REINFORCEMENT LEARNING WITH DISCRETE DIFFUSION POLICIES FOR COMBINATORIAL ACTION SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) struggles to scale to large, combinatorial action spaces common in many real-world problems. This paper introduces a novel framework for training discrete diffusion models as highly effective policies in these complex settings. Our key innovation is an efficient online training process that ensures stable and effective policy improvement. By leveraging policy mirror descent (PMD) to define an ideal, regularized target policy distribution, we frame the policy update as a distributional matching problem, training the expressive diffusion model to replicate this stable target. This decoupled approach stabilizes learning and significantly enhances training performance. Our method achieves state-of-the-art results and superior sample efficiency across a diverse set of challenging combinatorial benchmarks, including DNA sequence generation, RL with macro-actions, and multi-agent systems. Experiments demonstrate that our diffusion policies attain superior performance compared to other baselines.

1 INTRODUCTION

Reinforcement learning (RL) has been instrumental in pushing the boundaries of autonomous decision-making, achieving superhuman performance in a diverse range of complex sequential tasks (Silver et al., 2016; Vinyals et al., 2019; Schrittwieser et al., 2020). However, a significant frontier remains: scaling these successes to problems with vast, combinatorial discrete action spaces. Such challenges are not niche; they are central to many real-world applications, including planning with macro-actions in hierarchical RL (Sutton et al., 1999a; Durugkar et al., 2016), coordinating strategies in multi-agent systems (Hernandez-Leal et al., 2019), and generating slates in recommender systems (Ie et al., 2019). The sheer scale of these action spaces poses a fundamental challenge to standard RL algorithms, demanding highly efficient policy parameterizations and effective exploration strategies.

Prior approaches have attempted to mitigate this complexity by mapping actions to lower-dimensional subspaces (Stulp et al., 2012; Tennenholtz & Mannor, 2019), employing hierarchical training schemes (Nachum et al., 2018), or assuming specific structural properties of the action space (Carrara et al., 2019). While effective in certain contexts, these methods often rely on structural assumptions or inductive biases that may not hold in more general and complex problem settings. More recently, the success of autoregressive models (Vaswani et al., 2017) has inspired their use for policies over combinatorial actions (Chen et al., 2021; Wen et al., 2022b). Yet, these models suffer from two key limitations: high computational cost during inference due to their sequential generation process and the imposition of a causal action ordering, which is often an artificial and restrictive constraint.

Diffusion models have emerged as a powerful class of generative models, renowned for their ability to capture highly complex probability distributions without imposing a causal structure (Sohl-Dickstein et al., 2015; Ho et al., 2020). Recent extensions to discrete spaces have further broadened their applicability (Austin et al., 2021; Sun et al., 2022; Campbell et al., 2022; Shi et al., 2024). This inherent flexibility and expressiveness make them an ideal candidate for modeling policies in large, unstructured discrete action spaces. While diffusion models have been actively explored for synthesizing policies in continuous control (Wang et al., 2022; Ding et al., 2024; Ren et al., 2024; Ma et al., 2025), a principled and efficient framework for training discrete diffusion policies with RL remains unexplored.

In this work, we introduce a novel framework for training discrete diffusion models as highly effective policies for combinatorial action spaces. Our key innovation is an efficient online training process that ensures stable and effective policy improvement. We leverage policy mirror descent (PMD, Shani et al. (2020); Tomar et al. (2021); Lan (2023)) to define the ideal target policy distribution based on the PMD optimization objective. This reframes the policy update as a distributional matching problem, where we train our expressive diffusion model to replicate this stable target. This decoupled approach is critical: it separates the RL objective optimization from the complex task of representation learning, which we delegate to the diffusion model, thereby stabilizing the entire learning process and significantly enhancing performance.

Our core contributions are as follows: (1) We introduce RL-D^2, a new and efficient online training framework for using discrete diffusion models as policies in RL for combinatorial action spaces. Our core mechanism reframes the policy update as a distributional matching problem by using policy mirror descent (PMD) to define a stable target distribution, which significantly stabilizes learning. (2) We derive and analyze two practical policy improvement methods based on minimizing the forward and reverse Kullback-Leibler (KL) divergence to the PMD target. (3) Finally, we conduct extensive experiments across three distinct and challenging domains: DNA sequence generation (Gosai et al., 2023), long-horizon RL with macro-actions in Atari (Bellemare et al., 2013), and cooperative multi-agent RL in the challenging Google Research Football domain (Kurach et al., 2020). In all settings, our method achieves state-of-the-art results, demonstrating superior performance, scalability, and efficiency.

2 Related Work

Discrete Diffusion Models. Diffusion models for generating continuous data, such as images, typically rely on the gradual addition and removal of Gaussian noise to learn and synthesize complex probability distributions (Sohl-Dickstein et al., 2015; Ho et al., 2020). However, this paradigm is ill-suited for discrete data like text or biological sequences, where values are categorical and adding small amounts of continuous noise is not meaningful. To address this, *discrete diffusion models* extend the iterative refinement idea to discrete state spaces, with forward and backward processes using Markov chains where each transition in a sequence is sampled independently. Various approaches have explored different transition mechanisms and training objectives (Austin et al., 2021; Campbell et al., 2022; Sun et al., 2022; Lou et al., 2023; Shi et al., 2024). Among these, *absorbing (or masked) diffusion* has proven to be particularly effective (Sahoo et al., 2024; Ou et al., 2024). The success of these models has led to their application in a range of domains. In natural language processing, they have been adapted for complex generation tasks (Arriola et al., 2025; Ye et al., 2025; Nie et al., 2025). More relevant to our work, discrete diffusion has shown significant promise in bio-sequence modeling for generating novel proteins and DNA sequences with desired properties (Gruver et al., 2023; Wang et al., 2024a).

Reward-based Fine-tuning and RL for Discrete Diffusion. A key challenge, beyond unconditional generation, is adapting discrete diffusion to optimize for specific objectives. This has primarily been approached through reward-based fine-tuning, which adjusts the model's parameters to increase the likelihood of generating high-reward samples. For instance, (Wang et al., 2024a) enable direct reward backpropagation by leveraging the Gumbel-softmax trick, while ()zekri2025fine optimize the model by manipulating the score entropy (Lou et al., 2023). While effective, these methods can be viewed as forms of a single-step policy optimization. By contrast, the application of online RL to discrete diffusion is unexplored, and faces challenges such as exploration-exploitation trade-offs, computational efficiency, horizon-complexity trade-off.

3 Preliminaries

In this section, we review the necessary background. We first define the problem setup for RL with large, combinatorial action spaces. We then introduce policy mirror descent as the foundation for our policy improvement step, followed by a review of discrete diffusion models, which will serve as our policy parameterization.

3.1 Problem Setup

We consider a *Markov decision process (MDP)* defined by a tuple $(\mathcal{S}, \mathcal{A}^K, P, \gamma, r, \rho_0)$, where $\mathcal{S}$ is the state space, P is the transition function, r is a reward function, $\gamma \in [0,1)$ is the discount factor, and ρ_0 is the initial state distribution. The action space $\mathcal{A}^K$ is assumed to be large, with some combinatorial structure, i.e., $\mathbf{a} \in \mathcal{A}^K$ is a structured, "multi-component" object [1]. The reward function $r : \mathcal{S} \times \mathcal{A}^K \to \mathbb{R}$ and transition function $P : \mathcal{S} \times \mathcal{A}^K \mapsto \Delta_{\mathcal{S}}$ are defined w.r.t. $\mathcal{A}^K$.

This general setup encapsulates a range of different problems, one of which is *hierarchical RL* (Sutton et al., 1999a; Vezhnevets et al., 2017; Haarnoja et al., 2018), where each action $\mathbf{a} \in \mathcal{A}^K$ is a *macro-action*, or a sequence of K primitive actions, $\mathbf{a} = (a_1, \ldots, a_K)$.[2]. In this case, $r(s, \mathbf{a})$ and $P(s'|s, \mathbf{a})$ represent the total discounted reward and the final state after executing the entire K-step sequence. Other examples include multi-agent policy optimization (Hernandez-Leal et al., 2019), where $\mathbf{a} = (a_1, \ldots, a_K)$ is the joint action for K agents, where $a_i \in \mathcal{A}_i$ is the ith agent's action; slate recommendation (Ie et al., 2019), where a_i is the item at the ith position of a set/slate of recommendations of size K; and combinatorial sequential assignment (Carrara et al., 2019).

A policy $\pi : \mathcal{S} \mapsto \Delta_{\mathcal{A}^K}$ maps a state to a distribution over the action space. The state-action value function $q^\pi(s, \mathbf{a})$ is the expected return after taking action $\mathbf{a}$ in state s and following π thereafter:

$$q^\pi(s, \mathbf{a}) = r(s, \mathbf{a}) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, \mathbf{a}), \mathbf{a}' \sim \pi(\cdot|s')}[q^\pi(s', \mathbf{a}')]. \tag{1}$$

The state-value function is the expectation over actions, $v^\pi(s) := \mathbb{E}_{\mathbf{a} \sim \pi(\cdot|s)}[q^\pi(s, \mathbf{a})]$. The agent's goal is to find an optimal policy π^* within a policy class Π that maximizes the expected return: $\pi^* \in \arg\max_{\pi \in \Pi} \mathbb{E}_{s \sim \rho_0}[v^\pi(s)]$.

3.2 Policy Mirror Descent

Policy mirror descent (PMD) (Beck & Teboulle, 2003; Shani et al., 2020; Tomar et al., 2020; Lan, 2023) is a policy optimization method that provides a provably convergent stable and regularized policy improvement step. Given a current policy π_{old}, the PMD update finds a new policy π by solving:

$$\pi(\cdot|s) \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\mathbf{a} \sim \pi(\cdot|s)}[A^{\pi_{\text{old}}}(s, \mathbf{a})] - \lambda d_{KL}(\pi, \pi_{\text{old}}; s) \quad \forall s \in \mathcal{S} \tag{2}$$

where $A^{\pi_{\text{old}}}(s, \mathbf{a}) := q^{\pi_{\text{old}}}(s, \mathbf{a}) - v^{\pi_{\text{old}}}(s)$ is the advantage function, $\lambda > 0$ is a temperature parameter, and d_{KL} is the Kullback-Leibler (KL) divergence: $d_{KL}(\pi, \mu; s) := \sum_{\mathbf{a} \in \mathcal{A}^K} \pi(\mathbf{a}|s) \log \frac{\pi(\mathbf{a}|s)}{\mu(\mathbf{a}|s)}$.

The unique solution to this optimization problem is given by:

$$\pi_{\text{MD}}(\mathbf{a}|s) = \pi_{\text{old}}(\mathbf{a}|s) \exp(A^{\pi_{\text{old}}}(s, \mathbf{a})/\lambda) \;/\; Z(s), \tag{3}$$

where $Z(s) = \mathbb{E}_{\mathbf{a} \sim \pi_{\text{old}}}[\exp(q^{\pi_{\text{old}}}(s, \mathbf{a})/\lambda)]$ is the normalization constant, or partition function.

3.3 Masked Discrete Diffusion Processes

We provide a general background on discrete diffusion in this subsection, and refer the reader to Austin et al. (2021) and Appendix A for an exhaustive derivation of this method. A reader already familiar with discrete diffusion processes can skip directly to Sec. 4.

Discrete diffusion models are powerful generative models, well-suited for capturing complex distributions over structured, sequential data. We therefore focus on combinatorial action spaces that can be represented as a fixed-length sequence of K discrete actions, $\mathbf{a} = (a_0, \ldots, a_{K-1}) \in \mathcal{A}^K$. This formulation directly applies to the macro-action problem and can be adapted for other settings like multi-agent joint actions by imposing a consistent ordering on the agents. We use a masked diffusion process (Shi et al., 2024), which operates over an augmented vocabulary $\mathcal{A} \cup \{m\}$ that includes a mask action m.

[1]For simplicity we focus on power sets of $\mathcal{A}$, though more complex combinatorial action spaces can be used.

[2]We abuse terminology slightly. In general, macro-actions (or *options*) are general "local" policies with suitable termination conditions that can used within a larger hierarchical or abstract policy (Sutton et al., 1999b; Hauskrecht et al., 1998). However, the fixed sequence view of macros (a special case of the former) also appears in the literature (Durugkar et al., 2016).

