Scalable Decision-Making in Stochastic Environments through Learned Temporal Abstraction

Baiting Luo[1], Abhishek Dubey[1], Ayan Mukhopadhyay[1]
[1]Vanderbilt University
{baiting.luo, abhishek.dubey, ayan.mukhopadhyay}@vanderbilt.edu

Abstract

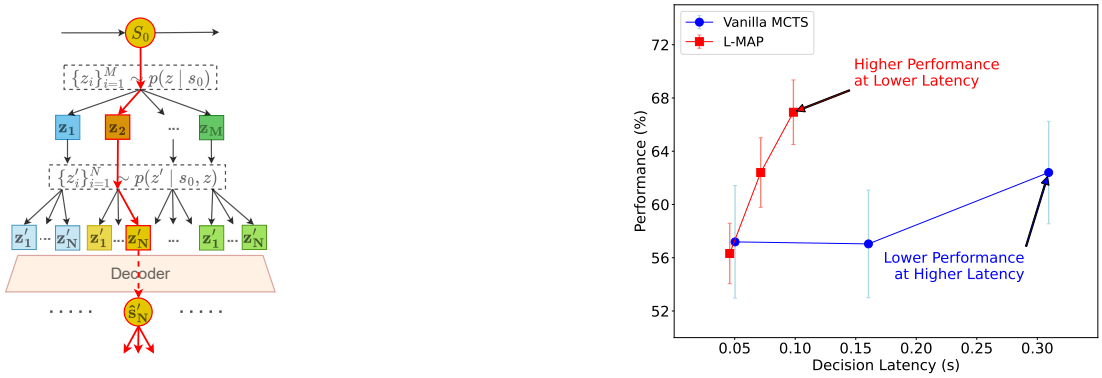
Sequential decision-making for embodied agents in high-dimensional continuous action spaces, particularly in stochastic environments, poses significant computational challenges. We explore this challenge in the traditional offline RL setting, where an agent must learn how to make decisions based on data collected through a stochastic behavior policy. We present Latent Macro Action Planner (L-MAP), which addresses this challenge by learning a set of temporally extended macro-actions through a state-conditional Vector Quantized Variational Autoencoder (VQ-VAE), effectively reducing action dimensionality. L-MAP employs a (separate) learned prior model that acts as a latent transition model and allows efficient sampling of plausible actions. During planning, our approach accounts for stochasticity in both the environment and the behavior policy by using Monte Carlo tree search (MCTS). In offline RL settings, including stochastic continuous control tasks, L-MAP efficiently searches over discrete latent actions to yield high expected returns. Empirical results demonstrate that L-MAP maintains low decision latency despite increased action dimensionality. Notably, across tasks ranging from continuous control with inherently stochastic dynamics to high-dimensional robotic hand manipulation, L-MAP significantly outperforms existing model-based methods and performs on-par with strong model-free actor-critic baselines, highlighting the effectiveness of the proposed approach in planning in complex and stochastic environments with high-dimensional action spaces.

1. Introduction

Planning-based reinforcement learning (RL) has achieved remarkable success in settings with discrete, low-dimensional actions such as chess, Go, Atari, and in simulated continuous-control benchmarks [15, 27–29, 33]. However, extending these methods to embodied agents operating with high-dimensional continuous action spaces, especially in stochastic environments, presents significant challenges. For instance, legged robots navigating outdoor terrain must adapt their locomotion policies in real time to handle disturbances like wind gusts or slippery surfaces, while drones executing delivery tasks face dynamically shifting weather conditions that alter aerodynamic constraints [6].

In this paper, we posit that planning in such challenging settings could greatly benefit from temporal abstractions, i.e., representations of multi-step primitive behaviors such as macro actions [5, 9, 30]. By leveraging these abstractions, planners can navigate high-dimensional continuous action spaces more efficiently, potentially mitigating the curse of dimensionality and reducing decision-making latency in stochastic environments. This paper considers the standard setting where an agent can access a set of trajectories (i.e., a sequence of state, action, and reward traces) collected through a fixed behavior policy. Given this setting, we propose the *Latent Macro Action Planner* (L-MAP), which constructs a lower dimensional representation of temporally extended primitive actions by using a state-conditioned Vector Quantized Variational AutoEncoder (VQ-VAE) [32]. The encoder integrates the current state and macro-action to generate a discrete latent code. Subsequently, a sequential model (in our case, a Transformer) is employed to autoregressively model the distribution of these latent codes, conditioned on the current state (and the behavior policy). This Transformer facilitates a two-step inference process: initially, given a state, it enables the sampling of probable latent macro-actions under the behavior policy, effectively acting as a prior policy. Subsequently, conditioned on both the state and the sampled macro-action, it generates subsequent latent codes that encapsulate information about expected returns and potential next states. This dual functionality of the Transformer enables efficient exploration of promising action trajectories while forming a compact representation of the plausible trajectories during planning.

As shown in Fig.1a, leveraging these models, we build a latent search space that serves as a structured initializa-



(a) Planning with pre-constructed search space.

(b) Decision latency vs. performance.

Figure 1. (a) Overview of planning over the pre-constructed search space. (b) As MCTS iterations increase (10,50,100), pre-constructing the search space reduces decision latency for a given performance.

tion for planning, encapsulating likely trajectories based on the learned environment dynamics. To address stochasticity and optimize decision-making, we integrate Monte Carlo Tree Search (MCTS) with progressive widening to efficiently navigate this latent space. Initially, the search concentrates on the prebuilt latent space, facilitating rapid decision-making grounded in learned abstractions. If additional computation time becomes available, we progressively widen the search tree to extend the search beyond the prebuilt latent space incrementally. This dynamic expansion strategy enables our method to balance rapid planning using learned abstractions with more exhaustive exploration when computational resources permit. As shown in Fig.1b, this strategy achieves better performance with lower decision latency compared to planning with vanilla MCTS. Upon selecting a latent macro-action, we operate in a *polling control* mode [12, 14] wherein MCTS returns only the first primitive action of the recommended macro-action. This approach allows for recovery from locally suboptimal decisions by performing planning at each time step.

