DISCRETE FLOW MATCHING FOR OFFLINE-TO-ONLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many reinforcement learning (RL) tasks have discrete action spaces, but most generative policy methods based on diffusion and flow matching are designed for continuous control. Meanwhile, generative policies usually rely heavily on offline datasets and offline-to online RL is itself challenging, as the policy must improve from new interaction without losing useful behavior learned from static data. To address those challenges, we introduce DRIFT, an online fine-tuning method that updates an offline pretrained continuous time Markov chain (CTMC) policy with an advantage-weighted discrete flow matching loss. To preserve useful pretrained knowledge, we add a path-space penalty that regularizes the full CTMC trajectory distribution, rather than only the final action distribution. For large discrete action spaces, we introduce a candidate-set approximation that updates the actor over a small subset of actions sampled from reference-policy rollouts and uniform exploration. Our theoretical analysis shows that focusing on a small set of high probability actions can reduce computation while keeping the approximation error small, making our method practical for large action spaces. We evaluate on macro-action MinAtar games where offline-pretrained DRIFT outperforms its online-only counterpart across all five games and achieves the highest mean return on three and on oracle free Jericho, DRIFT achieves a mean normalized score of 7.3% with a lightweight GRU encoder, outperforming the evaluated pretrained language-model baselines. Ablations demonstrate the importance of appropriately weighted path-space regularization and reference-guided candidate construction.

1 INTRODUCTION

Offline-to-online reinforcement learning (RL) asks a simple but important question: how can a policy learned from static data keep improving through real interaction? Offline RL trains agents from logged data without costly or risky exploration (Levine et al., 2020; Kumar et al., 2020; Kostrikov et al., 2022), but the learned policy is limited by the coverage and quality of that dataset. If good actions or important states are missing from the data, offline training alone cannot discover them. Offline-to-online fine-tuning addresses this limitation by using the offline policy as a strong initialization and then improving it with online experience (Nair et al., 2020; Lee et al., 2022; Nakamoto et al., 2023; Zhang et al., 2023). The challenge is to improve the policy without forgetting the useful behavior structure learned during offline pre-training.

Diffusion (Ho et al., 2020; Song et al., 2021) and flow-matching (Lipman et al., 2023; Liu, 2022) policies capture multimodal action distributions in offline data (Wang et al., 2023; Chen et al., 2023; Kang et al., 2023; Zhang et al., 2025a; Park et al., 2025), and can guide exploration during offline-to-online fine-tuning. However, these methods rely on continuous dynamics that do not naturally apply to discrete spaces, leaving out domains such as recommendation, combinatorial control, scheduling, and token-level decision making. Discrete flow matching (Gat et al., 2024; Campbell et al., 2024) replaces continuous flows with continuous-time Markov chain (CTMC) generators making generative policies possible in discrete action spaces (Khan et al., 2026). However, adapting pretrained CTMC policies to online RL while preserving their learned behavior and scaling to large action spaces remains challenging. During the online stage, the generator must move toward newly discovered high-reward actions, preserve useful pretrained behavior, and scale when the action space is large. Existing offline discrete flow methods do not address online adaptation, while offline-to-online methods are mostly designed for continuous spaces.

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

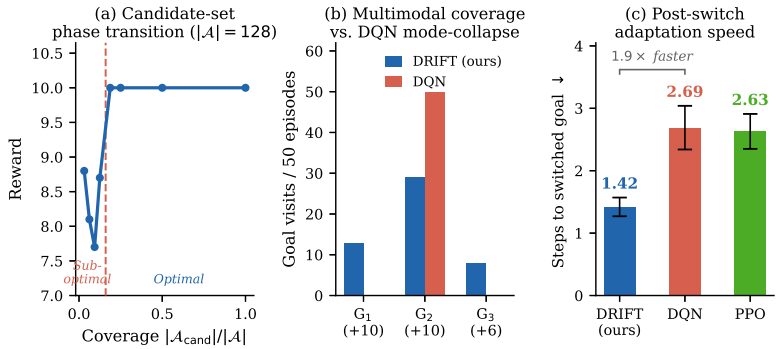


Figure 1: Three empirical motivations for our method. **(a)** The candidate-set approximation shows sharp performance phase transition at 16% action coverage. **(b)** Our CTMC policy maintains multimodal coverage, while DQN mode-collapses to single goal. **(c)** In a goal-switch environment, our stochastic policy adapts to new reward signal 1.9× faster than DQN and PPO.

Our method. Online fine-tuning of discrete generative policies faces two major challenges. The policy must discover better actions without forgetting useful behavior learned offline, while evaluating all possible actions becomes prohibitively expensive in large discrete action spaces. We introduce DRIFT to address both challenges. We combine advantage-weighted discrete flow matching with path-space regularization to improve the policy while limiting changes to its pretrained sampling dynamics. To scale to large action spaces, we update the policy using a small candidate set constructed from reference-policy rollouts and uniform exploration. However, excluding actions changes both the target policy and the CTMC transition rates, making it difficult to guarantee that the approximation remains accurate. Our theoretical analysis addresses this problem by showing that the error in the target generator is bounded by the probability mass excluded from the candidate set (Theorem 4.2). This provides a theoretical basis for reducing computation while controlling approximation error.

Our motivating experiments (Figure 1) show our generative policy maintains multimodal goal coverage where DQN (Mnih et al., 2015) mode-collapses, our stochastic generator adapts 1.9× faster than DQN and PPO (Schulman et al., 2017) after a reward shift, and the candidate-set approximation reaches optimal performance at just 16% action coverage in a 128-action gridworld, matching our coverage bound in Proposition 4.1. Together, these observations identify a common regime where standard discrete-action RL struggles: structured action spaces with multimodal actions, shifting rewards, or large action spaces where exhaustive evaluation is infeasible. This work is a first step in that direction which we start by evaluating how our method performs on standard benchmarks, comparing with baselines on Jericho text games, MinAtar, macro-action MinAtar and discretised MuJoCo tasks.

2 Related Work

Offline-to-online RL and discrete-action RL. Offline-to-online methods stabilize the transition from static data to live interaction through advantage weighting (Nair et al., 2020), balanced replay with pessimistic ensembles (Lee et al., 2022), calibrated value learning (Nakamoto et al., 2023), policy expansion (Zhang et al., 2023), and adaptive constraints (Li et al., 2026b). These approaches target continuous control or use standard policy classes. In discrete-action RL, DQN (Mnih et al., 2015), Double DQN (Van Hasselt et al., 2016), Rainbow (Hessel et al., 2018), A2C (Mnih et al., 2016), and PPO (Schulman et al., 2017) provide strong online baselines but do not model multimodal action distributions.

Generative policies for RL. Diffusion and flow-based policies capture multimodal behavior in offline data. Diffusion-QL (Wang et al., 2023), IDQL (Hansen-Estruch et al., 2023), efficient diffusion policies (Kang et al., 2023), energy-weighted flow matching (Zhang et al., 2025a), and Flow Q-Learning (Park et al., 2025) all improve offline RL through expressive action modeling with value guidance. Online fine-tuning of such policies has been explored in continuous control (Zhang et al.,

2025b; McAllister et al., 2025), but these methods require continuous action representations and do not extend to discrete decisions.

Discrete generative policies for RL. RL-D^2 (Ma et al., 2026) trains discrete diffusion policies for online RL by matching a policy mirror descent target using forward or reverse KL divergence, demonstrating strong performance on macro-action MinAtar and Atari. QDFM (Khan et al., 2026) uses value-weighted discrete flow matching to learn CTMC policies for offline RL. Inspired by these work, our method addresses offline-to-online adaptation of pretrained CTMC policies. We regularize the full CTMC path during fine-tuning rather than relying only on policy-level KL constraints, while improving online experience using path-space regularization and reference-guided candidate sets to improve pretrained policies in large discrete action spaces.

Discrete generative models and trust-region fine-tuning. Discrete diffusion and flow models replace Euclidean dynamics with Markov processes over finite state spaces for sequences (Gat et al., 2024), molecular design (Campbell et al., 2024), and biological generation (Holderrieth et al., 2025; Wang et al., 2025). Our path-space regularization builds on trust-region policy improvement (Peters et al., 2010b; Schulman et al., 2015; 2017) and KL-regularized fine-tuning from human feedback (Ziegler et al., 2019; Ouyang et al., 2022) requiring careful design considerations which we detail in the next sections.

3 Preliminaries

3.1 Markov Decision Process and Offline Data

We consider a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is a *finite* discrete action space with $K = |\mathcal{A}|$ actions, $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, P is the transition kernel, and $\gamma \in (0,1)$ is the discount factor. The state-action value function is $Q^\pi(s,a) = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \mid s_0 = s, a_0 = a\big]$.

An offline dataset $\mathcal{D}_{\text{off}} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^N$ is collected by an unknown behavior policy. The KL-regularized optimal policy admits the well-known Boltzmann form (Todorov, 2006; Haarnoja et al., 2017; Lu et al., 2023): $\pi^\star(a \mid s) \propto \pi_{\text{ref}}(a \mid s)\, \exp\big(A(s,a)/\beta\big)$, where $A(s,a) = Q(s,a) - V(s)$ is the advantage, π_{ref} is a reference policy, and $\beta > 0$ is a temperature that trades off reward maximization against proximity to π_{ref}.

3.2 Discrete Flow Matching via CTMCs

Discrete Flow Matching (DFM) (Campbell et al., 2024; Gat et al., 2024) extends the flow-matching paradigm from continuous Euclidean spaces to finite discrete spaces by replacing continuous vector fields used in standard flow matching with Continuous-Time Markov Chains (CTMCs).

A CTMC over the action space $\mathcal{A}$ is specified by a time-dependent rate matrix (or generator) $u_t(i \to j \mid s)$, which represents the instantaneous rate of jumping from action i to action j at time t, given state s. It satisfies the following constraints for every source action i:

$$u_t(i \to j \mid s) \geq 0 \;\; \forall\, j \neq i, \qquad \sum\nolimits_{j \in \mathcal{A}} u_t(i \to j \mid s) = 0. \tag{1}$$

The first constraint ensures nonnegative off-diagonal transition rates and the second ensures probability conservation. The marginal probability mass function p_t evolves according to the Kolmogorov forward equation: $\frac{d}{dt}\, p_t(j \mid s) = \sum_{i \in \mathcal{A}} p_t(i \mid s)\, u_t(i \to j \mid s)$.

DFM learns a parametric rate model u_θ by regressing it against a tractable *conditional* target generator derived from a prescribed probability path connecting a source distribution p_0 to a target distribution p_1. At inference time, actions are sampled by simulating the learned CTMC from $t = 0$ to $t = 1$ using Euler discretization, as explained in Appendix B.

3.3 Offline Pre-Training via Advantage-Weighted DFM

Our method requires a pretrained reference generator u_{ref} but does not depend on a particular pretraining procedure. In our experiments, we obtain it through advantage-weighted DFM, which

learns to transport a uniform prior toward a policy based on offline advantage estimates. Other pretrained discrete-action policies can also be distilled into a CTMC generator. Implementation details are provided in Appendix A.1.

4 Our Method: Online fine-tuning of Discrete Flow Matching Policy

Offline-pretrained policies are limited by the coverage and quality of their data (Levine et al., 2020). Online fine-tuning can discover better actions but risks forgetting useful pretrained behavior. We introduce DRIFT, which improves a pretrained CTMC generator using online rewards while preserving its learned behavior through a path-space trust region.

Overview. Starting from a reference generator u_{ref} obtained via advantage-weighted DFM pre-training (Section 3.3), DRIFT performs iterative online learning. Each iteration consists of four steps, summarized in Algorithm 1: (1) collect online transitions by rolling out the current CTMC generator in the environment (Line 3), (2) update critics and a value network from a mixed replay buffer (Lines 5–6), (3) construct a reward-weighted target policy and its corresponding DFM target generator over a candidate action set (Lines 8–11), (4) update the generator by minimizing a combined DFM loss plus path-space KL penalty (Lines 12–15). The reference generator is periodically refreshed to track improving performance (Line 17).