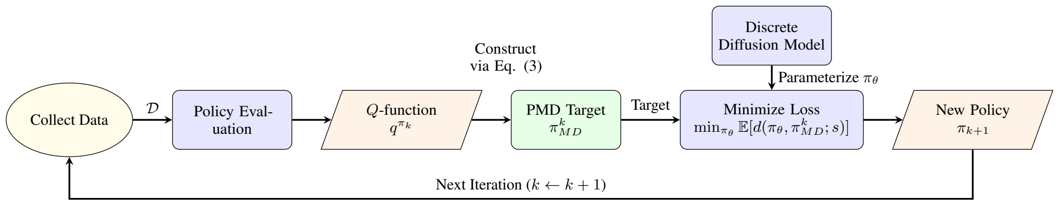

Figure 1: **Overview of the RL-D^2 Framework.** Our framework adopts a policy iteration structure. Following policy evaluation, which estimates the current Q-function, the policy improvement step is implemented as a distributional matching problem. Here, the discrete diffusion policy is trained to minimize the KL divergence (FKL or RKL) relative to an optimal target distribution (π^k_{MD}) derived via Policy Mirror Descent in equation 3.

Forward Process. The fixed forward process q gradually noises a clean macro-action $\mathbf{a}^0 \in \mathcal{A}^K$ into a fully masked sequence $\mathbf{a}^N$ over N discrete steps. This noising process is applied independently to each component $a_t \in \mathbf{a}$. The single-action transition is defined as $q(a^n = m | a^{n-1} \neq m) = 1 - \beta_n$ and $q(a^n = a^{n-1} | a^{n-1} \neq m) = \beta_n$, where $\{\beta_n\}_{n=1}^N$ is a fixed schedule. This defines a marginal distribution $q(a^n | a^0)$ where $a^n = a^0$ with probability α_n and $a^n = m$ with probability $1 - \alpha_n$, for a known noise schedule α_n.

Reverse Process. The learned reverse process $p_\theta(\mathbf{a}^{n-1} | \mathbf{a}^n, s)$ is trained to reverse this noising, conditioned on the state s. It iteratively denoises a sequence $\mathbf{a}^n$, starting from the pure noise prior $\mathbf{a}^N \sim p(\cdot|s)$, to generate a clean macro-action $\mathbf{a}^0 \sim \pi_\theta(\cdot|s)$. This process is parameterized by a model f_θ (e.g., a Transformer) that predicts the clean sequence $\mu_\theta(\mathbf{a}^n, n, s) \approx \mathbf{a}^0$ from any noised sequence $\mathbf{a}^n$ at step n.

Training Objective. The model f_θ is trained by maximizing the Evidence Lower Bound (ELBO), $\mathcal{L}_{\text{ELBO}}(\mathbf{a}^0, s; \theta)$, which is a lower bound on the log-likelihood $\log \pi_\theta(\mathbf{a}^0 | s)$. This objective trains the network to reconstruct the clean action $\mathbf{a}^0$ from its noised versions $\mathbf{a}^n$. For a macro-action $\mathbf{a}^0$ with K actions, the objective to maximize is a sum of weighted negative cross-entropy terms over all diffusion steps n and actions k:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{a}^0, s; \theta) = \sum_{n=1}^{N} \bar{\alpha}_n \mathbb{E}_{\mathbf{a}^n \sim q(\cdot|\mathbf{a}^0)} \Big[\sum_{k=0}^{K-1} \delta_{a_k^n, m} \cdot \log \mu_\theta(\mathbf{a}^n, n, s)_{a_k^0} \Big] \tag{4}$$

where $\bar{\alpha}_n$ is a weighting term derived from the noise schedule, $\delta_{a_k^n, m}$ is an indicator function that is 1 if the k-th action is masked (and 0 otherwise), and $\log \mu_\theta(\cdot)_{a_k^0}$ is the model's predicted log-probability for the original clean action a_k^0. The full derivation is detailed in Appendix A.

4 RL-D^2: Reinforcement Learning with Discrete Diffusion

We now introduce our framework for training discrete diffusion policies. Our approach follows a policy iteration structure. Let k be the current training iteration. The process alternates between (1) policy evaluation, which estimates the Q-function q^{π_k} for the current policy π_k, and (2) policy improvement. The core of our method lies in the latter improvement step.

We first define a target distribution, π^k_{MD}, which is the mirror descent iteration optimal solution from Eq. (3) calculated using $\pi_{\text{old}} \equiv \pi_k$ and $q^{\pi_{\text{old}}} \equiv q^{\pi_k}$. This transforms the policy improvement problem into a distributional approximation problem, a common paradigm in deep RL (Chan et al., 2022; Abdolmaleki et al., 2018). The new policy π_{k+1} is then obtained by finding the parameters θ that minimize a chosen divergence d to this target; namely,

$$\pi_{k+1} \in \arg\min_{\pi_\theta \in \Pi} \mathbb{E}_{s \sim \mathcal{D}} \left[d(\pi_\theta, \pi^k_{\text{MD}}; s) \right] \tag{IMPR. STEP}$$

where $\mathcal{D}$ is a distribution of states, typically from a replay buffer or current policy stationary state distribution. A flowchart summarizing our approach is presented in Fig. 1.

4.1 Reverse and Forward KL

The choice of the divergence d in (IMPR. STEP) is critical and defines the practical update rule. We focus on the Kullback-Leibler (KL) divergence (i.e., $d \equiv d_{KL}$). Specifically, we consider two variants of (IMPR. STEP) using forward KL and reverse KL divergences, which result in two different methods for policy improvement, as we explain below. We refer the reader to (Chan et al., 2022) for a thorough review of reverse and forward KL properties in RL.

Forward KL Divergence (FKL). The forward KL objective, $d_{KL}(\pi^k_{\text{MD}}, \pi_\theta; s)$, seeks a policy π_θ that covers the modes of the target distribution. This "mean-seeking" behavior can be beneficial for exploration, as it encourages the policy to maintain probability mass over all high-value actions (Chan et al., 2022). Minimizing this objective directly is intractable. Instead, we minimize a tractable bound derived by applying the diffusion model's ELBO inequality to the KL definition (see Appendix B.1). This results in the following weighted ELBO loss:

$$\mathcal{L}_{\text{FKL}}(\theta) = -\mathbb{E}_{s\sim\mathcal{D},\hat{\mathcal{A}}_s\sim\pi_k}\Big[\sum_{\mathbf{a}^0\in\hat{\mathcal{A}}_s} (\text{softmax}_{\mathbf{a}\in\hat{\mathcal{A}}_s}(A^{\pi_k}(s,\mathbf{a}^0)/\lambda)) \cdot \mathcal{L}_{\text{ELBO}}(\mathbf{a}^0, s; \theta)\Big]. \quad \text{(FKL Loss)}$$

Here, $\hat{\mathcal{A}}_s$ is a batch of macro-actions sampled from the "old" policy π_k (i.e., a target network, $\pi_{\theta_{\text{old}}}$), and $\mathcal{L}_{\text{ELBO}}$ is the weighted cross-entropy loss in equation 4. The softmax re-weights the sampled actions to approximate the target distribution π^k_{MD}. This objective effectively trains the diffusion model as a generative classifier, focusing the model's capacity on reconstructing high-value actions more frequently and is more easily adjusted to off-policy training.

Reverse KL Divergence (RKL). The reverse KL objective, $d_{KL}(\pi_\theta, \pi^k_{\text{MD}}; s)$, is equivalent to the original PMD optimization in Eq. (2) (see Appendix B.2). This objective has strong theoretical policy improvement guarantees (Chan et al., 2022) and results in a "mode-seeking" policy that focuses on the highest-value action. This objective can be written as follows:

$$\mathcal{L}_{\text{RKL}}(\theta) = \mathbb{E}_{s\sim\mathcal{D},\mathbf{a}\sim\pi_k}[-\eta(s,\mathbf{a};\theta)A^{\pi_k}(s,\mathbf{a}) + \lambda d_{KL}(\pi_\theta, \pi_k; s)], \quad \text{(RKL Loss)}$$

where here, $\eta(s,\mathbf{a};\theta) := \pi_\theta(\mathbf{a}|s)/\pi_k(\mathbf{a}|s)$ is an importance sampling (IS) ratio. In this case, $\mathcal{D}$ usual choice is the state occupancy measure of π_k (Schulman, 2015; Shani et al., 2020) for an on-policy training.

For diffusion policies, the likelihood $\pi_\theta(\mathbf{a}|s)$ is intractable, and thus so is the ratio η. Following Ren et al. (2024), we can construct an augmented MDP where states are $(s, \mathbf{a}^n)$ (an environment state and a noisy action at diffusion step n) and the advantage for a denoising step is defined by the final clean action's advantage, i.e., $A^{\pi_k}((s,\mathbf{a}^n),\mathbf{a}^{n-1}) \triangleq A^{\pi_k}(s,\mathbf{a}^0)$. This yields a tractable IS ratio based on the single-step reverse process:

$$\eta((s,\mathbf{a}^n),\mathbf{a}^{n-1};\theta) = \frac{p_\theta(\mathbf{a}^{n-1}|\mathbf{a}^n, s)}{p_k(\mathbf{a}^{n-1}|\mathbf{a}^n, s)}.$$

We refer to this ratio as "single-step ratio". Given a tractable estimator for η, we optimize the objective in Eq. (RKL Loss) using a PPO-style clipping mechanism (Schulman et al., 2017).

4.2 On-Policy Diffusion Learning

The standard ELBO objective, used in both (FKL Loss) and (RKL Loss), trains the model to denoise samples $(\mathbf{a}^n, \mathbf{a}^0)$ generated from the *fixed forward process* $q(\mathbf{a}^n|\mathbf{a}^0)$. However, this distribution of noised actions $\mathbf{a}^n$ may differ significantly from the actions the policy actually generates during its own generative process. To align the training distribution with the inference distribution, we propose *On-Policy Diffusion Learning*. Instead of starting from a clean action $\mathbf{a}^0 \sim \pi_k$ and adding noise, we generate the entire diffusion trajectory $(\mathbf{a}^N, \dots, \mathbf{a}^0)$ on-policy by sampling from the *learned reverse process* of the current policy, i.e., $\mathbf{a}^N \sim p(\cdot|s)$ and $\mathbf{a}^{n-1} \sim p_{\theta_k}(\cdot|\mathbf{a}^n, s)$. This yields $(\mathbf{a}^n, \mathbf{a}^0)$ pairs that are "on-policy" with respect to the policy's own generative dynamics, which we find enhances stability and sample efficiency. Note that the "on-policy" here only refers to the diffusion process, rather than the overall RL framework.

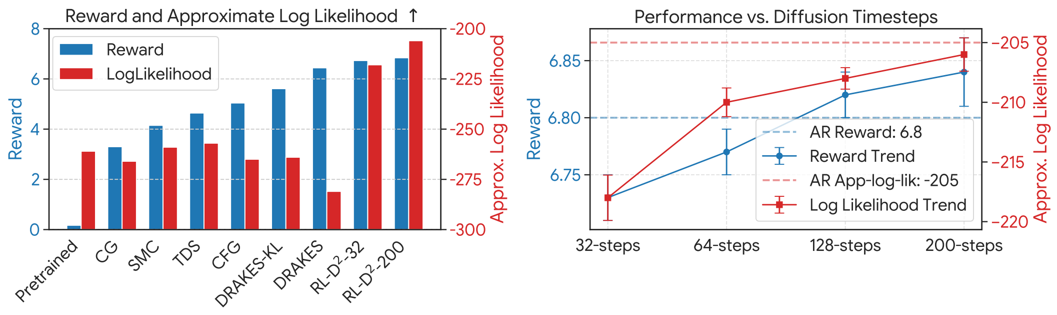

Figure 2: Reward and Approximate Likelihood of DNA generation. **Left:** The proposed RL-D^2 gets best performance on reward and log likelihood, even with fewer diffusion time steps. **Right:** The mean and 95% confidence intervals of reward and approximate log likelihood with various diffusion timesteps.

5 Experiments

We conduct a comprehensive set of experiments to evaluate our proposed framework for training discrete diffusion policies. Our evaluation spans three distinct and challenging domains to demonstrate the method's effectiveness, scalability, and versatility: (1) reward-based finetuning for DNA sequence generation, (2) online reinforcement learning with long-horizon in complex single-agent Atari environments, and (3) multi-agent cooperative learning with combinatorial joint action spaces.

5.1 DNA Sequence Generation: Single-step Policy Optimization

We first validate our approach on a reward-guided generation task, which serves as a single-step RL problem (i.e., combinatorial multi-armed bandit). The goal is to finetune a pretrained discrete diffusion model to generate DNA sequences that maximize a specific reward signal, verifying the effectiveness of our policy optimization algorithm.

We use a large public enhancer dataset of approximately 700,000 DNA sequences with length 200 (Gosai et al., 2023). A reward function, detailed in Appendix D.1, is defined to predict gene expression activity, we leverage the pre-trained reward model provided by (Wang et al., 2024a). Our primary metrics are the reward achieved and the approximate log-likelihood of the generated sequences, which measures their naturalness. We compare against controlled generation methods such as conditional guidance (CG) (Nisonoff et al., 2024), SMC and TDS (Wu et al., 2023) and classifier-free guidance (CFG) (Ho & Salimans, 2022), as well as a strong RL-based baseline, DRAKES (Wang et al., 2024a), that optimizes the policies by backpropagating reward through the reverse process using Gumbel-Softmax trick. For this task, we optimize our policy using the forward KL (FKL) objective Eq. (FKL Loss).