We evaluate L-MAP extensively in the offline RL setting across a diverse range of tasks. In stochastic MuJoCo environments [26], L-MAP consistently outperforms both model-based baselines like Trajectory Transformer (TT) [16] and Trajectory Autoencoding Planner (TAP) [17], as well as model-free methods such as Conservative Q-Learning (CQL) [21] and Implicit Q-Learning (IQL) [20]. This demonstrates L-MAP's robust capability in handling stochastic dynamics. For deterministic MuJoCo tasks, L-MAP shows comparable or superior performance to these baselines, highlighting that our planning approach effectively accounts for stochasticity in the behavior policy, leading to competitive performance even in deterministic environments. Notably, L-MAP scales effectively to high-dimensional tasks, as evidenced by its strong performance on the challenging Adroit hand manipulation tasks. Furthermore, L-MAP's use of temporal abstraction enables lower latency decision-making compared to methods like TT. These results underscore L-MAP's versatility and effectiveness across various types of control problems, from stochastic to deterministic environments, and from low to high-dimensional action spaces.

2. Preliminaries

We consider a continuous state and action space Markov Decision Process (MDP) defined by $\{\mathcal{S}, \mathcal{A}, P, r\}$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^l$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. To manage the complexity of these continuous spaces, we introduce macro actions, which are fixed-length sequences of primitive actions. A macro action $m \in \mathcal{M}$ is defined as $m = \langle a_t, \ldots, a_{t+L-1} \rangle$, where each $a_i \in \mathcal{A}$ and L is the length of the macro action. Our goal is to compute an optimal macro-level policy $\pi^* : \mathcal{S} \to \mathcal{P}_m$ that maximizes the expected discounted return $\mathbb{E}_\pi \left[R(s, \pi(s)) \right]$.

Trajectory Representation: Consider a trajectory τ of length $T = \kappa \cdot L$ ($\kappa \in \mathbb{N}^+$), which is composed of a sequence of states $s_t \in \mathcal{S}$, fixed-size macro actions $m_t \in \mathcal{M}$, and corresponding return-to-go estimates $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, formally represented as $\tau = (R_1, s_1, m_1, R_{L+1}, s_{L+1}, m_{L+1}, \ldots, R_{(\kappa-1)L+1}, s_{(\kappa-1)L+1}, m_{(\kappa-1)L+1})$.

3. Method

Planning in continuous action space is hard and computationally challenging, and full enumeration of all possible actions is infeasible. Discretizing the action space is one way to address this challenge, but in practice, enumerating a large set of discrete actions can also be challenging, particularly for online approaches. Sample-based methods offer an

efficient approach for handling large and complex domains. These methods sample a subset of actions rather than exhaustively enumerating all possibilities, reducing computational costs while computing optimal policies or value functions [15]. Building on these insights, we propose the Latent Macro Action Planner (L-MAP), which learns temporal abstractions in the form of macro-actions and plans using a latent transition model that serves as both a prior policy and a transition model.

3.1. Discretizing State-Macro Action Sequences with VQ-VAE

A key insight from prior work is that a learned state-conditioned discretization can be used to construct a discretization scheme with relatively few discrete actions while maintaining high granularity [17, 23]. As shown in Fig.2, our approach leverages a learned state-conditioned discretization to enable planning in a lower-dimensional discrete space. Specifically, our encoder processes sequences of state and macro-actions as input. For example, each token is defined as $x_t = (R_t, s_t, m_t)$ and its subsequent token as $x_{t+L} = (R_{t+L}, s_{t+L}, m_{t+L})$. The encoder function is defined as:

$$f_{\text{enc}}(x_t = (R_t, s_t, m_t), \\ x_{t+L} = (R_{t+L}, s_{t+L}, m_{t+L})) = (z_t, z_{t+L}) \tag{1}$$

where the transition chunk size is two, resulting in two latent codes assigned per chunk. To elaborate, the encoder first concatenates the input return-to-go estimates, states, and macro actions into two transition vectors. It then applies a sequence model, producing two latent feature vectors for each chunk of transitions.

In stochastic environments, executing the same macro-action m from state s can yield different returns R, introducing variability that complicates the vector quantization in VQ-VAE, i.e., note that using the full token $x_t = (R_t, s_t, m_t)$ directly can result in different latent codes z for identical (s_t, m_t) pairs solely due to differences in R_t. This challenge can cause the latent space to become fragmented and reflect return variability more than the underlying structure of available actions. Consequently, the agent might overestimate the returns during decision-making by emphasizing latent codes associated with higher observed returns, neglecting the true distribution of the primitive actions and their expected returns.

To address this issue, we aim to focus the vector quantization process primarily on representations of the state s and macro-actions m, while still preserving the ability to reconstruct the return R. To tackle this challenge, our approach involves creating two versions of each token x_t: the full input $x_t = (R_t, s_t, m_t)$ and a masked version $x_t^{\text{mask}} = (\text{mask}, s_t, m_t)$, where R_t is masked out. The encoder processes both x_t and x_t^{mask} to generate two embeddings, $z_e(x_t)$ and $z_e(x_t^{\text{mask}})$, respectively. We use $z_e(x_t^{\text{mask}})$ for vector quantization to obtain the quantized latent code $z_q(x_t^{\text{mask}})$. To ensure that the codebook embeddings incorporate information from the full input, including R_t, we update the embedding $\mathbf{e}_t$ of the quantized latent code towards $z_e(x_t)$. We modify the loss function by introducing an additional term that encourages the embedding from the masked input to be close to that from the full input. Specifically, we used the embedding from the full input $z_e(x)$ as the learning target for the embedding of the masked input $z_e(x^{\text{mask}})$. The modified loss function is:

$$\mathcal{L} = \log p(x \mid z_q(x^{\text{mask}})) + \|\text{sg}[z_e(x)] - \mathbf{e}\|_2^2 \\ + \beta\|z_e(x^{\text{mask}}) - \text{sg}[\mathbf{e}]\|_2^2 + \|z_e(x^{\text{mask}}) - z_e(x)\|_2^2 \tag{2}$$

where sg denotes the stopgradient operator and the additional term $\left\|z_e(x^{\text{mask}}) - z_e(x)\right\|_2^2$ acts as a regularizer that aligns the embeddings of the masked and full inputs.