Algorithm 1 DRIFT: Discrete flow matching with Regularised Information-theoretic Fine-Tuning (sketched version, full algorithm in Appendix H)

Require: Pre-trained generator u_{ref}, offline data $\mathcal{D}_{\mathrm{off}}$, temperature β, KL weight α, CTMC sub-steps M, candidate sizes $N_{\mathrm{roll}}, N_{\mathrm{rand}}$, refresh interval K, flow-time truncation $\delta \in (0,1)$

1: Initialize $u_\theta \leftarrow u_{\mathrm{ref}}$, critics Q_{ϕ_1}, Q_{ϕ_2}, value V_ψ, targets $Q_{\phi_k^-}, V_{\psi^-}$, buffer $\mathcal{B}$
2: **for** $n = 1$ **to** N_{steps} **do**
3: Simulate CTMC under u_θ from $X_0 \sim \mathrm{Unif}(\mathcal{A})$ for M sub-steps; execute $a = X_M$; store (s, a, r, s', d) in $\mathcal{B}$
4: Sample mixed minibatch: $(1-\rho)B$ from $\mathcal{B}$ and ρB from $\mathcal{D}_{\mathrm{off}}$
5: Update critics: $Q_{\phi_k} \leftarrow \arg\min \mathcal{L}_{Q_k}$
6: Update value: $V_\psi \leftarrow \arg\min \mathcal{L}_V$ using lagged $\tilde{\pi}^-$ (Eq. 9)
7: **for** each state s_b in actor batch **do**
8: Build $\mathcal{A}_{\mathrm{cand}}(s_b)$ via N_{roll} rollouts of frozen u_{ref} + N_{rand} uniform actions
9: Compute smoothed π_{ref} (Eq. 2) and clipped advantage $\bar{A}$
10: Set target policy $\tilde{\pi}(a \mid s_b) \propto \pi_{\mathrm{ref}}(a \mid s_b) \exp(\bar{A}(s_b, a)/\beta)$ (Eq. 3)
11: Construct bridge p_t (Eq. 4) and target generator u_t^* by independent coupling (Eq. 5)
12: Compute $\ell_{\mathrm{DFM}}(s_b)$ by sampling $t \sim \mathrm{Unif}(0, 1-\delta)$ and $i \sim p_t$ (Eq. 6).
13: Estimate path-space $\widehat{\mathrm{KL}}(s_b)$ against frozen u_{ref} (Eq. 7)
14: **end for**
15: Update u_θ by minimizing $\mathcal{L}_{\mathrm{actor}} = \mathcal{L}_{\mathrm{DFM}} + \alpha \widehat{\mathcal{L}}_{\mathrm{KL}}$ (Eq. 8)
16: Soft-update targets: $\phi_k^- \leftarrow (1-\tau)\phi_k^- + \tau\phi_k$, $\psi^- \leftarrow (1-\tau)\psi^- + \tau\psi$
17: If $n \bmod K = 0$: set $u_{\mathrm{ref}} \leftarrow u_\theta$
18: **end for**
19: **return** Fine-tuned generator u_θ

4.1 Environment Interaction and Replay

Actions are generated by simulating the current CTMC generator u_θ forward in flow time. Starting from $X_0 \sim \mathrm{Uniform}(\mathcal{A})$, we run M Euler sub-steps with step size $\Delta t = 1/M$ (Algorithm 2 in Appendix B). At each sub-step, the process either stays at the current action or jumps to a new one with probability proportional to the learned transition rates. The terminal action $a = X_M$ is executed in the environment, and the resulting transition (s, a, r, s', d) is stored in a replay buffer $\mathcal{B}$. To prevent the fine-tuning distribution from drifting too far from the pre-trained policy, each training minibatch mixes online transitions from $\mathcal{B}$ with a fraction ρ of offline transitions from $\mathcal{D}_{\mathrm{off}}$ (Line 4), following offline-to-online practice (Nakamoto et al., 2023; Lee et al., 2022).

4.2 Candidate Set Construction and Target Policy

Evaluating every action during online fine-tuning becomes prohibitively expensive in large discrete action spaces. To address this, we carefully construct a small candidate set $\mathcal{A}_{\mathrm{cand}}(s)$ from two

complementary sources. First, N_{roll} rollouts of the frozen reference generator provide actions favored by the pretrained policy, preserving useful offline knowledge. However, relying only on these actions may prevent the policy from discovering better actions missing from the offline data. We therefore add N_{rand} uniformly sampled actions to encourage exploration beyond the pretrained policy. This combination focuses computation on promising actions while allowing the policy to discover new ones.

Smoothed reference policy. The reference distribution over the candidate set is estimated from rollout frequencies with additive smoothing (Line 9):

$$\pi_{\text{ref}}(a \mid s) \;\propto\; \frac{\text{count}(a)}{N_{\text{roll}}} + \frac{\epsilon}{|\mathcal{A}_{\text{cand}}(s)|}, \qquad a \in \mathcal{A}_{\text{cand}}(s). \tag{2}$$

Here, the first term, $\text{count}(a)/N_{\text{roll}}$, estimates how frequently the pretrained policy selects action a, giving greater weight to actions favored by the reference generator. The second term, $\epsilon/|\mathcal{A}_{\text{cand}}(s)|$, adds a small positive weight to every candidate, including uniformly sampled actions that never appeared in the reference rollouts. Without this smoothing, such actions would receive zero probability and could not gain probability through the subsequent advantage-weighted update. The resulting weights are normalized over the candidate set to obtain a valid probability distribution.

Target policy. The target policy $\tilde{\pi}$ is the solution to the KL-regularized policy improvement problem (Ziegler et al., 2019; Peters et al., 2010a) (Line 10):

$$\tilde{\pi}(a \mid s) = \frac{\pi_{\text{ref}}(a \mid s)\, \exp\big(\bar{A}(s,a)/\beta\big)}{\sum_{a' \in \mathcal{A}_{\text{cand}}(s)} \pi_{\text{ref}}(a' \mid s)\, \exp\big(\bar{A}(s,a')/\beta\big)}. \tag{3}$$

Here, $\bar{A}(s,a)$ is the normalized and clipped conservative advantage defined in Section 4.3. We extend this target to the full action space by setting $\tilde{\pi}(a \mid s) = 0, a \notin \mathcal{A}_{\text{cand}}(s)$. The advantage term increases the probability of actions with higher estimated values, while β controls how strongly the target policy moves away from the reference distribution. The key question is whether restricting this update to a small candidate set introduces substantial error. Our analysis shows that this error depends on the target probability mass excluded from the candidate set (Proposition 4.1). Under the assumptions of Theorem 4.2, the resulting CTMC generator error is also bounded by this missing mass. Thus, the candidate set can remain small when it captures most of the target probability mass.

4.3 Critic and Value Network Updates

We maintain two independent critic networks Q_{ϕ_1}, Q_{ϕ_2} and a value network V_ψ. Using two critics helps reduce overestimation of action values, so taking their minimum can prevent the policy from favoring actions with inaccurately high estimated rewards. We update the critics using mixed offline and online replay, while a lagged target policy constructed from frozen networks provides stable supervision for the value network.

For the actor update, we compute the conservative advantage

$$A(s,a) = \min_{k=1,2} Q_{\phi_k}(s,a) - V_{\psi^-}(s).$$

We normalize the advantages over the candidate set and clip them to obtain

$$\bar{A}(s,a) = \text{clip}\left(\frac{A(s,a) - \mu_s}{\sigma_s + \varepsilon}, -c, c\right),$$

where μ_s and σ_s are the mean and standard deviation of the candidate advantages. The complete critic and value update equations are provided in Appendix A.2.

4.4 Actor Update: DFM Loss and Path-Space KL

The actor update has two components: a discrete flow matching loss that steers the generator toward $\tilde{\pi}$, and a path-space KL penalty that prevents catastrophic forgetting.

Linear bridge and target generator. We define a linear interpolation from the full-action uniform prior $p_0 = \mathrm{Uniform}(\mathcal{A})$ to the target policy:

$$p_t(a \mid s) = (1-t)\, p_0(a) + t\, \tilde{\pi}(a \mid s), \qquad \dot{p}_t(a \mid s) = \tilde{\pi}(a \mid s) - p_0(a). \tag{4}$$

For the candidate-set update, we view $\tilde{\pi}(\cdot \mid s)$ as a distribution on the full action space by setting $\tilde{\pi}(a \mid s) = 0$ for $a \notin \mathcal{A}_{\mathrm{cand}}(s)$. Because $\tilde{\pi}$ spreads mass across multiple actions, the Kolmogorov forward equation does not uniquely determine a generator consistent with $\dot{p}_t$. We resolve this via the *independent coupling transport* (Campbell et al., 2024; Holderrieth et al., 2025), which constructs a valid target generator u_t^* by routing outflow from each losing action to all gaining actions proportionally to their demand (Line 11). Concretely, for $i \neq j$:

$$u_t^*(i \to j \mid s) = \frac{(\dot{p}_t(i))^- \, (\dot{p}_t(j))^+}{p_t(i \mid s)\, Z_t}, \qquad Z_t = \sum\nolimits_a (\dot{p}_t(a))^+. \tag{5}$$

The validity of this generator and its mass conservation proof are provided in Appendix C.1.

Source actions outside the candidate set. Since the source distribution covers the full action space, intermediate actions may lie outside the candidate set. The independent coupling treats these actions as sources of probability mass and transfers their mass to candidate actions with positive demand. Further details are provided in Appendix C.1.

Discrete flow matching loss. For each state s_b, we sample a flow time $t \sim \mathrm{Uniform}(0, 1-\delta)$ and an intermediate action $i \sim p_t(\cdot \mid s_b)$, then regress the learned rates against the target rates (Line 12):

$$\mathcal{L}_{\mathrm{DFM}}(\theta) = \frac{1}{B} \sum_{b=1}^{B} \sum_{j \neq i_b} \left(u_\theta(i_b \to j, t_b \mid s_b) - u_{t_b}^*(i_b \to j \mid s_b) \right)^2. \tag{6}$$

Path-space KL regularization. Unlike the policy KL commonly used in RL, which compares two action distributions at a state, our KL compares two CTMC path distributions, so it constrains how probability mass moves over flow time. We penalize deviations across the *entire trajectory* of the CTMC, including both the jump destinations and the holding times (Line 13). For CTMCs, the KL divergence between path measures $\mathcal{P}_{u_\theta}$ and $\mathcal{P}_{u_{\mathrm{ref}}}$ admits a tractable decomposition via the Radon–Nikodym derivative (Kipnis & Landim, 2013; Zhang et al., 2025b):

$$\begin{aligned} \mathrm{KL}\big(\mathcal{P}_{u_\theta} \,\|\, \mathcal{P}_{u_{\mathrm{ref}}}\big) = \mathbb{E}_{\mathcal{P}_{u_\theta}}\Bigg[& \sum_{k:\mathrm{jumps}} \log \frac{u_\theta(X_{t_k^-} \to X_{t_k}, t_k \mid s)}{u_{\mathrm{ref}}(X_{t_k^-} \to X_{t_k}, t_k \mid s)} \\ & + \int_0^1 \big(\lambda_{\mathrm{ref}}(X_t, t \mid s) - \lambda_\theta(X_t, t \mid s) \big) dt \Bigg], \end{aligned} \tag{7}$$

where $\lambda_\theta(i, t \mid s) = \sum_{j \neq i} u_\theta(i \to j, t \mid s)$ is the total exit rate. This is a Monte Carlo plug-in surrogate where gradients flow only through the evaluated u_θ and λ_θ along the simulated path, so the term acts as a practical regularizer toward u_{ref} rather than as an unbiased gradient estimator of the path-space KL. For each sampled state s_b, we estimate this quantity with a Monte Carlo plug-in estimator $\widehat{\mathcal{L}}_{\mathrm{KL}}(s_b, \theta)$ along a simulated CTMC path.

Combined fine-tuning objective. The generator is then updated by minimizing

$$\mathcal{L}_{\mathrm{actor}}(\theta) = \mathcal{L}_{\mathrm{DFM}}(\theta) + \alpha \widehat{\mathcal{L}}_{\mathrm{KL}}(\theta), \tag{8}$$

where $\widehat{\mathcal{L}}_{\mathrm{KL}}(\theta) = \frac{1}{B} \sum_{b=1}^{B} \widehat{\mathcal{L}}_{\mathrm{KL}}(s_b, \theta)$ and $\alpha > 0$ controls the strength of the path-space trust region.

4.5 Rate Network Parameterization

The CTMC generator u_θ constraints (Eq. 1) are enforced by construction. The network outputs unconstrained logits $g_\theta(i \to j, t \mid s)$ for $j \neq i$, which are passed through a softplus to obtain nonnegative off-diagonal rates: $u_\theta(i \to j, t \mid s) = \mathrm{softplus}(g_\theta(i \to j, t \mid s))$. The diagonal is set to $u_\theta(i \to i, t \mid s) = -\sum_{j \neq i} u_\theta(i \to j, t \mid s)$, guaranteeing a valid generator at every parameter setting. Our formulation is independent of the network architecture, provided that the output rates satisfy the CTMC generator constraints.

4.6 Theoretical Analysis

We analyze the error caused by replacing the full action space with a candidate set. For the analysis, let $\tilde{\pi}(\cdot \mid s)$ denote the ideal target policy on the full action space, and define its candidate-restricted renormalization as

$$\tilde{\pi}_{\mathrm{cand}}(a \mid s) = \frac{\tilde{\pi}(a \mid s)\,\mathbf{1}\{a \in \mathcal{A}_{\mathrm{cand}}(s)\}}{\sum_{a' \in \mathcal{A}_{\mathrm{cand}}(s)} \tilde{\pi}(a' \mid s)}.$$

Proposition 4.1 (Coverage Error). *Let $\epsilon(s) = \sum_{a \notin \mathcal{A}_{\mathrm{cand}}(s)} \tilde{\pi}(a \mid s)$ be the target mass excluded by the candidate set. If $\epsilon(s) < 1$, then*

$$\|\tilde{\pi}_{\mathrm{cand}} - \tilde{\pi}\|_1 = 2\epsilon(s).$$

If the candidate set is built from N_{roll} independent samples from π_{ref} and N_{rand} independent uniform samples, then

$$\mathbb{E}[\|\tilde{\pi}_{\mathrm{cand}} - \tilde{\pi}\|_1] = 2 \sum_{a \in \mathcal{A}} \tilde{\pi}(a \mid s)\,(1 - \pi_{\mathrm{ref}}(a \mid s))^{N_{\mathrm{roll}}}\,(1 - 1/K)^{N_{\mathrm{rand}}}.$$

Proposition 4.1 shows that the expected approximation error decreases exponentially as the number of reference rollouts and uniform samples increases. This explains why a relatively small candidate set can accurately approximate the target policy when it captures most of its probability mass. The proof is provided in Appendix C.3.

