for CTMCs with discrete state spaces. Let $\mathcal{S}$ be a discrete state space, and $\mathcal{A}$ be an action space. With an abuse of notation, for continuous-time dynamics (and theoretical development), $t \in [0, T]$ where $T \in \mathbb{R}_+$ is a finite time horizon; while in the final algorithmic stage where the continuous time range is discretized, $t \in [T]$ denotes the time step with $T \in \mathbb{N}_+$ being the number of time discretization steps.

We first introduce a controlled transition-rate map $R : [0, T] \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{|\mathcal{S}|}$, such that for each fixed $(t, x, a) \in [0, T] \times \mathcal{S} \times \mathcal{A}$, it holds that

$$R(t, x, a)_y \geq 0 \quad \text{for } y \neq x,\ y \in \mathcal{S}, \quad \text{and} \quad R(t, x, a)_x = -\sum_{y \neq x} R(t, x, a)_y. \tag{1}$$

If we ignore the control a in (1), R is the rate matrix of a time-inhomogeneous CTMC, with the (x, y)-th entry given by $R(t, x)_y$ at a fixed time $t \in [0, T]$ (see e.g., Liggett (2010, Chapter 2)). Thus, (1) can be understood as an action-controlled rate matrix for any fixed t. For the realization of such an R, we give an example that is linear in a, encompassing a rich family of transition-rate maps.

Example 1 (Rate with Linear Action)**.** Let the action space be $\mathcal{A} \subset \mathbb{R}^d_+$. For any matrix-valued map $\Phi : [0, T] \times \mathcal{S} \to \mathbb{R}^{|\mathcal{S}| \times d}$ such that for any $(t, x) \in [0, T] \times \mathcal{S}$, all columns of $\Phi(t, x)$ sum to zero:

$$\sum_{y \in \mathcal{S}} \Phi(t, x)_{y,i} = 0, \quad \forall i \in [d],$$

$R(t, x, a) = \Phi(t, x) \cdot a$ is a valid rate map if the off-diagonal rates are nonnegative on $\mathcal{A}$: $R(t, x, a)_y \geq 0$ for all $y \neq x$ and $a \in \mathcal{A}$, because it holds that

$$\sum_{y \in \mathcal{S}} R(t, x, a)_y = \sum_{y \in \mathcal{S}} \sum_{i=1}^{d} \Phi(t, x)_{y,i}\, a_i = \sum_{i=1}^{d} a_i \sum_{y \in \mathcal{S}} \Phi(t, x)_{y,i} = 0.$$

As will be clear in Proposition 3, this linear action structure is satisfied by the discrete diffusion embedding. □

Next, given an action sequence $\mathbf{a} = (a_t)_{t \in [0,T]} \in \mathcal{A}^{[0,T]}$, we define the controlled state dynamics $(X^{\mathbf{a}}_t)_{t \in [0,T]}$ on $\mathcal{S}$ by a CTMC with infinitesimal transition probability:

$$p_{t+\Delta t|t}(y \mid x^{\mathbf{a}}_t) = \delta\{x^{\mathbf{a}}_t, y\} + R(t, x^{\mathbf{a}}_t, a_t)_y \cdot \Delta t + o(\Delta t), \quad y \in \mathcal{S},\ x^{\mathbf{a}}_0 \sim \rho, \tag{2}$$

where $\delta\{\cdot, \cdot\}$ denotes the Kronecker delta, $x^{\mathbf{a}}_t$ is a realization of $X^{\mathbf{a}}_t$, a_t stands for the action/control at time t, and ρ is the initial state distribution. The zero-sum condition in (1) ensures that (2) is a valid infinitesimal transition probability, and (2) means that $R(t, x^{\mathbf{a}}_t, a_t)_y$ is the rate at which the probability mass moves from state $x^{\mathbf{a}}_t$ to y under the control a_t.

Commonly, we consider stochastic policy in RL by exploration to interact with and learn the unknown environment through trial and error. At each time t with the current state x_t, an action a_t is generated or sampled from the distribution $\pi(\cdot \mid t, x_t)$. A policy $\pi(\cdot \mid t, x)$ is a probability distribution over $\mathcal{A}$ given the current time-state pair (t, x). Formally, $\pi(\cdot \mid \cdot, \cdot)$ is a function $\pi : [0, T] \times \mathcal{S} \to \Delta(\mathcal{A})$. Denote by $(X_t^\pi)_{t\in[0,T]}$ the state dynamics governed by a feedback policy $\pi(\cdot \mid \cdot, \cdot)$, which is a CTMC with infinitesimal transition probability:

$$p_{t+\Delta t|t}(y \mid x_t^\pi) = \delta\{x_t^\pi, y\} + R(t, x_t^\pi, a_t^\pi)_y \cdot \Delta t + o(\Delta t), \quad y \in \mathcal{S},\ x_0^\pi \sim \rho, \tag{3}$$

where x_t^π is a realization of X_t^π and $a_t^\pi \sim \pi(\cdot \mid t, x_t^\pi)$. To evaluate such a policy π, we consider an *exploratory version* of state dynamics $(\tilde{X}_t^\pi)_{t\in[0,T]}$ in the same spirit as RL in continuous time and space (Wang et al., 2020). Define its transition mechanism by

$$p_{t+\Delta t|t}(y \mid \tilde{x}_t^\pi) = \delta\{\tilde{x}_t^\pi, y\} + \tilde{R}(t, \tilde{x}_t^\pi; \pi(\cdot \mid t, \tilde{x}_t^\pi))_y \cdot \Delta t + o(\Delta t), \quad y \in \mathcal{S},\ \tilde{x}_0^\pi \sim \rho, \tag{4}$$

where $\tilde{x}_t^\pi$ is a realization of $\tilde{X}_t^\pi$ and $\tilde{R}(t, x; \pi(\cdot)) := \int_{\mathcal{A}} R(t, x, a)\pi(a)\,\mathrm{d}a = \mathbb{E}_{a\sim\pi}\, R(t, x, a)$. Note that the external randomization of action a is removed by taking the expectation, leaving a chain governed by the averaged transition rate $\tilde{R}$. Importantly, we have the following proposition that allows us to perform theoretical analysis with the action-averaged exploratory state dynamics (4), while observing real trajectories by simulating (3). The proof is deferred to Appendix A.1.

Proposition 1. *The CTMC $(X_t^\pi)_{t\in[0,T]}$ has the same distribution as the exploratory state dynamics $(\tilde{X}_t^\pi)_{t\in[0,T]}$ under another probability measure.*

Performance Metric. The goal is to find the optimal feedback policy π^* that maximize the expected reward:

$$\pi^* = \arg\max_\pi \mathbb{E}\left[\int_0^T r(t, X_t^\pi, a_t^\pi)\,\mathrm{d}t + h(X_T^\pi) \mid X_0^\pi \sim \rho\right], \tag{5}$$

where $r : [0, T] \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the running reward function, and $h : \mathcal{S} \to \mathbb{R}$ is the terminal reward function. Also denote the value function of a policy π by

$$\begin{aligned} V(t, x; \pi) &= \mathbb{E}\left[\int_t^T r(s, X_s^\pi, a_s^\pi)\,\mathrm{d}s + h(X_T^\pi) \mid X_t^\pi = x\right] \\ &= \mathbb{E}\left[\int_s^T \tilde{r}(s, \tilde{X}_s^\pi; \pi(\cdot \mid s, \tilde{X}_s^\pi))\,\mathrm{d}s + h(\tilde{X}_T^\pi) \mid \tilde{X}_t^\pi = x\right], \end{aligned}$$

where $\tilde{r}(t, x; \pi(\cdot)) := \int_{\mathcal{A}} r(t, x, a)\pi(a)\,\mathrm{d}a = \mathbb{E}_{a\sim\pi}\, r(t, x, a)$ is the exploratory reward for any policy $\pi \in \Delta(\mathcal{A})$.

2.2 Feynman-Kac Formula

Given $a \in \mathcal{A}$, let $\mathcal{L}^a$ be the infinitesimal generator associated with the process $(X_t^a)_{t\in[0,T]}$ defined by (2) (see e.g., Benton et al. (2024, Example 2)):

$$\mathcal{L}^a f(t, x) := \frac{\partial f}{\partial t}(t, x) + \langle R(t, x, a), f(t, \cdot)\rangle, \quad f : [0, T] \times \mathcal{S} \to \mathbb{R}.$$

Here $\langle \cdot, \cdot \rangle$ denotes inner product between two vectors, i.e., $\langle R(t, x, a), f(t, \cdot)\rangle = \sum_{y\in\mathcal{S}} R(t, x, a)_y f(t, y)$.

With the action-aware infinitesimal generator in place, we now state the Feynman-Kac formula for discrete state spaces. The proof is deferred to Appendix A.3.

Lemma 1 (Feynman-Kac Formula)**.** *There exists v satisfying*

$$\mathop{\mathbb{E}}_{a\sim\pi(\cdot|t,x)}(\mathcal{L}^a v(t,x) + r(t,x,a)) = 0, \quad v(T,x) = h(x), \forall x \in \mathcal{S}.$$

Then v is the value function: $v(t,x) = V(t,x;\pi), \forall (t,x) \in [0,T]\times\mathcal{S}$.

Remark 1. For any $(t,x,a) \in [0,T]\times\mathcal{S}\times\mathcal{A}$ and $f : [0,T]\times\mathcal{S}\to\mathbb{R}$, let

$$H(t,x,a,f) := \langle R(t,x,a), f(t,\cdot)\rangle + r(t,x,a).$$

Then H is the (generalized) Hamiltonian, an analogue to Hamiltonian in classical stochastic control. The optimal value function is given by

$$V^*(t,x) := \sup_{\mathbf{a}=(a_s)_{s\in[t,T]}} \mathbb{E}\left[\int_t^T r(s,X_s^{\mathbf{a}},a_s)\,\mathrm{d}s + h(X_T^{\mathbf{a}}) \mid X_t^{\mathbf{a}} = x\right],$$

where $(t,x) \in [0,T]\times\mathcal{S}$. The associated HJB equation for V^* is:

$$\frac{\partial v}{\partial t}(t,x) + \sup_{a\in\mathcal{A}} H(t,x,a,v) = 0, \quad v(T,x) = h(x),\ \forall x \in \mathcal{S},$$

and the optimal feedback policy is $a_t^* = a^*(t,x) = \arg\max_{a\in\mathcal{A}} H(t,x,a,V^*)$, where (t,x) is the current time-state pair.

In the standard control setting where the model is fully known (i.e., functional forms of R, r and h are specified), the opitmal policy can be derived at $t = T$ and will be carried out through $[0,T]$ by dynamic programming, so the learning and exploration are not needed. In the RL setting, an additional entropy regularizer is usually adopted to encourage exploration (but for our purpose of fine-tuning diffusion models, this term is not needed):

$$V_\gamma(t,x;\pi) := \mathbb{E}\left[\int_t^T (r(s,X_s^\pi,a_s^\pi) - \gamma\log\pi(a_s^\pi))\,\mathrm{d}s + h(X_T^\pi) \mid X_t^\pi = x\right].$$

The value function $V_\gamma^*(t,x) := \sup_\pi V_\gamma(t,x;\pi)$ satisfies the exploratory HJB equation (Tang et al., 2022):

$$\frac{\partial v}{\partial t}(t,x) + \sup_\pi \mathop{\mathbb{E}}_{a\sim\pi}(H(t,x,a,v) - \gamma\log\pi(a)) = 0, \quad v(T,x) = h(x),\ \forall x \in \mathcal{S},$$

and the optimal feedback policy is $\pi_\gamma^*(a \mid t,x) \propto \exp\left(\frac{1}{\gamma}H(t,x,a,V_\gamma^*)\right)$. □

2.3 Policy Gradient

In practice, the policy is parameterized by a function family $\{\pi^\theta(\cdot \mid \cdot,\cdot)\}_\theta$. We write $(X_t^\theta)_{t\in[0,T]}$ for the process $(X_t^{\pi^\theta})_{t\in[0,T]}$ governed by the policy π^θ. Its transition probability is:

$$p_{t+\Delta t|t}^\theta(y \mid x_t^\theta) = \delta\{x_t^\theta, y\} + \tilde{R}(t,x_t^\theta;\pi^\theta(\cdot \mid t,x_t^\theta))_y\cdot\Delta t + o(\Delta t), \quad y\in\mathcal{S},\ x_0^\theta\sim\rho,$$

where x_t^θ is a realization of X_t^θ, and we denote by p_t^θ its distribution (a probability mass function). Also denote its corresponding value function by $V^\theta(t,x) = V(t,x;\pi^\theta(\cdot \mid t,x))$ and $V^\theta = \mathbb{E}_{x\sim\rho}V^\theta(0,x)$. The following theorem computes $\nabla_\theta V^\theta$, whose proof is given in Appendix A.4.

Theorem 1 (Policy Gradient)**.** *We have*

$$\nabla_\theta V^\theta = \mathbb{E}\left[\int_0^T \nabla_\theta \log \pi^\theta(a_t^\theta \mid t, X_t^\theta) \cdot q(t, X_t^\theta, a_t^\theta; \pi^\theta)\, \mathrm{d}t\right],$$

where $q(t,x,a;\pi) := \mathcal{L}^a V(t,x;\pi) + r(t,x,a) = \frac{\partial V}{\partial t}(t,x;\pi) + \langle R(t,x,a), V(t,\cdot;\pi)\rangle + r(t,x,a) = \frac{\partial V}{\partial t}(t,x;\pi) + H(t,x,a,V(\cdot,\cdot;\pi))$.

Remark 2. The $q(t,x,a;\pi)$ function defined in Theorem 1 is the instantaneous advantage rate analogue to the continuous space counterpart (Jia and Zhou, 2023). Following the presentation of Jia and Zhou (2023), given π and $(t,x,a) \in [0,T) \times \mathcal{S} \times \mathcal{A}$, consider a perturbed policy $\hat{\pi}$ of π, which takes the constant action $a \in \mathcal{A}$ on $[t, t+\Delta t)$ for $\Delta t > 0$ and then follows π on $[t+\Delta t, T]$. Specifically, the corresponding state process $X^{\hat{\pi}}$, given $X_t^{\hat{\pi}} = x$, is broken into two pieces: on $[t, t+\Delta t)$, it is X^a following the rate matrix (1), and on $[t+\Delta t, T]$ it is X^π following the transition probability (3) with initial time-space pair $(t+\Delta t, X_{t+\Delta t}^a)$. The q-function measures the rate of the performance difference between the two policies π and $\hat{\pi}$ when $t \to 0$, as shown in the following proposition whose proof is provided in Appendix A.5.

Proposition 2. *Let (Δt-parametrized) Q-function, denoted by $Q_{\Delta t}(t,x,a;\pi)$, be the expected reward of the perturbed policy $\hat{\pi}$:*

$$Q_{\Delta t}(t,x,a;\pi) := \mathbb{E}\left[\int_t^{t+\Delta t} r(s, X_s^a, a)\, \mathrm{d}s + \int_{t+\Delta t}^T r(s, X_s^\pi, a_s^\pi)\, \mathrm{d}s + h(X_T^\pi) \mid X_t^{\hat{\pi}} = x\right].$$

Then we have:

$$q(t,x,a;\pi) = \lim_{\Delta t \to 0} \frac{Q_{\Delta t}(t,x,a;\pi) - V(t,x;\pi)}{\Delta t}.$$

By the zero-sum condition in (1), we get

$$q(t,x,a;\pi) = \frac{\partial V}{\partial t}(t,x;\pi) + \sum_{y \neq x} R(t,x,a)_y (V(t,y;\pi) - V(t,x;\pi)) + r(t,x,a).$$

This expression has an intuitive interpretation of the advantage rate. The first term is independent of the action and captures the temporal change of the value function. The second term is the value difference weighted by the transition rates, reflecting the space change of the value. In particular, if $V(t,y;\pi) - V(t,x;\pi) > 0$, so that transiting from x to y is advantageous, then a larger transition rate $R(t,x,a)_y$ leads to a higher q-value. The final term corresponds to the running reward and quantifies the immediate contribution of the time-state-action triple (t,x,a). □

Based on the derived q-function, we present a PPO algorithm in Algorithm 1, where the value and policy networks are optimized iteratively. Note that we subtract the action-independent term $\partial V/\partial t$ when computing q-function $q_{b,t}$. We also present a GRPO algorithm that provides process supervision (Shao et al., 2024; Wang et al., 2026a) in Algorithm 2, eliminating the need for a value network by directly leveraging the intermediate (running) reward r and the terminal reward h. While both algorithms employ a uniform time discretization by default, the continuous-time formulation allows for arbitrary discretization schemes. Consequently, more flexible time discretizations in the final algorithmic stage can be adopted for efficiency or accuracy.