As shown in Fig. 2 (left), our method achieves a new state-of-the-art, attaining the highest reward scores while simultaneously generating the most probable sequences (highest log-likelihood). This demonstrates that our FKL-based update effectively optimizes for the target reward without sacrificing generative quality. Moreover, Fig. 2 (right) showed that we can achieve consistently strong performance with diffusion timesteps much smaller than the sequence length 200, highlighting the inference-time efficiency compared to autoregressive (AR) generation. Finally, our approach is significantly more computationally efficient than DRAKES. As we don't need to backpropagate through the whole reverse process, we reduced GPU memory consumption from 66.4 GB to 10.6 GB, and computation time from 268 minutes to 97 minutes compared to DRAKES, making high-performance reward optimization more accessible.

5.2 Reinforcement Learning with Macro Actions

Next, we now evaluate our RL-D^2 in the challenging MinAtar (Young & Tian, 2019) and Atari (Bellemare et al., 2013) benchmarks, where the agent learns to make decisions over long horizons by generating macro-actions, i.e., sequences of primitive actions. Our experiments are designed to assess

Table 1: Atari performance. Mean and 95% confidential intervals of scores over the last 100 evaluation episodes with 3 random seeds during training on MinAtar

	ASTERIX	BREAKOUT	FREEWAY	SEAQUEST	SPACE INVADERS
DQN	**276.10** ± **40.67**	189.85 ± 26.46	**61.05** ± **0.45**	106.02 ± 6.77	1.12K ± 156.02
DQN-MACRO	44.84 ± 2.32	300.81 ± 49.38	57.27 ± 1.18	**171.80** ± **12.83**	801.94 ± 113.24
IMPALA	24.29 ± 1.81	0.99 ± 0.00	42.72 ± 3.12	42.43 ± 3.33	46.77 ± 0.00
IMPALA-MACRO	21.95 ± 1.90	7.36 ± 0.10	52.01 ± 2.31	58.34 ± 5.17	33.86 ± 0.00
RL-D^2 (OURS)	50.37 ± 1.83	**20.18K** ± **3.75K**	**61.20** ± **0.52**	**161.0** ± **13.04**	**178.9K** ± **64.63K**

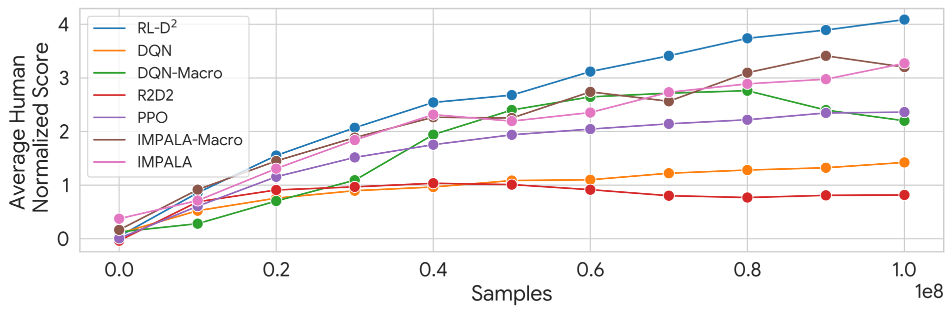

Figure 3: Atari performance. Performance improvement over the best baseline, evaluated by the percentage of human normalized scores.

the performance and scalability in using diffusion policies for complex planning tasks. For these tasks, we optimize our policy using the forward KL (FKL) objective in Eq. (FKL Loss).

We evaluate RL-D^2 on the MinAtar benchmark (Young & Tian, 2019), a suite of simplified Atari games that provide a controlled setting without partial observability. We employ a **macro-action length of** 4. We compare against DQN (Mnih et al., 2015), IMPALA (Espeholt et al., 2018), and their macro-action-enabled variants (*DQN-Macro*, *IMPALA-Macro*). In *DQN-Macro*, the Q-network's output dimension is modified to $|\mathcal{A}|^4$ to select one of all possible length-4 macro-actions. For *IMPALA-Macro*, the policy network's output is changed to $4 \times |\mathcal{A}|$, allowing it to sample the four actions of the macro-action independently at once. The detailed implementations are in Appendix D.2.

As shown in Table 1, RL-D^2 achieves substantially stronger performance in 4 out of 5 tasks in MinAtar. The substantial score improvements in BREAKOUT and SPACE INVADERS highlight the policy's ability to discover and represent complex, long-term strategies. While standard baselines adapted for macro-actions show modest gains, they are far outstripped by our approach, underscoring the necessity of an expressive generative model to effectively navigate large combinatorial action spaces.

We confirm our findings on the full Atari benchmark (Bellemare et al., 2013) with additional strong baselines including R2D2 (Kapturowski et al., 2018) and PPO (Schulman et al., 2017). As shown in Figure 3, our method achieves the highest average human-normalized score in average and outperforms strong baselines using both macro and single actions in 36 of 56 environments (the full results can be found in Appendix E.1), showcasing the strong performance of the proposed discrete diffusion.

Scalability and horizon-complexity trade-off. We investigate the scalability by varying the macro-action length. As the size of action space grows exponentially with respect to the macro action length but the horizon only shrinks linearly, solving the MDP becomes much more difficult with increasing macro action length. Fig. 4 (left) shows that with a fixed computational budget, performance peaks at a macro-action length of 4. However, the key advantage of our method is its scalability. As shown in Fig. 4 (right), when we scale up model capacity and data proportionally to the action space complexity, our diffusion policy's performance continues to improve, consistently surpassing baselines *IMPALA-Macro* and *DQN-Macro* that output the whole set of macro actions (We select Alien, BeamRider, Phoenix, Zaxxon to demonstrate scalability as RL-D^2 showed good performance with longer macro action length). *DQN-Macro* fails to fit in a reasonable learner with macro action 8

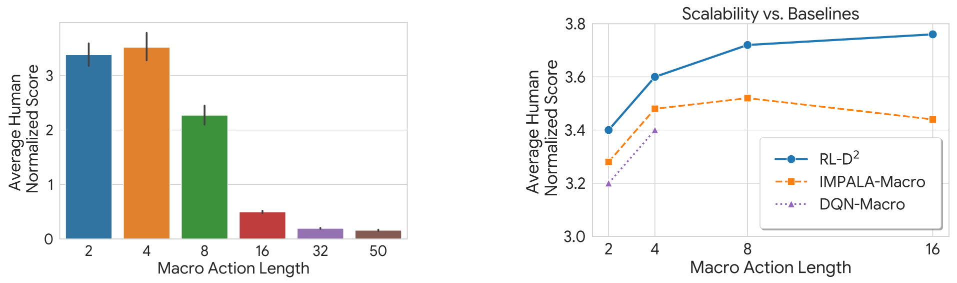

Figure 4: **Left:** Mean and 95% confidence intervals of averaged episode return over all 56 tasks to show the trade-off between planning horizon and model complexity with fixed network size and data. **Right:** The proposed method scales more effectively with increasing network size and data compared to baselines. *DQN-Macro* fails to learn in a reasonable amount of time as the action space grows too large with macro actions more than 4.

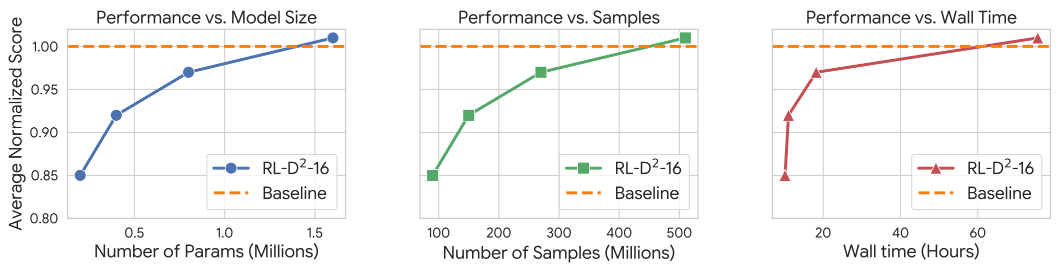

Figure 5: Mean episode return of RL-D^2 with 16 macro actions compared to the 8 macro actions as a function of model parameters, data samples, and training time, averaged over 4 tasks and 3 seed each.

and 16 as the size of action space increases exponentially. This demonstrates that our approach can effectively leverage increased resources to tackle more complex, longer-horizon problems. Moreover, we also show how our proposed approach scales up well with increasing computational resources like model sizes, samples, and training time in Fig. 5, while the baseline *IMPALA-Macro* fails to increase performance when the macro action length increase from 8 to 16.

Efficient and flexible sampling techniques. We evaluate the inference-time efficiency and flexibility of discrete diffusion policies. To make the difference clear, we extend to a long macro action setup with length 32. We leverage two techniques to further improve the sampling qualities of discrete diffusion models for these extra-long macro actions, **(1)** Top-p sampling or nuclear sampling (Holtzman et al., 2019) that selects actions from the smallest set of actions whose cumulative probabilities exceed a certain threshold; **(2)** Remasking diffusion process (Wang et al., 2025) that allows the actions to be re-masked and re-unmasked during the reverse process. The implementation details can be found in Appendix C.3.

As seen in Figure 6, top-p sampling enhanced the performance of RL-D^2 with fewer diffusion steps, such as 4 and 8, making inference-time more efficient without losing performance. Remasking sampling performs best when the number of timesteps is close to the sequence length. This highlights the flexibility of diffusion models.

More Experimental Results and Ablation studies.

In the appendix, we explore other techniques and perform ablation studies. This includes **a)** automatically tuning the temperature parameter λ in Eq. (3) by enforcing a hard KL constraint; **b)** leveraging discrete diffusion as a planner instead of committing to all macro actions generated, and **c)** the ablation study of on-policy diffusion training discussed in Sec. 4.2. **d)** comparing FKL with RKL with the same computation time. Please refer to Appendix E for the results and discussions.

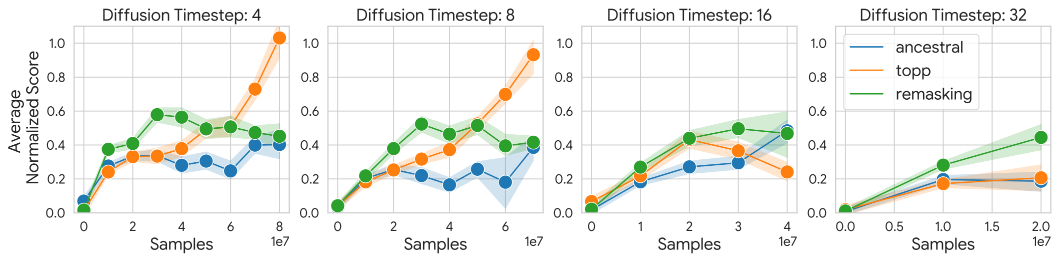

Figure 6: Mean and 95% confidence intervals of scores averaged over 4 tasks and 3 seed each normalized by best scores achieved with macro action length 32 as a function of diffusion timesteps and sampling techniques. Top-p sampling excels with very few steps while remasking improve the performance with more diffusion steps.

Table 2: Final performances of on Google Research Football academy scenarios, measured by maximum mean win rate scaled by 100 with confidence intervals.

	PASS & SHOOT	RUN PASS & SHOT	3 VS 1	CT-EASY	CT-HARD	CORNER	5 VS 5	11 VS 11	11 VS 11 SM
AUTOREGRESSIVE	78.2 ± 40.1	70.2 ± 35.2	75.8 ± 35.1	69.3 ± 30.6	33.6 ± 40.1	0.0 ± 0.0	71.1 ± 30.5	34.4 ± 14.4	0.0 ± 0.0
RL-D^2 RKL (OURS)	**100 ± 0.0**	**98.5 ± 1.6**	**98.6 ± 1.0**	**99.1 ± 0.6**	**97.7 ± 2.2**	**96.3 ± 3.4**	**99.6 ± 0.4**	**99.8 ± 0.2**	0.0 ± 0.0
RL-D^2 FKL (OURS)	**99.0 ± 1.0**	**98.6 ± 0.5**	**97.3 ± 2.0**	98.0 ± 1.1	**97.1 ± 2.9**	93.2 ± 2.9	98.4 ± 1.0	97.4 ± 0.9	**67.2 ± 8.1**
MAT	**97.9 ± 2.1**	**98.3 ± 1.2**	92.9 ± 1.1	87.9 ± 2.1	88.2 ± 3.8	95.3 ± 2.5	93.7 ± 0.9	92.0 ± 2.4	9.0 ± 3.7
MAPPO	99.5 ± 0.2	73.2 ± 3.6	93.2 ± 1.5	70.1 ± 3.8	63 ± 2.1	53.1 ± 5.3	95.4 ± 1.6	52.6 ± 3.1	5.0 ± 0.2

5.3 COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

Finally, we evaluate our framework in cooperative multi-agent reinforcement learning (MARL), where the combinatorial action space is the joint action of all agents. Efficiently searching this space is a primary challenge in MARL. By modeling the joint action distribution, our diffusion policy can capture complex inter-agent dependencies without relying on restrictive factorization assumptions.