Incorporating macro-actions within each token is critical, as it enables the model to capture temporal dependencies across multiple time steps without the need for downsampling. This approach is particularly important in stochastic settings, where downsampling techniques that aggregate states (as in Jiang et al. [17]) can obscure the stochasticity imposed by the environment's dynamics. The decoder takes the initial state and latent codes as inputs, and outputs the reconstructed trajectories:

$$f_{dec}(s_t, z_t, z_{t+L}) = (\hat{x}_t = (\hat{R}_t, \hat{s}_t, \hat{m}_t), \\ \hat{x}_{t+L} = (\hat{R}_{t+L}, \hat{s}_{t+L}, \hat{m}_{t+L})) \tag{3}$$

The decoding process can be seen as the inverse of the encoding process, except that the initial state s_t is merged into the embeddings of the codes with a linear projection before decoding.

Latent Transition Model: Following the discretization process, the subsequent step involves modeling sequences of latent codes in an autoregressive manner using a causal Transformer. The Prior Transformer is conditioned on the initial state s_t, achieved by adding the state feature to all token embeddings [17]. Primarily, it functions as a transition model in the latent space, enabling the sampling of the next latent code z_{i+1} conditioned on the current code z_i and state s. This transition, represented as $T : \mathcal{S} \times \mathcal{Z} \to \mathcal{Z}$, implicitly captures the full $\mathcal{R} \times \mathcal{S} \times \mathcal{M} \to \mathcal{R} \times \mathcal{S} \times \mathcal{M}$ transition in the original space, as each z encodes information about the return-to-go, state and macro-action. Additionally, $p(z \mid s)$ acts as a prior policy for efficient action sampling, allowing rapid selection of probable macro-actions based on learned behaviors from the offline dataset. By operating in the learned latent space, the model potentially reduces computational complexity compared to modeling transitions in the original state-action space, especially for high-dimensional environments. The discrete nature of the latent space allows for efficient sampling, which can be beneficial for downstream tasks such as planning.

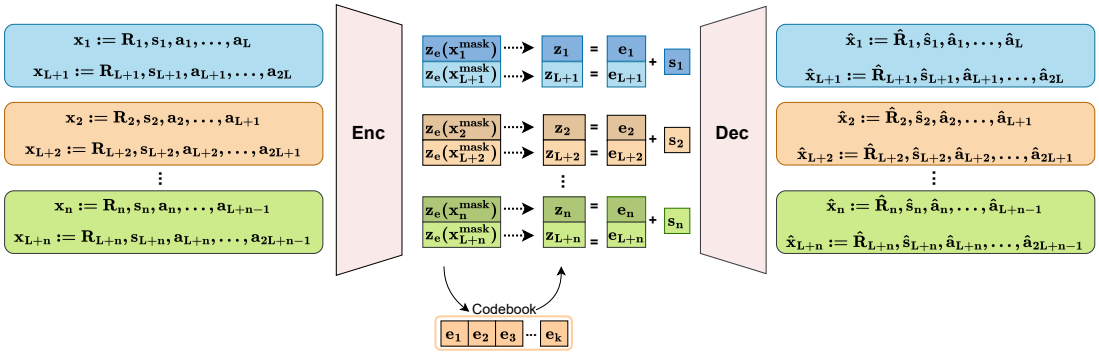


Figure 2. An overview of our VQ-VAE model that discretizes state-macro action sequences.

3.2. Planning with a Latent Macro Action Model

Planning in high-dimensional environments using learned discrete representations introduces uncertainties from multiple sources. First, the representation learning process introduces uncertainty due to the non-injective mapping from the high-dimensional state-action space to a lower-dimensional latent space. This can result in many-to-one correspondences, where multiple distinct high-dimensional inputs map to the same latent representation, creating apparent stochasticity even in deterministic environments. Second, the environment itself may be inherently stochastic. The detailed analysis of these uncertainty sources is provided in Appendix D.

We argue that taking expectations over latent transitions is beneficial in mitigating all these sources of uncertainty, regardless of whether the environment is deterministic or stochastic. By considering the expected outcomes over multiple latent transitions, we can average out the randomness introduced by the non-injective mapping and inherent stochasticity, leading to more reliable planning decisions. This insight applies broadly to planning methods that employ models with non-injective mapping characteristics. Building on this insight, we employ Monte Carlo Tree Search (MCTS) as our planning algorithm to mitigate the impact of stochasticity arising from non-injective mappings and potential environmental randomness. MCTS iteratively explores the latent space and takes expectations over transitions, allowing for robust planning in the presence of uncertainty.

Pre-constructing the Latent Search Space. Our approach leverages a learned latent transition model to generate and evaluate macro actions for planning efficiently. Starting from an initial state s_0, we sample M latent codes z, each representing a potential macro action. For each sampled latent code z, we sample N subsequent latent codes z' to simulate possible future trajectories, capturing the outcomes of these macro actions. We obtain the corresponding state-action transitions and return estimates by decoding these latent pairs (z, z') conditioned on s. To construct

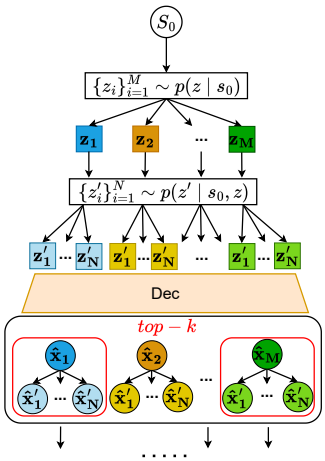


Figure 3. Pre-construction of the latent search space by sampling and evaluating latent macro-action codes, caching the top-k candidates, and recursively expanding the planning tree for efficient macro-level planning.

the planning tree efficiently, we cache the initial state s_0 along with the top-k latent codes z (and their associated information) based on the decoded returns, where $k = \lambda \times M$ and $\lambda \in (0, 1]$ controls the expansion ratio of the tree. The cached latent codes represent the most promising macro actions to consider from the initial state. The latent codes z' are then decoded to obtain a set of reconstructed tokens, i.e., (R, s, m). For each of these states s, we sample B latent codes z'', representing potential macro actions from s (note that B and M are exogenously defined hyper-parameters). This process is recursively applied, allowing us to expand the planning tree while controlling its growth through the parameter λ. By focusing on the most promising macro actions at each state, we maintain a compact and informative planning structure that efficiently explores the state-action space at a macro level.