Algorithm 1 Proximal Policy Optimization for Continuous-Time Discrete-Space RL

Input: Initial policy parameter θ_0 and value network parameter ϕ_0, trajectory batch size B, number of time discretization steps T, step size m, clip parameter ϵ.

$\theta \leftarrow \theta_0$, $\phi \leftarrow \phi_0$

repeat

$\theta_{\text{old}} \leftarrow \theta$, $\phi_{\text{old}} \leftarrow \phi$

Collect B trajectories $\{\tau_b\}_{b=1}^{B}$, where $\tau_b = \{x_{b,t}, a_{b,t}, r_{b,t}\}_{t=0}^{T-1} \cup \{t_{b,T}, x_{b,T}\}$ for $t \in [T-1] \cup \{0\}$ by running policy π^{θ_n}

Compute rewards-to-go $\hat{r}_{b,t} = m \sum_{t'=t}^{T-1} r_{b,t'} + r_{b,T}$ for all $b \in [B]$ and $t \in [T-1] \cup \{0\}$

Perform multiple value network updates starting from ϕ_{old}:

$$\phi \leftarrow \arg\min_{\phi} \frac{1}{B} \sum_{b=1}^{B} \frac{1}{T} \sum_{t=0}^{T-1} |V^{\phi}(tm, x_{b,t}) - \hat{r}_{b,t}|^2$$

Compute $q_{b,t} = \sum_{y \neq x_{b,t}} R(tm, x_{b,t}, a_{b,t})_y \cdot (V^{\phi_{n+1}}(tm, y) - V^{\phi_{n+1}}(tm, x_{b,t})) + r(tm, x_{b,t}, a_{b,t})$ for all $b \in [B]$ and $t \in [T-1] \cup \{0\}$

Perform multiple policy updates starting from θ_{old}:

$$\theta \leftarrow \arg\max_{\theta} \frac{1}{B} \sum_{b=1}^{B} \frac{1}{T} \sum_{t=0}^{T-1} \min(\rho_{b,t}^{\theta} \cdot q_{b,t}, \text{clip}(\rho_{b,t}^{\theta}, 1-\epsilon, 1+\epsilon) \cdot q_{b,t}),$$

where $\rho_{b,t}^{\theta} = \frac{\pi^{\theta}(a_{b,t}|tm, x_{b,t})}{\pi^{\theta_{\text{old}}}(a_{b,t}|tm, x_{b,t})}$

until Convergence

Output: θ

Note: Here for sampled trajectories, $x_{b,0} \sim \rho$, $a_{b,t} \sim \pi^{\theta_{\text{old}}}(\cdot \mid tm, x_{b,t})$, $x_{b,t+1} \sim \text{Cat}(\cdot; e_{x_{b,t}} + R(tm, x_{b,t}, a_{b,t}) \cdot m)$, $r_{b,t} = r(tm, x_{b,t}, a_{b,t})$ for all b, t, and $r_{b,T} = h(x_{b,T})$.

3 A General Framework for Fine-Tuning Score-Based Discrete Diffusion Models

3.1 Discrete Diffusion Models

In score-based discrete diffusion models (Lou et al., 2024; Sun et al., 2023; Campbell et al., 2022), the forward and reverse processes are formulated as CTMCs. Let $Q_t \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ be the transition rate matrix of the forward process $(Y_t)_{t \in [0,T]}$ at time $t \in [0,T]$ such that $\text{Law}(Y_0) = p_{\text{data}}$ and $\text{Law}(Y_T) \approx p_{\text{noise}}$. The reverse process $(X_t)_{t \in [0,T]}$ is the exact time reversal of $(Y_t)_{t \in [0,T]}$ (Kelly, 2011), that is, $X_t \overset{\text{d}}{=} Y_{T-t}$, with the rate matrix $Q_t^{\leftarrow}$ given by

$$Q_t^{\leftarrow}(x,y) = Q_{T-t}(y,x) \cdot s(T-t,x)_y \quad \text{for } y \neq x,\ y \in \mathcal{S}, \text{ and } Q_t^{\leftarrow}(x,x) = -\sum_{y \neq x} Q_t^{\leftarrow}(x,y),\ \forall x \in \mathcal{S}.$$

Here $s(t,x) := (q_t(y)/q_t(x))_{y \neq x} \in \mathbb{R}_+^{|\mathcal{S}|-1}$ is called *discrete score function* (Lou et al., 2024; Meng et al., 2022), which is a collection of probability ratios of forward marginal distributions. Here, q_t is the forward marginal at time t, i.e., the distribution (a probability mass function) of Y_t.

In practice, the score $s(t,x)$ is parameterized by a function family $\{s_\theta(t,x)\}_\theta$, and is then learned by score matching methods, such as denoising score entropy (Lou et al., 2024; Benton et al., 2024).

Algorithm 2 Group Relative Policy Optimization for Continuous-Time Discrete-Space RL

Input: Initial policy parameter θ_0, trajectory batch size B, trajectory group size G, number of time discretization steps T, step size m, clip parameter ϵ.

$\theta \leftarrow \theta_0$

repeat

$\theta_{\text{old}} \leftarrow \theta$

Sample B initial states $\{x_b\}_{b=1}^{B} \sim \rho$

For each x_b, collect G trajectories, $\{\tau_{b,g}\}_{g=1}^{G}$, starting from x_b, where $\tau_{b,g} = \{x_{b,g,t}, a_{b,g,t}, r_{b,g,t}\}_{t=0}^{T-1} \cup \{t_T, x_{b,g,T}, r_{b,g,T}\}$ by running policy $\pi^{\theta_{\text{old}}}$

Compute group mean $\mu_b = \frac{1}{GT}\sum_{g=1}^{G}\sum_{t=0}^{T-1} r_{b,g,t}$ and $\nu_b = \frac{1}{G}\sum_{g=1}^{G} r_{b,g,T}$, group standard deviation $\sigma_b = \sqrt{\frac{1}{GT}\sum_{g=1}^{G}\sum_{t=0}^{T-1}(r_{b,g,t}-\mu_b)^2}$ and $\omega_b = \sqrt{\frac{1}{G}\sum_{g=1}^{G}(r_{b,g,T}-\nu_b)^2}$, normalized reward $\hat{r}_{b,g,t} = \frac{r_{b,g,t}-\mu_b}{\sigma_b}$ and $\hat{r}_{b,g,T} = \frac{r_{b,g,T}-\nu_b}{\omega_b}$, and advantage $A_{b,g,t} = \frac{1}{T}\sum_{t'=t}^{T-1}\hat{r}_{b,g,t'} + \hat{r}_{b,g,T}$

Perform multiple policy updates starting from θ_{old}:

$$\theta \leftarrow \arg\max_{\theta} \frac{1}{B}\sum_{b=1}^{B}\frac{1}{G}\sum_{g=1}^{G}\frac{1}{T}\sum_{t=0}^{T-1}\min(\rho_{b,g,t}^{\theta}A_{b,g,t}, \text{clip}(\rho_{b,g,t}^{\theta}, 1-\epsilon, 1+\epsilon)A_{b,g,t}),$$

where $\rho_{b,g,t}^{\theta} = \frac{\pi^{\theta}(a_{b,g,t}|tm, x_{b,g,t})}{\pi^{\theta_{\text{old}}}(a_{b,g,t}|tm, x_{b,g,t})}$

until Convergence

Output: θ

Note: For sampled trajectories, $x_{b,g,0} = x_b$, $a_{b,g,t} \sim \pi^{\theta_{\text{old}}}(\cdot \mid tm, x_{b,g,t})$, $x_{b,g,t+1} \sim \text{Cat}(\cdot; e_{x_{b,g,t}} + R(tm, x_{b,g,t}, a_{b,g,t})\cdot m)$, $r_{b,g,t} = r(tm, x_{b,g,t}, a_{b,g,t})$, and $r_{b,g,T} = h(x_{b,g,T})$ for all $b \in [B], g \in [G], t \in [T-1]\cup\{0\}$.

Once we have a pretrained estimator $s_{\theta_{\text{pre}}}(\cdot,\cdot)$, we can perform generative modeling by discretizing the continuous-time reverse process:

$$p_{t+\Delta t|t}^{\theta_{\text{pre}}}(y \mid x_t) = \delta\{x_t, y\} + Q_{T-t}(y, x_t)s_{\theta_{\text{pre}}}(T-t, x_t)_y \cdot \Delta t + o(\Delta t), \quad y \neq x_t,\ y \in \mathcal{S},\ x_0 \sim p_{\text{noise}}, \tag{6}$$

where x_t is a realization of X_t and $p_t^{\theta_{\text{pre}}}$ is the marginal distribution of the pretraining sampling process with score $s_{\theta_{\text{pre}}}(\cdot,\cdot)$ at time t.

3.2 Score as Action

To draw connections between CTRL and discrete diffusion models, we compare the RL process (4) with the diffusion sampling process (6). Given the forward diffusion rate matrix $(Q_t)_{t\in[0,T]}$, we choose the transition-rate map R in the RL dynamics (4) as

$$R(t, x, a)_y = Q_{T-t}(y, x)\cdot a_y \quad \text{for } y \neq x,\ y \in \mathcal{S}, \tag{7}$$

for any $(t, x, a) \in [0, T] \times \mathcal{S} \times \mathcal{A}$. Here the action space is specified as $\mathcal{A} = \mathbb{R}_+^{|\mathcal{S}|-1}$, because we are only concerned with the off-diagnal entries. The following proposition shows that the rate map (7) belongs to the family of rate maps in Example 1. We give the proof in Appendix A.6.

Proposition 3. *Given any $(t, x) \in [0, T] \times \mathcal{S}$, index the action by the off-diagonal targets, $a =$*

$(a_y)_{y\neq x} \in \mathbb{R}_+^{|\mathcal{S}|-1}$ *(so* $d = |\mathcal{S}| - 1$ *in Example 1), and set*

$$\Phi(t,x)_{y,i} := \begin{cases} Q_{T-t}(i,x), & y = i, \\ -Q_{T-t}(i,x), & y = x, \\ 0, & \text{otherwise}, \end{cases} \qquad \text{for all } y \in \mathcal{S} \text{ and } i \in \mathcal{S} : i \neq x.$$

Then each column of $\Phi(t,x)$ *sums to zero, and the rate map* (7) *satisfies* $R(t,x,a) = \Phi(t,x) \cdot a$ *for all* $(t,x,a) \in [0,T] \times \mathcal{S} \times \mathcal{A}$.

Suppose that we have access to a pretrained score function $s_{\theta_{\text{pre}}}(\cdot,\cdot)$, and hence a pretrained model. Let $\pi^{\theta_{\text{pre}}}(\cdot \mid t,x)$ be a deterministic policy $\delta_{s_{\theta_{\text{pre}}}(T-t,x)}$; that is, if $a_t^{\theta_{\text{pre}}} \sim \pi^{\theta_{\text{pre}}}(\cdot \mid t, x_t)$, then $a_t^{\theta_{\text{pre}}} = s_{\theta_{\text{pre}}}(T-t, x_t)$. So the reverse CTMC (6) becomes

$$p^{\theta_{\text{pre}}}_{t+\Delta t|t}(y \mid x_t) = \delta\{x_t, y\} + R(t, x_t, a_t^{\theta_{\text{pre}}})_y \cdot \Delta t + o(\Delta t) \quad \text{for } y \in \mathcal{S}.$$

As a result, analogous to Zhao et al. (2025a), the score function is replaced by the action, and the problem of finding the optimal score becomes finding the optimal action/policy, which can be tackled by policy optimization.

3.3 Post-Training

Generally, for a parameterized policy $\pi^\theta(\cdot \mid \cdot,\cdot)$, we denote the control $a_t^\theta \sim \pi^\theta(\cdot \mid t, X_t^\theta)$, where X_t^θ is short for the reverse process $X_t^{\pi^\theta}$ driven by π^θ:

$$p^\theta_{t+\Delta t|t}(y \mid x_t^\theta) = \delta\{x_t^\theta, y\} + R(t, x_t^\theta, a_t^\theta)_y \cdot \Delta t + o(\Delta t) \quad \text{for } y \in \mathcal{S},\ x_0^\theta \sim p_{\text{noise}},$$

where x_t^θ is a realization of X_t^θ and p_t^θ is the distribution of X_t^θ. The post-training objective is:

$$\max_\theta \mathbb{E}\left[\text{TRF}(X_T^\theta) + \int_0^T \text{IRF}(X_t^\theta)\,\mathrm{d}t - \beta D_{\text{KL}}(\mathbb{P}^\theta \,\|\, \mathbb{P}^{\theta_{\text{pre}}})\right]. \tag{8}$$

Here $\text{TRF}(\cdot)$ denotes the terminal reward function for the final output X_T^θ, $\text{IRF}(\cdot)$ denotes the intermediate reward function (possibly time-dependent) for the intermediate state X_t^θ, $\mathbb{P}^\theta$ and $\mathbb{P}^{\theta_{\text{pre}}}$ are the path measures of the processes $(X_t^\theta)_{t\in[0,T]}$ and $(X_t^{\theta_{\text{pre}}})_{t\in[0,T]}$, respectively, and $\beta > 0$ is a penalty hyperparameter. The following theorem connects the post-training objective (8) with the RL objective (5). The proof will be given in Appendix A.7.

Theorem 2. *The KL divergence between* $\mathbb{P}^\theta$ *and* $\mathbb{P}^{\theta_{\text{pre}}}$ *can be expressed as:*

$$D_{\text{KL}}(\mathbb{P}^\theta \,\|\, \mathbb{P}^{\theta_{\text{pre}}}) = \mathop{\mathbb{E}}_{X_t^\theta \sim p_t^\theta} \int_0^T \sum_{y \neq X_t^\theta} Q_{T-t}(y, X_t^\theta) D_I\left(\mathop{\mathbb{E}}_{a_t^\theta \sim \pi^\theta(\cdot|t,X_t^\theta)} (a_t^\theta)_y \,\|\, s_{\theta_{\text{pre}}}(T-t, X_t^\theta)_y\right) \mathrm{d}t,$$

where $D_I(\cdot\,\|\,\cdot)$ *is the generalized I-divergence (Amari, 2012) given by* $D_I(x\,\|\,y) = -x + y + x\log(x/y)$.