We test our policy on the challenging Google Research Football benchmark (Kurach et al., 2020), using scenarios ranging from small-scale tasks (*3 vs 1*) to full-team games (*11 vs 11*). Our diffusion model generates the joint action for all controlled agents simultaneously. We adopt the feature extractor and scenario settings suggested by (Song et al., 2023); for further implementation details, please refer to appendix D.4. We evaluate our diffusion policy, examining both RKL and FKL objective variants against an autoregressive sampler baseline. Additionally, we compare our method against two centralized MARL policy baselines: Multi-Agent PPO (MAPPO) (Yu et al., 2022) and the **current state-of-the-art**, Multi-Agent Transformer (MAT) (Wen et al., 2022b). Note that MAT operates autoregressively, mitigating causal bias by training over random permutations of the agents' order. Finally, to assess sample efficiency, we conduct an ablation on the full game scenario (11 vs. 11) using significantly smaller budget of 100M environment steps (compared to the standard 1G) - marked as *11 vs 11 sm*.

As presented in Table 2, our discrete diffusion policy variants achieve the highest mean win rates across all scenarios. This advantage is particularly presented in the most challenging tasks requiring intricate team coordination, such as *5 vs 5*, *corner*, *ct-hard*, and the highly complex *11 vs 11* full game, where our method shows a clear advantage over state-of-the-art methods. regarding the baselines, we note that the naive autoregressive sampler lags significantly behind. This suggests that to match the performance of diffusion models, autoregressive methods require additional alignment techniques, such as the random permutations used in MAT, to mitigate causal bias. Furthermore, while the RKL objective yields superior results overall, FKL significantly outperforms RKL (and other baselines) in the small-budget regime *11 vs 11 sm*, approaching a 70% win rate. RKL is more exploratory, allowing it to achieve optimal behavior given sufficient samples, whereas the FKL exploits faster at the cost of less exploration. Moreover, we observe FKL is likely to collapse if the temperature tuning is not handled well (results shown in Appendix E.2.3). Overall, these results highlight the potential of discrete diffusion models to effectively generate highly coordinated joint actions in challenging multi-agent tasks.

5.4 Empirical Selection of FKL and RKL

We summarize our empirical observations about FKL or RKL to provide guidance to practitioners from the viewpoint of balancing the trade-off between computational efficiency and asymptotic performance.

Data efficiency: FKL demonstrates faster learning in the initial stages with fewer samples, as evidenced by the *11 vs 11 sm* scenario in Table 2 and the Atari benchmarks in Fig. 15. We attribute this to the temperature values employed. As shown in Fig. 9, FKL performs well only with a large KL constraint and small temperatures; this configuration promotes aggressive exploitation by rapidly shifting the policy toward being greedy and deterministic. In contrast, our RKL implementation utilizes importance sampling ratio clipping. This mechanism results in more gradual shifts in the policy distribution, thereby leading to a slower learning pace.

Asymptotic performance and stability. Table 2 demonstrates that RKL typically outperforms FKL when trained with large sample sizes. Moreover, the temperature schedule ablation for Google Football in Fig. 11 indicates that FKL is prone to collapse if the temperature is not appropriately selected. This slightly inferior performance of FKL may stem from two factors: (1) FKL optimizes the ELBO, a lower bound of policy loss, whereas RKL optimizes the unbiased policy mirror descent loss; and (2) the potential for collapse is caused by the fact that policy being too deterministic results in a lack of diversity in the replay buffer.

In summary, our empirical results have shown that, RL-D^2-FKL favors learning faster by heavy exploitation, but is less stable and requires careful temperature tuning. It is suitable for tasks with data and computational bottleneck such as Atari games. RL-D^2-RKL learns slightly slower but is more stable and have better asymptotic performance. It is suitable for tasks with cheap sampling cost, such as Google Football.

6 Summary

This paper introduces RL-D^2, a novel framework for reinforcement learning with discrete diffusion policies, aimed at solving decision making problems with large, combinatorial action spaces. In this framework, we propose to train diffusion models by fitting their output distribution to the analytic solution of the Policy Mirror Descent policy optimization algorithm. This is done by projecting the outputs of the model with either the forward or reverse KL divergences. Extensive experiments demonstrate that the proposed method achieves state-of-the-art performance across three challenging domains: reward-guided sequence generation, long-horizon planning with macro-actions, and cooperative multi-agent RL. The results suggest that the RL-D^2 framework provides a scalable and high-performing solution to a long-standing challenge in RL, effectively handling complex, combinatorial action spaces where traditional methods often fail.

Ethics Statement

Our work strictly follows the **ICLR Code of Ethics**. This study did not involve human subjects, personally identifiable information, or the use of proprietary data. All utilized datasets were sourced exclusively from publicly available resources that explicitly permit academic research. All authors confirm they have read and agree to comply with the ICLR Code of Ethics.

Reproducibility Statement

To ensure the reproducibility of our findings, we have provided comprehensive details of our proposed framework and experimental evaluation. The full derivations for the forward and reverse KL-divergence-based policy update rules, are presented in Sec. 4 and Appendix B. The general implementation details, model architectures, and key hyperparameters are described in Appendix C. Specific experimental setups for DNA sequence generation, Reinforcement Learning with macro-actions, and Multi-Agent Reinforcement Learning are detailed in Appendix D, which also includes descriptions of the baseline implementations and domain-specific hyperparameters. Complete experimental results and extensive ablation studies are provided in Appendix E.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

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.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18 (10):1196–1203, 2021.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.

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.

Nicolas Carrara, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric-Ambrym Maillard, and Olivier Pietquin. Budgeted reinforcement learning in continuous state space. In *Advances in Neural Information Processing Systems 32 (NeurIPS-19)*, 2019.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palanicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

Alan Chan, Hugo Silva, Sungsu Lim, Tadashi Kozuno, A Rupam Mahmood, and Martha White. Greedification operators for policy optimization: Investigating forward and reverse kl divergences. *Journal of Machine Learning Research*, 23(253):1–79, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *arXiv preprint arXiv:2405.16173*, 2024.

Ishan P Durugkar, Clemens Rosenbaum, Stefan Dernbach, and Sridhar Mahadevan. Deep reinforcement learning with macro-actions. arXiv:1606.04615, 2016.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

Sager J Gosai, Rodrigo I Castro, Natalia Fuentes, John C Butts, Susan Kales, Ramil R Noche, Kousuke Mouri, Pardis C Sabeti, Steven K Reilly, and Ryan Tewhey. Machine-guided design of synthetic cell type-specific cis-regulatory elements. *bioRxiv*, 2023.

Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36:12489–12517, 2023.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1851–1860. PMLR, 2018.

Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 220–229, Madison, WI, 1998.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

Matthew W Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Nikola Momchev, Danila Sinopalnikov, Piotr Stańczyk, Sabela Ramos, Anton Raichuk, Damien Vincent, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. SlateQ: A tractable decomposition for reinforcement learning with recommendation sets. In *Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, pp. 2592–2599, Macau, 2019.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4501–4510, 2020.

Guanghui Lan. Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes. *Mathematical programming*, 198(1):1059–1106, 2023.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of classification-based policy iteration algorithms. *Journal of Machine Learning Research*, 17(19):1–30, 2016.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Soft diffusion actor-critic: Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

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. *arXiv preprint arXiv:2406.03736*, 2024.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

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

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

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

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5668–5675, 2020.

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.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2256–2265. PMLR, June 2015.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yan Song, He Jiang, Haifeng Zhang, Zheng Tian, Weinan Zhang, and Jun Wang. Boosting studies of multi-agent reinforcement learning on google research football environment: The past, present, and future. *arXiv preprint arXiv:2309.12951*, 2023.

Freek Stulp, Evangelos A Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on robotics*, 28(6):1360–1370, 2012.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999a.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and Semi-MDPs: Learning, planning, and representing knowledge at multiple temporal scales. *Artificial Intelligence*, 112: 181–211, 1999b.

Guy Tennenholtz and Shie Mannor. The natural language of actions. In *International Conference on Machine Learning*, pp. 6196–6205. PMLR, 2019.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization, 2021. URL `https://arxiv.org/abs/2005.09814`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. *arXiv preprint arXiv:2410.13643*, 2024a.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*, 2025.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, and Shengbo Eben Li. Diffusion Actor-Critic with Entropy Regulator, December 2024b.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Muning Wen, Jakub Kuba, Ruiqing Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16706–16719, 2022a. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/6b0928eÑČÑĞĐřÑĄÑĆĐÿÑŔ82d7349b604bebc53aa1e-Abstract-Conference.html`.

Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022b.

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36:31372–31403, 2023.

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.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

A Masked Discrete Diffusion Process

Similar to the continuous diffusion, discrete diffusion is also composed by a fixed forward process and a learned reverse process. The forward process degrades a data sample $x^0 \sim q\left(x^0\right)$ into a sequence of progressively noisier latent variables $x^1, x^2, \ldots, x^N$ via a Markov chain,

$$q\left(x^{1:N} \mid x^0\right) = \prod_{n=1}^{N} q\left(x^n \mid x^{n-1}\right), \text{where} \quad q\left(x^n \mid x^{n-1}\right) = \text{Cat}\left(x^n; p = \mathbf{Q}_n^\top x^{n-1}\right)$$

where $\mathbf{Q}_n$ is the transition matrix with $\left[\mathbf{Q}_n\right]_{ij} = q\left(x^n = j \mid x^{n-1} = i\right)$. Specifically, we focus on a family of discrete diffusion processes called masked diffusion models (Austin et al., 2021; Campbell et al., 2022; Shi et al., 2024), where an additional `[MASK]` action is added to the action space. Denote the mask action as a special $\mathtt{m}$ action, the transition kernel of the forward process is defined as

$$\mathbf{Q}_n = \beta_n \mathbf{I} + (1 - \beta_n)\mathbf{1}\mathbf{e}_m^\top,$$

In another word,

$$q\left(x^n \mid x^{n-1}\right) = \begin{cases} 1 - \beta_n & \text{if } x^{n-1} \neq \mathtt{m} \text{ and } x^n = \mathtt{m} \\ \beta_n & \text{if } x^{n-1} \neq \mathtt{m} \text{ and } x^n = x^{n-1} \\ 1 & \text{if } x^{n-1} = x^n = \mathtt{m} \\ 0 & \text{otherwise} \end{cases}$$

which means the action is masked out with $1 - \beta_n$ probability, otherwise stays as the same. The masking schedule is defined as $\alpha_n := \prod_{i=1}^{n}(1 - \beta_i)$. Once the action is masked, it stays as the masked action $\mathtt{m}$. The learned reverse Markov process $p_\theta(x^{0:N}) = p(x^N)\prod_{n=1}^{N} p_\theta(x^{n-1}|x^n)$ gradually denoises (unmasks) the latent variables towards the data distribution. The reversal model estimates the posterior:

$$q(x^{n-1}|x^n, x^0) = \begin{cases} \text{Cat}\left(x^{n-1}; \bar{\alpha}_n x^0 + (1 - \bar{\alpha}_n)\mathbf{e}_m\right) & x^n = \mathbf{e}_m, \\ \text{Cat}\left(x^{n-1}; x^n\right) & x^n \neq \mathbf{e}_m \end{cases}$$

where $\bar{\alpha}_n := \frac{\alpha_{n-1} - \alpha_n}{1 - \alpha_n}$, through the parameterized model $p_\theta(x^{n-1}|x^n) := q(x^{n-1}|x^n, \mu_\theta(x^n, n))$, where

$$\mu_\theta(x^n, n) = \begin{cases} \text{softmax}(f_\theta(x^n, n)) & x^n = \mathtt{m}, \\ x^n & x^n \neq \mathtt{m}. \end{cases}$$

is the clean sample mean-value estimator induced by a trained model f_θ optimized by maximizing the Evidence Lower Bound (ELBO)

$$\mathcal{L}_{\text{ELBO}}(x^0; \theta) = \sum_{n=1}^{N} \bar{\alpha}_n \mathbb{E}_{x^n \sim q_{n|0}}\left[\delta_{x^n, \mathtt{m}} \cdot (x^0)^\top \log \mu_\theta(x^n, n)\right], \tag{5}$$

where $\delta_{x,y}$ is an indicator function and $q_{n|0} := q(x^n|x^0)$. The ELBO acts as a lower bound for the expected log-likelihood

$$\log p_\theta(x^0) \geq \mathcal{L}_{ELBO}(x^0; \theta). \tag{6}$$

Throughout this work, we abuse the notation such that x^n can be either an integer or its corresponding one-hot vector, whenever it is clear from the context.