Theorem 4.2 (Generator Stability). *Let u_t^* and $u_t^{*,\mathrm{cand}}$ be the target generators induced by $\tilde{\pi}$ and $\tilde{\pi}_{\mathrm{cand}}$, respectively. Assume both bridges use the same source distribution $p_0 = \mathrm{Unif}(\mathcal{A})$. Fix $\delta \in (0, 1)$ and suppose that for all $t \in [0, 1-\delta]$, $p_t(i),\ p_t^{\mathrm{cand}}(i) \geq p > 0, Z_t,\ Z_t^{\mathrm{cand}} \geq Z > 0$. Then, for every source action i,*

$$\sum_{j \neq i} \left| u_t^{*,\mathrm{cand}}(i \to j) - u_t^*(i \to j) \right| \leq \frac{C}{p^2 Z^2}\, \|\tilde{\pi}_{\mathrm{cand}}(\cdot \mid s) - \tilde{\pi}(\cdot \mid s)\|_1 ,$$

for a universal constant C.

Theorem 4.2 shows that as the candidate set captures more target-policy probability mass, the resulting CTMC generator approaches the full-action generator. The proof is provided in Appendix C.4.

5 Experiments

We evaluate three aspects of DRIFT: its ability to adapt pretrained policies across different structured discrete action spaces, the contribution of path-space regularization and reference-guided candidate selection, and its practical scalability as the action space grows. Macro-action MinAtar (Section 5.1) tests combinatorial action spaces ranging from 1,296 to approximately 1.68 million actions, while oracle-free Jericho (Section 5.2) tests language-based action generation without access to admissible-action lists. Alongside returns and learning curves, we report controlled ablations, candidate coverage, and wall-clock measurements. Additional benchmarks and implementation details are provided in the appendix.

5.1 Macro-action MinAtar

Setup. A macro executes K primitive actions without observing intermediate frames, so $\mathcal{A} = \{0, \ldots, 5\}^K$, and all methods use semi-Markov targets. Every offline-to-online method starts from the same 100K-transition dataset per game and macro length, collected by a behaviour double DQN, and then interacts for 1.2M primitive frames, so all methods receive equal experience whatever their macro length. We evaluate every 40K frames with 20 episodes over five seeds and report returns.

Baselines. The offline-to-online baselines are RL-D^2-O2O (Ma et al., 2026), Q-chunking (Li et al., 2026a), QDFM (Khan et al., 2026), BraVE (Landers et al., 2026), Cal-QL (Nakamoto et al., 2023), IQL (Kostrikov et al., 2022) and SPA (Li et al., 2026b). The online-only baselines are DRIFT without offline data, RL-D^2 as published, DQN-Macro with one output per macro, IMPALA-Macro (Espeholt

Table 1: Macro-action MinAtar at K=4 ($|\mathcal{A}|$=1,296). Methods with an offline phase show the score after pre-training → the final score after 1.2M online primitive frames. Mean ± std over five seeds. Bold marks the best macro-action score per game. Primitive-action references do not depend on K and are reported in Table 3.

Method	Asterix	Breakout	Freeway	Seaquest	Space Invaders
Macro-action, offline-to-online					
DRIFT (ours)	0.4 → **2.1±0.3**	0.5 → 15.4±3.6	11.3 → **38.1±0.5**	0.6 → **7.0±0.2**	4.2 → 36.0±1.7
RL-D^2-O2O	0.4 → 1.2±0.1	0.9 → 16.4±0.2	9.4 → 32.8±2.3	0.3 → 5.9±0.1	4.2 → **40.2±2.8**
Q-chunking	0.4 → 0.9±0.3	0.9 → **20.4±0.7**	25.4 → 25.1±0.2	0.8 → 6.9±0.1	4.8 → 30.4±1.7
QDFM	0.6 → 1.3±0.3	0.0 → 14.7±2.4	23.2 → 25.4±1.7	0.5 → 1.5±0.2	3.4 → 24.4±2.7
BraVE	0.5 → 0.9±0.0	0.0 → 11.2±3.2	0.0 → 14.4±3.5	0.3 → 4.5±0.2	1.4 → 25.5±1.8
Cal-QL	0.6 → 0.6±0.1	0.7 → 5.5±0.2	13.3 → 3.0±0.7	0.3 → 5.7±0.3	2.4 → 18.5±2.5
IQL	0.5 → 0.7±0.1	0.6 → 6.9±2.1	0.3 → 0.2±0.1	0.1 → 1.2±0.2	4.5 → 9.0±0.5
SPA	0.8 → 0.8±0.0	1.2 → 11.0±1.6	18.9 → 22.5±0.8	0.6 → 1.8±0.1	5.6 → 18.1±1.0
Macro-action, online only					
DRIFT, no offline data	0.8±0.3	3.8±0.8	19.9±4.7	4.0±0.1	26.7±4.1
RL-D^2	1.0±0.2	8.7±1.1	11.5±11.5	3.5±0.5	31.7±1.1
DQN-Macro	0.8±0.0	7.2±0.6	7.7±3.7	4.3±0.1	10.6±2.0
IMPALA-Macro	0.4±0.1	3.7±0.3	17.8±0.4	4.9±0.3	17.3±1.0
MASP	0.7±0.2	1.1±0.3	6.9±6.9	2.7±0.1	3.0±1.0

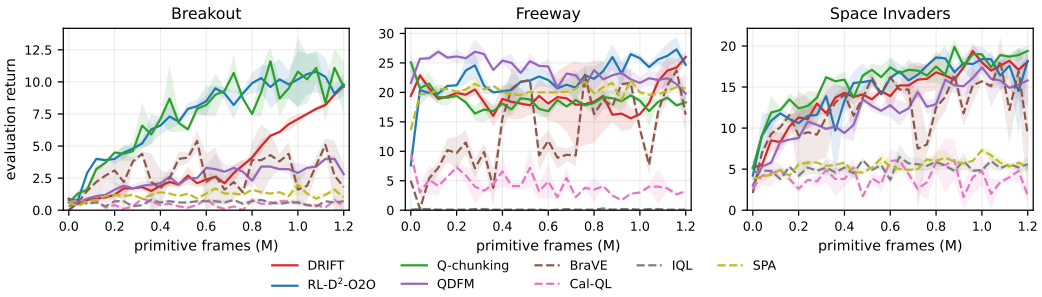


Figure 2: Learning curves at K=8 ($|\mathcal{A}|$≈1.68 million) for the methods with an offline phase, on the three games that separate methods. The first point is the score after pre-training and shading spans the five seeds. Table 3 gives final scores for all games.

et al., 2018) and MASP (Hosu et al., 2026), the last at K=4 only because its similarity matrix has $|\mathcal{A}|^2$ entries. DRIFT uses B=64, β=0.5, M=10 and α=0.01.

Results at K=4. As we can see from Table 1, DRIFT achieves the highest macro-action mean scores on Freeway (38.1 versus 32.8 for RL-D^2-O2O), Asterix (2.1), and Seaquest (7.0). Q-chunking leads the macro-action methods on Breakout (20.4), while RL-D^2-O2O leads on Space Invaders (40.2). Temporal abstraction is game-dependent: primitive R2D2 reaches 25.2 on Breakout, exceeding every macro-action method. Offline initialization improves DRIFT's final mean scores across all five games, including gains of 18.2 on Freeway and 9.3 on Space Invaders relative to online-only DRIFT.

Scaling to K=8. Here, the action space grows to approximately 1.68 million actions. Figure 2 shows the offline-to-online learning curves on Breakout, Freeway, and Space Invaders. DRIFT remains competitive with the discrete generative baselines and achieves a higher final return on Freeway than RL-D^2-O2O (26.0 versus 24.7). Complete results across all five games are provided in Appendix D.2.

Ablations. Removing path-space KL reduces returns from 38.1 to 25.9 on Freeway and from 36.0 to 20.8 on Space Invaders. These results support path-space regularization as an important component of offline-to-online adaptation, although excessively strong regularization can restrict policy improvement. Replacing reference-guided candidates with uniform sampling further reduces returns to 11.1 and 8.0, respectively. Increasing the candidate budget generally improves performance, highlighting the trade-off between computation and candidate coverage. Full ablation results are provided in Appendix D.5.

Table 2: Offline to online on five Jericho games without the admissible-action oracle, at B=128. Each cell reads as the score after offline pre-training, then the score after a further 100K online steps, the latter as mean ± std over 5 seeds. Bold marks the best final mean among methods without the oracle, and the best average gain. †From Tuyls et al. (2022), Table 1, which uses the oracle and 800K steps and reports no offline phase.

Method	Zork1	Deephome	Detective	Ludicorp	Pentari	Avg. %max
No state-level admissible-action oracle						
DRIFT (ours)	1.2 → **18.5±1.5**	1.0 → 1.0±0.0	20.8 → 42.8±5.2	6.0 → 7.5±0.1	1.6 → **9.8±1.2**	2.5 → **7.3% (+4.8)**
DRIFT-no path-KL	1.4 → 14.0±2.0	1.0 → 1.0±0.0	23.5 → 40.0±4.0	3.2 → 5.6±0.1	1.9 → 3.8±0.2	2.4 → 4.9% (+2.5)
DRIFT-no offline data	11.4±1.3	1.0±0.0	20.0±4.0	3.6±0.1	2.8±0.2	3.1
DRRN-Struct	0.2 → 12.0±1.5	1.0 → 1.0±0.0	26.5 → **46.5±1.0**	2.1 → 2.3±0.1	0.5 → 0.0±0.0	2.0 → 3.6% (+1.6)
CALM	0.0 → 0.0±0.0	1.0 → 1.0±0.0	19.5 → 27.0±13.0	6.0 → 6.4±0.1	0.0 → 0.0±0.0	2.0 → 2.4% (+0.4)
CSM	0.0 → 0.0±0.0	1.0 → 1.0±0.0	19.5 → 41.2±1.2	6.0 → **8.4±0.9**	0.0 → 0.0±0.0	2.0 → 3.5% (+1.5)
With the admissible-action oracle, reference only						
DRRN (oracle actions)	10.0 → 12.3±2.3	20.2 → 18.4±1.1	40.2 → 89.0±16.0	15.6 → 16.0±1.1	15.4 → 30.0±0.0	10.6 → 17.6% (+7.0)
DRRN†	40.3	58.8	289.9	17.1	45.6	37.6%
XTX†	103.4	77.7	312.2	78.8	49.6	53.1%
Maximum game score	350	300	360	150	70	100%

5.2 TEXT GAMES WITHOUT THE ADMISSIBLE-ACTION ORACLE

Prior work on Jericho, and our experiments in Appendix E.2, give every agent the admissible actions that Jericho computes by simulating candidates in the emulator. We remove that oracle, so an action is a template from the game grammar with up to two object slots filled with words from the observation, room description and inventory. DRRN-Struct isolates the generator, sharing DRIFT's action space, encoders, budget and offline data but selecting among uniform samples by Q-value. We also reimplement CALM (Yao et al., 2020) and the self-imitation part of CSM (Shi et al., 2023) with GPT-2, and train DRRN (He et al., 2016) on admissible lists for reference. All methods use the same offline data and 100K online steps at B=128 with five seeds.

Without the admissible-action oracle, performance is substantially lower than the published oracle-based references (Table 2). DRIFT achieves a mean normalized score of 7.3%, with final scores of 18.5 on Zork1 and 9.8 on Pentari, exceeding the other oracle-free methods on both games. DRRN-Struct scores higher on Detective (46.5 versus 42.8), while CSM leads on Ludicorp (8.4 versus 7.5). All oracle-free methods remain at 1.0 on Deephome, compared with 18.4 for DRRN using oracle actions. Relative to DRRN-Struct, DRIFT issues accepted commands more frequently on Zork1 (0.262 versus 0.014) and achieves higher admissible-action recall on four of the five games. Increasing the candidate budget from $B = 32$ to $B = 256$ improves Detective recall from 0.123 to 0.519 and its final score from 34.2 to 51.2. Further results are provided in Appendix E.1. Under matched runtime profiling, DRIFT also requires approximately half the median training time of DRRN-Struct (3.2 versus 6.4 hours); see Appendix E.1.

Additional results and takeaways. DRIFT also performs well on standard MinAtar and discretized D4RL tasks (Appendix F). Together with our macro-action MinAtar and oracle-free Jericho results, these experiments demonstrate the applicability of a common offline-to-online framework across small, combinatorial, and language-based discrete action spaces. Our ablations further support the benefits of combining path-space regularization with reference-guided candidate selection.

6 CONCLUSION, LIMITATIONS AND FUTURE WORK

We introduced DRIFT for offline-to-online fine-tuning of discrete CTMC policies using advantage-weighted flow matching and path-space regularization. Our reference-guided candidate-set mechanism reduces computation in large action spaces, with theoretical bounds on its approximation error. On macro-action MinAtar, offline-pretrained DRIFT improves over its online-only counterpart across all five games and achieves the highest macro-action mean return on three at $K = 4$. On Jericho without the admissible-action oracle, DRIFT achieves a mean normalized score of 7.3%, outperforming the evaluated oracle-free baselines. DRIFT remains sensitive to the regularization weight and candidate-set coverage, and its performance varies across environments. Future work includes adaptive regularization, more efficient candidate selection, and evaluation on more combinatorial control tasks.