By Theorem 2, the objective in (8) is

$$\mathop{\mathbb{E}}_{X_t^\theta \sim p_t^\theta} \left[\underbrace{\text{TRF}(X_T^\theta)}_{h(X_T^\theta)} + \int_0^T \left(\underbrace{\text{IRF}(X_t^\theta) - \beta \sum_{y \neq X_t^\theta} Q_{T-t}(y, X_t^\theta) D_I\Big(\mathop{\mathbb{E}}_{a_t^\theta \sim \pi^\theta(\cdot|t,X_t^\theta)} (a_t^\theta)_y \,\|\, s_{\theta_{\text{pre}}}(T-t, X_t^\theta)_y\Big)}_{\tilde{r}(t, X_t^\theta; \pi^\theta(\cdot|t,X_t^\theta))} \right) \mathrm{d}t \right]. \tag{9}$$

The objective (9) coincides with the RL objective (5). With this expression, for any $(t, x) \in [0, T] \times \mathcal{S}$, we define the value function

$$V^\theta(t,x) = \mathop{\mathbb{E}}_{X_s^\theta \sim p_s^\theta} \Bigg[\mathrm{TRF}(X_T^\theta) + \int_t^T \Bigg(\mathrm{IRF}(X_s^\theta) - \beta \sum_{y \neq X_s^\theta} Q_{T-s}(y, X_s^\theta) D_I \Bigg(\mathop{\mathbb{E}}_{a_s^\theta \sim \pi^\theta(\cdot \mid s, X_s^\theta)} (a_s^\theta)_y \,\Bigg\|\, s_{\theta_{\mathrm{pre}}}(T-s, X_s^\theta)_y \Bigg) \Bigg) \mathrm{d}s \,\Bigg|\, X_t^\theta = x \Bigg]. \quad (10)$$

To design the fine-tuning algorithms, it suffices to inject the functions R, r, and h in (7) and (10) into Algorithms 1 and 2, and replace the initial distribution ρ with p_{noise} and the initial parameter θ_0 with θ_{pre}. Notably, our framework naturally accommodates arbitrary intermediate rewards, does not require the reward function to be differentiable, and avoids the non-differentiability issue of sampling from a categorical distribution when maximizing the reward (see Wang et al. (2025, Section 4.2)).

4 Fine-Tuning Masked Diffusion Models

Although our framework applies to general high-dimensional CTMC-based diffusion models as discussed in Section 3, our focus will be on the widely adopted masked diffusion models (MDMs). In this setting, the score function has a direct correspondence with the denoiser, allowing the policy to be naturally parameterized as a probability distribution over the simplex.

Notations. We use lowercase letters to denote scalars, and boldface lowercase letters to denote vectors. The i-th entry of a vector $\mathbf{x}$ is denoted by $\mathbf{x}^i$, or simply x^i when the context is clear. We use $\mathbf{x}^{\setminus i}$ to refer to all dimensions of $\mathbf{x}$ except the i-th, and $\mathbf{x}^{\setminus i} \odot \hat{x}^i$ to denote a vector whose i-th dimension takes the value $\hat{x}^i$, while the other dimensions remain as $\mathbf{x}^{\setminus i}$. For a positive integer n, we denote $\mathbf{1}_n \in \mathbb{R}^n$ as the column vector of ones, and $I_n \in \mathbb{R}^{n \times n}$ as the identity matrix. The notation $\mathbf{e}_j$ refers to a one-hot column vector with a 1 in the j-th position. We use $\mathrm{Cat}(\cdot; \mathbf{p})$ for a categorical distribution over a one-hot column vector with probabilities given by the column vector $\mathbf{p}$. $\mathbf{1}\{\cdot\}$ is the indicator function.

4.1 Preliminaries on Masked Diffusion Models

Let $\mathcal{S} = (\mathcal{V} \cup \{\mathtt{m}\})^L$ be the state space, where $\mathcal{V} := \{1, \cdots, V\}$ is the vocabulary space, $\mathtt{m}$ is the special "mask" token, and L is the sequence length. The continuous time horizon is set to 1. We use superscript $i \in [L]$ for the sequence index, subscript $j \in \mathcal{V}$ for the token vocabulary index, subscript t for the current time (continuous-time, $t \in [0, 1]$) or time step (discrete-time, $t \in [T]$), and subscript $g \in [G]$ for a specific sample trajectory.

Forward Process. The forward process is factorized as $p_{t|0}(\mathbf{x}_t \mid \mathbf{x}_0) = \prod_{i=1}^L p_{t|0}(x_t^i \mid x_0^i)$, where

$$p_{t|0}(x_t^i \mid x_0^i) = \mathrm{Cat}(x_t^i; \alpha_t \mathbf{e}_{x_0^i} + (1 - \alpha_t)\mathbf{e}_{\mathtt{m}}), \quad t \in [0, 1].$$

α_t is the noise scheduled to satisfy $\alpha_0 \approx 1$ and $\alpha_1 \approx 0$. A common choice is $\alpha_t = 1 - t$ (Nie et al., 2026b), which we adopt throughout this work. The forward process interpolates the clean data and pure mask noise. The resulting forward rate matrix for each dimension is

$$Q_t^{\mathrm{tok}} = -\frac{1}{\alpha_t} \frac{\partial \alpha_t}{\partial t} (\mathbf{1}_{V+1} \cdot \mathbf{e}_{\mathtt{m}}^\top - I_{V+1}) \in \mathbb{R}^{(V+1) \times (V+1)},$$

and the full-dimensional rate matrix $Q_t = \oplus_{i=1}^{L} Q_t^{\text{tok}} \in \mathbb{R}^{(V+1)^L \times (V+1)^L}$ is the Kronecker sum of L dimension-wise rate matrices, with non-diagnal entries given by

$$Q_t(\mathbf{x}, \mathbf{x}^{\backslash i} \odot \mathtt{m}) = -\frac{1}{\alpha_t}\frac{\partial \alpha_t}{\partial t} = \frac{1}{1-t}, \quad x^i \neq \mathtt{m}.$$

Reverse Process. Given $\mathbf{x}_0 \sim p_{\text{data}}$, for $0 \leq s \leq t$, the reverse transition probability is

$$p_{s|t,0}(x_s^i \mid \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \text{Cat}(x_s^i; \mathbf{e}_{x_t^i}), & x_t^i \neq \mathtt{m}; \\ \text{Cat}(x_s^i; \frac{1-\alpha_s}{1-\alpha_t}\mathbf{e}_{\mathtt{m}} + \frac{\alpha_s - \alpha_t}{1-\alpha_t}\mathbf{e}_{x_0^i}), & x_t^i = \mathtt{m}. \end{cases}$$

Note that conditional on $\mathbf{x}_0$, if $x_t^i = \mathtt{m}$, then with probability $\frac{\alpha_s - \alpha_t}{1-\alpha_t}$, x_t^i will jump to x_0^i at time s; once x_t^i is unmasked, it remains the same until $t = 0$. Thus, one can parameterize a denoiser $\mathbf{p}_\theta(\mathbf{x}_t)$, which takes all the dimensions of $\mathbf{x}_t$ as the input and predicts masked tokens ($x_t^i = \mathtt{m},\ i \in [L]$) simultaneously. To be more specific, for any $i \in [L]$, a neural network $\mathbf{p}_\theta^{(i)}(\mathbf{x}_t)$, the i-th output of $\mathbf{p}_\theta(\mathbf{x}_t)$, outputs a probability distribution of x_0^i:

$$\mathbf{p}_\theta^{(i)}(\mathbf{x}_t) = \begin{cases} \text{softmax}(\mathbf{f}_\theta^{(i)}(\mathbf{x}_t)), & i \in [L] : x_t^i = \mathtt{m}, \\ \mathbf{e}_{x_t^i}, & i \in [L] : x_t^i \neq \mathtt{m} \end{cases} \in \Delta^{V-1},$$

where $\Delta^{V-1} := \{\mathbf{x} \in \mathbb{R}_+^V : \sum_{i=1}^V \mathbf{x}^i = 1\} \subset \mathbb{R}^V$ is the $(V-1)$-dimensional simplex. For $i : x_t^i = \mathtt{m}$, $\mathbf{f}_\theta^{(i)}(\mathbf{x}_t) \in \mathbb{R}^V$ is the model logits and $\mathbf{p}_\theta^{(i)}(\mathbf{x}_t) \in \Delta^{V-1}$ is a probability vector with the j-th component, denoted by $p_\theta^{(i)}(\mathbf{x}_t)_j$ or $p_\theta^{(i)}(j \mid \mathbf{x}_t)$, predicting the conditional probability $p_{0|t}(x_0^i = j \mid \mathbf{x}_t)$ for all $j \in \mathcal{V}$. Notably, by the identity (Shi et al., 2024; Ou et al., 2025; Hasan et al., 2026):

$$s(t, \mathbf{x}_t)_{i,j} := \frac{p_t(\mathbf{x}_t^{\backslash i} \odot j)}{p_t(\mathbf{x}_t)} = \delta\{\mathtt{m}, j\} + \frac{\alpha_t}{1-\alpha_t} p_{0|t}(x_0^i = j \mid \mathbf{x}_t), \quad x_t^i = \mathtt{m},$$

parameterizing the denoising probability is equivalent to parametrizing the score function up to some scalar:

$$s_\theta(t, \mathbf{x}_t)_{i,j} = \frac{\alpha_t}{1-\alpha_t} p_\theta^{(i)}(\mathbf{x}_t)_j, \quad x_t^i = \mathtt{m},\ j \in \mathcal{V}.$$

Here the denoiser network does not depend on the time t because the partially unmasked sequence $\mathbf{x}_t$ implicitly contained the time information (Shi et al., 2024; Kim et al., 2025; Nie et al., 2026b). In the remainder of this work, we always consider the denoiser parameterization $\mathbf{p}_\theta(\mathbf{x}_t)$ (which will be replaced by actions) instead of the score parameterization. Also, the reverse rate matrix is

$$Q_{1-t}^{\leftarrow}(\mathbf{x}, \mathbf{x}^{\backslash i} \odot j) = -\frac{1}{\alpha_t}\frac{\partial \alpha_t}{\partial t} s(t, \mathbf{x})_{i,j} = -\frac{1}{1-\alpha_t}\frac{\partial \alpha_t}{\partial t} p_{0|t}(x_0^i = j \mid \mathbf{x}) = \frac{1}{t} p_{0|t}(x_0^i = j \mid \mathbf{x}), \quad x^i = \mathtt{m},\ j \in \mathcal{V},$$

Hence, the reverse process parametrized by $\mathbf{p}_\theta$ has rate matrix

$$Q_t^\theta(\mathbf{x}, \mathbf{x}^{\backslash i} \odot j) = \frac{1}{1-t} p_\theta^{(i)}(\mathbf{x})_j, \quad x^i = \mathtt{m},\ j \in \mathcal{V}. \tag{11}$$

4.2 Post-Training

Policy Parameterization. In RL, an action $\mathbf{a}_t$ is sampled from a parameterized stochastic policy to encourage exploration: $\mathbf{a}_t \sim \pi^\theta(\cdot \mid t, \mathbf{x}_t)$ given the current time-state pair $(t, \mathbf{x}_t)$. Since the reverse

masked diffusion dynamics (11) is fully determined by the denoiser probability vector, we consider the action space $\mathcal{A} = (\Delta^{V-1})^L$. Then $\pi(\cdot \mid t, \mathbf{x})$ is a distribution over the product simplex $(\Delta^{V-1})^L$. For simplicity, we let the parameterized policy π^θ factorize over all dimensions:

$$\pi^\theta(\mathbf{a}_t \mid t, \mathbf{x}_t) = \prod_{i=1}^{L} \pi^{\theta,(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t),$$

where $\pi^{\theta,(i)}(\cdot \mid t, \mathbf{x}_t)$ is a distribution over Δ^{V-1}, and $\mathbf{a}_t^{(i)} \in \Delta^{V-1}$ is a probability vector over $\mathcal{V}$ for all $i \in [L]$. At time t, we sample from $\mathrm{Cat}(\cdot; \mathbf{a}_t^{(i)})$, instead of $\mathrm{Cat}(\cdot; \mathbf{p}_\theta^{(i)}(\mathbf{x}_t))$, to predict the clean token at the i-th position. So $\mathbf{a}_t^{(i)}$ can be viewed as performing exploration over the denoiser $\mathbf{p}_\theta^{(i)}(\mathbf{x}_t)$. As a result, we need to specify the dimension-wise policies $\pi^{\theta,(i)}(\cdot \mid t, \mathbf{x})$ for all i.

We emphasize that the policy parameterization must be chosen carefully. In particular, $\pi^\theta(\cdot \mid t, \mathbf{x})$ should have a tractable likelihood, and the resulting probability ratio $\rho_t^\theta := \pi^\theta(\mathbf{a}_t \mid t, \mathbf{x}_t)/\pi^{\theta_{\text{old}}}(\mathbf{a}_t \mid t, \mathbf{x}_t)$ should be amenable to stable optimization. Several policy parameterizations that perform full-logit exploration and induce distributions supported on the full simplex, including Dirichlet, temperature softmax, and logistic-normal policies, are summarized in Appendix B.1, together with their practical implementation details. However, for masked dLLMs with a large vocabulary size ($|\mathcal{V}| \approx 1.2 \times 10^5$), it is inefficient to explore the full simplex: the learned denoiser distribution $\mathbf{p}_\theta$ is typically highly concentrated, with probabilities spanning several orders of magnitude and effectively residing on a much lower-dimensional region of the simplex, often near a small number of vertices. As discussed in Appendix B.1, applying full-simplex exploration in this regime amounts to exploring the entire vocabulary, which can lead to unstable optimization due to the large vocabulary size. Motivated by the empirical success of d1 (Zhao et al., 2025c), we consider a policy family whose support is concentrated on the V vertices of the simplex Δ^{V-1}, assigning probability mass directly to the vertices rather than exploring its interior through a continuous density. Nevertheless, the CTRL perspective suggests that broader exploration of the denoiser distribution still be beneficial. Consequently, while vertex-supported policies are used for optimization, we retain simplex-level exploration during rollout generation in our experiments.

Following prior works (Zhao et al., 2025c,b; Wang et al., 2026b,d; Oba et al., 2026), we focus on the GRPO algorithm for fine-tuning masked dLLMs to avoid training an expensive value network in PPO, though we also describe a possible value function approximation without training a network in Appendix C. Let $(\mathbf{x}_t)_{t=0}^T$ be a decoding trajectory, where $\mathbf{x}_0$ is a prompt concatenated with mask tokens of the completion length L, $\mathbf{x}_T$ is the final output corresponding to the prompt, and $T \in \mathbb{Z}_+$ is the number of denoising steps. The trajectory can be generated by any inference strategy, while we adopt a semi-autoregressive strategy with low-confidence remasking and random sampling (Nie et al., 2026b; Zhao et al., 2025c). Specifically, at time step t, we first select the k masked positions with the highest confidence score in the current block $B_t \subset [L]$, denoted by

$$\mathcal{M}_t^{\text{TopkConf}} := \left\{ i \in B_t : x_t^i = \mathtt{m},\ c_t^i \in \mathrm{Topk}\left(\{c_t^i\}_{i \in B_t : x_t^i = \mathtt{m}} \right) \right\}, \quad c_t^i := \max_{j \in \mathcal{V}} p_\theta^{(i)}(\mathbf{x}_t)_j,$$

to unmask, and then sample according to $\mathrm{Cat}(\cdot; \mathbf{a}_t^{(i)})$ for all $i \in \mathcal{M}_t^{\text{TopkConf}}$:

$$\mathbf{a}_t^{(i)} \sim \pi^{\theta,(i)}(\cdot \mid t, \mathbf{x}_t) \quad \text{then} \quad \mathbf{a}_t^{(i)} = \begin{cases} \mathbf{e}_j \quad \text{w.p. } p_\theta^{(i)}(\mathbf{x}_t)_j,\ j \in \mathcal{V}, & \text{for } i \in \mathcal{M}_t^{\text{TopkConf}} \\ \mathbf{e}_{x_t^i}, & \text{for } i \notin \mathcal{M}_t^{\text{TopkConf}} \end{cases} \in \Delta^{V-1}. \quad (12)$$

The corresponding intermediate and terminal reward functions are given by (9):

$$\tilde{r}(t, \mathbf{x}_t; \pi^\theta(\cdot \mid t, \mathbf{x}_t)) = \mathrm{IRF}(\mathbf{x}_t) - \frac{\beta}{1-t} \sum_{i: x_t^i = \mathtt{m}} D_{\mathrm{KL}}(\mathbf{p}_\theta^{(i)}(\mathbf{x}_t) \,\|\, \mathbf{p}_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{x}_t)) \quad \text{and} \quad h(\mathbf{x}_1) = \mathrm{TRF}(\mathbf{x}_1), \tag{13}$$

and the probability ratio is

$$\rho_t^\theta = \frac{\pi^\theta(\mathbf{a}_t \mid t, \mathbf{x}_t)}{\pi^{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid t, \mathbf{x}_t)} = \frac{\prod_{i \in \mathcal{M}_t^{\mathrm{TopkConf}}} \pi^{\theta,(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t)}{\prod_{i \in \mathcal{M}_t^{\mathrm{TopkConf}}} \pi^{\theta_{\mathrm{old}},(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t)} = \prod_{i \in \mathcal{M}_t^{\mathrm{TopkConf}}} \frac{p_\theta^{(i)}(x_T^i \mid \mathbf{x}_t)}{p_{\theta_{\mathrm{old}}}^{(i)}(x_T^i \mid \mathbf{x}_t)}. \tag{14}$$

Thus, the GRPO loss is

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{(\mathbf{x}_{g,0}, \mathbf{x}_{g,1}, \cdots, \mathbf{x}_{g,T})_{g=1}^G \sim \pi^{\theta_{\mathrm{old}}}} \left[\frac{1}{G} \sum_{g=1}^{G} \frac{1}{T} \sum_{t=1}^{T} \min\left(\rho_{g,t}^\theta A_{g,t}, \mathrm{clip}\left(\rho_{g,t}^\theta, 1-\epsilon, 1+\epsilon \right) A_{g,t} \right) \right], \tag{15}$$

where $T = L/k$, $\rho_{g,t}^\theta = \prod_{i \in \mathcal{M}_{g,t}^{\mathrm{TopkConf}}} p_\theta^{(i)}(x_{g,T}^i \mid \mathbf{x}_{g,t}) / p_{\theta_{\mathrm{old}}}^{(i)}(x_{g,T}^i \mid \mathbf{x}_{g,t})$, and the computation of $A_{g,t}$ is implied by (13) and Algorithm 2.