B Proofs and Derivations

B.1 Proof to the Forward KL Loss Eq. (FKL Loss)

Proof. Denoting the forward KL:

$$\begin{aligned}
d_{KL}(\pi_{MD}, \pi_\theta; s) &= \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_{MD}(\mathbf{a}|s) \log \frac{\pi_{MD}(\mathbf{a}|s)}{\pi_\theta(\mathbf{a}|s)} \\
&= \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_{MD}(\mathbf{a}|s) \log \pi_{MD}(\mathbf{a}|s) - \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_{MD}(\mathbf{a}|s) \log \pi_\theta(\mathbf{a}|s) \\
&= -\mathcal{H}(\pi_{MD}(\cdot|s)) - \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_{\text{old}}(\mathbf{a}|s) \frac{\exp(q^{\pi_{\text{old}}}(\mathbf{a}, s)/\lambda)}{Z(s)} \log \pi_\theta(\mathbf{a}|s) \\
&\leq - \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_{\text{old}}(\mathbf{a}|s) \frac{\exp(q^{\pi_{\text{old}}}(\mathbf{a}, s)/\lambda)}{Z(s)} \mathcal{L}_{ELBO}(\mathbf{a}, s; \theta) \\
&= \mathbb{E}_{\mathbf{a}\sim\pi_{\text{old}}} \left[- \frac{\exp(q^{\pi_{\text{old}}}(\mathbf{a}, s)/\lambda)}{Z(s)} \mathcal{L}_{ELBO}(\mathbf{a}, s; \theta) \right],
\end{aligned}$$

where the first equality is the KL divergence definition, the third equality come from the definition of entropy $\mathcal{H}$ and the definition of the MD policy in Eq. (3), the inequality comes from the ELBO inequality w.r.t $\log \pi_\theta(\mathbf{a}|s)$ of the discrete diffusion policy and from the fact that entropy is has a non-negative value. □

That is, we can bound the forward KL metric using the self-imitation objective. Given the support of π_{old}: $\mathcal{A}(\pi_{\text{old}}; s) = \{\mathbf{a}^0 | \pi_{\text{old}}(\mathbf{a}^0|s) > 0, \mathbf{a}^0 \in \mathcal{A}^k\}$, the self-imitation loss is a weighted average of the discrete diffusion loss with softmax weights w_λ over this support. The SI loss is a generalized version of the classification policy iteration (Lazaric et al., 2016), which is the solution w.r.t non-regularized MD policy:

Remark 1 (Classification Discrete Diffusion Loss)**.** *Consider the limit MD policy's temperature $\lambda \to 0$, we get that:*

$$\lim_{\lambda\to 0} \mathbb{E}_{\boldsymbol{a}\sim\pi_{\text{old}}} \left[- \frac{\exp(q^{\pi_{\text{old}}}(\boldsymbol{a}, s)/\lambda)}{Z(s)} \mathcal{L}_{ELBO}(\boldsymbol{a}, s; \theta) \right] = -\frac{1}{n^*} \sum_{\boldsymbol{a}^*\in\mathcal{A}^*(\pi_{\text{old}};s)} \mathcal{L}_{ELBO}(\boldsymbol{a}^*, s; \theta),$$

where, $\mathcal{A}^*(\pi; s) := \arg\max_{\boldsymbol{a}\in\mathcal{A}(\pi_{\text{old}};s)} q^\pi(\boldsymbol{a}, s)$ and $n^* := |\mathcal{A}^*(\pi; s)|$. This is a classification policy iteration (Lazaric et al., 2016) of the discrete diffusion policy .

Overall, the self-imitation loss encapsulate a weighted behavioral cloning objective w.r.t MD policy. In practice, computing w_λ is non-trivial, as sampling actions from the whole action space may be expensive, especially in the domains consider in this work. Therefore, a practical approach would be to estimate w_λ by sampling a subset of actions $\hat{\mathcal{A}}_s \sim \pi_{\text{old}}(\cdot|s)$, where $\hat{\mathcal{A}}_s := \{\mathbf{a}_i\}_{i=1}^M, \mathbf{a}_i \sim \pi_{\text{old}}(\cdot|s)$ and perform a softmax over their q-function values, which effectively estimates the normalization factor over the sampled set:

$$Z(s) \approx \frac{1}{M} \sum_{\mathbf{a}\in\hat{\mathcal{A}}_s} \exp(q^{\pi_{\text{old}}}(s, \mathbf{a})/\lambda) = \frac{\hat{Z}(s)}{M}.$$

This gives us the next approximated loss:

$$\mathcal{L}_{FKL}(\theta) = -\mathbb{E}_{s\sim\mathcal{D}, \hat{\mathcal{A}}_s\sim\pi_{\text{old}}} \left[\sum_{\mathbf{a}^0\in\hat{\mathcal{A}}_s} \hat{w}_\lambda(s, \mathbf{a}^0) \mathcal{L}_{ELBO}(\mathbf{a}^0, s; \theta) \right],$$

where $\hat{w}_\lambda(s, \mathbf{a}) := \frac{\exp(q^{\pi_{\text{old}}}(s,\mathbf{a})/\lambda)}{\hat{Z}(s)}$ and $\mathcal{D}$ is the replay buffer.

B.2 Proof to The Reverse KL Loss Eq. (RKL Loss)

Proof. Denoting the reverse KL:

$$\begin{aligned}
d_{KL}(\pi_\theta, \pi_{MD}; s) &= \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_\theta(\mathbf{a}|s) \log \frac{\pi_\theta(\mathbf{a}|s)}{\pi_{MD}(\mathbf{a}|s)} \\
&= \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_\theta(\mathbf{a}|s) \log \pi_\theta(\mathbf{a}|s) - \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_\theta(\mathbf{a}|s) \log \pi_{MD}(\mathbf{a}|s) \\
&= \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_\theta(\mathbf{a}|s) \log \pi_\theta(\mathbf{a}|s) - \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_\theta(\mathbf{a}|s) \log \pi_{\text{old}}(\mathbf{a}|s) - \lambda^{-1} \sum_{\mathbf{a}\in\mathcal{A}^K} \pi_\theta(\mathbf{a}|s) q^{\pi_{\text{old}}}(\mathbf{a}, s) + Z(s) \\
&= d_{KL}(\pi_\theta, \pi_{\text{old}}; s) - \lambda^{-1} \mathbb{E}_{\mathbf{a}\sim\pi_\theta}[q^{\pi_{\text{old}}}(\mathbf{a}, s)] + Z(s)
\end{aligned}$$

Since the normalization factor is independent of θ

$$\arg\min_\theta d_{KL}(\pi_\theta, \pi_{MD}; s) = \arg\max_\theta \mathbb{E}_{\mathbf{a}\sim\pi_\theta}[q^{\pi_{\text{old}}}(\mathbf{a}, s)] - \lambda d_{KL}(\pi_\theta, \pi_{\text{old}}; s)$$

which is the mirror decent objective regularized with π_{old}. □

B.3 Using ELBO as an Estimator of Importance Sampling Ratios

The importance sampling ratio $\eta(s, \mathbf{a}; \theta)$ in equation RKL Loss can be estimated by a biased estimation using $\hat{\eta}_{ELBO}(s, \mathbf{a}; \theta) = \exp\left(\mathcal{L}_{ELBO}(\mathbf{a}^0, s; \theta) - \mathcal{L}_{ELBO}(\mathbf{a}^0, s; \theta_k)\right)$. To show the bias factor, we can reformulate the ELBO such that:

$$\mathcal{L}_{ELBO}(\mathbf{a}^0, s; \theta) = \underbrace{\log \pi_\theta(\mathbf{a}^0|s)}_{\text{Log-likelihood}} - \underbrace{\sum_{n=2}^{N} \mathbb{E}_{\mathbf{a}^n\sim q_{n|0}}[d_{KL}(q(\mathbf{a}^{n-1}|\mathbf{a}^n, \mathbf{a}^0, s), p_\theta(\mathbf{a}^{n-1}|\mathbf{a}^n, s))]}_{\text{Bias}(\mathbf{a}^0, s; \theta)}.$$

Examining $\hat{\eta}_{ELBO}$:

$$\begin{aligned}
\hat{\eta}_{ELBO}(s, \mathbf{a}^0; \theta) &= \exp\{\log \pi_\theta(\mathbf{a}^0|s) - \log \pi_{\theta_{\text{old}}}(\mathbf{a}^0|s) + Bias(\mathbf{a}^0, s; \theta_{\text{old}}) - Bias(\mathbf{a}^0, s; \theta)\} \\
&= \frac{\pi_\theta(\mathbf{a}^0|s)}{\pi_{\theta_{\text{old}}}(\mathbf{a}^0|s)} \exp\{Bias(\mathbf{a}^0, s; \theta_{\text{old}}) - Bias(\mathbf{a}^0, s; \theta)\} \\
&= \eta(s, \mathbf{a}^0; \theta)\Gamma(s, \mathbf{a}^0; \theta).
\end{aligned}$$

where $\Gamma(s, \mathbf{a}^0; \theta) := \exp\{Bias(\mathbf{a}^0, s; \theta_{\text{old}}) - Bias(\mathbf{a}^0, s; \theta)\}$.

We show empirical results in Appendix E.6, which is not as good as augmented MDP approach mentioned in Sec. 4.1 due to the biased nature.

C Implementation Details

C.1 RL-D^2 Implementation Details for Atari

We follow similar off-policy distributed RL framework as R2D2 (Kapturowski et al., 2018) implemented on ACME (Hoffman et al., 2020). In Atari games, we leverage the same recurrent feature extraction in (Kapturowski et al., 2018) by unrolling an LSTM network. We leverage the priority experience replay (Horgan et al., 2018). The hyperparameters are listed in Table 3.

Model Architechtures. The same 3-layer convolutional network structure as (Kapturowski et al., 2018; Mnih et al., 2015) is used for all the algorithms, followed by an LSTM with 512 hidden units, which feeds into an actor and value networks implemetned as transformers.

The learnable parameters inside our transformer include the input embedding, linear projections for conditioning, weights/biases in the multi-head attention and feed-forward networks within each transformer block. Key learnable parameters are also the adaptive normalization layers (lns) that generate dynamic shift, scale, and gate values based on the conditioning. Finally, the output projection is learnable. Conditioning is introduced via a FiLM-like (Feature-wise Linear Modulation)

Table 3: RL-D^2 Hyperparameters with FKL

Parameters	Value	Parameters	Value
Number of samples	1e8	Sample-to-insert Ratio	4.0
Number of parallel actors	16	Mini Batch size	64
Unroll length	40	Burn-in length	8
Q-network	Transformer	Target update rate	0.005
Policy network	Transformer	Actor learning rate	1×10^{-4}
Tempearture learning rate	1×10^{-2}	Critic leaning rate	1×10^{-4}
Transformer hidden dim	80	Transformer layers	3
Transformer heads	1	Discounted factor	0.997
Priority exponent	0.99	Replay buffer size	5×10^{6}

mechanism. The objective then passes through small linear networks to produce shift, scale, and gate parameters. These dynamically modulate activations after layer normalization in both the attention and feed-forward sub-layers. For example, inputs to sub-layers become `norm(h) * (scale + 1.0) + shift`, with gate controlling residual connections. This enables the transformer to adapt its internal computations layer-wise based on external conditions.

C.2 Temperature Tuning By Hard KL Divergence Constraints

Autotuning the temperature parameters λ has been a challenging problem for RL with diffusion policies, as the output log probabilities are unknown. Existing methods leverage Gaussian mixture fitting (Wang et al., 2024b), uniform data insertion (Ding et al., 2024), and data processing inequalities (Celik et al., 2025).