Reproducibility Statement

We provide anonymized source code and scripts for offline data collection, training, evaluation, and reproducing the reported tables and figures in the supplementary material. The full training procedure is presented in Appendix H, with pre-training and critic-update details in Appendix A. All theoretical assumptions and complete proofs are provided in Appendix C. The experimental protocols for macro-action MinAtar and oracle-free Jericho are described in Sections 5.1 and 5.2, with additional details in Appendices D and E.1. These include offline dataset construction, evaluation protocols, implementation choices, and candidate-set analyses. We use shared offline datasets and interaction budgets for comparable methods and report the number of random seeds and variability for each experiment.

AI Use Statement

We used large language models, to assist with manuscript drafting and editing, literature discovery, and feedback on methodological and experimental design. AI tools also assisted with code debugging, and the presentation of arguments. The authors reviewed the AI-assisted material and take full responsibility for the methods, implementation, experimental results, mathematical claims, and final manuscript.

References

Prithviraj Ammanabrolu and Matthew Hausknecht. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*, 2020.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning*, pp. 5453–5512. PMLR, 2024.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

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.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7903–7910, 2020.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1621–1630, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ionel-Alexandru Hosu, Traian Rebedea, and Razvan Pascanu. Meta-learning how to share credit among macro-actions. *Advances in Neural Information Processing Systems*, 38:90378–90403, 2026.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 67195–67212, 2023.

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.

Fairoz Nower Khan, Nabuat Zaman Nahim, Ruiquan Huang, Haibo Yang, and Peizhong Ju. Flow matching for offline reinforcement learning with discrete actions. *arXiv preprint arXiv:2602.06138*, 2026.

Claude Kipnis and Claudio Landim. *Scaling limits of interacting particle systems*. Springer Science & Business Media, 2013.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Matthew Landers, Taylor Killian, Hugo Barnes, Tom Hartvigsen, and Afsaneh Doryab. Brave: Offline reinforcement learning for discrete combinatorial action spaces. *Advances in Neural Information Processing Systems*, 38:71415–71444, 2026.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026a.

Songlin Li, Wei Xiao, Hao Wu, Xiaodan Zhang, Daolong An, and Shuai Lü. State proficiency-based adaptive fine-tuning for offline-to-online reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23169–23176, 2026b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023.

Haitong Ma, Ofir Nabati, Aviv Rosenberg, Bo Dai, Oran Lang, Craig Boutilier, Na Li, Shie Mannor, Lior Shani, and Guy Tenneholtz. Reinforcement learning with discrete diffusion policies for combinatorial action spaces. In *Forty-third International Conference on Machine Learning*, 2026.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 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.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning*. PMLR, 2025.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010a.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, 2010b.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3 edition, 1976.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

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

Zijing Shi, Yunqiu Xu, Meng Fang, and Ling Chen. Self-imitation learning for action generation in text-based games. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 703–726, 2023.

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

Emanuel Todorov. Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19, 2006.

Jens Tuyls, Shunyu Yao, Sham M Kakade, and Karthik R Narasimhan. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations*, 2022.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Avantika Lal, Tommi Jaakkola, Sergey Levine, Aviv Regev, Tommaso Biancalani, et al. Fine-tuning discrete diffusion models via reward optimization with applications to dna and protein design. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep calm and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8736–8754, 2020.

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

Haichao Zhang, Wei Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

APPENDIX

A ADDITIONAL TRAINING DETAILS

A.1 OFFLINE PRE-TRAINING

Our Stage 1 pre-training produces a reference generator u_{ref} from the offline dataset $\mathcal{D}_{\mathrm{off}}$. This procedure is an *advantage-weighted DFM* variant.

Critic pre-training. We pre-train two critic networks Q_{ϕ_1}, Q_{ϕ_2} and a value network V_ψ using standard TD learning on $\mathcal{D}_{\mathrm{off}}$:

$$y_b = r_b + \gamma\, V_{\psi^-}(s'_b)\,(1 - d_b), \qquad \mathcal{L}_{Q_k}(\phi_k) = \frac{1}{B}\sum_b \big(Q_{\phi_k}(s_b, a_b) - \mathrm{sg}[y_b]\big)^2.$$

The value network is trained via an advantage-weighted target. This differs from the Boltzmann soft-value backup with in-support sampling.

Generator pre-training. We compute a target policy $\tilde{\pi}_{\text{pre}}(a \mid s) \propto \exp(\bar{A}_{\text{off}}(s,a)/\beta)$ over the *full* action space using the pre-trained critics, where $\bar{A}_{\text{off}}$ is the clipped, normalized pessimistic advantage. The generator is then trained via standard DFM to flow from $p_0 = \text{Uniform}(\mathcal{A})$ to $\tilde{\pi}_{\text{pre}}$ using the independent coupling (Eq. 5).

The pre-trained generator captures high-value behaviors from the offline data and provides a strong initialization for online fine-tuning.

A.2 Critic and Value Network Updates

We maintain two independent critic networks Q_{ϕ_1}, Q_{ϕ_2} and a value network V_ψ, together with frozen target copies $Q_{\phi_k^-}$ and V_{ψ^-}.

Critic update. Each critic is trained by minimizing the squared Bellman error against a one-step TD target computed from the frozen value network:

$$\mathcal{L}_{Q_k}(\phi_k) = \frac{1}{B}\sum_{b=1}^{B}\left(Q_{\phi_k}(s_b,a_b) - \text{sg}\left[r_b + \gamma V_{\psi^-}(s_b')(1-d_b)\right]\right)^2,$$

where $\text{sg}[\cdot]$ denotes stop-gradient. Using the frozen value network avoids a full CTMC rollout at every critic update.

Value update. The value network is trained using a lagged target policy constructed from frozen networks. We first compute

$$A^-(s,a) = \min_k Q_{\phi_k^-}(s,a) - V_{\psi^-}(s).$$

These advantages are normalized over the candidate set and clipped to $[-c,c]$, giving $\bar{A}^-(s,a)$. The lagged target policy is

$$\tilde{\pi}^-(a \mid s) \propto \pi_{\text{ref}}(a \mid s)\exp\left(\bar{A}^-(s,a)/\beta\right), \qquad a \in \mathcal{A}_{\text{cand}}(s). \tag{9}$$

Both the lagged target policy and the value target are computed over the candidate set, avoiding full-action enumeration. The value loss is

$$\mathcal{L}_V(\psi) = \frac{1}{B}\sum_{b=1}^{B}\left(V_\psi(s_b) - \text{sg}\left[\sum_{a\in\mathcal{A}_{\text{cand}}(s_b)}\tilde{\pi}^-(a \mid s_b)\min_k Q_{\phi_k^-}(s_b,a)\right]\right)^2.$$

All target networks are soft-updated:

$$\phi_k^- \leftarrow (1-\tau)\phi_k^- + \tau\phi_k, \qquad \psi^- \leftarrow (1-\tau)\psi^- + \tau\psi.$$

B CTMC Simulation Procedure

Euler discretization. Euler discretization approximates a continuous-time Markov chain by dividing the time interval $[0,1]$ into M small steps of size $\Delta t = 1/M$. At each step $t_m = m\Delta t$, the process either stays at its current action i or jumps to another action j. The probability of jumping from i to j is approximated by $\Delta t\, u_{t_m}(i \to j \mid s)$, while the probability of staying at i is $1 - \Delta t \sum_{j\neq i} u_{t_m}(i \to j \mid s)$. Repeating this procedure for M steps produces the final action X_M. We require $\Delta t \sum_{j\neq i} u_{t_m}(i \to j \mid s) \leq 1$ to ensure valid transition probabilities.

Algorithm 2 describes the Euler discretization used to sample actions from the learned CTMC generator. This procedure is used both during environment interaction (Line 3 of Algorithm 1) and when constructing candidate sets via reference rollouts (Line 8).

The scheme corresponds to the standard Euler discretization of a CTMC (Lipman et al., 2024, Eq. (6.6)). For sufficiently regular and bounded transition rates, it has a local error of $O(\Delta t^2)$ and a global error of $O(\Delta t)$. Validity requires $\lambda\Delta t \leq 1$, which we ensure by choosing M sufficiently large.

Algorithm 2 CTMC Euler Simulation

Require: State s, generator u, sub-steps M, step size $\Delta t = 1/M$
Sample $X_0 \sim \mathrm{Uniform}(\mathcal{A})$
for $m = 0$ **to** $M - 1$ **do**
  Set $t_m \leftarrow m\,\Delta t$ and current action $i \leftarrow X_m$
  Compute total exit rate $\lambda \leftarrow \sum_{j \neq i} u(i \to j, t_m \mid s)$
  if $\lambda = 0$ **then**
    $X_{m+1} \leftarrow i$ {no transition possible}
  else
    With probability $1 - \lambda\,\Delta t$: $X_{m+1} \leftarrow i$ {stay}
    With probability $\lambda\,\Delta t$: sample $X_{m+1} \sim u(\cdot, t_m \mid s)/\lambda$ {jump}
  end if
end for
return Terminal action X_M

C Proofs and Derivations

C.1 Independent Coupling: Construction and Mass Conservation

Construction. Given the linear bridge $p_t(a \mid s) = (1-t)\,p_0(a) + t\,\tilde{\pi}(a \mid s)$ with $p_0 = \mathrm{Uniform}(\mathcal{A})$, its time derivative is $\dot{p}_t(a) = \tilde{\pi}(a \mid s) - p_0(a)$. We decompose $\dot{p}_t$ into its positive and negative parts:

$$(\dot{p}_t(a))_+ = \max(\dot{p}_t(a), 0), \quad (\dot{p}_t(a))_- = \max(-\dot{p}_t(a), 0), \quad Z_t = \sum_a (\dot{p}_t(a))_+.$$

The independent coupling assigns rates proportional to the product of outflow from i and inflow to j:

$$u_t^*(i \to j \mid s) = \frac{(\dot{p}_t(i))^-\,(\dot{p}_t(j))^+}{p_t(i \mid s)\,Z_t}, \qquad i \neq j,$$

with $u_t^*(i \to j \mid s) = 0$ whenever $p_t(i \mid s) = 0$ or $Z_t = 0$. This construction is valid because it produces nonnegative off-diagonal rates and the diagonal is set by conservation.

Source actions outside the candidate set. The source distribution $p_0 = \mathrm{Uniform}(\mathcal{A})$ covers the full action space, whereas the target $\tilde{\pi}$ is supported only on $\mathcal{A}_{\mathrm{cand}}(s)$. Consequently, an intermediate action $i \sim p_t(\cdot \mid s)$ may lie outside the candidate set, especially for small t. For such an action,

$$\tilde{\pi}(i \mid s) = 0, \qquad \dot{p}_t(i \mid s) = -\frac{1}{|\mathcal{A}|}, \qquad (\dot{p}_t(i \mid s))^- = \frac{1}{|\mathcal{A}|}.$$

Thus, i acts as a source of probability mass. Under the independent-coupling generator in Eq. 5, this mass is transferred to actions j with positive demand, satisfying

$$(\dot{p}_t(j \mid s))^+ > 0 \quad \Longleftrightarrow \quad \tilde{\pi}(j \mid s) > \frac{1}{|\mathcal{A}|}.$$

Such actions must belong to $\mathcal{A}_{\mathrm{cand}}(s)$. Hence, actions outside the candidate set are not ignored. They supply probability mass to candidate actions that need to gain probability.

Interpretation in the fine-tuning context. The independent coupling spreads each source's outflow across all sinks proportionally to demand. In the fine-tuning setting, this means that actions gaining probability under $\tilde{\pi}$ receive mass from *all* actions losing probability, rather than from a single greedy source. This produces a dense, stable supervision signal that reduces susceptibility to mode collapse during fine-tuning.

Proposition C.1 (Mass Conservation)**.** *Assume $p_t(i) > 0$ for actions with outgoing mass and $Z_t = \sum_a (\dot{p}_t(a))_+ > 0$. Then the independent-coupling generator*

$$u_t^*(i \to j) = \frac{(\dot{p}_t(i))_-\,(\dot{p}_t(j))_+}{p_t(i)\,Z_t}, \qquad i \neq j,$$

satisfies the Kolmogorov forward equation.

Proof of Proposition C.1 (Mass Conservation). Substituting the generator into the Kolmogorov forward equation, the net flow into state j is:

$$\begin{aligned}\mathrm{Net}(j) &= \sum_{i\neq j} p_t(i)\, u_t^*(i\to j) - p_t(j)\sum_{k\neq j} u_t^*(j\to k)\\ &= \sum_{i\neq j}\frac{(\dot p_t(i))^-\,(\dot p_t(j))^+}{Z_t} - \frac{(\dot p_t(j))^-}{Z_t}\sum_{k\neq j}(\dot p_t(k))^+.\end{aligned}$$

Since $\sum_a \dot p_t(a) = 0$, total positive and negative parts are equal: $\sum_a(\dot p_t(a))^+ = \sum_a(\dot p_t(a))^- = Z_t$. Therefore $\sum_{i\neq j}(\dot p_t(i))^- = Z_t - (\dot p_t(j))^-$ and $\sum_{k\neq j}(\dot p_t(k))^+ = Z_t - (\dot p_t(j))^+$. Substituting and simplifying:

$$\mathrm{Net}(j) = \frac{(\dot p_t(j))^+\big(Z_t - (\dot p_t(j))^-\big) - (\dot p_t(j))^-\big(Z_t - (\dot p_t(j))^+\big)}{Z_t} = (\dot p_t(j))^+ - (\dot p_t(j))^- = \dot p_t(j). \quad \square$$

C.2 Path-Space KL Derivation

The KL divergence between CTMC path measures is a standard result from the theory of Markov jump processes (Norris, 1998; Kipnis & Landim, 2013). We provide a self-contained derivation for completeness.