Practical Considerations. Our CTRL-based GRPO loss in (15) cannot be evaluated in a single causal forward pass over an entire denoising trajectory due to the evaluation of $\{\rho_{g,t}^\theta\}_{t \in [T]}$. In online GRPO-based methods for dLLMs, this leads to a fundamental trade-off: on one hand, accurate likelihood computation under a CTRL formulation and the incorporation of intermediate rewards to optimize intermediate steps both require conditioning on the intermediate state $\mathbf{x}_t$ at each time step t; on the other hand, this necessitates multiple forward passes along the trajectory, incurring computational and memory costs proportional to the prompt batch size, group size G, and number of denoising steps T, as all intermediate activations must be stored for backpropagation. Existing approaches make different compromises. For instance, d1 (Zhao et al., 2025c) computed the loss using a single forward pass by conditioning only on the perturbed prompt. DTRPO (Zhang et al., 2026) estimated the trajectory probability via a single forward pass of a re-masked final state; however, it is designed for offline DPO-style objectives and is difficult to extend to online GRPO settings. D2 (Wang et al., 2026c) utilized block composite likelihood as an estimator to reduce the number of forward passes.

In practice, to reduce the number of required forward passes in our framework, we propose trajectory subsampling, which yields an unbiased estimator of the full GRPO loss in (15). At each optimizer update we draw a subset $\mathcal{T} \subset [T]$ of size $N = |\mathcal{T}|$ uniformly at random without replacement, the same subset being reused for the policy, old, and reference passes so the ratio (14) is consistent. We then form the subsampled estimator

$$\hat{\mathcal{L}}_{\mathcal{T}}(\theta) = \mathop{\mathbb{E}}_{(\mathbf{x}_{g,0}, \mathbf{x}_{g,1}, \cdots, \mathbf{x}_{g,T})_{g=1}^G \sim \pi^{\theta_{\mathrm{old}}}} \left[\frac{1}{G} \sum_{g=1}^{G} \frac{1}{N} \sum_{t \in \mathcal{T}} \min\left(\rho_{g,t}^\theta A_{g,t}, \mathrm{clip}\left(\rho_{g,t}^\theta, 1-\epsilon, 1+\epsilon \right) A_{g,t} \right) \right],$$

which reduces the number of forward passes from T to N per rollout. One can easily see that this estimator is unbiased by the fact that $\mathbb{P}(t \in \mathcal{T}) = N/T$: $\mathbb{E}_{\mathcal{T}}[\hat{\mathcal{L}}_{\mathcal{T}}(\theta)] = \mathcal{L}(\theta)$. When N is too small, the model fails to reliably converge to competitive performance due to inaccurate loss estimation. Conversely, increasing N substantially raises the number of forward passes, thereby reducing training efficiency. Our experiments indicate that $N = 8$ provides a favorable trade-off, achieving performance comparable to $N = 16$ and $N = 32$ while requiring substantially fewer function evaluations. More theoretical derivation and practical implementation details including the ablation study on N are provided in Appendix B.2.

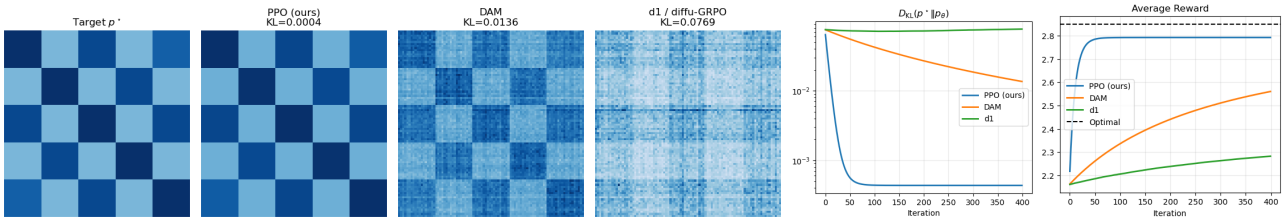

Figure 1: Three-way comparison. PPO and DAM converge toward p^*; d1 plateaus far above it.

5 Experiments

We showcase our method for fine-tuning MDMs on two benchmarks: 2-dimensional synthetic example with PPO, and higher-dimensional reasoning and coding dataset for dLLMs with GRPO.

5.1 Synthetic Example

Setup. We consider a synthetic example described in Discrete Adjoint Matching (DAM) (So et al., 2026), where we fine-tune an MDM on a 90×90 discrete grid so that its terminal sampling distribution matches a prescribed *checkerboard* target (the first subfigure in Figure 1). The target is the entropy-regularized optimum $p^* \propto p^{\text{base}}\, e^{h/\beta}$. We solve the task with our proposed PPO algorithm.

We use a two-dimensional discrete space $\mathcal{X} = \{\mathtt{m}, 1, 2, \ldots, 90\}^2$, and a state is a pair $\mathbf{x} = (x^1, x^2)$ in which each coordinate is either masked or holds a token. The MDM begins at the fully masked state $\mathbf{X}_0 = (\mathtt{m}, \mathtt{m})$ and unmasks exactly one token per step, so there are $T = 2$ denoising steps. The base/pretrained denoiser is uniform: at every masked position it predicts a uniform distribution over the V tokens, so the base terminal distribution p_1^{base} is uniform on the grid and all target structure is induced by the reward. We use the softmax exponential-temperature exploration policy described in Appendix B.1.2. The PPO advantage is the value difference across one unmasking step. Because only the states reachable in the two-step process matter, we store an explicit logit table indexed by the context (position, other coordinate), where the other coordinate is a revealed token in $\{1, \cdots, 90\}$ or the mask token, and we evaluate the value V^θ exactly by computing the surrogate's expectation over all V tokens for each visited context in closed-form. Further implementation details are provided in Appendix D.1.

Results. We select DAM (So et al., 2026) and d1 (Zhao et al., 2025c) as the baselines. The experimental results are presented in Figure 1. PPO aligns most closely with p^*, both visually and numerically. In this $L = 2$ tabular setting, PPO uses an exact critic, so it converges faster and tighter than DAM's sample-based adjoint. d1 was designed for large-vocabulary reasoning with informative prompts, so this synthetic distribution-matching task isolates the objective rather than d1's intended use case.

5.2 Mathematical Reasoning and Coding Tasks

Setup. We evaluate our GRPO algorithm on four standard mathematical reasoning tasks: GSM8K (Cobbe et al., 2021), which consists of multi-step grade-school math word problems; MATH500 (Lightman et al., 2024), a collection of high-school competition mathematics problems; 3-number Countdown, a combinatorial arithmetic game in which models must reach a target number using basic arithmetic operations on three given numbers; and 4×4 Sudoku, which requires constraint

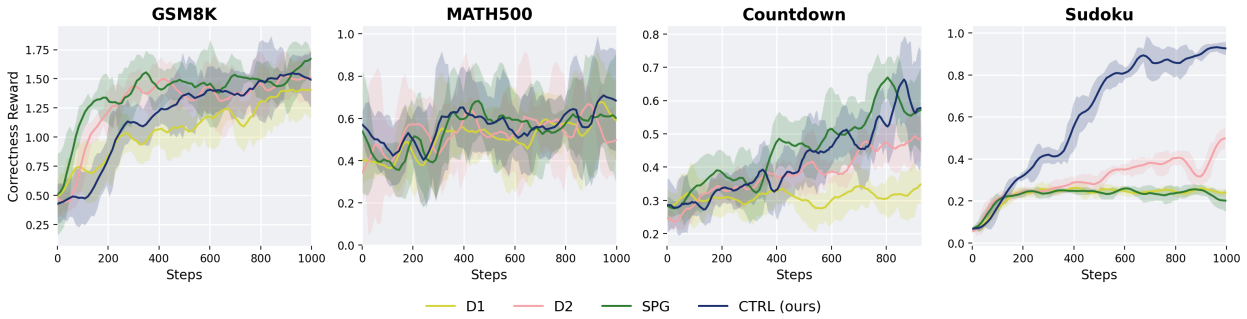

Figure 2: Correctness reward dynamics of CTRL during RL training with a completion length of 128, compared with d1, d2, and SPG.

	GSM8K		MATH500		Countdown		Sudoku	
Model / Seq Len	**128**	**256**	**128**	**256**	**128**	**256**	**128**	**256**
LLaDA-8B-Instruct	67.9	76.1	26.8	33.4	21.2	17.2	11.5	7.1
D1	70.9	76.6	28.2	36.6	28.4	29.5	22.9	15.7
D2	74.8	78.9	32.2	37.8	47.8	49.2	29.9	21.4
SPG	75.7	78.1	32.0	37.8	64.9	57.0	26.1	26.0
CTRL	74.3	80.3	32.6	38.2	58.7	52.8	88.2	51.7

Table 1: Evaluation results in 1000 training steps for each method on reasoning tasks. All evaluations are conducted using the evaluation code from d1 with 0-shot prompting. CTRL significantly outperforms all baselines on Sudoku and achieves the best performance on GSM8K and MATH500.

satisfaction and systematic reasoning to complete the grid. We employ LLaDA-8B-Instruct (Nie et al., 2026b) without supervised fine-tuning as the base model, and adopt the experiment setting in d1 (Zhao et al., 2025c). We choose d1 (Zhao et al., 2025c), SPG with mixture (Wang et al., 2026b), and d2 (Wang et al., 2026c) as the baselines: d1 is the first GRPO-based fine-tuning algorithm for dLLMs, and SPG and d2 are included because they are representative methods that outperform d1, while requiring multiple forward passes during training. For each method, we fine-tune a separate model on each task with completion lengths of 128 and 256 in 1000 steps (due to the time cost), and evaluate each resulting model at completion lengths of 128 and 256, accordingly.

Results. The correctness reward curves during the RL training with a completion length of 128 are depicted in Figure 2. For the Sudoku experiments, we note that SPG uses 3-shot prompting during training, whereas all four methods in our experiments are trained with 0-shot prompting. Despite this more challenging setting, CTRL consistently converges to a correctness reward of 0.92 on Sudoku in 1000 training steps, whereas d2 converges much more slowly, and both d1 and SPG plateau at around 0.25. We note that our success on Sudoku can be attributed to using the intermediate reward as process supervision, which guides the model to generate valid answers (see Figure 5 in Appendix). Further experimental details including the specification of intermediate reward functions are provided in Appendix D.2. Table 1 reports the best evaluation results achieved by each method over all checkpoints. For fair comparison, all evaluations are conducted with 0-shot prompting, using the evaluation code from d1 with a batch size of 8. Across nearly all experimental settings except Countdown, CTRL achieves comparable or superior accuracy to all baselines, demonstrating its

	HumanEval		MBPP	
Model / Seq Len	**128**	**256**	**128**	**256**
LLaDA-8B-Instruct	24.8	34.8	39.7	40.5
D1	29.3	39.0	42.0	45.5
D2	39.6	48.7	45.6	46.8
SPG	29.3	40.2	44.4	44.8
CTRL	62.8	66.2	52.9	57.7

Table 2: Evaluation results in 1000 training steps for each method on coding tasks. CTRL consistently outperforms all baselines on HumanEval and MBPP benchmarks.

effectiveness for fine-tuning moderate-sized dLLMs with moderate sequence lengths. In particular, CTRL substantially outperforms the competing methods on Sudoku with a completion length of 128; all methods collapse when the completion length is increased to 256, suggesting that shorter sequence lengths are better suited to this task. By contrast, the performance gains on GSM8K and MATH500 are more modest across different baseline methods, which likely reflects the limited pretraining capability of the base model.

Extension to Coding. We further extend CTRL to coding tasks, fine-tuning the LLaDA-8B-Instruct base model on the KodCodeLight-RL-10K dataset (Xu et al., 2025) and evaluating on HumanEval and MBPP benchmarks. As shown in Table 2, CTRL consistently improves the accuracy on all benchmarks over the baselines across different generation lengths, demonstrating its strong ability in the coding domain.

6 Conclusion and Future Work

We introduce a theoretical framework of reinforcement learning in continuous time and discrete spaces, with applications to fine-tuning score-based discrete diffusion models, especially masked diffusion models by setting the score/denoiser as the action. Our framework naturally accommodates intermediate rewards into the RL objective, and leads to algorithms that do not require differentiable rewards. We demonstrate the effectiveness of our methods by both solving low-dimensional entropy-regularized optimization problems and fine-tuning diffusion large language models on reasoning and coding tasks. Future work will focus on developing refined techniques to reduce the number of function evaluations associated with the computation of importance sampling probability ratios under the continuous-time RL framework, and improve the computational efficiency of PPO-based RL post-training for large-scale problems.

Acknowledgement

Tang is supported by NSF CAREER Award DMS-2538791 and the Tang Family Assistant Professorship. Sheng and Zhang are supported by NSF Grant DMS-2206038. This research is part of a Columbia-CityU/HK collaborative project that is supported by InnoHK Initiative, The Government of the HKSAR and the AIFT Lab.

References

Shun-ichi Amari. *Differential-geometrical methods in statistics*, volume 28. Springer Science & Business Media, 2012.

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

Joe Benton, Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. From denoising diffusions to denoising markov models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(2):286–301, 2024.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, and Biqing Qi. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

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

Stewart N Ethier and Thomas G Kurtz. *Markov processes: characterization and convergence*. John Wiley & Sons, 2009.

Griffin Floto, Thorsteinn Jonsson, Mihai Nica, Scott Sanner, and Eric Zhengyu Zhu. Diffusion on the probability simplex. *arXiv preprint arXiv:2309.02530*, 2023.

Xuefeng Gao, Jiale Zha, and Xun Yu Zhou. Reward-directed score-based diffusion models via q-learning. *Journal of Machine Learning Research*, 26(302):1–46, 2025.

Mohsin Hasan, Viktor Ohanesian, Artem Gazizov, Yoshua Bengio, Alán Aspuru-Guzik, Roberto Bondesan, Marta Skreta, and Kirill Neklyudov. Discrete feynman-kac correctors. *arXiv preprint arXiv:2601.10403*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models. *Advances in Neural Information Processing Systems*, 38:152677–152710, 2026.