We noticed a simple approach using duality by enforcing a hard constraint on the KL divergence based on Abdolmaleki et al. (2018). Consider the policy mirror descent with hard constraints,

$$\begin{aligned}&\max_{\pi} \mathbb{E}_{\mathbf{a}\sim\pi}[A^{\pi_{\text{old}}}(s,\mathbf{a})]\\ &\text{s.t. } d_{\text{KL}}(\pi,\pi_{\text{old}}) \le \epsilon\end{aligned} \tag{7}$$

To solve it, we construct the Lagrangian

$$L(\pi,\lambda,\eta) = \mathbb{E}_{a\sim\pi}[A^{\pi_{\text{old}}}(s,\mathbf{a})] + \lambda(\epsilon - d_{\text{KL}}(\pi,\pi_{\text{old}})) + \eta(1 - \sum_a \pi(\mathbf{a}|s))$$

Gradient to the primal objective,

$$\partial\pi L = A^{\pi_{\text{old}}}(s,\mathbf{a}) - \lambda(\log\frac{\pi(\mathbf{a}|s)}{\pi_{\text{old}}(a|s)} + 1) + \eta$$

Let it equal 0 we get the primal optimal solution is

$$\pi = \pi_{\text{old}}(a|s)\exp(\frac{A^{\pi_{\text{old}}}(s,\mathbf{a})}{\lambda})\exp(\frac{\eta-\lambda}{\lambda})$$

As we have the normalization constraint, we have

$$\exp(-\frac{\eta-\lambda}{\lambda}) = \sum_{\mathbf{a}} \pi_{\text{old}}(a|s)\exp(\frac{A^{\pi_{\text{old}}}(s,\mathbf{a})}{\lambda}) := Z$$

Therefore, we have

$$\eta = \lambda(1 - \log Z)$$

Substituting this back to the Lagrangian, we have

$$\begin{aligned}
g(\lambda) =& \lambda\epsilon + \eta + \sum_{\mathbf{a}} \pi(\mathbf{a}|s)\left(A^{\pi_{\text{old}}}(s,\mathbf{a}) - \lambda \log\left(\exp(A^{\pi_{\text{old}}}(s,\mathbf{a})/\lambda)\right) + \lambda \log Z - \eta\right) \\
=& \lambda\epsilon + \sum_{\mathbf{a}} \pi(\mathbf{a}|s)(\lambda \log Z) \\
=& \lambda\epsilon + \lambda \log Z \\
=& \lambda\epsilon + \lambda \log \sum_{\mathbf{a}} \pi_{\text{old}}(a|s) \exp(\frac{A^{\pi_{\text{old}}}(s,\mathbf{a})}{\lambda}) \\
\approx& \lambda\epsilon + \lambda \log \frac{1}{N} \sum_{i=1}^{N} \exp(\frac{A^{\pi_{\text{old}}}(s,\mathbf{a}_i)}{\lambda}) \qquad a_i \sim \pi_{\text{old}}(\mathbf{a}_i|s) \\
=& \lambda\epsilon + \lambda\, \text{logsumexp}(\frac{A^{\pi_{\text{old}}}(s,\mathbf{a}_i)}{\lambda}) - \lambda \log N
\end{aligned}$$

Therefore, we can update the temperature parameter by $\min g(\lambda)$, which can be used in discrete and continuous diffusion.

C.3 Sampling Techniques.

- **Top-p sampling.** For each action that is sampled to be unmasked, we select the smallest set of actions whose cumulative probability computed from f_θ exceeds $P = 0.98$. Then we re-normalize the distribution including only these actions and sample from this re-normalized distribution.
- **Re-masking.** Using the same techniques in (Wang et al., 2025), we don't need to change the ELBO Eq. (5) as well as the FKL policy loss Eq. (FKL Loss). We only need to change

D Experiments

D.1 DNA Generation Setup

Dataset. The experiment is based on a large, publicly available dataset of enhancers, which contains activity measurements for approximately 700,000 DNA sequences, each 200 base pairs long, within human cell lines. The dataset contains the expressive level, which is also used to train our reward models. A masked discrete diffusion model was pretrained on the complete set of sequences.

Reward models. Following established conventions in (Wang et al., 2024a), the dataset was then divided by chromosome to train two distinct reward models, or "oracles". These oracles, built on the Enformer (Avsec et al., 2021) architecture, were designed to predict the enhancer activity level in the HepG2 cell line; one oracle was used to fine-tune the models, while the other was reserved for evaluation.

Evaluation Metrics. To conduct a thorough assessment of each model's ability to generate effective enhancers, the following metrics were employed:

1. Predicted Activity (Reward): This metric measures the enhancer activity level in the HepG2 cell line as predicted by the evaluation reward oracle. It's important to note that the models were fine-tuned using a separate oracle trained on a different chromosomal subset of the data.
2. Approximated Log-Likelihood (App-Log-Lik): The log-likelihood of the generated sequences was calculated with respect to the pretrained model. This measures how "natural" the sequences are; a low likelihood would indicate that the model over-optimized the reward and generated out-of-distribution sequences.

D.2 Implementation of the Macro baselines

We conducted the following algorithm that converts baseline algorithms to the setup with macro actions.

- **DQN-Macro:** For DQN, we directly make the $Q-$network output to be in the shape of $|\mathcal{A}|^K$, which is the size of the total combinatorial action space.
- **IMPALA-Macro:** Instead of output $|\mathcal{A}|$ logits for a single action, the actor networks predict $K \times |\mathcal{A}|$, where each $|\mathcal{A}|$-dim vector is the logits for one action in the macro actions.

D.2.1 Hyperparameter Search for the DQN-Macro

We conduct hyperparameter search for the baseline **DQN-Macro** algorithm to see that whether the our macro baselines prefer different hyper parameters from the original algorithm. we conduct search on the following 3 parameters:

- MLP layers. The original paper include a 1-layer MLP head over the deep residual feature extractors. We increase the number of layers to **2 and 3** to see whether the increase number of parameters benefit macro actions.
- Prioritized experience replay exponents. Current one used for DQN is 0.6 and we try 0.4.
- The value of ϵ in the ϵ-greedy exploration. Current one used for DQN is 0.01 and we try 0.001 and 0.1.

The results are shown in Fig. 7, which shows that the default DQN parameters, also the one we used in our main results, still perform the best for the DQN with macro actions.

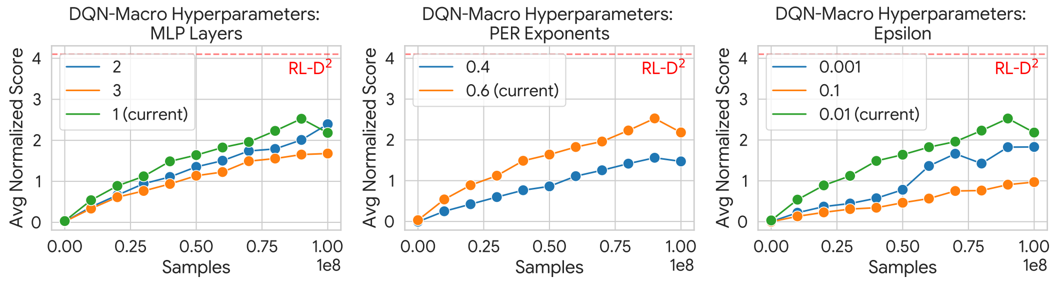

Figure 7: DQN-Macro hyperparameter search. The results shows averaged human normalized score over 10 Atari environments, each with 3 random seeds. Red dash line is the proposed RL-D^2 for reference.

D.3 Scalability and Horizon-Complexity Trade-off Setup

Compared to the hyper parameters in Table 3, we scale the computation by the following conditions:

- **Number of environment steps:** 5×10^8 maximum for macro action length 8 and 16.
- **Model sizes:** We change the heads of the transformers, 2 heads for macro aciton 4, 4 heads for macro aciton 8, 6 heads for macro aciton length 16.
- **Batch size.** We change the mini-batch size to 128, resulting in a total effective batch size of 4096 for macro action length 8 and 16.

D.4 RL-D^2 Implementation of Google Research Football

For the Google Research Football we used the same features using in (Song et al., 2023) and the same scenarios settings for training and evaluation. We implemented the RKL version of RL-D^2 with a single-step ratio in a PPO (Schulman et al., 2017) framework. For the state embeddings we used an additional transformer similar to the one used for the diffusion process. The transformer outputs an state embedding for each player, which are fed as a condition for the action-transformer while also fed to a value-head MLP network that outputs a value for each player. The value is trained as mentioned in (Wen et al., 2022a). We trained the model and baselines over 1G enviornment steps for *11 vs 11* and 500M for the rest of the scenarios.

Table 4: Google Research Football Hyperparameters

Parameters	Value	Parameters	Value
Critic LR	5e-4	Sample-to-insert Ratio	1.0
Actor LR	5e-4	Batch size	256
Discount factor	0.995	Number of mini-batch	1
Number of actors	256	Max grad norm	0.5
Entropy coeff	1e-2	Discount factor	0.995
Training epochs	10	Rollout size	1024
Diffusion steps	Num of players	Ratio clip	0.2