Selection. Starting from the cached tree structure, MCTS iteratively expands and evaluates nodes, allowing for a more comprehensive exploration of the state-action space. For each state s in the tree, MCTS selects one of the top-k cached latent codes z based on the Upper Confidence Bounds for Trees (UCT) [19]: $\text{UCT}(s,z) = Q(s,z) + c\sqrt{\frac{\log(N(s))}{N(s,z)}}$ where $Q(s,z)$ represents the value of executing macro action z in state s (estimated through the decoded return-to-go), $N(s)$ denotes the number of times state s has been visited, $N(s,z)$ denotes the number of times macro action$z$ has been chosen in state s, and c is an exploration coefficient. **Progressively Widening the State Space for Search.** Despite these powerful abstraction techniques, the search space remains challenging due to the underlying high-dimensional nature of the original state space, residual stochastic characteristics of transitions in the abstracted space, and the complexity of long-horizon planning scenarios. If we were to apply MCTS directly to this abstracted space, we would encounter two main issues: inefficient utilization of our pre-built search space, with the search potentially diverging prematurely into unexplored regions, and difficulty in building sufficiently deep trees for high-quality long-term decision-making, particularly in areas of high stochasticity or uncertainty [8]. Therefore, we use progressive widening to extend MCTS to incrementally expand the search tree. It balances the exploration of new states with the exploitation of already visited states based on two hyperparameters: $\alpha \in [0,1]$ and $\epsilon \in \mathbb{R}^+$. Let $|\mathcal{C}(s,z)|$ denote the number of children for the state-action pair (s,z). The key idea is to alternate between adding new child nodes and selecting among existing child nodes, depending on the number of times a state-action pair (s,z) has been visited. A new state is added to the tree if $|\mathcal{C}(s,z)| < \epsilon \cdot N(s,z)^\alpha$, where $N(s,z)$ is the number of times the state-action pair has been visited. The hyperparameter α controls the propensity to select among existing children, with $\alpha = 0$ leading to always selecting among existing child and $\alpha = 1$ leading to vanilla MCTS behavior (always adding a new child). In this way, we could enhance our approach by efficiently utilizing the pre-built search space, prioritizing the exploration of promising macro actions while allowing for incremental expansion of the search tree. This technique enables our method to make quick decisions in an anytime manner, leveraging the cached information, and further refine the planning tree if additional time is available.

Expansion. In our approach, the *expansion* phase differs from standard MCTS by performing *parallel expansion* of multiple nodes from a leaf node. From the leaf node, a set of B latent codes $\{z^{(i)}\}_{i=1}^{B}$ is sampled, each representing a distinct macro action, drawn from a latent transition model $p(z \mid s)$ to ensure diverse action space coverage. For each sampled macro action $z^{(i)}$, N subsequent latent codes $\{z'^{(i,j)}\}_{j=1}^{N}$ are sampled according to $z'^{(i,j)} \sim p(z' \mid z^{(i)}, s)$, for $j = \{1,\dots,N\}$, modeling potential outcomes and capturing the stochastic nature of macro actions. These latent transitions are then decoded to obtain the resulting next states $\{s'^{(i,j)}\}_{j=1}^{N}$ for each macro action. Finally, the search tree is expanded by adding all L child nodes $\{(s'^{(i,j)}, z'^{(i,j)})\}_{j=1}^{N}$ for each macro action $z^{(i)}$ to the current leaf node s. This breadth-wise expansion enables simultaneous exploration of multiple promising macro actions, enhancing the diversity and comprehensiveness of the search and facilitating efficient exploration in complex environments.

Backpropagation. Following the expansion phase, where multiple macro actions are expanded simultaneously, the *backpropagation* step updates the estimated Q-values based on the return-to-go as shown in Fig.4.

4. Experiments

The empirical evaluation of L-MAP consists of three sets of tasks from D4RL [11]: gym locomotion control, AntMaze, and Adroit. We compare L-MAP to a range of prior offline RL algorithms, including both model-free actor-critic methods [20, 21] and model-based approaches [16, 17, 26]. Our work is conceptually most related to the Trajectory Transformer (TT; Janner et al. [16]) and the Trajectory Autoencoding Planner (TAP; Jiang et al. [17]), which are model-based planning methods that predict and plan in continuous state and action spaces. These two baselines serve as our main points of comparison for deterministic environments.

To demonstrate L-MAP's ability to make performant decisions in stochastic environments, we compare it with One Risk to Rule Them All (1R2R; Rigter et al. [26]), a risk-averse model-based algorithm designed for stochastic domains, and model-free actor-critic methods Conservative Q-Learning (CQL; Kumar et al. [21]) and Implicit Q-Learning (IQL; Kostrikov et al. [20]). We evaluate L-MAP on Stochastic MuJoCo tasks [26], which serve as a proof of concept in the stochastic continuous control domain.

We then test L-MAP on Adroit, which presents a challenge with its high state and action dimensionality. Finally, we evaluate L-MAP on AntMaze, a sparse-reward continuous-control problem. In this task, L-MAP achieves similar performance to TT, surpassing model-free methods. Through these diverse evaluations, we aim to demonstrate L-MAP's versatility and effectiveness across different types of control problems, including stochastic environments, high-dimensional spaces, and sparse-reward scenarios. Additionally, we conduct an ablation study to analyze the impact of key components in L-MAP; detailed results of this study can be found in Appendix A.