Yanwei Jia and Xun Yu Zhou. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *Journal of Machine Learning Research*, 23(275):1–50, 2022a.

Yanwei Jia and Xun Yu Zhou. Policy evaluation and temporal-difference learning in continuous time and space: A martingale approach. *Journal of Machine Learning Research*, 23(154):1–55, 2022b.

Yanwei Jia and Xun Yu Zhou. q-learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61, 2023.

Frank P Kelly. *Reversibility and Stochastic Networks*. Cambridge University Press, 2011.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, and Stefano Ermon. Mercury: Ultra-fast language models based on diffusion. *arXiv e-prints*, pages arXiv–2506, 2025.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DjJmre5IkP.

Thomas M. Liggett. *Continuous time Markov processes*, volume 113 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2010. An introduction.

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 *International Conference on Learning Representations*, volume 2024, pages 39578–39601, 2024.

Aiwei Liu, Minghua He, Shaoxun Zeng, Sijun Zhang, Linhao Zhang, Chuhan Wu, Wei Jia, Yuan Liu, Xiao Zhou, and Jie Zhou. Wedlm: Reconciling diffusion language models with standard causal attention for fast inference. *arXiv preprint arXiv:2512.22737*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2024.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete score matching: Generalized score matching for discrete data. *Advances in Neural Information Processing Systems*, 35:34532–34545, 2022.

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 *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kpgURPRMGf.

Shen Nie, Qiyang Min, Shaoxuan Xu, Zihao Huang, Yuxuan Song, Yong Shan, Yankai Lin, Wayne Xin Zhao, Chongxuan Li, and Ji-Rong Wen. Improved large language diffusion models. *arXiv preprint arXiv:2606.25331*, 2026a.

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. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026b.

Daisuke Oba, Hiroki Furuta, and Naoaki Okazaki. Diffusion-state policy optimization for masked diffusion language models. *arXiv preprint arXiv:2602.06462*, 2026.

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

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled RL for diffusion LLMs emerges from a sequence-level perspective. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=S5YeC9llIL.

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. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Pierre H Richemond, Sander Dieleman, and Arnaud Doucet. Categorical sdes with simplex diffusion. *arXiv preprint arXiv:2210.14784*, 2022.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and Yang Wu. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

Oswin So, Brian Karrer, Chuchu Fan, Ricky T. Q. Chen, and Guan-Horng Liu. Discrete adjoint matching. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VXB4xxAgOf.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=BYWWwSY2G5s.

Wenpin Tang and Xun Yu Zhou. Regret of exploratory policy improvement and q-learning. *arXiv preprint arXiv:2411.01302*, 2024.

Wenpin Tang, Yuming Paul Zhang, and Xun Yu Zhou. Exploratory HJB equations and their convergence. *SIAM Journal on Control and Optimization*, 60(6):3191–3216, 2022.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=L2rfd2Czbj.

Chao Wang, Hongtao Tian, Tao Yang, Yunsheng Shi, Ting Yao, and Wenbo Ding. Process advantage signal shaping: A paradigm-agnostic middleware for process-supervised rl in llm reasoners. *arXiv preprint arXiv:2606.29296*, 2026a.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Hanchen Wang, and Tommaso Biancalani. Fine-tuning discrete diffusion

models via reward optimization with applications to dna and protein design. In *International Conference on Learning Representations*, volume 2025, pages 47871–47899, 2025.

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 *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=18j5Q49GwN.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improved techniques for training reasoning diffusion language models. In *The Forty-third International Conference on Machine Learning*, 2026c.

Haoran Wang and Xun Yu Zhou. Continuous-time mean–variance portfolio selection: A reinforcement learning framework. *Mathematical Finance*, 30(4):1273–1308, 2020.

Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(198): 1–34, 2020.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://openreview.net/forum?id=KNAyc9DMe3.

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

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6980–7008, 2025.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. MMaDA: Multimodal large diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=wczmXLuLGd.

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.

Oussama Zekri and Nicolas Boullé. Fine-tuning discrete diffusion models with policy gradient methods. *Advances in Neural Information Processing Systems*, 38:152868–152906, 2026.

Wenxuan Zhang, Lemeng Wu, Changsheng Zhao, Ernie Chang, Mingchen Zhuge, Zechun Liu, Andy Su, Hanxian Huang, Jun Chen, and Chong Zhou. dtrpo: Trajectory reduction in policy optimization of diffusion large language models. *arXiv preprint arXiv:2603.18806*, 2026.

Zikun Zhang, Zixiang Chen, and Quanquan Gu. Convergence of score-based discrete diffusion models: A discrete-time analysis. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pq1WUegkza.

Hanyang Zhao, Wenpin Tang, and David Yao. Policy optimization for continuous reinforcement learning. *Advances in Neural Information Processing Systems*, 36:13637–13663, 2023.

Hanyang Zhao, Haoxian Chen, Ji Zhang, David D Yao, and Wenpin Tang. Scores as actions: a framework of fine-tuning diffusion models by continuous-time reinforcement learning. *arXiv preprint arXiv:2409.08400*, 2024.

Hanyang Zhao, Haoxian Chen, Ji Zhang, David Yao, and Wenpin Tang. Score as action: Fine tuning diffusion generative models by continuous-time reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=V5HEX6stS2.

Hanyang Zhao, Dawen Liang, Wenpin Tang, David Yao, and Nathan Kallus. Diffpo: Training diffusion llms to reason fast and furious via reinforcement learning. *arXiv preprint arXiv:2510.02212*, 2025b.

Hanyang Zhao, Wenpin Tang, and David D Yao. Policy optimization for reinforcement learning in continuous time and space. *Applied Mathematics & Optimization*, 94(1):16, 2026.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=7ZVRlBFuEv.

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

A Proofs

A.1 Proof of Proposition 1

A.1.1 Heuristic Law-of-Large-Numbers Argument

Analogous to the law-of-large-numbers argument in Wang and Zhou (2020), we first give a heuristic proof. Fix $(t,x) \in [0,T] \times \mathcal{S}$. Consider N independent copies of the controlled problem run under the *same* feedback policy $\pi(\cdot \mid t,x)$. In copy i, an action $a_t^i \sim \pi(\cdot \mid t,x)$ is drawn independently, and over $[t, t+\Delta t)$ the copy attempts a jump to $y \neq x$ with probability $R(t,x,a_t^i)_y\,\Delta t + o(\Delta t)$. Let

$$J_t^i(y) := \mathbf{1}\{\text{copy } i \text{ jumps to } y \text{ over } [t, t+\Delta t)\}.$$

Then $\mathbb{E}[J_t^i(y) \mid a_t^i] = R(t,x,a_t^i)_y\,\Delta t + o(\Delta t)$, and taking expectation over $a_t^i \sim \pi$,

$$\mathbb{E}[J_t^i(y)] = \Big(\int_{\mathcal{A}} R(t,x,a)_y\,\pi(a \mid t,x)\,\mathrm{d}a\Big)\Delta t + o(\Delta t) = \tilde{R}(t,x;\pi(\cdot \mid t,x))_y\,\Delta t + o(\Delta t).$$

Since the copies are i.i.d., the law of large numbers gives, as $N \to \infty$,

$$\frac{1}{N}\sum_{i=1}^{N} J_t^i(y) \xrightarrow{\text{a.s.}} \mathbb{E}[J_t^i(y)] = \tilde{R}(t,x;\pi(\cdot \mid t,x))_y\,\Delta t + o(\Delta t).$$

Thus the empirical infinitesimal jump frequency from x to y, averaged over the externally randomized copies, equals the exploratory rate $\tilde{R}(t,x;\pi(\cdot \mid t,x))_y$.

A.2 Rigorous Proof

Now we give a rigorous proof by showing that both processes solve the same martingale problem, which for time-inhomogeneous CTMCs on a discrete space has a unique solution.

The controlled process X^π is built by: in state x at time t, first sample $a \sim \pi(\cdot \mid t, x)$, then evolve infinitesimally under generator $\mathcal{L}^a$. To identify the law of X^π we compute the infinitesimal evolution of $\mathbb{E}[f(t, X_t^\pi)]$, i.e. the generator obtained after averaging over the action.

Let $f : [0, T] \times \mathcal{S} \to \mathbb{R}$ be bounded. By (3), conditioning on the state $x_t^\pi = x$ and the sampled action $a_t^\pi = a$,

$$\mathbb{E}\left[f(t + \Delta t, X_{t+\Delta t}^\pi) \mid x_t^\pi = x,\, a_t^\pi = a\right] = \sum_{y \in \mathcal{S}} \Big(\delta\{x, y\} + R(t, x, a)_y\, \Delta t + o(\Delta t)\Big) f(t + \Delta t, y).$$

Using $\sum_y \delta\{x, y\} f(t + \Delta t, y) = f(t + \Delta t, x) = f(t, x) + \frac{\partial f}{\partial t}(t, x)\, \Delta t + o(\Delta t)$ and $\langle R(t, x, a),\, f(t + \Delta t, \cdot)\rangle = \langle R(t, x, a),\, f(t, \cdot)\rangle + o(1)$, this becomes

$$f(t, x) + \Big(\tfrac{\partial f}{\partial t}(t, x) + \langle R(t, x, a),\, f(t, \cdot)\rangle\Big)\Delta t + o(\Delta t) = f(t, x) + \mathcal{L}^a f(t, x)\, \Delta t + o(\Delta t).$$

Now average over $a \sim \pi(\cdot \mid t, x)$. By linearity of $\mathcal{L}^a$ in $R(t, x, a)$, and finiteness of $\mathcal{S}$ (so the sum over y and the expectation over a may be interchanged),

$$\mathop{\mathbb{E}}_{a \sim \pi(\cdot \mid t, x)}\left[\mathcal{L}^a f(t, x)\right] = \frac{\partial f}{\partial t}(t, x) + \Big\langle \underbrace{\mathop{\mathbb{E}}_{a \sim \pi(\cdot \mid t, x)} R(t, x, a)}_{= \tilde{R}(t, x; \pi(\cdot \mid t, x))},\; f(t, \cdot)\Big\rangle.$$

Define the averaged generator

$$\tilde{\mathcal{L}} f(t, x) := \mathop{\mathbb{E}}_{a \sim \pi(\cdot \mid t, x)}\left[\mathcal{L}^a f(t, x)\right] = \frac{\partial f}{\partial t}(t, x) + \langle \tilde{R}(t, x; \pi(\cdot \mid t, x)),\, f(t, \cdot)\rangle.$$

Therefore, we have

$$\mathbb{E}\left[f(t + \Delta t, X_{t+\Delta t}^\pi) \mid x_t^\pi = x\right] = f(t, x) + \tilde{\mathcal{L}} f(t, x)\, \Delta t + o(\Delta t). \tag{16}$$

On the other hand, the exploratory process $\tilde{X}^\pi$ jumps directly with rate $\tilde{R}(t, x; \pi(\cdot \mid t, x))$, with no external action sampling. Repeating the identical one-step computation using (4),

$$\mathbb{E}\left[f(t + \Delta t, \tilde{X}_{t+\Delta t}^\pi) \mid \tilde{X}_t^\pi = x\right] = f(t, x) + \tilde{\mathcal{L}} f(t, x)\, \Delta t + o(\Delta t). \tag{17}$$

Comparing (16) and (17), both processes have the same infinitesimal generator $\tilde{\mathcal{L}}$, and thus have equal one-step transition rates.

To conclude equality of the full path laws, we invoke uniqueness of the martingale problem. Define

$$M_t^f := f(t, X_t) - f(0, X_0) - \int_0^t \tilde{\mathcal{L}} f(s, X_s)\, \mathrm{d}s$$

for bounded $f : [0, T] \times \mathcal{S} \to \mathbb{R}$. A process X with $X_0 \sim \rho$ *solves the martingale problem for* $(\tilde{\mathcal{L}}, \rho)$ if M_t^f is a martingale for every bounded f. Hence, it suffices to prove that both X^π and $\tilde{X}^\pi$ solve this martingale problem. For $\tilde{X}^\pi$, this is the standard Dynkin's formula (see, e.g., Benton et al. (2024,

Lemma 3)) applied to the CTMC with rate $\tilde{R}$, so $M_t^f(\tilde{X}^\pi)$ is a martingale by construction. For X^π, we must check the controlled process is also a martingale solution for the averaged generator $\tilde{\mathcal{L}}$. Let $\mathcal{F}_t = \sigma(X_s^\pi : s \le t)$. For $u < t$, we have

$$\mathbb{E}\left[M_t^f(X^\pi) - M_u^f(X^\pi) \mid \mathcal{F}_u\right] = \mathbb{E}\left[f(t, X_t^\pi) - f(u, X_u^\pi) - \int_u^t \tilde{\mathcal{L}} f(s, X_s^\pi)\,\mathrm{d}s \,\middle|\, \mathcal{F}_u\right].$$

Partition $[u, t]$ into mesh Δs. By (16), the conditional increment of f over each subinterval, after averaging the external action a_s^π (sampled independently of the past given the current state), is

$$\mathbb{E}\left[f(s + \Delta s, X_{s+\Delta s}^\pi) - f(s, X_s^\pi) \mid \mathcal{F}_s\right] = \tilde{\mathcal{L}} f(s, X_s^\pi)\,\Delta s + o(\Delta s).$$

Crucially, the averaging in (16) uses the Markov property: the action a_s^π at time s depends on the past only through X_s^π, so conditioning on $\mathcal{F}_s$ and then on a_s^π collapses to the state-conditional average $\mathbb{E}_{a\sim\pi(\cdot|s,X_s^\pi)}$, yielding $\tilde{\mathcal{L}}$. Summing over the partition and letting $\Delta s \to 0$ (dominated convergence is valid since $\mathcal{S}$ is finite and the rates are bounded), the telescoping sum gives

$$\mathbb{E}\left[f(t, X_t^\pi) - f(u, X_u^\pi) \mid \mathcal{F}_u\right] = \mathbb{E}\Big[\int_u^t \tilde{\mathcal{L}} f(s, X_s^\pi)\,ds \,\Big|\, \mathcal{F}_u\Big],$$

and thus $\mathbb{E}[M_t^f - M_u^f \mid \mathcal{F}_u] = 0$. Hence, X^π also solves the martingale problem for $(\tilde{\mathcal{L}}, \rho)$.