Tasks	DQN	DQN-Macro	IMPALA	IMPALA-Macro	PPO	R2D2	RL-D^2
Alien	11.49k ± 105.46	2.63k ± 43.32	9.04k ± 201.70	12.40k ± 183.48	17.84k ± 480.29	8.09k ± 89.89	22.53k ± 848.31
Amidar	3.43k ± 37.04	691.56 ± 29.63	3.06k ± 12.48	5.60k ± 39.90	1.44k ± 1.89	1.64k ± 35.44	6.44k ± 71.30
Assault	5.92k ± 137.76	6.46k ± 433.42	18.34k ± 884.87	18.29k ± 467.93	5.23k ± 121.56	3.50k ± 96.09	20.99k ± 286.49
Asterix	4.56k ± 73.22	24.42k ± 611.26	361.80k ± 6.94k	28.07k ± 563.07	37.57k ± 1.92k	4.67k ± 64.12	133.56k ± 7.46k
Asteroids	1.51k ± 23.60	2.11k ± 15.32	5.71k ± 112.65	8.95k ± 145.30	14.49k ± 266.01	2.01k ± 32.60	82.24k ± 5.90k
Atlantis	984.11k ± 13.14k	802.66k ± 50.57k	1030.98k ± 6.81k	864.89k ± 1.24k	710.85k ± 18.11k	1082.26k ± 40.63k	1003.63k ± 1.79k
BankHeist	1.84k ± 18.02	1.25k ± 19.15	1.50k ± 2.70	1.10k ± 4.31	485.20 ± 4.21	944.20 ± 3.53	1.70k ± 6.59
BattleZone	116.15k ± 1.48k	43.66k ± 837.10	68.43k ± 1.12k	166.30k ± 2.96k	54.66k ± 472.35	76.09k ± 1.69k	197.94k ± 3.08k
BeamRider	3.55k ± 99.35	5.62k ± 815.84	21.07k ± 837.41	14.82k ± 143.88	29.47k ± 774.52	3.10k ± 78.03	28.98k ± 1.10k
Berzerk	340.95k ± 24.42k	1.11k ± 19.22	1.49k ± 45.20	7.43k ± 269.77	1.29k ± 33.72	110.48k ± 42.94	802.99k ± 64.87k
Bowling	266.38 ± 0.01	41.94 ± 6.70	70.00 ± 0.00	54.80 ± 0.13	149.03 ± 0.25	197.18 ± 0.17	266.38 ± 0.00
Boxing	97.13 ± 0.20	99.31 ± 0.10	100.00 ± 0.00	98.60 ± 0.02	98.99 ± 0.11	97.89 ± 0.12	99.51 ± 0.00
Breakout	76.39 ± 2.94	401.10 ± 1.61	675.48 ± 18.21	161.19 ± 5.48	394.17 ± 0.69	124.08 ± 1.44	424.15 ± 0.00
Centipede	36.42k ± 541.25	6.26k ± 214.74	8.08k ± 381.00	27.66k ± 835.60	27.60k ± 380.41	20.66k ± 247.27	68.41k ± 910.77
ChopperCommand	13.18k ± 320.75	3.59k ± 300.73	23.86k ± 630.67	15.74k ± 709.66	2.35k ± 115.21	2.52k ± 148.92	34.36k ± 609.01
CrazyClimber	93.75k ± 1.37k	136.37k ± 3.07k	136.05k ± 949.62	107.05k ± 1.55k	70.01k ± 1.19k	118.39k ± 1.04k	113.69k ± 759.99
Defender	17.60k ± 256.97	51.94k ± 496.01	427.49k ± 23.25k	33.85k ± 618.50	55.75k ± 606.50	38.70k ± 961.61	138.16k ± 7.10k
DemonAttack	3.89k ± 61.93	26.78k ± 5.85k	132.40k ± 126.41	44.64k ± 1.58k	9.91k ± 313.53	2.89k ± 35.35	55.93k ± 1.34k
DoubleDunk	-0.36 ± 0.06	-3.18 ± 2.10	23.46 ± 0.07	0.00 ± 0.00	5.60 ± 0.36	-0.86 ± 0.34	24.00 ± 0.00
Enduro	651.59 ± 8.84	2.22k ± 58.92	8.16 ± 0.49	1.15k ± 19.29	1.64k ± 48.10	852.35 ± 29.13	1.45k ± 50.17
FishingDerby	68.54 ± 0.86	26.46 ± 0.52	44.99 ± 0.66	45.31 ± 0.68	0.20 ± 0.67	20.74 ± 0.99	80.64 ± 0.00
Freeway	33.82 ± 0.00	32.61 ± 0.07	32.73 ± 0.03	33.46 ± 0.03	32.68 ± 0.06	34.00 ± 0.00	33.84 ± 0.00
Frostbite	9.41k ± 14.63	3.93k ± 194.22	862.00 ± 48.06	9.00k ± 2.64	3.97k ± 429.59	10.47k ± 78.34	14.39k ± 143.25
Gopher	2.72k ± 120.47	27.47k ± 3.09k	89.58k ± 3.91k	27.69k ± 626.48	5.38k ± 132.20	3.95k ± 274.47	67.92k ± 1.32k
Gravitar	2.68k ± 11.18	1.10k ± 19.89	4.27k ± 2.64	4.87k ± 28.31	4.57k ± 5.56	3.08k ± 28.87	4.62k ± 13.83
Hero	22.90k ± 17.41	10.69k ± 452.97	28.98k ± 5.71	36.74k ± 7.49	14.08k ± 10.82	32.41k ± 685.43	28.94k ± 14.34
IceHockey	12.45 ± 0.70	4.62 ± 0.67	26.42 ± 0.15	25.69 ± 0.20	15.98 ± 0.45	-0.15 ± 0.16	46.60 ± 0.14
Jamesbond	1.41k ± 50.94	584.00 ± 9.20	1.74k ± 66.32	75.06k ± 80.92	1.60k ± 155.90	1.08k ± 18.32	18.21k ± 3.59k
Kangaroo	14.67k ± 113.45	8.96k ± 273.42	14.50k ± 0.00	14.22k ± 8.61	2.00k ± 0.00	12.80k ± 127.16	15.26k ± 10.64
Krull	69.80k ± 1.86k	9.36k ± 132.50	10.02k ± 38.49	83.59k ± 2.56k	8.97k ± 83.18	61.42k ± 1.50k	385.51k ± 3.70k
KungFuMaster	26.71k ± 236.87	36.65k ± 718.07	55.69k ± 1.09k	15.98k ± 206.24	31.15k ± 465.11	54.57k ± 1.02k	63.76k ± 789.60
MontezumaRevenge	835.28 ± 36.72	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	400.00 ± 0.00	400.00 ± 0.00	2.50k ± 7.26
MsPacman	20.18k ± 162.54	3.90k ± 130.60	8.85k ± 77.01	7.77k ± 63.70	21.04k ± 250.98	9.82k ± 68.46	22.87k ± 3.39
NameThisGame	6.86k ± 68.39	14.91k ± 442.47	15.64k ± 90.75	13.69k ± 109.46	8.40k ± 68.86	7.15k ± 81.37	12.66k ± 197.84
Phoenix	4.98k ± 66.88	14.11k ± 1.41k	192.34k ± 11.13k	6.37k ± 38.35	43.91k ± 3.04k	5.61k ± 46.40	202.39k ± 5.33k
Pitfall	0.00 ± 0.00	-2.48 ± 0.94	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Pong	13.31 ± 0.12	19.70 ± 0.13	20.47 ± 0.09	12.25 ± 0.22	21.00 ± 0.00	19.73 ± 0.11	21.00 ± 0.00
PrivateEye	35.22k ± 4.16	100.00 ± 0.00	389.34 ± 268.73	99.64 ± 0.02	1.59k ± 0.57	35.12k ± 28.48	15.18k ± 4.02
Qbert	29.68k ± 62.59	9.00k ± 431.38	15.66k ± 386.65	475.04 ± 0.03	5.49k ± 263.05	16.68k ± 213.54	30.85k ± 57.30
Riverraid	7.48k ± 81.04	17.07k ± 492.67	18.44k ± 100.97	9.87k ± 39.11	8.01k ± 56.74	11.33k ± 90.61	17.94k ± 98.91
RoadRunner	317.32k ± 9.62k	52.61k ± 630.59	59.20k ± 368.83	514.91k ± 7.59k	23.88k ± 212.10	77.74k ± 1.95k	563.86k ± 2.53k
Robotank	33.33 ± 0.72	66.72 ± 0.49	71.97 ± 0.33	66.67 ± 0.17	63.62 ± 0.51	31.22 ± 0.20	66.74 ± 0.79
Seaquest	3.97k ± 37.56	24.67k ± 5.63k	27.12k ± 824.83	10.85k ± 134.19	6.87k ± 72.78	3.63k ± 194.09	144.62k ± 4.65k
Skiing	-4.43k ± 5.26	-29.41k ± 266.48	-9.01k ± 0.07	-8.95k ± 0.00	-15.00k ± 111.88	-27.93k ± 185.34	-4.41k ± 7.47
Solaris	15.00k ± 517.51	3.26k ± 181.94	2.22k ± 113.16	2.41k ± 47.98	6.00k ± 146.73	3.02k ± 160.32	14.39k ± 263.47
SpaceInvaders	2.02k ± 30.80	6.22k ± 714.13	41.54k ± 8.75k	9.92k ± 329.91	3.87k ± 56.20	1.75k ± 59.99	10.86k ± 480.94
StarGunner	1.40k ± 22.05	96.97k ± 7.53k	142.15k ± 838.43	29.10k ± 850.92	33.55k ± 427.98	2.23k ± 42.72	52.69k ± 448.91
Tennis	0.00 ± 0.00	20.77 ± 0.54	0.00 ± 0.00	0.00 ± 0.00	-2.34 ± 0.58	-1.69 ± 0.51	22.20 ± 0.17
TimePilot	34.33k ± 404.16	10.66k ± 159.67	55.38k ± 918.01	109.15k ± 4.05k	33.82k ± 544.74	11.37k ± 222.24	42.26k ± 722.78
Tutankham	160.03 ± 1.36	213.78 ± 8.55	240.49 ± 9.66	187.71 ± 0.03	190.20 ± 1.81	120.27 ± 3.63	230.73 ± 0.84
UpNDown	76.44k ± 910.55	65.52k ± 9.48k	422.67k ± 2.82k	322.33k ± 3.00k	240.21k ± 9.79k	127.06k ± 2.61k	264.48k ± 2.48k
Venture	2.05k ± 12.84	1.50k ± 35.34	20.00 ± 0.00	1.99k ± 2.95	1.41k ± 24.61	1.50k ± 24.88	2.06k ± 3.76
VideoPinball	155.83k ± 3.47k	341.15k ± 59.03k	542.75k ± 15.87k	392.46k ± 17.02k	63.37k ± 4.76k	117.67k ± 3.64k	545.38k ± 76.91k
WizardOfWor	31.94k ± 1.13k	16.52k ± 1.16k	19.41k ± 371.79	39.40k ± 406.34	9.68k ± 1.12k	10.94k ± 529.67	58.41k ± 588.34
YarsRevenge	82.41k ± 1.41k	70.73k ± 1.80k	125.27k ± 1.07k	139.64k ± 1.55k	70.65k ± 600.98	75.95k ± 785.33	165.95k ± 245.07
Zaxxon	18.84k ± 311.75	12.58k ± 727.69	34.99k ± 233.45	45.80k ± 276.10	30.07k ± 262.07	13.28k ± 505.46	30.60k ± 310.47

Table 5: Full Performance of Atari Games.

E Additional Experimental Results

E.1 Full Results for Atari Games

Please refer to Table 5 for the full results of Atari games, and Fig. 8 for the comparison with the best baselines. We outperm all the baselines in 36 out of 56 Atari environments.

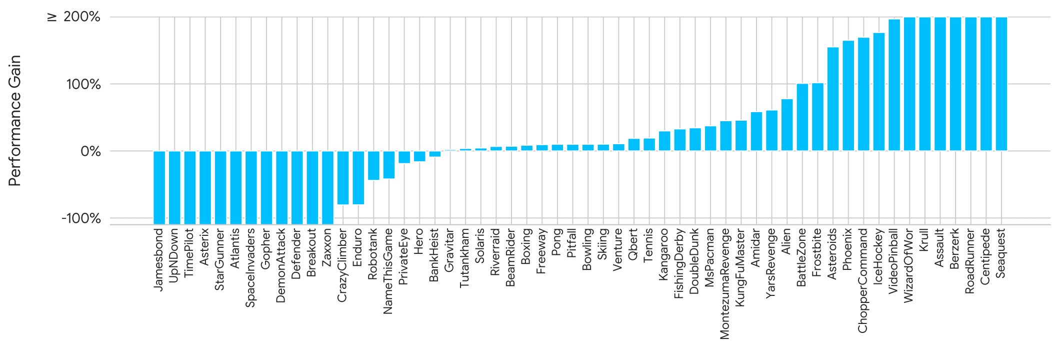

Figure 8: Mean human normalized score of RL-D^2 compared to the best baselines in each Atari task.

E.2 Ablation of temperature tuning

We conduct ablation studies on the temperature tuning discussed in Appendix C.2 on multiple benchmarks including MinAtar, Atari and Google Football.

E.2.1 MinAtar

For MinAtar, the current temperature is updated following a KL constraint is set to 1.0. We compare auto-tuning with fixed temperatures. The results are shown in Fig. 9, showing the auto-tuning consistently outperforms fixed temperature.

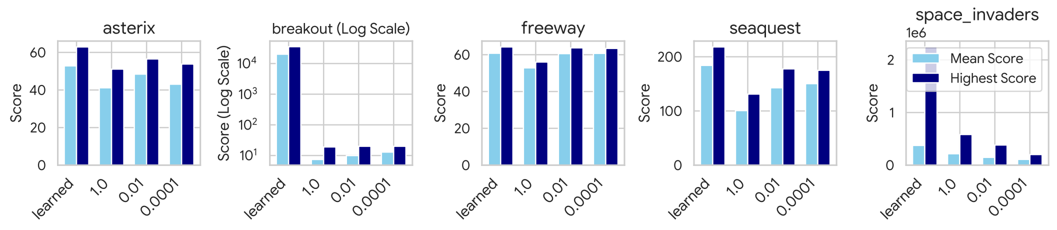

Figure 9: Ablation studies of temperature tuning with fixed temperature variable. Bars indicates the mean episode returns over last 100 evaluations over 3 seeds.

E.2.2 Atari Games

For Atari games, we use KL constraint schedule linearly decaying from 1 to 0.1 in the first 10^7 samples. The intuition behind this selection is that, in vast combinatorial discrete spaces, a larger initial KL constraint allows the policy to deviate significantly from initialization, enabling the broad exploration necessary to discover high-value actions.

We compare auto-tuning with other two temperature control scheme: Fixed KL constraints 1.0 and 0.1; The results are shown in Fig. 10, showing that smaller KL constraints like 0.1 fails to learn in the initial phase. Large KL constraints like 1.0 will cause instabilities after 50M steps, which makes the performance worse. The linear decay achieves initial fast learning and overall stability.

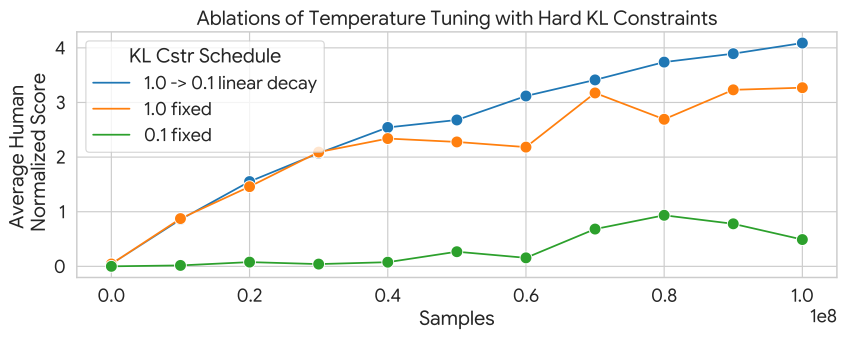

Figure 10: Ablation studies of temperature tuning with hard KL constraints, compared with different KL constraint schedules. The results shows averaged human normalized score over 10 Atari environments, each with 3 random seeds.

E.2.3 Google Football

We compare three different KL constraint schedule in Google Football, 0.1, 1.0, and 10. the results are shown in Fig. 11, which shows another failure mode of collapsing for larger KL constraints. The reason might be that larger KL constraints push the policy to be highly deterministic. Therefore, the collected data lost diversity, making the training failed.

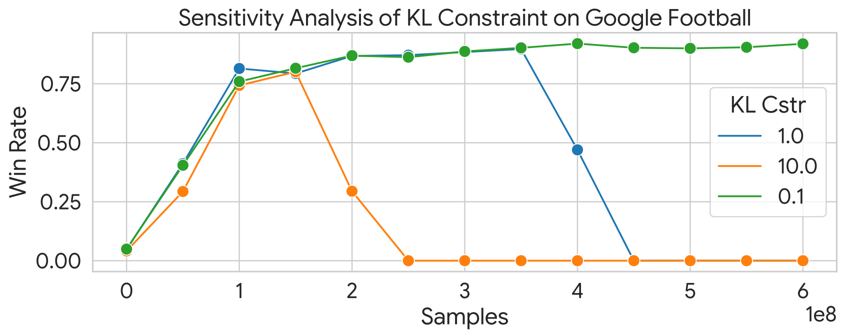

Figure 11: Sensitivity studies of temperature tuning with hard KL constraints on Google Football 11 vs 11. The results shows averaged win rate over 5 random seeds.