Let $\mathcal{P}_{u_\theta}$ and $\mathcal{P}_{u_{\mathrm{ref}}}$ denote the path measures of CTMCs with generators u_θ and u_{ref}, respectively, both starting from the same initial distribution. A CTMC path over $[0,1]$ consists of an initial state, a sequence of jump times $0 < t_1 < t_2 < \cdots < t_{N_{\mathrm{jump}}} \le 1$, and the corresponding jump destinations. Between jumps, the chain holds at its current state.

The log-likelihood of a path under generator u decomposes as:

$$\log \mathcal{P}_u(\text{path}) = \sum_{k=1}^{N_{\mathrm{jump}}} \log u(X_{t_k^-}\to X_{t_k}, t_k \mid s) - \int_0^1 \lambda_u(X_t, t\mid s)\, dt,$$

where the first term accounts for the jumps that occurred and the second term accounts for the holding (no-jump) periods. The KL divergence follows directly:

$$\begin{aligned}\mathrm{KL}(\mathcal{P}_{u_\theta}\,\|\,\mathcal{P}_{u_{\mathrm{ref}}}) &= \mathbb{E}_{\mathcal{P}_{u_\theta}}\left[\log\frac{\mathcal{P}_{u_\theta}(\text{path})}{\mathcal{P}_{u_{\mathrm{ref}}}(\text{path})}\right]\\ &= \mathbb{E}_{\mathcal{P}_{u_\theta}}\left[\sum_{k:\mathrm{jumps}}\log\frac{u_\theta(X_{t_k^-}\to X_{t_k}, t_k)}{u_{\mathrm{ref}}(X_{t_k^-}\to X_{t_k}, t_k)} + \int_0^1\big(\lambda_{\mathrm{ref}}(X_t,t) - \lambda_\theta(X_t,t)\big)dt\right].\end{aligned}$$

Monte Carlo estimation. In practice, we estimate this by simulating one trajectory $X_0, X_1, \ldots, X_M$ from the current generator u_θ (with stop-gradient applied to the trajectory) and computing:

$$\widehat{\mathrm{KL}}(s) = \sum_{k:\mathrm{jumps}}\log\frac{u_\theta(X_{t_k^-}\to X_{t_k}, t_k\mid s)}{u_{\mathrm{ref}}(X_{t_k^-}\to X_{t_k}, t_k\mid s)} + \sum_{m=0}^{M-1}\big(\lambda_{\mathrm{ref}}(X_m, t_m\mid s) - \lambda_\theta(X_m, t_m\mid s)\big)\,\Delta t.$$

Gradients flow through $\log u_\theta$ at jump times and through λ_θ along the trajectory, while the trajectory itself is treated as fixed.

C.3 Proof of Proposition 4.1 (Coverage Error)

Proof. Let $\tilde\pi$ be the ideal full-action target policy and let

$$\epsilon(s) = \sum_{a\notin\mathcal{A}_{\mathrm{cand}}(s)}\tilde\pi(a\mid s).$$

Assume $\epsilon(s) < 1$. The candidate-restricted policy is

$$\tilde\pi_{\mathrm{cand}}(a\mid s) = \frac{\tilde\pi(a\mid s)\,\mathbf{1}\{a\in\mathcal{A}_{\mathrm{cand}}(s)\}}{1-\epsilon(s)}.$$

Therefore,

$$\|\tilde{\pi}_{\text{cand}} - \tilde{\pi}\|_1 = \sum_{a \in \mathcal{A}_{\text{cand}}} \left| \frac{\tilde{\pi}(a)}{1-\epsilon} - \tilde{\pi}(a) \right| + \sum_{a \notin \mathcal{A}_{\text{cand}}} \tilde{\pi}(a)$$
$$= \sum_{a \in \mathcal{A}_{\text{cand}}} \tilde{\pi}(a) \cdot \frac{\epsilon}{1-\epsilon} + \epsilon = (1-\epsilon) \cdot \frac{\epsilon}{1-\epsilon} + \epsilon = 2\epsilon.$$

For the expectation, action a is excluded only if it is missed by all N_{roll} independent samples from π_{ref} and all N_{rand} independent uniform samples. Hence

$$\mathbb{P}(a \notin \mathcal{A}_{\text{cand}}(s)) = (1 - \pi_{\text{ref}}(a \mid s))^{N_{\text{roll}}} (1 - 1/K)^{N_{\text{rand}}}.$$

Thus,

$$\mathbb{E}[\epsilon(s)] = \sum_{a \in \mathcal{A}} \tilde{\pi}(a \mid s)(1 - \pi_{\text{ref}}(a \mid s))^{N_{\text{roll}}} (1 - 1/K)^{N_{\text{rand}}},$$

and the result follows from $\|\tilde{\pi}_{\text{cand}} - \tilde{\pi}\|_1 = 2\epsilon(s)$. □

C.4 Proof of Theorem 4.2 (Generator Stability)

Proof. The proof proceeds in three parts. First, we set up the notation and record elementary boundedness facts that will be used throughout. Second, we decompose the generator difference into three terms, each measuring a distinct source of perturbation. Third, we bound each term separately and combine the results.

Step 1: Setup and elementary bounds. Let

$$d = \|\tilde{\pi}_{\text{cand}} - \tilde{\pi}\|_1.$$

By construction, both the full and candidate bridges use the same source $p_0 = \text{Uniform}(\mathcal{A})$ (Eq. 4). Therefore,

$$\dot{p}_t^{\text{cand}} - \dot{p}_t = \tilde{\pi}_{\text{cand}} - \tilde{\pi}, \qquad \|\dot{p}_t^{\text{cand}} - \dot{p}_t\|_1 = d.$$

For any fixed source action i and destination action $j \neq i$, let $p_t(i)$ and $p_t^{\text{cand}}(i)$ denote the full and candidate bridge probabilities of action i at flow time t. The maps $x \mapsto x_+ = \max\{x, 0\}$ and $x \mapsto x_- = \max\{-x, 0\}$ are 1-Lipschitz (Rudin, 1976), so

$$\sum_a \left| (\dot{p}_t^{\text{cand}}(a))_+ - (\dot{p}_t(a))_+ \right| \leq d, \qquad \sum_a \left| (\dot{p}_t^{\text{cand}}(a))_- - (\dot{p}_t(a))_- \right| \leq d.$$

The normalizers are

$$Z_t = \sum_a (\dot{p}_t(a))_+, \qquad Z_t^{\text{cand}} = \sum_a (\dot{p}_t^{\text{cand}}(a))_+,$$

which equal the corresponding total outgoing masses by $\sum_a \dot{p}_t(a) = 0$. Hence $Z_t = \frac{1}{2}\|\dot{p}_t\|_1$ and similarly for Z_t^{cand}.

To simplify notation, we abbreviate the four quantities entering each rate (the source supply, sink demand, source probability, and total moving mass) using unsubscripted letters for the candidate bridge and a 0-subscript for the full bridge:

$$A := (\dot{p}_t^{\text{cand}}(i))_-, \quad A_0 := (\dot{p}_t(i))_-, \quad B_j := (\dot{p}_t^{\text{cand}}(j))_+, \quad B_j^0 := (\dot{p}_t(j))_+,$$

and

$$P := p_t^{\text{cand}}(i), \quad P_0 := p_t(i), \quad Z := Z_t^{\text{cand}}, \quad Z_0 := Z_t.$$

With this notation,

$$u_t^{*,\text{cand}}(i \to j) = \frac{AB_j}{PZ}, \qquad u_t^*(i \to j) = \frac{A_0 B_j^0}{P_0 Z_0}.$$

The lower bounds in the theorem statement give $P, P_0 \geq \underline{p} > 0$ and $Z, Z_0 \geq \underline{Z} > 0$. Furthermore, since $\|\dot{p}_t\|_1 \leq \|\tilde{\pi}\|_1 + \|p_0\|_1 = 2$, we have

$$A,\, A_0,\, B_j,\, B_j^0 \in [0, 1], \qquad Z,\, Z_0 \in [0, 1], \qquad P,\, P_0 \in [0, 1].$$

These elementary boundedness facts will be used repeatedly below.

Step 2: Three-term decomposition. Inserting and subtracting two intermediate terms,

$$\frac{AB_j}{PZ} - \frac{A_0 B_j^0}{P_0 Z_0} = \frac{(A - A_0)B_j}{PZ} + \frac{A_0(B_j - B_j^0)}{PZ} + A_0 B_j^0 \left(\frac{1}{PZ} - \frac{1}{P_0 Z_0} \right).$$

Summing over $j \neq i$ and applying the triangle inequality,

$$\sum_{j \neq i} \left| u_t^{*,\mathrm{cand}}(i \to j) - u_t^*(i \to j) \right| \leq \mathrm{(I)} + \mathrm{(II)} + \mathrm{(III)},$$

where

$$\mathrm{(I)} := \frac{|A - A_0|}{PZ} \sum_{j \neq i} B_j, \quad \mathrm{(II)} := \frac{A_0}{PZ} \sum_{j \neq i} |B_j - B_j^0|, \quad \mathrm{(III)} := A_0 \sum_{j \neq i} B_j^0 \left| \frac{1}{PZ} - \frac{1}{P_0 Z_0} \right|.$$

Each term measures a distinct perturbation: (I) the change in source supply, (II) the change in sink demand, and (III) the change in the denominator.

Step 3a: Bound on Term (I). Since $\sum_{j \neq i} B_j \leq Z$, the factor Z cancels:

$$\mathrm{(I)} \leq \frac{|A - A_0|}{P}.$$

Using $P \geq \underline{p}$ and the 1-Lipschitz property of $x \mapsto x_-$,

$$|A - A_0| = \left| (\dot{p}_t^{\mathrm{cand}}(i))_- - (\dot{p}_t(i))_- \right| \leq \left| \dot{p}_t^{\mathrm{cand}}(i) - \dot{p}_t(i) \right| = \left| \tilde{\pi}_{\mathrm{cand}}(i) - \tilde{\pi}(i) \right| \leq d,$$

where the final inequality uses that the pointwise difference is dominated by the L_1 norm. Therefore,

$$\mathrm{(I)} \leq \frac{d}{\underline{p}}.$$

Step 3b: Bound on Term (II). Since $A_0 \leq 1$, $P \geq \underline{p}$, and $Z \geq \underline{Z}$,

$$\mathrm{(II)} \leq \frac{1}{\underline{p}\,\underline{Z}} \sum_{j \neq i} |B_j - B_j^0|.$$

The 1-Lipschitz property of $x \mapsto x_+$ gives $|B_j - B_j^0| \leq |\dot{p}_t^{\mathrm{cand}}(j) - \dot{p}_t(j)| = |\tilde{\pi}_{\mathrm{cand}}(j) - \tilde{\pi}(j)|$ for each j, so summing yields $\sum_{j \neq i} |B_j - B_j^0| \leq d$. Therefore,

$$\mathrm{(II)} \leq \frac{d}{\underline{p}\,\underline{Z}}.$$

Step 3c: Bound on Term (III). We expand the denominator difference as

$$\frac{1}{PZ} - \frac{1}{P_0 Z_0} = \frac{P_0 Z_0 - PZ}{PZ\, P_0 Z_0} = \frac{(P_0 - P)Z_0 + P(Z_0 - Z)}{PZ\, P_0 Z_0}.$$

Taking absolute values, the denominator is bounded below by $PZ\, P_0 Z_0 \geq \underline{p}^{\,2}\, \underline{Z}^{\,2}$. For the numerator, since $P \leq 1$ and $Z_0 \leq 1$,

$$\left| (P_0 - P)Z_0 + P(Z_0 - Z) \right| \leq |P_0 - P| + |Z_0 - Z|.$$

Using the shared-source structure of the bridges,

$$P_0 - P = p_t(i) - p_t^{\mathrm{cand}}(i) = t\,(\tilde{\pi}(i) - \tilde{\pi}_{\mathrm{cand}}(i)),$$

so $|P_0 - P| \leq |\tilde{\pi}(i) - \tilde{\pi}_{\mathrm{cand}}(i)| \leq d$ (pointwise dominated by L_1). Likewise, the 1-Lipschitz property applied coordinate-wise and summed gives

$$|Z_0 - Z| = \left| \sum_a (\dot{p}_t(a))_+ - \sum_a (\dot{p}_t^{\mathrm{cand}}(a))_+ \right| \leq \sum_a \left| (\dot{p}_t(a))_+ - (\dot{p}_t^{\mathrm{cand}}(a))_+ \right| \leq d.$$

Hence the numerator is bounded by $2d$. The prefactor satisfies $A_0 \leq 1$ and $\sum_{j \neq i} B_j^0 \leq Z_0 \leq 1$, so $A_0 \sum_{j \neq i} B_j^0 \leq 1$. Therefore,

$$\mathrm{(III)} \leq \frac{2d}{\underline{p}^{\,2}\, \underline{Z}^{\,2}}.$$

Step 4: Combining the three bounds. Adding the three contributions,

$$\sum_{j\neq i}\left|u_t^{*,\mathrm{cand}}(i\to j)-u_t^*(i\to j)\right|\le \frac{d}{\underline{p}}+\frac{d}{\underline{p}\,\underline{Z}}+\frac{2d}{\underline{p}^2\,\underline{Z}^2}.$$

Since $\underline{p},\underline{Z}\in(0,1]$,

$$\frac{1}{\underline{p}}\le\frac{1}{\underline{p}^2\,\underline{Z}^2},\qquad \frac{1}{\underline{p}\,\underline{Z}}\le\frac{1}{\underline{p}^2\,\underline{Z}^2}.$$

All three terms can therefore be upper bounded by the common denominator, yielding

$$\sum_{j\neq i}\left|u_t^{*,\mathrm{cand}}(i\to j)-u_t^*(i\to j)\right|\le\frac{4}{\underline{p}^2\,\underline{Z}^2}\,\|\tilde\pi_{\mathrm{cand}}-\tilde\pi\|_1.$$

This establishes the theorem with universal constant $C=4$. □

Remark C.2. The theorem isolates the error caused by restricting the terminal target policy to $\mathcal{A}_{\mathrm{cand}}$. When the candidate bridge also changes the source distribution from p_0 to p_0^{cand}, an additional term $\|p_0^{\mathrm{cand}}-p_0\|_1$ enters the bound through $\|\dot p_t^{\mathrm{cand}}-\dot p_t\|_1$.