Finally, for a CTMC on a finite state space with measurable, bounded time-dependent rate matrix $\tilde{R}(t, \cdot; \pi)$, the martingale problem for $(\tilde{\mathcal{L}}, \rho)$ is well posed: there exists a unique law on $D([0, T], \mathcal{S})$ solving it (Ethier and Kurtz, 2009, Theorem 4.1). Since both X^π and $\tilde{X}^\pi$ solve the same well-posed martingale problem $(\tilde{\mathcal{L}}, \rho)$, by uniqueness, they induce the same law on path space:

$$\mathrm{Law}\big((X_t^\pi)_{t\in[0,T]}\big) = \mathrm{Law}\big((\tilde{X}_t^\pi)_{t\in[0,T]}\big).$$

□

A.3 Proof of Lemma 1

Proof of Lemma 1. Note that

$$\begin{aligned}
\mathop{\mathbb{E}}_{a\sim\pi(\cdot|t,x)}(\mathcal{L}^a v(t,x) + r(t,x,a)) &= \int_{\mathcal{A}} (\mathcal{L}^a v(t,x) + r(t,x,a))\pi(a \mid t,x)\,\mathrm{d}a \\
&= \frac{\partial v}{\partial t}(t,x) + \langle \tilde{R}(t,x;\pi(\cdot \mid t,x)), v(t,\cdot)\rangle + \tilde{r}(t,x;\pi(\cdot \mid t,x)) \\
&= \tilde{\mathcal{L}} v(t,x) + \tilde{r}(t,x;\pi(\cdot \mid t,x)),
\end{aligned}$$

where we denote $\tilde{\mathcal{L}} v(t,x) = \mathbb{E}_{a\sim\pi(\cdot|t,x)}\,\mathcal{L}^a v(t,x)$. Then Lemma 1 follows from Benton et al. (2024, Theorem 5). □

A.4 Proof of Theorem 1

Proof of Theorem 1. The proof is similar to that of Theorem 5 in Jia and Zhou (2022a). By Lemma 1, we replace $v(t,x)$ by $V^\theta(t,x)$ and obtain that for any $(t,x) \in [0,T] \times \mathcal{S}$,

$$\int_{\mathcal{A}} (\mathcal{L}^a V^\theta(t,x) + r(t,x,a))\pi^\theta(a \mid t,x)\,\mathrm{d}a = 0.$$

Taking derivative w.r.t. θ on both sides, we have

$$\begin{aligned}
&\int_{\mathcal{A}} \left[(\mathcal{L}^a \nabla_\theta V^\theta(t,x)) \pi^\theta(a \mid t,x) + (\mathcal{L}^a V^\theta(t,x) + r(t,x,a)) \nabla_\theta \pi^\theta(a \mid t,x) \right] \mathrm{d}a \\
= &\int_{\mathcal{A}} \left[\left(\mathcal{L}^a \nabla_\theta V^\theta(t,x) + (\mathcal{L}^a V^\theta(t,x) + r(t,x,a)) \nabla_\theta \log \pi^\theta(a \mid t,x) \right) \pi^\theta(a \mid t,x) \right] \mathrm{d}a = 0.
\end{aligned}$$

By again Lemma 1, we have

$$\nabla_\theta V^\theta(t,x) = \mathbb{E}\left[\int_t^T (\mathcal{L}^a V^\theta(s,x) + r(s,x,a)) \nabla_\theta \log \pi^\theta(a \mid s,x) \, \mathrm{d}s \mid X_t^\theta = x \right].$$

By letting $t = 0$ and integrating x, we conclude the proof. □

A.5 Proof of Proposition 2

Proof of Proposition 2. The proof is similar to that of Theorem 3 in Jia and Zhou (2023). We have

$$\begin{aligned}
&Q_{\Delta t}(t,x,a;\pi) \\
= &\mathbb{E}\left[\int_t^{t+\Delta t} r(s, X_s^a, a) \, \mathrm{d}s + \mathbb{E}\left[\int_{t+\Delta t}^T r(s, X_s^\pi, a_s^\pi) \, \mathrm{d}s + h(X_T^\pi) \mid X_{t+\Delta t}^a \right] \mid X_t^{\hat{\pi}} = x \right] \\
= &\mathbb{E}\left[\int_t^{t+\Delta t} r(s, X_s^a, a) \, \mathrm{d}s + V(t+\Delta t, X_{t+\Delta t}^a; \pi) \mid X_t^{\hat{\pi}} = x \right] \\
= &\mathbb{E}\left[\int_t^{t+\Delta t} r(s, X_s^a, a) \, \mathrm{d}s + V(t+\Delta t, X_{t+\Delta t}^a; \pi) - V(t, X_t^a; \pi) \mid X_t^{\hat{\pi}} = x \right] + V(t,x;\pi) \\
= &\mathbb{E}\left[\int_t^{t+\Delta t} r(s, X_s^a, a) \, \mathrm{d}s + \int_t^{t+\Delta t} \mathcal{L}^a V(s, X_s^a; \pi) \, \mathrm{d}s \mid X_t^{\hat{\pi}} = x \right] + V(t,x;\pi) \\
= &V(t,x;\pi) + q(t,x,a;\pi) \cdot \Delta t + o(\Delta t),
\end{aligned}$$

where the second to the last equality follows from Dynkin's formula (Benton et al., 2024, Lemma 3) and the last equality is because of the definition of q and the approximation of the integral. □

A.6 Proof of Proposition 3

Proof of Proposition 3. For the off-diagonal target $y = i \neq x$, the only column contributing to row y is column i, giving $(\Phi a)_y = \Phi_{y,y}\, a_y = Q_{T-t}(y,x)\, a_y$, matching the embedding. For the diagonal row $y = x$, $(\Phi a)_x = \sum_{i \neq x} \Phi_{x,i}\, a_i = -\sum_{i \neq x} Q_{T-t}(i,x)\, a_i = -\sum_{y \neq x} R(t,x,a)_y$, matching the conservative diagonal. Each column i has exactly two nonzero entries, $+Q_{T-t}(i,x)$ in row i and $-Q_{T-t}(i,x)$ in row x, so its column sum is $Q_{T-t}(i,x) - Q_{T-t}(i,x) = 0$. □

A.7 Proof of Theorem 2

Proof of Theorem 2. The proof follows by applying Lemma 1 in Zhang et al. (2025), where two CTMCs with rate matrices $\tilde{R}(t,x;\pi^\theta(\cdot \mid t,x))_y = Q_{T-t}(y,x)\, \mathbb{E}_{a \sim \pi^\theta(\cdot \mid t,x)}\, a_y$ and $\tilde{R}(t,x;\pi^{\theta_{\mathrm{pre}}}(\cdot \mid t,x)) = Q_{T-t}(y,x) s_{\theta_{\mathrm{pre}}}(T-t,x)_y$ are applied to Girsanov's theorem. □

B Policy Parameterization Choices

B.1 Policy over the Full Simplex

In this section, we present several policy parameterizations that explore the full simplex (and therefore full-vocabulary exploration) and are applicable to fine-tuning MDMs when the vocabulary size V is reasonably small.

B.1.1 Dirichlet Policy

We first introduce an unbiased policy by leveraging Dirichlet distribution. Consider

$$\mathbf{a}_t^{(i)} \sim \pi^{\theta,(i)}(\cdot \mid t, \mathbf{x}_t) := \begin{cases} \mathrm{Dir}(\cdot; k_t \cdot \mathbf{p}_\theta^{(i)}(\mathbf{x}_t)), & i \in [L] : x_t^i = \mathtt{m}, \\ \delta_{\mathrm{Cat}(\cdot;\mathbf{e}_{x_t^i})}, & i \in [L] : x_t^i \neq \mathtt{m} \end{cases} \in \Delta(\Delta^{V-1}).$$

Here, $\mathbf{p}_\theta^{(i)}(\mathbf{x}_t)$ determines the mean of the Dirichlet distribution and the time-dependent positive scalar $k_t \in \mathbb{R}_+$ controls its variance. Then we have $\mathbb{E}_{\mathbf{a}_t^{(i)} \sim \pi^{\theta,(i)}(\cdot|t,\mathbf{x}_t)} \mathbf{a}_t^{(i)} = \mathbf{p}_\theta^{(i)}(\mathbf{x}_t) \in \Delta^{V-1}$.

Remark 3 (Interpretation and Intuition of Using Dirichlet Distribution)**.** The Dirichlet distribution is defined over the simplex as the conjugate prior of the categorical distribution, playing a role somewhat analogous to that of the Gaussian distribution in continuous diffusion models (Richemond et al., 2022). Note that given $\alpha \in \mathbb{R}_+^V$, for any $k \in \mathbb{R}_+$, it holds that $\mathbb{E}(\mathrm{Dir}(\cdot;\alpha)) = \mathbb{E}(\mathrm{Dir}(\cdot; k\cdot\alpha)) = (\alpha_1/\alpha_0, \cdots, \alpha_V/\alpha_0)^\top$ with $\alpha_0 := \sum_{j=1}^V \alpha_j$. Thus, we can tune k to control the variance of π^θ while preserving the mean. Further let $\tilde{\alpha}_j := \alpha_j/\alpha_0$, then $\mathrm{Var}(\mathrm{Dir}(\cdot;\alpha)) = (\tilde{\alpha}_j(1-\tilde{\alpha}_j)/(1+\alpha_0))_{j\in[V]}$, from which we can see that if fixing the mean $(\mathbf{p}_\theta(\mathbf{x}_t))$, tuning k (k_t) small leads to a large variance and thus adds more exploration; in contrast, tuning k large reduces stochasticity. On the other hand, given a fixed α_0 (k_t), if $\tilde{\alpha}_j$ ($p_\theta(\mathbf{x}_t)_j$) is too large or small (close to 1 or 0), meaning that the model gives a very high or low confidence for the token j, then the variance of a_j is small and thus the predicted probability value of the token j is more deterministic; else if $\tilde{\alpha}_j$ lies in the middle of $[0,1]$, meaning that the model is not confident enough on the outcome of token j, then the variance of a_j is large and more exploration and stochasticity are injected in the dynamics by the algorithm. □

Value and Reward Functions. For Dirichlet policy, the value function in (10) becomes

$$\begin{aligned} & V^\theta(t, \mathbf{x}) \\ &= \mathop{\mathbb{E}}_{\mathbf{X}_s \sim p_s^\theta} \left[\mathrm{TRF}(\mathbf{X}_1) + \int_t^1 \left(\mathrm{IRF}(\mathbf{X}_s) - \frac{\beta}{1-s} \sum_{i:X_s^i=\mathtt{m}} \sum_{j \in \mathcal{V}} D_I(p_\theta^{(i)}(\mathbf{X}_s)_j \,\|\, p_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{X}_s)_j) \right) \mathrm{d}s \,\middle|\, \mathbf{X}_t = \mathbf{x} \right] \\ &= \mathop{\mathbb{E}}_{\mathbf{X}_s \sim p_s^\theta} \left[\mathrm{TRF}(\mathbf{X}_1) + \int_t^1 \left(\mathrm{IRF}(\mathbf{X}_s) - \frac{\beta}{1-s} \sum_{i:X_s^i=\mathtt{m}} D_{\mathrm{KL}}(\mathbf{p}_\theta^{(i)}(\mathbf{X}_s) \,\|\, \mathbf{p}_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{X}_s)) \right) \mathrm{d}s \,\middle|\, \mathbf{X}_t = \mathbf{x} \right]. \end{aligned}$$

Consequently, for $t \in [0,1)$, the intermediate and terminal reward functions are specified as

$$\tilde{r}(t, \mathbf{x}_t; \pi^\theta(\cdot \mid t, \mathbf{x}_t)) = \mathrm{IRF}(\mathbf{x}_t) - \frac{\beta}{1-t} \sum_{i:x_t^i=\mathtt{m}} D_{\mathrm{KL}}(\mathbf{p}_\theta^{(i)}(\mathbf{x}_t) \,\|\, \mathbf{p}_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{x}_t)) \quad \text{and} \quad h(\mathbf{x}_1) = \mathrm{TRF}(\mathbf{x}_1).$$

We can therefore define

$$r(t, \mathbf{x}_t, \mathbf{a}_t) := \mathrm{IRF}(\mathbf{x}_t) + \frac{\beta}{1-t} \sum_{i: x_t^i = \mathtt{m}} \left(D_{\mathrm{KL}}(\mathbf{a}_t^{(i)} \,\|\, \mathbf{p}_\theta^{(i)}(\mathbf{x}_t)) - D_{\mathrm{KL}}(\mathbf{a}_t^{(i)} \,\|\, \mathbf{p}_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{x}_t)) \right)$$

$$= \mathrm{IRF}(\mathbf{x}_t) - \frac{\beta}{1-t} \sum_{i: x_t^i = \mathtt{m}} \sum_{j \in \mathcal{V}} (\mathbf{a}_t^{(i)})_j \log \frac{p_\theta^{(i)}(\mathbf{x}_t)_j}{p_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{x}_t)_j}, \quad \mathbf{a}_t \sim \pi^\theta(\cdot \mid t, \mathbf{x}_t).$$

We can see that a large r corresponds to a large $D_{\mathrm{KL}}(\mathbf{a}_t^{(i)} \,\|\, \mathbf{p}_\theta^{(i)}(\mathbf{x}_t))$, encouraging exploration beyond the current mean $\mathbf{p}_\theta$, while maintaining a small $D_{\mathrm{KL}}(\mathbf{a}_t^{(i)} \,\|\, \mathbf{p}_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{x}_t))$, thereby keeping the exploration close to the pretrained model $\mathbf{p}_{\theta_{\mathrm{pre}}}$.

q-Function. For $\mathbf{a}_t \sim \pi^\theta(\cdot \mid t, \mathbf{x}_t)$, we can write

$$q(t, \mathbf{x}_t, \mathbf{a}_t; \pi^\theta(\cdot \mid t, \mathbf{x}_t)) = \mathrm{IRF}(\mathbf{x}_t) + \frac{1}{1-t} \sum_{i: x_t^i = \mathtt{m}} \sum_{j \in \mathcal{V}} (\mathbf{a}_t^{(i)})_j \left(V^\theta(t, \mathbf{x}_t^{\setminus i} \odot j) - V^\theta(t, \mathbf{x}_t) - \beta \log \frac{p_\theta^{(i)}(\mathbf{x}_t)_j}{p_{\theta_{\mathrm{pre}}}^{(i)}(\mathbf{x}_t)_j} \right).$$

Probability Ratio. To sample $\mathbf{a} \sim \pi^\theta(\cdot \mid t, \mathbf{x}) = \mathrm{Dir}(\cdot; \alpha_\theta(t, \mathbf{x}))$ where $\alpha_\theta(t, \mathbf{x}) \in \mathbb{R}_+^V$, one can first sample V independent Gamma random variables $g_j \sim \Gamma(\alpha_\theta(t, x)_j, 1)$, and then set $a_j = g_j / \sum_{k=1}^V g_k$ for all $j \in [V]$. Since the density of $\mathrm{Dir}(\cdot; \alpha)$ is

$$\mathrm{Dir}(a; \alpha) = \frac{\Gamma(\sum_{j=1}^V \alpha_j)}{\prod_{j=1}^V \Gamma(\alpha_j)} \prod_{j=1}^V a_j^{\alpha_j - 1},$$

we have

$$\log \mathrm{Dir}(a; \alpha) = \sum_{j=1}^V (\alpha_j - 1) \log a_j - \sum_{j=1}^V \log \Gamma(\alpha_j) + \log \Gamma(\sum_{j=1}^V \alpha_j).$$

Then we obtain the probability ratio

$$\begin{aligned} \frac{\pi^\theta(a)}{\pi^{\theta_{\mathrm{old}}}(a)} &= \exp(\log \pi^\theta(a) - \log \pi^{\theta_{\mathrm{old}}}(a)) \\ &= \exp\left(\sum_{j=1}^V (\alpha_j^\theta - \alpha_j^{\theta_{\mathrm{old}}}) \log a_j - \sum_{j=1}^V \log \frac{\Gamma(\alpha_j^\theta)}{\Gamma(\alpha_j^{\theta_{\mathrm{old}}})} + \log \frac{\Gamma(\sum_{j=1}^V \alpha_j^\theta)}{\Gamma(\sum_{j=1}^V \alpha_j^{\theta_{\mathrm{old}}})} \right). \end{aligned}$$

Specific to MDMs, we have

$$\begin{aligned} \rho_t^\theta &= \frac{\pi^\theta(\mathbf{a}_t \mid t, \mathbf{x}_t)}{\pi^{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid t, \mathbf{x}_t)} \\ &= \prod_{i \in B_t : x_t^i = \mathtt{m}} \frac{\pi^{\theta,(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t)}{\pi^{\theta_{\mathrm{old}},(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t)} \\ &= \prod_{i \in B_t : x_t^i = \mathtt{m}} \exp\left(\sum_{j \in \mathcal{V}} k_t (p_\theta^{(i)}(\mathbf{x}_t)_j - p_{\theta_{\mathrm{old}}}^{(i)}(\mathbf{x}_t)_j) \log(\mathbf{a}_t^{(i)})_j - \sum_{j \in \mathcal{V}} \log \frac{\Gamma(k_t \cdot p_\theta^{(i)}(\mathbf{x}_t)_j)}{\Gamma(k_t \cdot p_{\theta_{\mathrm{old}}}^{(i)}(\mathbf{x}_t)_j)} \right) \\ &= \exp\left(\sum_{i \in B_t : x_t^i = \mathtt{m}} \sum_{j \in \mathcal{V}} \left(k_t (p_\theta^{(i)}(\mathbf{x}_t)_j - p_{\theta_{\mathrm{old}}}^{(i)}(\mathbf{x}_t)_j) \log(\mathbf{a}_t^{(i)})_j - \log \frac{\Gamma(k_t \cdot p_\theta^{(i)}(\mathbf{x}_t)_j)}{\Gamma(k_t \cdot p_{\theta_{\mathrm{old}}}^{(i)}(\mathbf{x}_t)_j)} \right) \right), \end{aligned} \tag{18}$$

where $B_t \subset [L]$ is the sequence indices of the current block corresponding to the current time step t.