Hyperparameters As for the L-MAP-specific hyperparameters, we set our macro action length to 3. The planning

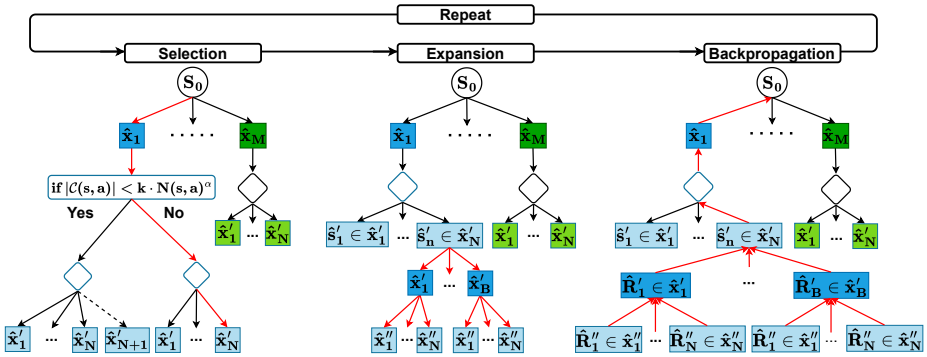


Figure 4. Illustration of our MCTS process for macro-level planning. The algorithm iteratively selects actions using the UCT policy, applies progressive widening to balance exploration and exploitation, performs parallel expansion of multiple macro actions and their potential outcomes, and backpropagates estimated Q-values to efficiently explore and refine the planning tree.

horizon in the raw action space is set to 9 for gym locomotion tasks and 15 for Adroit tasks. These horizons are either smaller or equal to those used in TT and TAP. Our choice of parameters is to ensure a control rate of approximately 10 Hz for locomotion tasks. For each task, we conduct experiments with 3 different training seeds, and each seed is evaluated for 20 episodes.

Stochastic Mujoco On the Stochastic MuJoCo tasks, with results presented in Table 1, L-MAP **consistently outperforms the model-based baselines, TAP and TT, across all datasets and environments**, demonstrating its superior capacity to handle stochasticity in continuous control tasks. Notably, L-MAP achieves the highest performance in multiple datasets for both the Hopper and Walker2D environments. When compared to 1R2R, a risk-averse model-based algorithm specifically designed for stochastic domains, L-MAP shows competitive or superior results in most cases. An exception is the Medium-Replay-High Hopper dataset, where 1R2R attains a higher score. This suggests that while L-MAP exhibits robustness across a variety of stochastic settings, there are specific scenarios where risk-averse strategies like 1R2R may hold an advantage. Additionally, L-MAP generally outperforms the model-free methods, CQL and IQL. However, CQL surpasses L-MAP in the Medium-Expert-Mod Hopper dataset. It is worth noting that L-MAP is the only method among all baselines that achieves performance comparable to CQL in this specific setting.

Adroit Control In the Adroit robotic control tasks, which are characterized by their high-dimensional state and action spaces, our proposed method, L-MAP, **demonstrates strong and competitive performance** as shown in Table 2. Across the Human, Cloned, and Expert datasets, L-MAP exhibits notable effectiveness compared to both model-based approaches (TAP and TT) and model-free methods (CQL, IQL, and Behavior Cloning (BC)[1]).

In the Human dataset, which includes suboptimal human demonstrations, L-MAP achieves the highest score in the Door environment and performs well in other tasks. Although IQL leads in the Pen task and CQL leads in the Hammer and Relocate tasks, L-MAP maintains competitive results, particularly surpassing TT and BC in most environments. This suggests that L-MAP effectively utilizes suboptimal data to make robust decisions in complex settings. For the Cloned dataset, which contains a mix of optimal and suboptimal trajectories, L-MAP secures top performance in the Pen and Relocate tasks. In the Expert dataset, comprised of optimal demonstrations, L-MAP attains the highest scores in the Pen and Relocate environments while remaining competitive in the Hammer and Door tasks. Overall, L-MAP achieves the highest average score of 51.40 across all datasets and environments, and 18.79 across non-expert datasets, highlighting its effectiveness in handling varying levels of data optimality. Furthermore, the experimental results indicate that L-MAP effectively manages the complexities of high-dimensional Adroit environments. Incorporating more action information into the single token does not detract from performance; instead, it appears to enhance the model's ability to learn nuanced temporal dependencies required for successful task execution.

AntMaze In the AntMaze environments—a set of sparse-reward continuous-control tasks where an agent must navigate a robotic ant to a target location, L-MAP demonstrates strong and competitive performance as shown in Table 3. These tasks are particularly challenging due to the sparse rewards and the presence of suboptimal trajectories that lead to various goals other than the target position used during testing.

Similar to TAP, our approach integrates goal positions

[1]We included Behavior Cloning (BC) as an additional baseline since the original 1R2R method was not evaluated for Adroit tasks.