In summary, the temperature tuning is significant to the performance of RL-D^2-FKL. When selecting the KL constraints, we should consider two key factors: (1) The initial KL constraint should be large so that the policy can start learning; (2) Avoid collapsing in the later training phase by enforcing not too large KL constraints. Although not necessary, a decay schedule is very helpful to stabilize the training and get better performance.

E.3 Discrete Diffusion as Planner for Causal Action Spaces

In applications of macro actions in Atari games, we can just commit to the first action rather than all the macro actions. Therefore, it is common to plan for a longer trajectory and only commits to the first action, such as model predictive control and Monte-Carlo tree search (Garcia et al., 1989; Silver et al., 2016). However, if we would like to implement planning in online RL, the parameterization of the planning trajectory is not trivial. If we use autoregressive models as our the parameterization to generate the trajectory, the next action will depend only on the current state and not depend on the planned trajectory, leaving future actions useless.

Benefiting from the non-casual unmasking of discrete diffusion models, we can directly use the discrete diffusion to parameterize the planner. **Note that this is a total different setup and algorithm from the main text Sec. 5.2**. We show the planner performance with the same set of hyperparameters in Table 3 with varying planning steps shown in Fig. 12. The performance increase with increasing

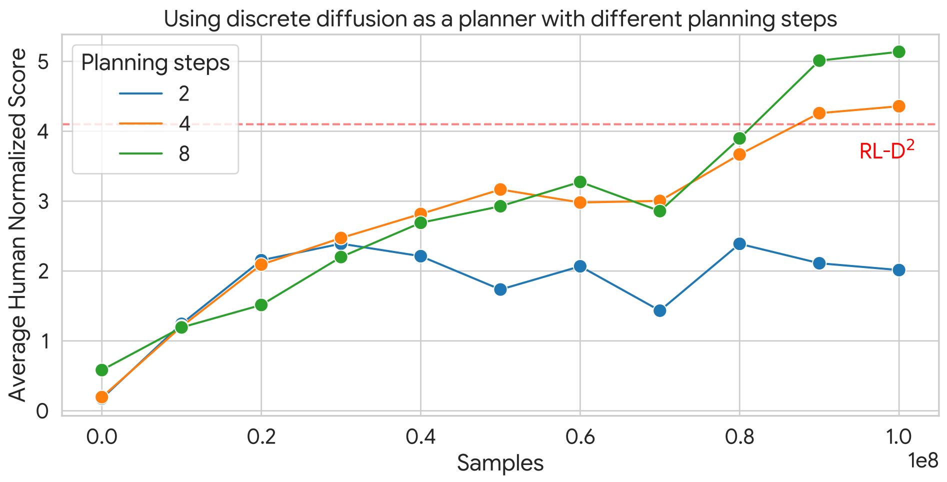

Figure 12: Performance with using discrete diffusion as a planner with different length of planning steps, averaged over 10 Atari games with 3 random seed each game.

planning length, showcasing that only committing to the first action increase the robustness of committing all macro actions. The planner is also scalable with respect to the increasing size of action spaces.

Algorithm 1 Discrete Diffusion as Planners

```
Require: Planning length $K$, current policy $\pi_{\text{old}}$, replay buffer $\mathcal{D}$, current value function $q^{\pi_{\text{old}}}$.
1: # Policy updates in training.
2: For states $s \sim \mathcal{D}$, sample macro actions $\mathbf{a} = (a_1, \ldots, a_k)$, compute $\mathcal{L}_{\text{ELBO}}$ with Eq. (5).
3: Take the first one to compute value function $q^{\pi_{\text{old}}}(s, a_1)$.
4: Optimize the policy by self-imitation loss Eq. (FKL Loss).
5: # Inference.
6: Sample macro actions $(a_1, \ldots, a_k)$ and only take $a = a_0$.
```

E.4 Ablation of On-Policy Diffusion Training.

We conduct ablation studies on the on-policy training discussed in Sec. 4.2 and the results are shown in Fig. 13, which shown the on-policy diffusion training help improve the performance.

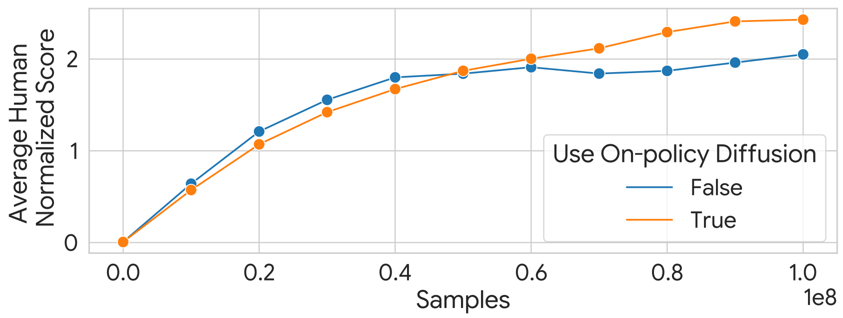

Figure 13: Ablation studies of on-policy diffusion training. The curves indicates mean reward using macro length 8 over 10 representative Atari environments, 3 seed, and 10 consecutive evaluation episodes.

E.5 Ablation of Network Architecture.

We compare using multi-layer perceptrons (MLP) versus transformers (Vaswani et al., 2017) for parameterization of the Q and policy networks. The two networks share the same amount of parameters around 4×10^5. We can see the transformer consistently outperform MLP.

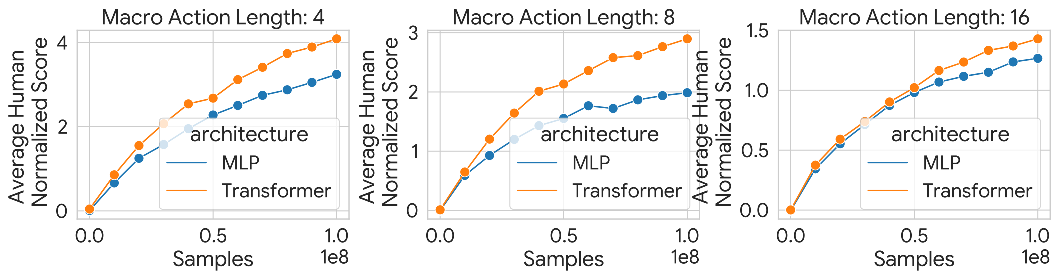

Figure 14: Ablation studies of on-policy diffusion training. The curves indicates mean reward using macro length 8 over 10 representative Atari environments, 3 seed, and 10 consecutive evaluation episodes.

E.6 Importance Sampling Ratio Estimation

We compare two methods to handle the unknown log probabilities for RL-D^2-RKL. (1) augmented MDP in Sec. 4.1 and (2) ELBO-based estimation in Appendix B.3. The results are shown in Table 6. The ELBO estimator does not show good performance due to estimating the importance sampling ratio using the ratio of two lower bounds. Therefore, all the results in the main text are using the augmented MDP approach and we defer the ELBO-based estimation to the Appendix B.3 just for reference.

	pass & shoot	run pass & shot	3 vs 1	ct-easy	ct-hard	corner	5 vs 5	11 vs 11
Augmented MDP	**100 $\pm$ 0.0**	**98.5 $\pm$ 1.6**	**98.6 $\pm$ 1.0**	**99.1 $\pm$ 0.6**	**97.7 $\pm$ 2.2**	**96.3 $\pm$ 3.4**	**99.6 $\pm$ 0.4**	**99.8 $\pm$ 0.2**
ELBO	90.0 $\pm$ 10.0	62.6 $\pm$ 20.9	72.6 $\pm$ 14.0	7.9 $\pm$ 3.5	6.3 $\pm$ 0.6	13.4 $\pm$ 6.2	46.0 $\pm$ 13.4	0.0$\pm$ 0.0

Table 6: Comparing two importance sampling ratio estimation for RL-D^2-RKL on Google Football. The augmented MDP approach performs much better than ELBO-based importance sampling ratio estimation.

E.7 Comparing FKL v.s. RKL with Same Computation on Atari Games

The actual bottleneck of Atari games is computations due to using a deep residual network to handle image inputs. Therefore, to compare the performance and stability of FKL and RKL in Atari games, we align the training wall time while adapting the batch size. The results are showin in Fig. 15. We can see that in general FKL has a better performance than RKL, further verifying the fact that FKL learn faster than RKL we observed in Table 2. Moreover, FKL is less sensitive batch sizes, showing only marginal increase when increasing batch sizes. RKL show large performance improvement when increasing batch size from 256 to 512[3], showing RKL favors larger batches because the high-variance nature of augmented MDP method.

F Baseline Selection Protocol

Our baseline selection protocol was designed to rigorously evaluate the performance, scalability, and versatility of $RL - D^2$ across the distinct challenges presented by combinatorial action spaces. We selected a diverse set of baselines, ranging from established standards in reinforcement learning to

[3]One sample in the batch is a 32-step trajectory rather than a transition pair, which is a common practice in training pipelines used for Atari games. Therefore, the actual number of transition pairs is 32 times more.

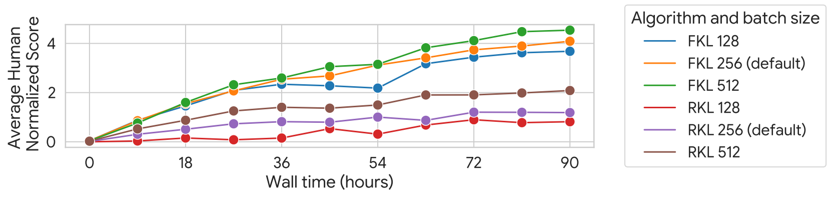

Figure 15: Comparing FKL and RKL, on-policy, in Atari games with different learner batch sizes. The curves shows averaged human normalized score over 10 atari games with 3 random seeds each game.

domain-specific state-of-the-art methods, ensuring that our comparisons were fair, comprehensive, and directly addressed the central claims of our work.

The selection was tailored to the three specific experimental domains:

F.1 DNA Sequence Generation

This domain tests the single-step policy optimization (combinatorial bandit) capabilities of our framework. The baselines were chosen to cover two main categories:

- **Controlled Generation Methods:** We included standard guidance-based techniques (CG, SMC, TDS, and CFG). These methods are not based on RL policy optimization but are common for guiding generative models toward desired properties. This comparison validates RL-D^2 against non-RL finetuning approaches.
- **State-of-the-Art RL Finetuning:** We included **DRAKES**, a strong, recent baseline that also uses RL to optimize a generative model for sequence generation. DRAKES employs the Gumbel-Softmax trick to enable reward backpropagation through the generative process. This provides a direct comparison against another RL-based approach for finetuning discrete generative models.

F.1.1 Macro Actions

This domain tests the ability of RL-D^2 to handle online RL in complex, long-horizon tasks by modeling sequences of primitive actions.

- **Standard RL Baselines:** We first included strong, general-purpose RL algorithms (DQN, IMPALA, PPO, R2D2) that operate on a single-action level. These baselines establish a performance reference and demonstrate the inherent difficulty of the tasks without temporal abstraction.
- **Adapted Macro-Action Baselines:** To create a direct comparison, we adapted standard algorithms to handle macro-actions, as detailed in **Appendix D.2**:
  - **DQN-Macro:** This baseline represents a "naive" approach, where the Q-network's output layer is expanded to $|\mathcal{A}|^K$. This directly exposes the challenge of exponential scaling that RL-D^2 is designed to overcome.
  - **IMPALA-Macro:** This baseline represents a "factored" approach, where the policy network outputs $K \times |\mathcal{A}|$ logits, assuming conditional independence between actions in the macro-action sequence.

  These adaptations allow us to test our hypothesis that an expressive, non-causal generative model (our diffusion policy) can outperform both naive exponential-space methods and simple independent factorization methods.

F.2 Cooperative Multi-Agent Reinforcement Learning

This domain tests RL-D^2 on modeling the combinatorial *joint action* of multiple agents.

- **Autoregressive (AR) Policy:** We selected a strong **autoregressive transformer** policy as our primary baseline. This is a dominant and highly effective paradigm for modeling joint actions, where the action for each agent is sampled sequentially, conditioned on the actions of previous agents.
- This comparison is central to our motivation. The introduction (Section 1) explicitly notes that AR models impose an artificial causal ordering. By comparing RL-D^2 (a non-causal generative model) against a strong AR baseline, we directly test our claim that diffusion's flexible, non-causal generation process is a superior parameterization for modeling complex inter-agent dependencies in MARL.