B.1.2 Temperature Softmax Policy

Although Dirichlet policy is unbiased, the probability ratio has a complicated expression and suffers from high numerical instability, leading to unstable training. In dLLMs, people typically sample rollouts according to $\mathrm{softmax}(\mathbf{f}_\theta^{(i)}(\mathbf{x}_t)/\tau)$ (Zhao et al., 2025c), where the temperature $\tau \geq 0$ control the sampling stochasticity (and exploration), and higher temperature leads to higher stochasticity: if $\tau = 0$, it reduces to perform deterministic greedy sampling by unmasking the token with the largest logit value, $\arg\max_{j\in\mathcal{V}} f_\theta^{(i)}(\mathbf{x}_t)_j$; if $\tau = 1$, then it is vanilla sampling according to the softmax applied to the model logits $\mathbf{f}_\theta^{(i)}(\mathbf{x}_t)$; and if $\tau = +\infty$, it corresponds to sampling from the uniform distribution over $\mathcal{V}$, which is totally random. Zhao et al. (2025c) chose $\tau = 0.9$ for training. Here, since we view the probability simplex Δ^{V-1} as the action space, dividing the logits by a single τ followed by applying softmax cannot make the resulting probability vectors cover the whole high-dimensional simplex Δ^{V-1} by varying a one-dimensional τ. Thus, we turn to consider *dimension-aware* temperature softmax exploration:

$$\mathbf{a}_t^{(i)} \sim \pi^{\theta,(i)}(\cdot \mid t, \mathbf{x}_t) \quad \text{then} \quad \mathbf{a}_t^{(i)} = \begin{cases} \mathrm{softmax}\left((f_\theta^{(i)}(\mathbf{x}_t)_j/\tau_j)_{j\in\mathcal{V}}\right), & i \in [L] : x_t^i = \mathtt{m}, \\ \mathrm{Cat}(\cdot; \mathbf{e}_{x_t^i}), & i \in [L] : x_t^i \neq \mathtt{m} \end{cases} \in \Delta^{V-1}.$$

where $\tau_j \overset{\text{i.i.d.}}{\sim} p,\ j \in \mathcal{V}$ and p is a continuous probability distribution over $(0, +\infty)$ which is to be picked. Here, each dimension of the logit $f_\theta^{(i)}(\mathbf{x}_t)_j$ is perturbed by a different but i.i.d. τ_j, therefore enhancing the degree of freedom of τ to V while avoiding increasing much complexity by specifying only a one-dimensional probability density p, in contrast to a complicated $(V-1)$-dimensional Dirichlet density. The dimension-aware temperature softmax approach also allows us to give an estimation of the probability ratio. Suppose $\mathbf{a}_t^{(i)} = \mathrm{softmax}((f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j/\tau_j)_{j\in\mathcal{V}})$ for some $\{\tau_j\}_{j\in\mathcal{V}}$. One can see that there exists $\tau_j' = f_\theta^{(i)}(\mathbf{x}_t)_j \cdot \tau_j / f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j$ such that $\mathbf{a}_t^{(i)} = \mathrm{softmax}((f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j/\tau_j)_{j\in\mathcal{V}}) = \mathrm{softmax}((f_\theta^{(i)}(\mathbf{x}_t)_j/\tau_j')_{j\in\mathcal{V}})$. Hence, we obtain that

$$\begin{aligned} \rho_t^\theta = \frac{\pi^\theta(\mathbf{a}_t \mid t, \mathbf{x}_t)}{\pi^{\theta_{\text{old}}}(\mathbf{a}_t \mid t, \mathbf{x}_t)} &= \prod_{i\in B_t : x_t^i = \mathtt{m}} \frac{\pi^{\theta,(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t)}{\pi^{\theta_{\text{old}},(i)}(\mathbf{a}_t^{(i)} \mid t, \mathbf{x}_t)} \\ &\approx \prod_{i\in B_t : x_t^i = \mathtt{m}} \prod_{j\in\mathcal{V}} \frac{f_\theta^{(i)}(\mathbf{x}_t)_j}{f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j} \cdot \frac{p(\tau_j')}{p(\tau_j)} \\ &= \prod_{i\in B_t : x_t^i = \mathtt{m}} \prod_{j\in\mathcal{V}} \frac{f_\theta^{(i)}(\mathbf{x}_t)_j}{f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j} \cdot \frac{p(f_\theta^{(i)}(\mathbf{x}_t)_j \cdot \tau_j / f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j)}{p(\tau_j)}, \end{aligned} \tag{19}$$

where the approximation arises because the softmax function is not injective. One can also consider adaptive τ_t, where $(\tau_t)_j \overset{\text{i.i.d.}}{\sim} p_t,\ j \in \mathcal{V}$ for a time-dependent continuous distribution p_t. In particular, if we choose time-dependent p_t as $\mathrm{Exp}(\lambda_t)$ which we refer to as exponential-temperature softmax policy, then

$$\rho_t^\theta \approx \exp\left(\sum_{i\in B_t : x_t^i = \mathtt{m}} \sum_{j\in\mathcal{V}} \left(\log \frac{f_\theta^{(i)}(\mathbf{x}_t)_j}{f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j} + \lambda_t(\tau_t)_j \left(1 - \frac{f_\theta^{(i)}(\mathbf{x}_t)_j}{f_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t)_j}\right)\right)\right).$$

B.1.3 Logistic-Normal Policy

Although the temperature-softmax policy is easy to implement, its probability ratio cannot be computed exactly because the softmax mapping is not injective: $\mathrm{softmax}(\mathbf{f}) = \mathrm{softmax}(\mathbf{f} + c)$ for any $c \in \mathbb{R}$. We therefore consider the logistic-normal distribution as the policy distribution, which admits an analytical probability density (Floto et al., 2023).

Logistic Transform $\mathrm{LT} : \mathbb{R}^{V-1} \to \Delta^{V-1}$ defined by

$$\mathrm{LT}(\mathbf{y}) := \begin{cases} \frac{e^{y_i}}{1+\sum_{j=1}^{V-1} e^{y_j}}, & \text{if } i \in [V-1], \\ \frac{1}{1+\sum_{j=1}^{V-1} e^{y_j}}, & \text{if } i = V \end{cases}$$

is a bijection. At time t given partially masked sequence (state) $\mathbf{x}_t$, we have model logits $\mathbf{f}_\theta(\mathbf{x}_t) \in \mathbb{R}^V$. Vanilla sampling: $\mathrm{softmax}(\mathbf{f}_\theta(\mathbf{x}_t))$. Now consider exploring this logits via Logistic-Normal. Let $p_\theta(\mathbf{x}_t)_j := f_\theta(\mathbf{x}_t)_j - f_\theta(\mathbf{x}_t)_V$ (subtracting a baseline to reduce one dimension) for all $j \in [V-1]$. Then $\mathbf{p}_\theta(\mathbf{x}_t) \in \mathbb{R}^{V-1}$. Then let the action be

$$\mathbf{a}_t \sim \mathrm{LT}(\mathcal{N}(\mathbf{p}_\theta(\mathbf{x}_t), \sigma_t^2 I_{V-1})).$$

If $\sigma_t = 0$, then $\mathbf{a}_t = \mathrm{softmax}(\mathbf{f}_\theta(\mathbf{x}_t))$. The density of $\mathbf{a}_t$ has a closed form (see, e.g., Floto et al. (2023, Equation (4))). Now given stored $\mathbf{a} \sim \pi^{\theta_{\text{old}}}$, let $\mathbf{y} = \mathrm{LT}^{-1}(\mathbf{a}) \in \mathbb{R}^{V-1}$ and suppose $y_j = p_{\theta_{\text{old}},j} + \sigma\epsilon_j$ for $j = 1, \cdots, V-1$ (In practice, we store ϵ_j and $p_{\theta_{\text{old}},j}$ for all $j = 1, \cdots, V-1$ during rollout), we have the probability ratio

$$\begin{aligned} \rho^\theta = \frac{\pi^\theta(\mathbf{a}|\mathbf{x})}{\pi^{\theta_{\text{old}}}(\mathbf{a}|\mathbf{x})} &= \exp\left(\frac{1}{2\sigma^2}\sum_{j=1}^{V-1}\left((y_j - p_{\theta_{\text{old}},j})^2 - (y_j - p_{\theta,j})^2\right)\right) \\ &= \exp\left(\frac{1}{2\sigma^2}\sum_{j=1}^{V-1}\left(\sigma^2\epsilon_j^2 - (p_{\theta_{\text{old}},j} - p_{\theta,j} + \sigma\epsilon_j)^2\right)\right), \end{aligned} \tag{20}$$

in which everything is dependent on t.

Remark 4. We empirically observe that the three policies described above lead to unstable training. Since these policies are supported over the entire simplex, the probability ratios in (18), (19), and (20) involve summation over the entire vocabulary, which can result in exploding gradient norms when the vocabulary is large. Furthermore, the probability ratios may become excessively large during training. In (18), this is caused by the singularity of $\log\Gamma(\cdot)$, while in (19), it arises when the old logits are close to zero. Consequently, most probability ratios are clipped into the interval $[1-\epsilon, , 1+\epsilon]$ by PPO or GRPO, leading to negligible policy updates and ineffective learning. Although these policies are not suitable for training, we empirically observe that exploring the entire simplex during inference can be beneficial for generating training rollouts. □

B.2 Policies over the Simplex Vertices

Motivated by these observations stated in Remark 4, in this section we introduce a class of policies that support on the vertices of the simplex. This design avoids the optimization issues associated with full-simplex policies and yields more stable and effective training. We first recall the d1 loss (Eq (4) in Zhao et al. (2025c)) is equivalent to (if we do not consider random masking of prompts)

$$\mathcal{L}_{d1}(\theta) := \mathop{\mathbb{E}}_{q\sim\mathcal{D},\{(\mathbf{x}_{g,0},\mathbf{x}_{g,1},\cdots,\mathbf{x}_{g,T})\}_{g=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{g=1}^{G}\frac{1}{L}\sum_{t=1}^{L}\min\left(\rho_{g,t}^\theta A_g, \mathrm{clip}\left(\rho_{g,t}^\theta, 1-\epsilon, 1+\epsilon\right)A_g\right)\right], \tag{21}$$

where

$$\rho_{g,t}^{\theta} = \frac{p_{\theta}^{(i_{g,t})}(x_{g,T}^{i_{g,t}} \mid \mathbf{x}_{g,0})}{p_{\theta_{\text{old}}}^{(i_{g,t})}(x_{g,T}^{i_{g,t}} \mid \mathbf{x}_{g,0})}. \tag{22}$$

This loss is autoregressive (AR)-style, where the inner summation in (21) is computed to average the sequence-length steps. Given this insight, the corresponding d1 sampling procedure here should be understood in the sense that exactly one token is unmasked at each time step; in (22), $i_{g,t} \in [L]$ is the masked token index/position with the highest confidence in the current block at time step t for the g-th denoising trajectory and $x_{g,t}^{i_{g,t}} = \mathtt{m}$ is to be unmasked at time t.

However, the actual policy for the d1 sampling is diffusion-style (see Appendix D in Zhao et al. (2025c)), i.e., two masked tokens with the highest confidence score are unmasked at one time step:

$$\mathbf{a}_t^{(i)} \sim \pi^{\theta,(i)}(\cdot \mid t, \mathbf{x}_t) \quad \text{then} \quad \mathbf{a}_t^{(i)} = \begin{cases} \mathbf{e}_j & \text{w.p. } p_{\theta}^{(i)}(\mathbf{x}_t)_j,\ j \in \mathcal{V}, \\ \mathbf{e}_{x_t^i}, & \end{cases} \begin{array}{ll} \text{for } i \in \mathcal{M}_t^{\text{Top2Conf}}, \\ \text{for } i \notin \mathcal{M}_t^{\text{Top2Conf}} \end{array} \in \Delta^{V-1}. \tag{23}$$

Note that the action distribution for a position $i \in \mathcal{M}_t^{\text{Top2Conf}}$ is supported on the V vertices of the probability simplex Δ^{V-1}. Apparently, there is a gap between the diffusion-style sampling procedure (23) and the AR-style loss construction for the d1 loss (21). To be more specific, if we look at the loss (21) in which the probability ratios are only conditioned on the initial state/prompt $\mathbf{x}_0$ and the inner sum is averaged over the sequence length steps, the sampling procedure is as if

$$\mathbf{a}_t^{(i)} \sim \pi^{\theta,(i)}(\cdot \mid t, \mathbf{x}_t) \quad \text{then} \quad \mathbf{a}_t^{(i)} = \begin{cases} \mathbf{e}_j & \text{w.p. } p_{\theta}^{(i)}(\mathbf{x}_0)_j,\ j \in \mathcal{V}, \\ \mathbf{e}_{x_t^i}, & \end{cases} \begin{array}{ll} \text{for } i = i_t, \\ \text{for } i \neq i_t \end{array} \in \Delta^{V-1}, \tag{24}$$

which in fact/in reality is not this case as shown in (23). The "as if" argument is: since the token $x_T^{i_t} \in \mathcal{V}$ is sampled from $\mathbf{a}_t^{(i_t)} \in \Delta^{V-1}$, by (24) we have $\mathbf{a}_t^{(i_t)} = \mathbf{e}_{x_T^{i_t}}$, and thus the policy probability ratio

$$\frac{\pi^{\theta}(\mathbf{a}_t \mid t, \mathbf{x}_t)}{\pi^{\theta_{\text{old}}}(\mathbf{a}_t \mid t, \mathbf{x}_t)} = \frac{\pi^{\theta,(i_t)}(\mathbf{a}_t^{(i_t)} \mid t, \mathbf{x}_t)}{\pi^{\theta_{\text{old}},(i_t)}(\mathbf{a}_t^{(i_t)} \mid t, \mathbf{x}_t)} = \frac{p_{\theta}^{(i_t)}(x_T^{i_t} \mid \mathbf{x}_0)}{p_{\theta_{\text{old}}}^{(i_t)}(x_T^{i_t} \mid \mathbf{x}_0)},$$

which coincides with the probability ratio (22) of d1. Motivated by our continuous-time RL perspective, we bridge this gap by revising the d1 loss into our loss (15), which is consistent with the underlying continuous-time RL dynamics and the inference procedure.

However, evaluating (15) exactly requires one forward pass on each reconstructed state $\mathbf{x}_{g,t}$, for every step $t = 1, \ldots, T$ with gradients enabled. With T on the order of 128–512 this is the dominant cost in both memory and wall-clock time: the autograd graph for every step is held until the backward pass. To address this, one can consider gradient checkpointing across the step loop by recomputing each step's forward during backward instead of storing all activations. This keeps the math identical and brings peak memory back to roughly one forward graph at a time, at the cost of one extra forward pass per step. The extra recomputation makes the policy pass slower, with roughly $2\times$ the forwards for that pass. The other practical option is to subsample the trajectory described in Section 4.2, compute the loss on a random subset of N denoising steps each update, trading exactness for memory. Trajectory subsampling reduces the number of evaluated steps per optimizer update from T to a small $N \ll T$, giving an approximately T/N reduction in the number of policy forward/backward passes, at the cost of a stochastic but unbiased estimate (Proposition 4) of the per-rollout mean. In Figure 3, we show the accuracy trade-off of CTRL with different values of N in the GSM8K benchmark. We found that $N = 8$ is a favorable balance between the accuracy and training efficiency, which reduces the convergence time without compromising the final reward.