		Model-Based				Model-Free	
Dataset Type	**Env**	**L-MAP**	**TAP**	**TT**	**1R2R**	**CQL**	**IQL**
Medium-Expert-Mod	Hopper	106.11 ± 2.16	40.86 ± 5.42	56.10 ± 3.33	52.19 ± 8.37	**106.17 ± 2.16**	60.61 ± 3.46
Medium-Expert-Mod	Walker2D	**93.43 ± 1.41**	91.40 ± 1.42	80.93 ± 2.60	56.48 ± 7.51	91.44 ± 1.44	86.66 ± 1.84
Medium-Mod	Hopper	55.07 ± 3.06	43.64 ± 2.25	44.49 ± 2.47	**65.24 ± 3.31**	49.92 ± 3.00	56.00 ± 3.60
Medium-Mod	Walker2D	52.94 ± 1.57	44.46 ± 1.82	43.61 ± 2.15	**65.16 ± 2.84**	49.38 ± 2.02	48.82 ± 2.31
Medium-Replay-Mod	Hopper	**52.30 ± 2.65**	38.10 ± 3.22	37.85 ± 1.19	22.82 ± 2.08	40.53 ± 1.52	49.12 ± 3.38
Medium-Replay-Mod	Walker2D	51.44 ± 1.65	43.49 ± 2.27	27.43 ± 3.33	**52.23 ± 2.22**	40.24 ± 1.67	40.77 ± 2.72
Medium-Expert-High	Hopper	66.93 ± 3.46	37.31 ± 3.66	58.04 ± 3.60	37.99 ± 2.71	**68.03 ± 3.94**	44.83 ± 2.58
Medium-Expert-High	Walker2D	**97.18 ± 2.08**	91.09 ± 2.78	50.01 ± 3.51	32.38 ± 4.55	83.18 ± 3.70	68.61 ± 3.33
Medium-High	Hopper	**55.32 ± 3.56**	43.93 ± 2.66	41.26 ± 5.53	33.99 ± 0.92	45.21 ± 2.97	49.69 ± 2.47
Medium-High	Walker2D	**68.87 ± 2.21**	52.20 ± 2.76	59.84 ± 5.03	32.13 ± 4.51	61.49 ± 3.24	47.53 ± 3.05
Medium-Replay-High	Hopper	58.05 ± 3.36	48.69 ± 2.97	39.24 ± 2.16	**68.25 ± 3.78**	51.70 ± 3.09	43.27 ± 2.78
Medium-Replay-High	Walker2D	**65.87 ± 3.07**	55.15 ± 3.29	16.55 ± 2.17	65.63 ± 3.41	50.33 ± 3.88	45.13 ± 2.38
Mean		**68.63**	52.53	46.28	48.71	61.47	53.42

Table 1. Results for Stochastic MuJoCo under various dataset/noise settings.

		Model-Based Approaches			Model-Free Approaches		
Dataset Type	**Env**	**L-MAP**	**TAP**	**TT**	**CQL**	**IQL**	**BC**
Human	Pen	**76.26 ± 8.58**	66.86 ± 8.41	36.4	37.5	71.5	34.4
Human	Hammer	1.71 ± 0.12	1.57 ± 0.09	0.8	**4.4**	1.4	1.5
Human	Door	**11.24 ± 1.11**	9.51 ± 1.10	0.1	9.9	4.3	0.5
Human	Relocate	0.09 ± 0.02	0.06 ± 0.01	0.0	**0.2**	0.1	0.0
Cloned	Pen	**60.68 ± 7.88**	46.44 ± 7.54	11.4	39.2	37.3	56.9
Cloned	Hammer	**2.43 ± 0.29**	1.32 ± 0.12	0.5	2.1	2.1	0.8
Cloned	Door	13.22 ± 1.34	**13.45 ± 1.43**	–0.1	0.4	1.6	–0.1
Cloned	Relocate	**0.15 ± 0.13**	–0.23 ± 0.01	–0.1	–0.1	–0.2	–0.1
Expert	Pen	**126.60 ± 5.60**	112.16 ± 6.57	72.0	107.0	–	85.1
Expert	Hammer	127.16 ± 0.29	**128.79 ± 0.52**	15.5	86.7	–	125.6
Expert	Door	105.24 ± 0.10	**105.86 ± 0.08**	94.1	101.5	–	34.9
Expert	Relocate	**107.57 ± 0.76**	106.21 ± 1.61	10.3	95.0	–	101.3
Mean (All)		**51.40**	49.33	20.08	40.32	14.76	36.73
Mean (Non-Expert)		**18.79**	17.37	6.13	11.70	14.76	11.74

Table 2. Adroit robotic hand control results.

into the observation space, allowing it to condition trajectory generation on specific goals. This conditioning narrows the focus of sampled trajectories towards the target direction, simplifying the planning process. Instead of using the IQL critic for value estimation, L-MAP leverages Monte Carlo planning to provide refined value estimates. This alternative approach avoids the additional computational cost of sampling with a separate Q-network, as required by TT (+Q).

Our method achieves an average success rate of 83.33% across all AntMaze environments, which is comparable to the 84.00% average of TT (+Q). Notably, L-MAP outperforms TT (+Q) in the more complex Large-Play and Large-Diverse environments, achieving success rates of 78.33% and 81.67% respectively, compared to TT (+Q)'s 66.7% and 60.0%. This indicates that L-MAP is particularly effective in larger mazes where navigation complexity is higher. While TT (+Q) attains perfect success rates in smaller environments like Umaze and Medium-Diverse, L-MAP still performs exceptionally well with success rates of 93.33% and 88.33% in these settings. This consistency suggests that our method is robust across different scales of environment complexity.

5. Related Work

Recent advancements in reinforcement learning focus on learning temporally extended action primitives to reduce decision-making horizons and improve learning efficiency. Both model-free and model-based methods leverage temporal abstraction to manage task complexity.

Model-free methods such as CompILE [18], RPL [13],

Table 3. Performance comparison on AntMaze environments. This evaluation demonstrates that our approach can achieve comparable performance to TT with a separate Q network, while being more efficient during sampling and decision-making.

Dataset Environment	**BC**	**CQL**	**IQL**	**TT (+Q)**	**TAP**	**L-MAP**
Umaze AntMaze	54.6	74.0	87.5	**100.0 ± 0.0**	78.33 ± 5.32	93.33 ± 3.22
Medium-Play AntMaze	0.0	61.2	71.2	**93.3 ± 6.4**	43.33 ± 6.40	75.00 ± 6.85
Medium-Diverse AntMaze	0.0	53.7	70.0	**100.0 ± 0.0**	30.00 ± 5.92	88.33 ± 4.14
Large-Play AntMaze	0.0	15.8	39.6	66.7 ± 12.2	63.33 ± 6.22	**78.33 ± 5.32**
Large-Diverse AntMaze	0.0	14.9	47.5	60.0 ± 12.7	66.67 ± 6.09	**81.67 ± 5.00**
Mean	10.92	43.92	55.16	**84.00**	56.33	83.33

OPAL [1], ACT [34], and PRISE [35] leverage temporal abstraction in various ways. For instance, CompILE learns latent codes representing variable-length behavior segments, enabling cross-task generalization. RPL employs a hierarchical policy architecture to simplify long-horizon tasks by decomposing them into sub-policies. OPAL introduces a continuous space of primitive actions to reduce distributional shift in offline RL, enhancing policy robustness. PRISE applies sequence compression to learn variable-length action primitives, improving behavior cloning by capturing essential behavioral patterns. These approaches demonstrate the versatility of temporal abstraction in addressing different challenges in reinforcement learning, particularly in managing the complexity inherent in sequential decision-making.