Corollary C.3 (Uniform-in-Time Stability)**.** *Assume that the full and candidate bridges use the same source distribution p_0. Fix $\delta\in(0,1)$ and suppose that, for all $t\in[0,1-\delta]$,*

$$p_t(i),\ p_t^{\mathrm{cand}}(i)\ge\underline{p}>0,\qquad Z_t,\ Z_t^{\mathrm{cand}}\ge\underline{Z}>0.$$

Then, for every such t and every $i\in\mathcal{A}_{\mathrm{cand}}$,

$$\sum_{j\neq i}\left|u_t^{*,\mathrm{cand}}(i\to j)-u_t^*(i\to j)\right|\le\frac{C}{\underline{p}^2\,\underline{Z}^2}\,\|\tilde\pi_{\mathrm{cand}}-\tilde\pi\|_1,$$

where $C>0$ is a universal constant.

If $\underline{p}$ and $\underline{Z}$ are deterministic constants, then combining this bound with Proposition 4.1 gives

$$\mathbb{E}\left[\sum_{j\neq i}\left|u_t^{*,\mathrm{cand}}(i\to j)-u_t^*(i\to j)\right|\right]\le\frac{2C}{\underline{p}^2\,\underline{Z}^2}\sum_{a\in\mathcal{A}}\tilde\pi(a\mid s)(1-\pi_{\mathrm{ref}}(a\mid s))^{N_{\mathrm{roll}}}(1-1/K)^{N_{\mathrm{rand}}}.$$

Remark C.4. Using a uniform prior p_0 and truncating the flow at $t\le 1-\delta$ ensures $p_t(i)\ge\delta/|\mathcal{A}|$, providing a practical lower bound for $\underline{p}$.

D Additional Macro-Action MinAtar Experiments

D.1 Construction and protocol

A macro of length K maps to a K-tuple of primitive actions via base-6 decomposition and executes them in sequence, accumulating reward. The agent observes only the state after the last one. If the episode ends part-way through, the remaining primitives are not executed. All methods use semi-Markov targets, so the reward is $\sum_{t<n}\gamma^t r_t$ for the n frames executed and the bootstrap is discounted by γ^n. Episodes are capped at 5,000 primitive frames, which is treated as termination for bootstrapping. The cap is recorded so that truncations can be distinguished offline.

The offline dataset for each (game, K) contains 100K macro transitions. A behaviour double DQN is trained online on the *primitive* action set with ε annealed from 1.0 to 0.1 over the first half of collection, and its action stream is grouped into consecutive K-macros within each episode. A chunk that ends early is kept and padded with no-ops to length K. This is the correct semi-Markov transition for the padded macro and preserves terminal transitions, which dropping incomplete chunks would remove. The dataset is generated once per (game, K) from a fixed seed and shared by every method and seed.

D.2 RESULTS AT K=8

At $K = 8$, the action space grows to approximately 1.68 million actions. DRIFT achieves the highest offline-to-online macro-action returns on Asterix, Freeway, and Seaquest, while RL-D^2-O2O and Q-chunking obtain slightly higher scores on Breakout and Space Invaders, respectively. Offline initialization substantially improves DRIFT over its online-only counterpart across all five games, including Freeway (26.0 versus 7.2) and Space Invaders (18.1 versus 2.8). However, primitive-action methods remain stronger on several games, highlighting the task-dependent benefits of temporal abstraction. Primitive-action DQN (Mnih et al., 2015), IMPALA, PPO (Schulman et al., 2017) and R2D2 (Kapturowski et al., 2018) measure the value of temporal abstraction itself.

Table 3: Macro-action MinAtar at K=8 ($|\mathcal{A}|$=1,679,616), same format as Table 1. MASP is omitted because its similarity matrix would have $|\mathcal{A}|^2 \approx 2.8 \times 10^{12}$ entries.

Method	Asterix	Breakout	Freeway	Seaquest	Space Invaders
Macro-action, offline-to-online					
DRIFT (ours)	0.6 → **2.0±0.1**	0.9 → 9.7±0.4	19.4 → **26.0±0.8**	0.5 → **1.9±0.1**	5.2 → 18.1±0.9
RL-D^2-O2O	0.5 → 1.1±0.0	0.8 → **9.8±0.9**	7.6 → 24.7±0.5	0.2 → 0.8±0.1	4.2 → 18.2±1.2
Q-chunking	0.6 → 0.7±0.0	0.6 → 9.6±2.0	25.1 → 18.3±1.5	0.7 → 0.6±0.1	5.1 → **19.4±0.6**
QDFM	0.8 → 1.0±0.2	0.0 → 2.8±0.1	21.6 → 19.8±3.2	0.6 → 1.3±0.2	3.0 → 15.8±3.1
BraVE	0.4 → 0.5±0.1	0.0 → 1.8±1.8	5.0 → 16.2±3.6	0.0 → 0.4±0.1	2.1 → 9.2±7.8
Cal-QL	0.4 → 0.7±0.1	0.6 → 0.3±0.3	9.6 → 3.2±0.2	0.3 → 0.0±0.0	2.6 → 1.8±1.0
IQL	0.3 → 0.5±0.1	0.3 → 0.7±0.1	0.1 → 0.1±0.1	0.0 → 0.1±0.0	4.6 → 5.6±0.9
SPA	0.5 → 0.6±0.1	0.6 → 1.0±0.2	13.6 → 20.7±0.2	0.2 → 0.4±0.1	3.8 → 5.2±0.5
Macro-action, online only					
DRIFT, no offline data	0.9±0.1	3.6±0.0	7.2±0.3	0.5±0.0	2.8±0.4
RL-D^2	0.8±0.3	4.5±0.1	10.8±10.8	0.8±0.0	23.9±1.7
DQN-Macro	0.6±0.1	7.4±1.4	0.9±0.9	0.3±0.0	4.5±2.0
IMPALA-Macro	0.2±0.1	3.2±0.1	8.0±7.6	0.7±0.1	16.2±1.4
Primitive actions (reference)					
DQN	0.6±0.2	24.7±4.4	4.0±0.7	2.5±0.2	44.1±18.2
IMPALA	0.9±0.2	4.6±0.4	22.3±1.3	1.9±0.1	28.0±2.3
PPO	2.2±0.2	5.3±0.1	19.8±19.8	2.6±1.2	40.6±3.1
R2D2	0.6±0.2	25.2±0.4	12.4±0.9	1.4±0.1	23.4±2.9

D.3 LEARNING CURVES AT K=4

Figure 3 shows that performance varies across games. Q-chunking leads on Breakout, while DRIFT achieves the highest final return on Freeway. On Space Invaders, RL-D^2-O2O overtakes DRIFT during online fine-tuning and achieves a higher final return. Overall, DRIFT improves throughout training, although no method consistently dominates across all three games.

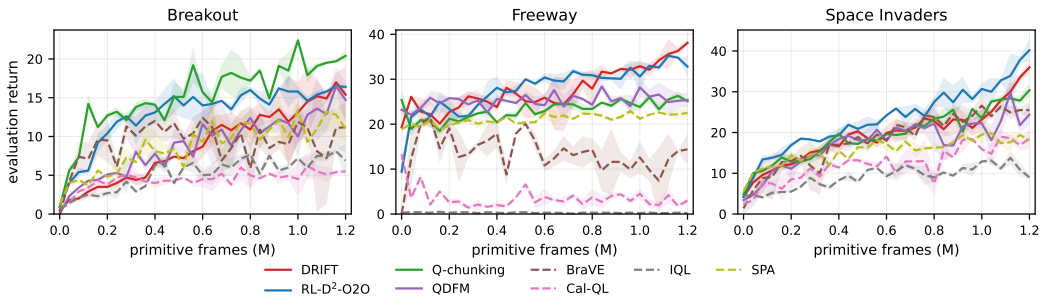


Figure 3: Offline-to-online learning curves at K=4 on the three games that separate methods. The first point of each curve is the score after offline pre-training, and shading spans the five seeds.

D.4 CANDIDATE COVERAGE

For every DRIFT run we log the mean candidate-set size, the fraction of fresh reference-generator rollouts whose terminal action lies inside the candidate set, and, where the action space is enumerable (K=4), the exact mass that the candidate set captures of the full-space target $\tilde{\pi} \propto \exp(\bar{A}/\beta)$. Table 4 reports these against the budget on Freeway.

Table 4: Candidate budget on Freeway at K=4, α=0.01, five seeds. Target mass is the exact probability of the candidate set under the full-space target, and $B/|\mathcal{A}|$ is what uniformly random candidates would capture.

B	Final score	Target mass	$B/\lvert\mathcal{A}\rvert$	Mean $\lvert\mathcal{A}_{\text{cand}}\rvert$
32	36.9±0.2	0.15	0.025	32
64	38.1±0.5	0.26	0.049	64
128	38.6±1.6	0.33	0.099	128
256	39.6±1.0	0.38	0.198	256

Although the candidate set captures only 26% of the measured full-action target mass at the default budget of 64, DRIFT achieves a Freeway return of 38.1. Increasing the budget to 256 raises the captured mass to 38% and the return to 39.6. These results suggest that effective control does not necessarily require closely approximating the entire full-action target distribution.

D.5 ABLATION STUDIES

We examine the effects of path-space regularization, candidate-set construction, and candidate budget at $K = 4$.

Table 5: DRIFT ablations at K=4, final score as mean ± std over 5 seeds. The default uses α=0.01 and 64 candidates from reference-generator rollouts. Reference-free candidates keep the dataset action and fill the set uniformly.

Component	Setting	Freeway	Space Invaders
Path-space KL	DRIFT, α=0	25.9±0.9	20.8±3.9
	DRIFT, α=0.001	28.9±1.9	23.4±0.7
	DRIFT, α=0.01 (default)	38.1±0.5	36.0±1.7
	DRIFT, α=0.03	29.5±0.9	27.1±0.6
	DRIFT, α=0.1	26.8±0.6	26.0±1.1
	DRIFT, α=0.3	26.8±0.3	21.2±0.1
Candidate set	Rollout candidates, B=64 (default)	38.1±0.5	36.0±1.7
	Full action space (all 1,296)	41.0±0.2	24.6±1.5
	Reference-free candidates	11.1±1.6	8.0±0.6
Budget B	B=32	36.9±0.2	33.5±0.3
	B=64	38.1±0.5	36.0±1.7
	B=128	38.6±1.6	41.0±2.9
	B=256	39.6±1.0	43.6±1.8

Results. Table 5 shows that $\alpha = 0.01$ achieves the highest returns among the tested regularization weights on both games. Removing reference-guided candidates substantially reduces performance, while increasing the candidate budget consistently improves returns. Full-action enumeration improves Freeway but performs worse on Space Invaders, demonstrating that evaluating more actions does not necessarily improve policy performance.

D.6 RUNTIME AND SCALING.

Table 6 compares methods profiled on the same H200 GPU with a fixed budget of 1.2M primitive frames. Although DRIFT is more expensive than RL-D^2-O2O, its measured runtime does not increase as sharply with macro-action space size as that of DQN-Macro. Longer macros also require fewer decisions under a fixed primitive-frame budget.

Table 6: Median training time in hours on H200 GPUs, over five games and seeds.

Method	$K = 4$	$K = 8$
DRIFT	3.2	1.9
RL-D^2-O2O	1.8	1.4
DQN-Macro	0.1	0.5

Baselines. BraVE is ported from the authors' code with their default hyperparameters, using a prefix tree over macro positions in place of their binary sub-action tree, and the shared discount and semi-Markov targets. Cal-QL uses the original Monte-Carlo calibration, with Monte-Carlo returns for online data filled in when each episode ends. SPA's actor weighting follows the published form. Its proficiency-adaptive offline replay fraction is replaced by the shared fixed fraction. Q-chunking uses the masked discrete-diffusion behaviour model as its chunk policy, with best-of-32 selection for acting and for the TD backup. MASP is Rainbow-lite (double, duelling, n-step, prioritised replay, ε-greedy) rather than full Rainbow.

Compute. Runs were distributed over H200, A100, V100 and P100 GPUs as availability allowed. The runtime table reports only the runs on a single GPU type, since wall-clock is not comparable across them.