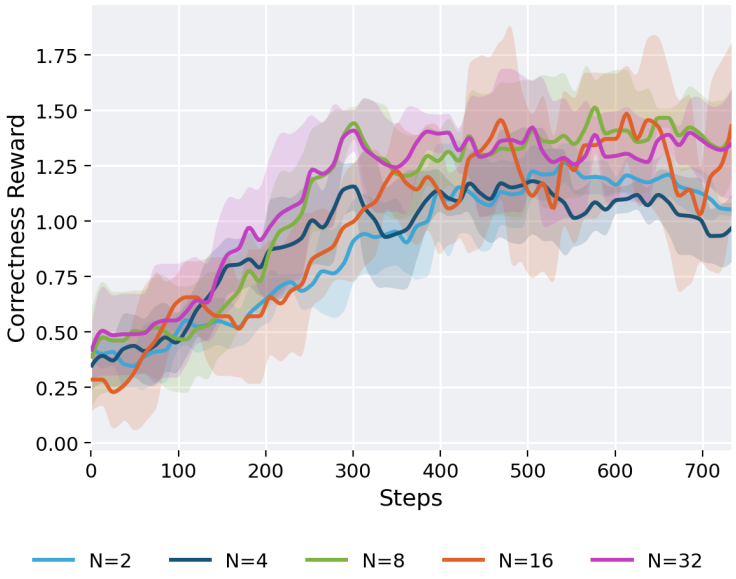

Figure 3: Ablation on N on GSM8K.

Proposition 4. *Write* $\ell_t(\theta) := \min(\rho_t A_t, \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t)$ *for the per-step GRPO surrogate and* $\bar{\ell}_t(\theta) := \sum_{t=1}^{T} \ell_t(\theta)/T$. *For a subset* $\mathcal{T}$ *drawn uniformly without replacement from* $[T]$ *with* $|\mathcal{T}| = N$,

$$\mathop{\mathbb{E}}_{\mathcal{T}}\left[\widehat{\ell}_{\mathcal{T}}(\theta)\right] \;=\; \bar{\ell}(\theta).$$

Proof of Proposition 4. For uniform sampling without replacement, each index t has marginal inclusion probability $\mathbb{P}(t \in \mathcal{T}) = N/T$. Hence,

$$\mathop{\mathbb{E}}_{\mathcal{T}}\left[\widehat{\ell}_{\mathcal{T}}(\theta)\right] = \frac{1}{N}\sum_{t=1}^{T}\mathbb{P}(t\in\mathcal{T})\,\ell_t(\theta) = \frac{1}{N}\sum_{t=1}^{T}\frac{N}{T}\,\ell_t(\theta) = \frac{1}{T}\sum_{t=1}^{T}\ell_t(\theta) = \bar{\ell}(\theta).$$

□

C PPO Implementation Insights for DLLMs

In this section, we present an alternative approach to value function approximation for PPO that avoids training a value network by leveraging sampled rollouts and a terminal reward function, inspired by DISPO (Oba et al., 2026). In PPO, computing the intermediate advantage requires estimating the value function: for $\mathbf{a}_t \sim \pi^\theta(\cdot \mid t, \mathbf{x}_t)$, we can write

$$q(t, \mathbf{x}_t, \mathbf{a}_t; \pi^\theta(\cdot \mid t, \mathbf{x}_t)) = \frac{1}{1-t}\sum_{i\in\mathcal{M}_t^{\text{TopkConf}}}\sum_{j\in\mathcal{V}}(\mathbf{a}_t^{(i)})_j\left(V^\theta(t, \mathbf{x}_t^{\setminus i}\odot j) - V^\theta(t,\mathbf{x}_t) - \beta\log\frac{p_\theta^{(i)}(\mathbf{x}_t)_j}{p_{\theta_{\text{pre}}}^{(i)}(\mathbf{x}_t)_j}\right).$$

One may drop the scalar factor $\frac{1}{1-t}$ and set $\beta = 0$ first, then we have $q_t := q(t, \mathbf{x}_t, \mathbf{a}_t; \pi^\theta(\cdot \mid t, \mathbf{x}_t)) \approx \sum_{i\in\mathcal{M}_t^{\text{TopkConf}}}\sum_{j\in\mathcal{V}}(\mathbf{a}_t^{(i)})_j\left(V^\theta(t, \mathbf{x}_t^{\setminus i}\odot j) - V^\theta(t,\mathbf{x}_t)\right)$. Noting that $\mathbf{a}_t^{(i)}$ is a probability distribution

over $\mathcal{V}$, we can write

$$\begin{aligned} q_t &\approx \sum_{i \in \mathcal{M}_t^{\text{TopkConf}}} \mathop{\mathbb{E}}_{j \sim \mathbf{a}_t^{(i)}} \left(V^\theta(t, \mathbf{x}_t^{\setminus i} \odot j) - V^\theta(t, \mathbf{x}_t) \right) \\ &\approx \sum_{i \in \mathcal{M}_t^{\text{TopkConf}}} \left(V^\theta(t, \mathbf{x}_t^{\setminus i} \odot x_{t+1}^i) - V^\theta(t, \mathbf{x}_t) \right) \\ &\approx V^\theta(t, \mathbf{x}_{t+1}) - V^\theta(t, \mathbf{x}_t), \end{aligned}$$

where the last approximation is because we simultaneously unmask the k tokens x_{t+1}^i (for $i \in \mathcal{M}_t^{\text{TopkConf}}$) at time step t. To estimate the value functions, in the rollouts, for each time step t, we resample Z outputs $\{\mathbf{o}_{t,z}\}_{z=1}^Z$ from $\mathbf{x}_t$ by one-step generation using the current logits $\mathbf{f}_\theta(\mathbf{x}_t)$, and then estimate $V^\theta(t, \mathbf{x}_t)$ via the Monte Carlo average $\frac{1}{Z}\sum_{z=1}^Z \text{RM}(\mathbf{o}_{t,z})$. Hence, we have the estimator

$$q_t \approx \frac{1}{Z} \sum_{z=1}^{Z} \left(\text{RM}(\mathbf{o}_{t+\Delta t,z}) - \text{RM}(\mathbf{o}_{t,z}) \right).$$

In a rollout generated by $\pi^{\theta_{\text{old}}}$, given $\mathbf{x}_t$ at time t, we resample Z outputs $\{\mathbf{o}_{t,z}\}_{z=1}^Z$ where each $\mathbf{o}_{t,z}$ fills all the masked positions of $\mathbf{x}_t$ via one-step generation according to $\text{softmax}(\mathbf{f}_{\theta_{\text{old}}}^{(i)}(\mathbf{x}_t))$ $(i \in [L] : x_t^i = \texttt{m})$, and compute $V_t := \frac{1}{Z}\sum_{z=1}^Z \text{RM}(\mathbf{o}_{t,z})$. We store only V_t and V_{t+1} in order to compute $q_t = V_{t+1} - V_t$. After all $\{q_t\}_{t\in[T]}$ have been computed and stored, the intermediate values $\{V_t\}_{t\in[T]}$ are discarded, thereby reducing the memory footprint. However, we found that this estimator for q leads to unstable training and exhibits high variance, possibly because its estimates are coarse at small t during the early denoising steps due to one-step generation. We leave further stabilization techniques and engineering improvements for PPO to future work.

D Experiment Details

D.1 Synthetic Example

DAM gives the entropy-regularized optimum $p^*(\mathbf{X}_1) \propto p^*(\mathbf{X}_1)e^{h(\mathbf{X}_1)}$:

$$p^*(i,j) \propto p^{\text{base}}(i,j)\, \exp(h(i,j)/\beta) = \frac{\exp(h(i,j)/\beta)}{\sum_{a,b} \exp(h(a,b)/\beta)}, \tag{25}$$

since p^{base} is uniform.

Reward Functions. Following So et al. (2026), the (terminal) reward $h(\cdot)$ is assigned via upweighting the diagonal blocks by 4.6, superdiagonal and subdiagonal blocks by 4.0, and other off-diagonal blocks by 3.4.

Implementation. All experiments are conducted on CPU. The hyperparameters for the three methods are summarized in Table 3.

Evaluation. After each iteration we compute the exact terminal grid $p_\theta(i,j)$ and report $\text{KL}(p^* \,\|\, p_\theta)$ against (25), the average reward $\mathbb{E}_{p_\theta}[h]$, and the aggregated mass of each of the 25 blocks. Table 4 reports the quantitative convergence results in terms of KL divergence and average reward, corresponding to the last two subfigures in Figure 1.

Quantity	PPO	DAM	D1
Objective	value matching	adjoint matching	reward max. (GRPO)
Critic	exact tabular	none (adjoint)	none (critic-free)
Estimator	exact expectation	MC adjoint, K=12	one-step log-prob
Group size G	—	—	64
Inner updates μ	4	1	4
Clip ϵ	0.2	—	0.2
KL weight β	6.0	6.0	6.0
Learning rate	0.3	0.3	0.3

Table 3: Hyperparameters. The problem definition is identical across methods, so performance differences are attributable to the algorithm.

Method	$\mathrm{KL}(p^* \,\|\, p_\theta)$	avg. reward
PPO	0.00044	2.793
DAM	0.004	2.560
D1	0.0769	2.282
optimal p^*	0	2.850

Table 4: Final performance on the checkerboard task (400 iterations).

D.2 Mathematical Reasoning

Inference. Following d1 (Zhao et al., 2025c), our training rollouts are generated by Top2 confidence (i.e., $k = 2$ in the main text) and random sampling using the semi-autoregressive decoding strategy. Specifically, we have $T = L/2$ and the 2 tokens with the highest confidence within the current block are unmasked, with the unmasked tokens sampled from $\mathbf{p}_\theta(\mathbf{x}_t)$, at each time step. The block size is set to 32, and once all the tokens in the current block are unmasked, we move to the next block of 32 tokens. To encourage exploration, we adopt a dimension-aware temperature-softmax strategy, where the temperature is sampled from a time-independent exponential distribution, described in Appendix B.1.2, to encourage exploration:

$$\mathbf{p}_\theta^{(i)}(\mathbf{x}_t) = \mathrm{softmax}((f_\theta^{(i)}(\mathbf{x}_t)/\tau_j)_{j\in\mathcal{V}}), \quad i \in \mathcal{M}_t^{\mathrm{Top2Conf}},$$

where $\tau_j \overset{\text{i.i.d.}}{\sim} \mathrm{Exp}(\lambda),\ j \in \mathcal{V}$. In Figure 4, we show the ablation results with different choices of λ on GSM8K with a sequence length of 128. We can see that $\lambda = 2.0$ yields a slightly better result, which we take in our experiments. When λ is either too large or too small, the exploration becomes excessive, causing the model logits to be overly perturbed and consequently degrading generation quality.

The terminal reward functions $\mathrm{TRF}(\cdot)$ for the four tasks are verifiable reward functions taken the same as d1 (see Appendix D.1.1 in Zhao et al. (2025c)). Our intermediate verifiable reward functions $\mathrm{IRF}(\cdot)$ are specified as follows:

- **GSM8K.** The intermediate reward consists of three non-positive components:
  1. an *answer-region legality penalty*, applying -1.0 if the `<answer>···</answer>` region is fully visible but contains any visible character that does not belong to the integer

category (letters, currency symbols, commas, decimal points, or digit runs split by visible content);

2. a *trailing-content penalty* of -0.001 per visible character after `</answer>`. The maximal amount of penalty from this component is capped at -0.5 ;
3. a *tag-structure penalty* of -0.125 per duplicated fully visible tag and -0.5 for an impossible tag order (e.g. `</answer>`$\cdots$`<answer>`). Masked positions are never penalized.

- **MATH500.** The intermediate reward consists of three non-positive components:
  1. an *answer-region legality penalty*, adapted to the `\boxed{}` format of the terminal reward: applying -1.0 if the `<answer>`$\cdots$`</answer>` region is fully visible but contains no `\boxed{`$\cdots$`}` expression;
  2. a *trailing-content penalty* of -0.001 per visible character after `</answer>`. The maximal amount of penalty from this category is capped at -0.5;
  3. a *tag-structure penalty* of -0.125 per duplicated fully visible tag and -0.5 for an impossible tag order. Masked positions are never penalized.
- **Countdown.** The sequence is split at the first fully visible `<answer>` tag, and the intermediate reward consists of two components, with target correctness left to TRF$(\cdot)$:
  1. an *equation-shaping award* on the reasoning region: each unique, fully visible arithmetic equation is verified with exact rational arithmetic and awarded $+0.05$ for correctness and -0.5 otherwise. Partially masked equations are skipped;
  2. an *answer-region legality penalty*, applying a flat -1.0 if any illegal visible character appears (only digits, $+$, $-$, $*$, $/$, parentheses, and whitespace are permitted) or if a fully determined digit run does not match an available number.
- **Sudoku.** The intermediate penalty is evaluated in the following steps:
  1. Pad or truncate the extracted answer to a 16-character grid as in the terminal validator, and only the cells from positions empty in the original puzzle are inspected;
  2. Check each inspected cell for *value legality*, only a value belongs to $\{1, 2, 3, 4\}$ is legal;
  3. Compute intermediate penalty according to

$$r_{\text{intermediate}} = \frac{\#\{\text{invalid inspected grids}\}}{\#\{\text{all inspected grids}\}}.$$

All four IRF's are mask-tolerant: they score partially denoised sequences by grading only visible content. The intermediate scores are scaled by a tunable weight hyperparameter $\alpha_{\text{intermediate}}$ (0.05 by default).

Implementation. We build our implementation on top of the same codebase of d1[1]. Our CTRL uses Low-Rank Adaptation (LoRA) (Hu et al., 2022) with a rank of $r = 128$ and scaling factor $\alpha = 64$, and AdamW (Loshchilov and Hutter, 2019) parameters $(\beta_1, \beta_2) = (0.9, 0.99)$ with a weight decay of 0.1 and learning rate 3×10^{-6}. All our training is conducted on 7 NVIDIA L40S GPUs, batch size of 6 per GPU and use gradient accumulation steps of 2. For GRPO hyperparameters,

[1] https://github.com/dllm-reasoning/d1

the group size $G = 6$, the number of inner gradient update is 6, and the clip parameter $\epsilon = 0.5$. The same configuration is also used for the three baseline training, and we set $N = 16$ in d2 (Wang et al., 2026c) as the authors suggest.

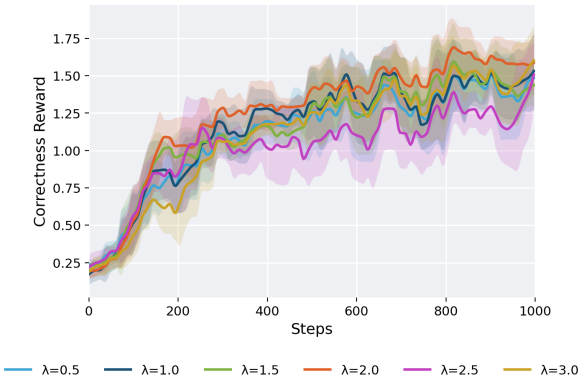

Figure 4: Ablation on λ on GSM8K.

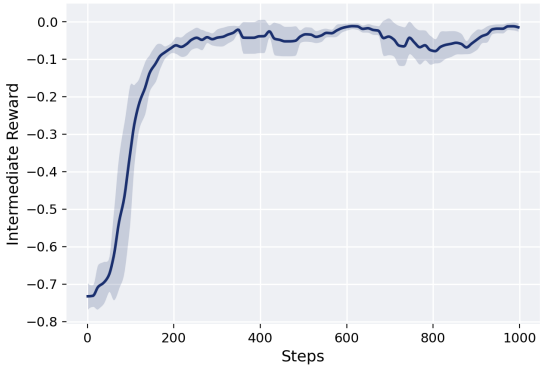

Figure 5: Convergence of intermediate reward of Sudoku.