From a model-based perspective, recent work has treated reinforcement learning as a sequence modeling problem, utilizing Transformer architectures to model entire trajectories of states, actions, rewards, and values. This approach is exemplified by methods like Trajectory Transformer (TT) [36], and TAP [17]. TAP, in particular, shares conceptual similarities with our proposed method, L-MAP, in its use of efficient planning solutions for complex action spaces. These sequence modeling approaches have shown promise in capturing long-term dependencies and handling the variability in trajectories, but they often face challenges in stochastic environments where the outcome is not solely determined by the agent's actions. As highlighted by Paster et al. [25], reinforcement learning via supervised learning methods may replicate suboptimal actions that accidentally led to good outcomes due to environmental randomness. To address this issue, they proposed ESPER, a solution inspired by the decision transformer framework [7]. ESPER mitigates the influence of stochasticity on policy learning in discrete action spaces by clustering trajectories and conditioning on average cluster returns.

From a planning perspective, our work relates to methods like MuZero [27], stochastic MuZero [3], and Vector Quantized Models for Planning [24], which primarily operate in discrete action spaces and online settings, limiting their applicability to continuous control tasks in offline RL. MuZero Unplugged [28] extended MuZero to the offline setting and adapted to low-dimensional continuous action spaces using factorized policy representations [31]. However, scaling to high-dimensional action spaces is challenging due to computational infeasibility and imprecise action selection [23]. Additionally, MuZero Unplugged focuses on deterministic environments and may struggle in highly stochastic continuous settings.

Our method, L-MAP, extends these concepts to high-dimensional continuous action spaces by effectively handling stochasticity and complexity. Using an encoder to group similar state-macro-action pairs and reconstructing return-to-go estimates via a decoder within the VQ-VAE framework, L-MAP captures essential dynamics while abstracting unnecessary details. This approach models future returns more accurately in stochastic settings. Combined with planning algorithms, L-MAP refines expected return estimates, bridging the gap between temporal abstraction techniques and robust performance in stochastic environments. Our latent code representation and transition model reduce the need to learn separate policy, dynamics, and value components in the offline setting, increasing planning efficiency and accounting for environmental stochasticity, thereby enhancing generalization across complex tasks.

6. Discussion and Limitations

In conclusion, we introduced the Latent Macro Action Planner (L-MAP), which leverages temporal abstractions learned with a state-conditioned VQ-VAE to construct a discrete latent space of macro-actions. This approach enables efficient planning in high-dimensional continuous action spaces within stochastic environments. Future directions include exploring transfer learning to handle new tasks, and adapting L-MAP to online learning scenarios through strategies such as risk-averse exploration [22]. These enhancements would enable continuous improvement and help tackle more complex challenges, ultimately improving generalization and efficiency in complex, real-world settings.

References

[1] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: offline primitive discovery for accelerating offline reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 8

[2] Robert Almgren and Neil Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001. 12

[3] Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K. Hubert, and David Silver. Planning in stochastic environments with a learned model. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 8

[4] Wenhang Bao and Xiao-yang Liu. Multi-agent deep reinforcement learning for liquidation strategy analysis. *arXiv preprint arXiv:1906.11046*, 2019. 12

[5] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13:341–379, 2003. 1

[6] Ashwin Carvalho, Yiqi Gao, Stéphanie Lefèvre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. 2014. 1

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097, 2021. 8

[8] Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*, pages 433–445. Springer, 2011. 5

[9] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000. 1

[10] Damien Ernst, Guy-Bart Stan, Jorge Goncalves, and Louis Wehenkel. Clinical data based optimal sti strategies for hiv: a reinforcement learning approach. *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 667–672, 2006. 12

[11] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. 5, 13

[12] Thomas Gabor, Jan Peter, Thomy Phan, Christian Meyer, and Claudia Linnhoff-Popien. Subgoal-based temporal abstraction in monte-carlo tree search. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5562–5568. ijcai.org, 2019. 2

[13] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, pages 1025–1037. PMLR, 2019. 7

[14] Ruijie He, Emma Brunskill, and Nicholas Roy. Efficient planning under uncertainty with macro-actions. *J. Artif. Intell. Res.*, 40:523–570, 2011. 2

[15] Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 4476–4486. PMLR, 2021. 1, 3

[16] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1273–1286, 2021. 2, 5

[17] Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 2, 3, 5, 8

[18] Thomas Kipf, Yujia Li, Hanjun Dai, Vinícius Flores Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter W. Battaglia. Compile: Compositional imitation learning and execution. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 3418–3428. PMLR, 2019. 7

[19] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, pages 282–293. Springer, 2006. 5

[20] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 2, 5

[21] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2, 5

[22] Baiting Luo, Yunuo Zhang, Abhishek Dubey, and Ayan Mukhopadhyay. Act as you learn: Adaptive decision-making in non-stationary markov decision processes. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 1301–1309. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024. 8

[23] Jianlan Luo, Perry Dong, Jeffrey Wu, Aviral Kumar, Xinyang Geng, and Sergey Levine. Action-quantized offline

reinforcement learning for robotic skill learning. In *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, pages 1348–1361. PMLR, 2023. 3, 8

[24] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aäron van den Oord, and Oriol Vinyals. Vector quantized models for planning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8302–8313. PMLR, 2021. 8

[25] Keiran Paster, Sheila A. McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. 8

[26] Marc Rigter, Bruno Lacerda, and Nick Hawes. One risk to rule them all: A risk-sensitive perspective on model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. 2, 5, 12

[27] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588 (7839):604–609, 2020. 1, 8