D.7 Macro-Action MinAtar ($|\mathcal{A}| = 216$)

Construction. Each macro action $m \in \{0, \ldots, 215\}$ maps to a unique 3-tuple (a_1, a_2, a_3) via base-6 decomposition. Executing m runs the three base actions sequentially in the environment, accumulating reward. The agent observes only the final state after all three steps. If the episode terminates mid-sequence the cumulative reward up to termination is returned.

This construction expands the action space from 6 to $6^3 = 216$ while preserving the original game dynamics. DRIFT operates on candidate sets of budget 40 (18.5% coverage of the macro-action space). The CTMC generator uses learned action embeddings (128 dimensions) with bilinear rate computation, replacing the one-hot input used for standard MinAtar. The Q-network output layer is $128 \to 216$.

Table 7: Macro-Action MinAtar ($|\mathcal{A}|$=216, k=3): mean online score, 5 seeds. DRIFT uses candidate budget 40 (18.5% coverage). Best per game in **bold**.

Game	**DRIFT**	DRIFT (full)	DQN	DQN-Sub	PPO
Breakout	**18.40**	17.80	11.98	7.45	5.83
Asterix	**0.78**	0.56	0.67	0.53	0.65
Freeway	5.56	5.21	0.66	0.50	**7.90**
Seaquest	**1.20**	0.85	0.52	0.98	0.66
Space Invaders	11.45	8.40	10.36	10.77	**14.25**
Average	**7.48**	6.56	4.84	4.04	5.85

Results. DRIFT has a higher average return than DQN and PPO across the five games here (Table 7). DRIFT with candidate sets (7.48 average) and DRIFT without candidate sets (6.56 average) perform similarly since the action space is relatively smaller.

Implications Overall our macro-action MinAtar results highlight that the candidate-set mechanism provides theoretical guarantees (Proposition 4.1, Theorem 4.2) and is validated in this benchmark, showing its practical benefit when the CTMC generator is well-initialised even with 216 actions and only 8 states per actor update.

E ADDITIONAL JERICHO EXPERIMENTS

E.1 JERICHO WITH NO-ORACLE DETAILS

Templates come from Jericho's template generator for each game, extended with the 12 compass and movement commands where absent. Objects are words of the observation, the response to `look` and the response to `inventory` that the game dictionary does not mark as verbs, directions, prepositions or meta words, capped at 24 per state. Episodes last at most 200 steps, and reaching this cap is not treated as termination when bootstrapping. The language-model proposers condition on the previous action, the current observation, the room description and the inventory, which differs from the original context of the previous observation, the previous action and the current observation. Jericho's admissible-action computation uses spaCy for object extraction, which we replace with a lookup in the game dictionary. Jericho still filters every candidate by simulating it, so the admissible lists stay exact up to the objects considered.

Table 8: Valid-command rate and admissible recall@B at B=128, mean over five seeds. The valid-command rate is the fraction of online training steps whose command the game accepted, meaning the score changed, the episode ended, or the world state changed. Recall is measured after the fact with the oracle on up to 25 states per evaluation visited by that method's own policy. *CALM and CSM candidate sets always contain the 12 compass and movement commands, which inflates their recall.

Method	Metric	Zork1	Deephome	Detective	Ludicorp	Pentari
DRIFT (ours)	valid	0.262	0.023	0.392	0.107	0.058
	recall	0.151	0.046	0.395	0.373	0.158
DRRN-Struct	valid	0.014	0.005	0.168	0.010	0.028
	recall	0.104	0.000	0.606	0.255	0.150
CALM	valid	0.116	0.000	0.366	0.084	0.046
	recall*	0.717	0.000	0.793	0.990	0.387
CSM	valid	0.116	0.000	0.506	0.118	0.046
	recall*	0.717	0.000	0.942	0.843	0.387

Table 9: Candidate budget for DRIFT on Jericho, separate run with five seeds per cell.

Game	Metric	B=32	B=64	B=128	B=256
Zork1	score	15.0±3.5	15.5±4.5	18.5±1.5	18.8±1.2
	recall	0.066	0.076	0.151	0.222
Detective	score	34.2±0.2	40.5±2.0	42.8±5.2	51.2±6.2
	recall	0.123	0.250	0.395	0.519

Runtime. Under matched profiling conditions, DRIFT has a median runtime of 3.2 hours, compared with 6.4 hours for DRRN-Struct, 5.1 hours for CALM-style action generation, and 6.8 hours for CSM-style action generation. These measurements cover five games and seeds.

E.2 JERICHO WITH THE ADMISSIBLE-ACTION ORACLE.

We initially evaluated DRIFT on ten text-based games from the Jericho benchmark (Hausknecht et al., 2020) to test whether the offline-to-online fine-tuning framework transfers to a qualitatively different domain. In these games the agent receives textual observations and must select a text command from a variable-size set of valid actions at each step.

Results. DRIFT achieves the highest average normalized score (23.2%) across all ten games, outperforming every baseline including the results for CALM (12.6%) and KG-A2C (19.2%). This is notable because CALM uses a fine-tuned GPT-2 language model for action generation while DRIFT uses a simple GRU encoder. DRIFT achieves its strongest gains on games with deeper puzzle chains. On Deephome it scores 35.8 compared to 6.0 for almost every other baseline, a 6× improvement. On Detective it solves the majority of the game (291.5 out of 360) and on Ludicorp it more than doubles

Table 10: Jericho text-game results (raw score, single seed, 300K steps). Best score per game in **bold** among our methods. †Published scores from Yao et al. (2020).

	Our experiments					*Prior work*†	
Game (max)	**DRIFT**	DRRN	DQN	IQL	AWAC	CALM	KG-A2C
Deephome (300)	**35.8**	6.0	6.0	6.0	3.5	1.0	1.0
Detective (360)	**291.5**	50.0	70.0	50.0	66.5	289.7	207.9
Enchanter (400)	**20.0**	**20.0**	**20.0**	**20.0**	17.0	19.1	12.1
Ludicorp (150)	**13.8**	6.0	6.0	6.0	11.4	10.1	17.8
Omniquest (50)	8.0	**10.0**	**10.0**	5.0	5.5	6.9	3.0
Pentari (70)	**26.8**	25.0	25.0	25.0	11.75	0.0	50.7
Temple (35)	**8.0**	5.0	**8.0**	**8.0**	7.4	0.0	7.6
Zork1 (350)	25.0	**35.0**	25.0	25.0	1.25	30.4	34.0
Zork3 (7)	2.5	**3.0**	0.0	0.0	0.0	0.5	0.0
Ztuu (100)	**5.0**	**5.0**	**5.0**	**5.0**	2.0	3.7	9.2
Avg. norm. (%)	**23.2**	15.3	12.1	10.6	8.3	12.6	19.2

DQN, DRRN, and IQL while exceeding the published CALM score. However, DRRN outperforms DRIFT on Zork1 (35.0 vs 25.0) and Omniquest (10.0 vs 8.0), and four games show floor effects where most methods converge to the same score within 300K steps. Path-space regularization helps prevent forgetting in sparse-reward settings by anchoring the fine-tuned generator to the pretrained reference, allowing DRIFT to preserve successful strategies while exploring and notably, it is one of the only two methods with a positive score on every game.

Environment and Action Space. Jericho (Hausknecht et al., 2020) provides a Python interface to Z-machine interactive fiction games. At each step the agent receives a textual observation and issues a text command from a set of valid actions identified by the game engine via `get_valid_actions()`. The number of valid actions per state ranges from 3 to 10 on average across our ten evaluation games. This is smaller than the 30–200 range reported in prior work (Yao et al., 2020; Ammanabrolu & Hausknecht, 2020) because those methods augment the action space with language model or template-based generation, while our setup uses only the built-in parser output.

Architecture. All methods share the same text-encoding architecture, adapted from DRRN (He et al., 2016). Words are tokenised via hash-based indexing (vocabulary size 5003, using MD5 hashing) and embedded into 64 dimensions. A single-layer GRU with hidden size 128 produces fixed-size representations for both observations and actions. Each network (Q1, Q2, V, generator) maintains its own GRU encoder parameters.

The Q-network scores each state action pair via $Q(s, a) = \mathrm{MLP}([\mathbf{h}_s; \mathbf{h}_a])$ with one hidden layer of 128 units and ReLU activation, naturally handling variable-size action sets by scoring each pair independently. The value network computes $V(s) = \mathrm{MLP}(\mathbf{h}_s)$ with the same hidden layer structure.

The CTMC generator uses a bilinear-attention design

$$u_\theta(i \to j, t \mid s) = \mathrm{softplus}\big(\mathbf{c}(\mathbf{h}_s, \mathbf{h}_{a_i}, t) \cdot \mathbf{h}_{a_j}\big),$$

where the context vector $\mathbf{c}$ is produced by an MLP from the concatenation of the observation embedding, source action embedding, and scalar flow time t. Rates to each target action are computed via dot products with target action embeddings. Self-transition rates ($j = i$) are masked to zero. This design handles arbitrary action set sizes without a fixed output head.

Candidate Set Construction. In practice, the valid action sets in our ten Jericho games average 3 to 9 actions per state, which falls below the candidate budget. Therefore, in this case our method uses all valid actions directly without subsampling. The candidate-set mechanism thus operates in its full-coverage regime ($\varepsilon(s) = 0$ in Proposition 4.1), and thus Jericho with the Admissible-Action Oracle primarily shows DRIFT's advantage from the path-space KL regularization and the flow-matching actor update rather than from action-space reduction.

Offline Data Collection. For each game, we collect 200 episodes of offline data using an ε-greedy DQN that trains online during collection. The exploration rate ε is annealed from 1.0 to 0.1 over the first half of the collection phase. The resulting transitions are cached to disk and reused across all methods to ensure identical offline data.

Training Protocol. All methods receive the same offline dataset per game and run for 300,000 online interaction steps with a maximum episode length of 200 steps. Critic pre-training runs for 5,000 gradient steps and generator pre-training runs for 2,000 gradient steps, both on the offline dataset. Evaluation uses 20 complete episodes at the end of training. Results are reported for a single seed (42) due to the high computational cost of text-game experiments.

Comparison with Prior Work. The CALM (Yao et al., 2020) and KG-A2C (Ammanabrolu & Hausknecht, 2020) scores in Table 10 are taken from published results under different experimental conditions.

CALM uses a fine-tuned GPT-2 language model pre-trained on 426 human gameplay transcripts from ClubFloyd to generate action candidates. It does not use the valid-action handicap provided by the game engine. KG-A2C builds a dynamic knowledge graph from game text using Stanford OpenIE, encodes it with a graph neural network, and trains with A2C. It uses the valid-action handicap and template-based action generation. Our methods (DRIFT and all baselines) use the valid-action handicap, GRU text encoders without any pre-trained language model, and 300K online steps. This is a simpler and more self-contained setup that ensures fair comparison among the evaluated methods.

Other recent work on Jericho includes XTX (Tuyls et al., 2022) (ICLR 2022 Spotlight) which achieves state-of-the-art scores via multi-stage episodic control (for example 103 on Zork1) and CSM (Shi et al., 2023) which improves CALM via confidence-based self-imitation. These methods use substantially more compute and game-specific engineering. We report their existence for context rather than direct comparison.

Compute. These Jericho experiments ran on a cluster with NVIDIA Tesla P4 GPUs (12 GB VRAM), 4 CPU cores, and 16 GB RAM per job. Table 11 reports per-game runtimes. The Jericho with the Admissible-Action Oracle (10 games $\times$ 5 methods $=$ 50 runs) consumed approximately 850 GPU-hours.

Table 11: Mean wall-clock runtime per Jericho run (one game, one method, seed 42).

Game	DRIFT	DRRN	DQN	IQL	AWAC
Deephome	40.0 h	34.1 h	32.4 h	31.3 h	30.4 h
Detective	16.9 h	14.5 h	14.0 h	11.1 h	11.4 h
Enchanter	16.6 h	18.5 h	15.7 h	12.1 h	11.6 h
Ludicorp	19.6 h	18.7 h	19.1 h	13.9 h	15.7 h
Omniquest	37.6 h	17.6 h	19.2 h	38.8 h	23.3 h
Pentari	15.2 h	14.2 h	15.0 h	10.0 h	13.1 h
Temple	37.3 h	38.4 h	35.0 h	34.5 h	33.7 h
Zork1	17.7 h	12.2 h	13.2 h	9.9 h	2.5 h
Zork3	13.7 h	10.7 h	10.4 h	7.1 h	8.2 h
Ztuu	16.5 h	33.3 h	51.7 h	10.8 h	13.0 h

F Additional Small-Action Benchmarks

F.1 MinAtar

We also evaluated DRIFT against nine offline-to-online baselines on MinAtar (Young & Tian, 2019) using the same offline dataset and 300K online steps.

Results. As shown in Table 12, DRIFT improves on all five games, achieves the best score on Breakout, Asterix, and Seaquest, and remains competitive on Freeway and Space Invaders. Unlike AWAC and SPA, which collapse to offline scores on four games, DRIFT avoids catastrophic forgetting

Table 12: MinAtar offline→online results (mean over 5 seeds). Best online score per game in **bold**. †Methods that fail to improve over their offline score.

Game	**DRIFT**	Cal-QL	CQL	IQL	AWAC	DQN	Rainbow	PPO	PEX	SPA
Breakout	0.43→**17.10**	1.19→14.61	0.62→14.69	0.01→17.01	0.50→0.51†	0.38→14.25	0.29→12.48	—→0.52	0.32→16.73	0.19→0.24†
Asterix	0.50→**1.17**	0.67→1.04	0.58→1.15	0.66→1.13	0.38→0.53†	0.55→1.06	0.55→0.91	—→0.50	0.53→1.05	0.44→0.53†
Freeway	1.26→25.00	19.89→25.71	22.97→**26.50**	2.54→23.56	15.85→15.69†	2.99→25.38	0.31→24.57	—→0.13	1.89→22.98	0.95→0.76†
Seaquest	0.15→**1.84**	0.46→0.86	0.42→0.95	0.12→0.90	0.28→0.20†	0.24→0.89	0.38→0.58	—→0.13	0.37→1.00	0.33→0.35†
Space Inv.	2.23→42.64	0.27→**54.09**	0.00→52.56	0.00→41.60	4.52→4.52†	0.00→49.04	0.19→35.52	—→2.96	0.00→48.58	0.00→0.00†
Average	0.91→17.55	4.50→19.26	4.92→19.17	0.67→16.84	4.31→4.29	0.83→18.12	0.34→14.81	—→0.85	0.62→18.07	0.38→0.38

through path-space KL regularization to the pretrained reference. Although its generative overhead is less beneficial in this small-action regime, DRIFT's consistent gains show reliable fine-tuning.

Environment and Offline Data MinAtar (Young & Tian, 2019) implements five simplified Atari games on a 10×10 grid with binary feature channels: Breakout (3 actions), Asterix (5), Freeway (3), Seaquest (6), Space Invaders (4). Observations are $10\times10\times n_{\text{ch}}$ binary tensors (n_{ch}=4–10 depending on game); rewards are sparse integers.

Every method receives the *same* offline dataset per game: 100,000 transitions from a behavioural DQN with ε linearly annealed from 1.0 to 0.1 over the first 50,000 steps. The DQN trains online during collection ($\eta=3\times10^{-4}$, target soft-update $\tau=0.005$, 50K-transition replay buffer). The dataset is cached to disk and reused across all methods and seeds to guarantee identical offline data.

Network Architecture All methods share the same encoder: flatten the $10\times10\times n_{\text{ch}}$ observation, then $\text{Linear}(100\cdot n_{\text{ch}}, 256)\to\text{ReLU}\to\text{Linear}(256, 128)\to\text{ReLU}$, yielding a 128-dim representation. Q-networks add $\text{Linear}(128, |\mathcal{A}|)$; value networks add $\text{Linear}(128, 1)$.

The DRIFT generator (CTMCGen) appends a rate head: $\text{Linear}(128+|\mathcal{A}|+1, 128)\to\text{ReLU}\to\text{Linear}(128, |\mathcal{A}|)\to\text{Softplus}$. The input concatenates the encoded observation, a one-hot current-action vector, and the scalar flow time t. The diagonal is masked to zero (no self-transitions), yielding non-negative off-diagonal rates.

Training Protocol. Offline phase: Critics are trained for 2,000 gradient steps on the shared offline dataset. Method-specific offline components run in parallel: IQL runs 2,000 expectile regression steps; AWAC runs 1,000 actor-critic steps; DRIFT trains the CTMC generator for 1,000 epochs of advantage-weighted DFM. Each method is then evaluated over 100 episodes to obtain the offline score.

Online phase: Each method interacts with the environment for 300,000 steps. Every training minibatch mixes online and offline data: $(1-\rho)\cdot B$ transitions from the online replay buffer and $\rho\cdot B$ from the offline dataset. The final policy is evaluated over 100 episodes.

F.2 D4RL

We further evaluated on three MuJoCo locomotion tasks from D4RL (Hopper, Walker2d, and HalfCheetah) each with two dataset qualities (medium, expert). Continuous action spaces are discretised into $k=22$ clusters via k-means on the offline dataset and scores follow the standard D4RL normalisation.

Results. DRIFT improves over its offline initialisation on all six tasks, achieving the best online score on Hopper (both qualities), Walker-medium, and HalfCheetah-expert. The largest gains appear on Hopper, where DRIFT improves by $+40.8$ (medium) and $+41.7$ (expert) normalised points. DRIFT's online improvement magnitude ($+20.7$ avg) is much better than CQL ($+10.8$) and Cal-QL ($+11.7$).

Discretisation Procedure. We discretise the continuous action spaces of Hopper (3-dim), Walker2d (6-dim), and HalfCheetah (6-dim) using MiniBatchKMeans with $k=22$ clusters fitted on all actions in the offline dataset. Each continuous action is replaced by its nearest cluster index; during online interaction the agent selects a cluster index and executes the corresponding centroid. The median inter-centroid distance ranges from 0.27 (Hopper) to 0.46 (HalfCheetah), confirming reasonable but

Table 13: D4RL discretised results: *offline→online* normalised score (mean, 5 seeds). Best online score per task in **bold**. All methods use k-means discretisation (k=22).

Task	**DRIFT**	CQL	IQL	AWAC	Cal-QL	DQN	PPO	PEX	SPA
Hopper-med	7.1→**47.9**	28.8→35.8	0.4→27.9	26.0→26.7	28.8→44.1	0.4→23.7	—→3.1	0.4→43.1	0.4→20.4
Hopper-exp	5.4→**47.1**	8.8→29.3	0.1→17.2	11.5→11.8	8.8→25.7	0.3→21.7	—→3.7	0.1→39.5	0.3→20.3
Walker-med	2.0→**15.9**	2.5→12.6	0.3→13.0	4.2→5.8	2.5→11.7	1.8→12.3	—→7.2	0.3→11.8	1.8→11.8
Walker-exp	4.1→14.8	4.0→10.7	0.2→10.1	9.4→9.8	4.0→**12.4**	0.2→6.5	—→7.0	0.2→7.2	0.2→4.2
Cheetah-med	2.2→11.1	3.3→15.8	0.2→7.8	13.1→13.5	3.3→15.8	2.1→**16.9**	—→4.9	0.2→14.6	2.1→9.1
Cheetah-exp	1.5→**9.7**	0.7→8.6	0.2→6.8	1.1→1.2	0.7→8.5	0.1→4.7	—→5.9	0.2→7.7	0.1→7.1
Average	**3.7→24.4**	8.0→18.8	0.2→13.8	10.9→11.5	8.0→19.7	0.8→14.3	—→5.3	0.1→20.6	0.8→12.2

coarse coverage of the continuous action manifold. We use the Minari versions of the D4RL datasets (v0.5.3) with `medium` and `expert` qualities, each containing ≈1M transitions.

Training Protocol. All methods share the same budget: 1M offline gradient steps for critic pre-training, followed by 300k online environment steps. The critic is a 2-layer MLP (256 hidden units, batch size 256, Adam η=3×10^{-4}). DRIFT's generator uses the same architecture with actor batch size 8 and 500K generator pre-training steps. The online phase uses a mixed replay buffer with 50% offline data and 200K capacity.

Limitations of the Discretised Setting D4RL was designed for continuous-action algorithms. Discretisation via k-means introduces an approximation ceiling that affects all methods equally, but also obscures each method's native strengths. With $|\mathcal{A}|$=22, the full action space can be enumerated cheaply, removing DRIFT's candidate-set advantage entirely. However, even in this setting our method DRIFT shows good performance and stable offline to online improvements. DRIFT's structural benefits are designed for larger action spaces, as validated by the macro-action MinAtar and Jericho results.

G Broader Impact

This paper presents a reinforcement learning algorithm for discrete action spaces. The method is evaluated on game benchmarks and does not target any specific real-world deployment. We discuss potential positive and negative implications below.

On the positive side, improving offline-to-online RL reduces the amount of costly or risky online interaction needed to train effective policies. This is valuable in domains where exploration is expensive or dangerous, such as robotics, healthcare, and autonomous systems. The path-space KL regularization specifically addresses catastrophic forgetting, which is a safety-relevant failure mode in deployed systems that must be updated without losing previously learned safe behaviors.

On the negative side, more capable RL agents could be applied in domains with societal risks. Recommendation systems with large discrete action spaces (a motivating application for candidate sets) can amplify filter bubbles or manipulate user behavior if reward signals are misaligned with user welfare. Autonomous agents in text-based interfaces could be adapted for automated social engineering or spam generation, though our GRU-based agents are far below the capability threshold for such misuse. More broadly, any improvement in RL sample efficiency lowers the barrier to training agents in sensitive domains without adequate safety testing.

We believe the benefits of publishing this work outweigh the risks. The method is general-purpose and not designed for any harmful application. The benchmarks we use (MinAtar, Jericho text games, discretised MuJoCo) are standard academic testbeds with no direct path to misuse. We release our code to support reproducibility and enable the research community to build on and scrutinize our results.

H Full Training Algorithm

Algorithm 3 provides the complete training procedure with all implementation details.

Algorithm 3 DRIFT: Full Training Procedure

Require: Replay source $\mathcal{D}_{\text{off}}$, environment, discount γ, temperature β, KL weight α, soft-update rate τ, advantage clip c, smoothing ϵ, CTMC sub-steps M, candidate sizes $N_{\text{roll}}, N_{\text{rand}}$, reference refresh interval K, offline mix fraction ρ, learning rates $\eta_Q, \eta_V, \eta_\theta$, flow-time truncation $\delta \in (0, 1)$

Initialize critics Q_{ϕ_1}, Q_{ϕ_2}, value V_ψ, generator u_θ
Initialize target copies $Q_{\phi_k^-} \leftarrow Q_{\phi_k}$, $V_{\psi^-} \leftarrow V_\psi$
Initialize reference generator $u_{\text{ref}} \leftarrow u_\theta$
Initialize replay buffer $\mathcal{B}$ with a subset of $\mathcal{D}_{\text{off}}$
Observe initial state s
for $n = 1$ **to** N_{steps} **do**
  Environment interaction: simulate CTMC under u_θ from $X_0 \sim \text{Unif}(\mathcal{A})$ for M Euler sub-steps and execute terminal action $a = X_M$
  Observe (s, a, r, s', d), add it to $\mathcal{B}$, and set $s \leftarrow s'$ (reset if d)
  Mixed minibatch: sample $\lfloor (1-\rho)B \rfloor$ transitions from $\mathcal{B}$ and $\lfloor \rho B \rfloor$ transitions from $\mathcal{D}_{\text{off}}$
  Critic update: set $y \leftarrow r + \gamma V_{\psi^-}(s')(1-d)$ and update Q_{ϕ_k} toward y for $k = 1, 2$
  Value update: compute lagged advantage $\bar{A}^-$ and $\tilde{\pi}^-(a \mid s) \propto \exp(\bar{A}^-(s,a)/\beta)$ from $\{Q_{\phi_k^-}, V_{\psi^-}\}$, then update

$$V_\psi(s) \approx \text{sg}\left[\sum_a \tilde{\pi}^-(a \mid s) \min_k Q_{\phi_k^-}(s,a)\right]$$

  Actor update:
  for each sampled state s_b in the actor batch **do**
    Construct candidate set $\mathcal{A}_{\text{cand}}(s_b)$ using N_{roll} rollouts of frozen u_{ref} and N_{rand} uniform actions
    Define

$$\pi_{\text{ref}}(a \mid s_b) \propto \frac{\text{count}(a)}{N_{\text{roll}}} + \frac{\epsilon}{|\mathcal{A}_{\text{cand}}(s_b)|}, \qquad a \in \mathcal{A}_{\text{cand}}(s_b)$$

    Compute clipped advantage $\bar{A}(s_b, \cdot)$ from $\{Q_{\phi_k}, V_{\psi^-}\}$ over $\mathcal{A}_{\text{cand}}(s_b)$ and set

$$\tilde{\pi}(a \mid s_b) = \frac{\pi_{\text{ref}}(a \mid s_b) \exp(\bar{A}(s_b, a)/\beta)}{\sum_{a' \in \mathcal{A}_{\text{cand}}(s_b)} \pi_{\text{ref}}(a' \mid s_b) \exp(\bar{A}(s_b, a')/\beta)}$$

    Define the bridge: $p_t(a \mid s_b) = (1-t)\, p_0(a) + t\, \tilde{\pi}(a \mid s_b)$, where $p_0(a) = \text{Unif}(\mathcal{A})$.
    Sample $t \sim \text{Unif}(0, 1-\delta)$, $i \sim p_t(\cdot \mid s_b)$.
    Construct target generator $u_t^*(i \to \cdot \mid s_b)$ by independent coupling on $\mathcal{A}_{\text{cand}}(s_b)$
    Accumulate DFM loss

$$\ell_{\text{DFM}}(s_b) = \sum_{j \in \mathcal{A}_{\text{cand}}(s_b),\, j \neq i} \big(u_\theta(i \to j, t \mid s_b) - u_t^*(i \to j \mid s_b)\big)^2$$

    Accumulate path-space KL estimate $\widehat{\text{KL}}(s_b)$ using frozen u_{ref}
  end for
  Update θ using

$$\frac{1}{B}\sum_b \ell_{\text{DFM}}(s_b) + \alpha \frac{1}{B}\sum_b \widehat{\text{KL}}(s_b)$$

  Soft-update targets: $\phi_k^- \leftarrow (1-\tau)\phi_k^- + \tau\phi_k$, $\psi^- \leftarrow (1-\tau)\psi^- + \tau\psi$
  Refresh reference generator every K steps: if $n \bmod K = 0$, set $u_{\text{ref}} \leftarrow u_\theta$
end for
return Fine-tuned generator u_θ