[28] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27580–27591, 2021. 8

[29] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. 1, 11

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

[31] Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5981–5988. AAAI Press, 2020. 8

[32] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017. 1

[33] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25476–25488, 2021. 1

[34] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. 8

[35] Ruijie Zheng, Ching-An Cheng, Hal Daumé III, Furong Huang, and Andrey Kolobov. PRISE: llm-style sequence compression for learning temporal action abstractions in control. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. 8

[36] Wenxuan Zhou, Sujay Bajracharya, and David Held. PLAS: latent action space for offline reinforcement learning. In *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, pages 1719–1735. PMLR, 2020. 8

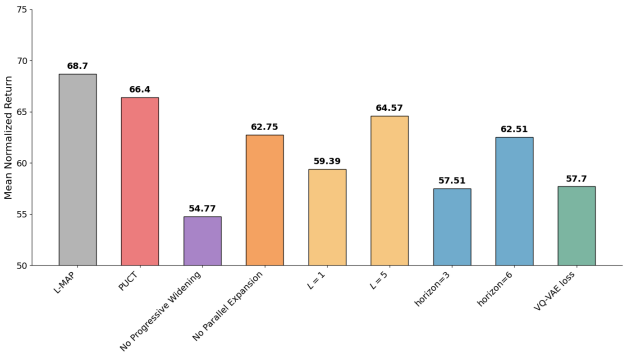


Figure 5. Results of ablation studies, where the height of the bar is the mean normalized scores on high noise gym locomotion control tasks.

A. Ablation Study

We present analyses and ablations of key hyperparameters such as macro action length, planning horizon, the use of pUCT [29] versus UCT, and the effect of our customized VQ-VAE loss function. Figure 5 summarizes the results from ablation studies conducted on high-noise stochastic MuJoCo tasks.

Macro Action Length

We tested macro action lengths $L = 1$, $L = 3$, and $L = 5$ to evaluate their impact on L-MAP's performance. The highest mean score of 68.7 was achieved with $L = 3$. Increasing L to 5 reduced the mean score to 64.57, while decreasing it to 1 further dropped it to 59.39. This indicates that a macro action length of 3 optimally balances temporal abstraction and adaptability. A moderate length allows the model to capture important action sequences while remaining responsive to environmental changes. Shorter lengths may fail to model temporal dependencies effectively, while longer lengths may hinder quick adaptation in stochastic environments.

Planning Horizon

We assessed the effect of planning horizon by varying the number of planning steps in L-MAP. Reducing the planning horizon to 3 steps (expanding a single latent variable) decreased the mean score to 57.51, compared to 68.7 with the default longer planning horizon. This demonstrates that a longer planning horizon significantly enhances performance by enabling the model to better anticipate future events and handle uncertainty in high-noise stochastic environments.

Tree Search Algorithm: UCT vs. pUCT

We compared standard UCT and pUCT as tree search algorithms in L-MAP. UCT achieved a mean score of 68.7, slightly outperforming pUCT, which scored 66.4. While both methods are effective, UCT performs marginally better in this context. A possible explanation is that pUCT leverages a learned prior policy to guide exploration, making it sensitive to the quality of the prior. If the prior policy is suboptimal, pUCT may be less effective due to this dependency.

VQ-VAE Loss Function We compared our loss function with the standard loss function without masking (mean scores: 68.7 vs 57.7). Our approach outperforms the standard loss by focusing primarily on state and action during vector quantization. This results in less skewed reconstructed returns and a more coherent latent space, accurately capturing action and state distributions. Consequently, the model generates more reliable latent representations for reconstruction.

Progressive Widening

We evaluated the impact of progressive widening on MAP's performance. Removing progressive widening led to a significant drop in the mean score from 68.70 to 54.77. This substantial decrease demonstrates the importance of controlled state space expansion during planning for a large search space. Progressive widening enables MAP to balance between exploiting existing states in the pre-built search space and incrementally adding new states. Without progressive widening, the search suffers from excessive branching, making it difficult to build sufficiently deep trees for meaningful planning in areas of high stochasticity.

Parallel Expansion

We assessed the contribution of parallel expansion by comparing L-MAP's performance with and without this feature. Removing parallel expansion reduced the mean score from 68.7 to 62.75, yielding performance similar to reducing the planning horizon to six steps. This comparison reveals that parallel expansion primarily affects the algorithm's ability to efficiently explore the search space. Given the same number of MCTS iterations, removing parallel expansion results in less exploration of possible trajectories, reducing the algorithm's planning capability to that of a shorter horizon. This demonstrates that parallel expansion is crucial for maximizing the effectiveness of each MCTS iteration by enabling broader simultaneous exploration of potential outcomes.

B. Additional Deterministic Environment experiments: D4RL MuJoCo

On the deterministic MuJoCo tasks, particularly when compared to established model-free approaches such as CQL and IQL, L-MAP demonstrates **notable performance in environments like Walker2D and Hopper**, matching or exceeding these baselines even in dense reward scenarios as shown in Table 4. This highlights L-MAP's effectiveness across various task structures. When compared to TT, L-MAP consistently delivers comparable results. However, L-MAP offers a significant practical advantage: its use of **temporal abstraction enables lower latency decision-making**

for equivalent planning horizons, resulting in improved efficiency during deployment. Furthermore, L-MAP generally outperforms TAP, suggesting that even in deterministic environments, the expectation-based planning approach proves advantageous by accounting for stochasticity in the behavior policy. This leads to more robust policies and, consequently, superior results.

C. Additional Stochastic Environment experiments: HIV treatment and Currency Exchange

The HIV Treatment environment, originally introduced by Ernst et al. [10], simulates treatment planning where an agent controls two drug types (RTI and PI) in a 6-dimensional state space representing cell and virus concentrations. The stochasticity arises from varying drug efficacy at each step. The Currency Exchange environment, based on the Optimal Liquidation problem [2, 4], involves converting currency under stochastic exchange rates that follow an Ornstein-Uhlenbeck process. Both environments were adapted by Rigter et al. [26] to the offline RL setting, with datasets collected using partially trained and random policies respectively.

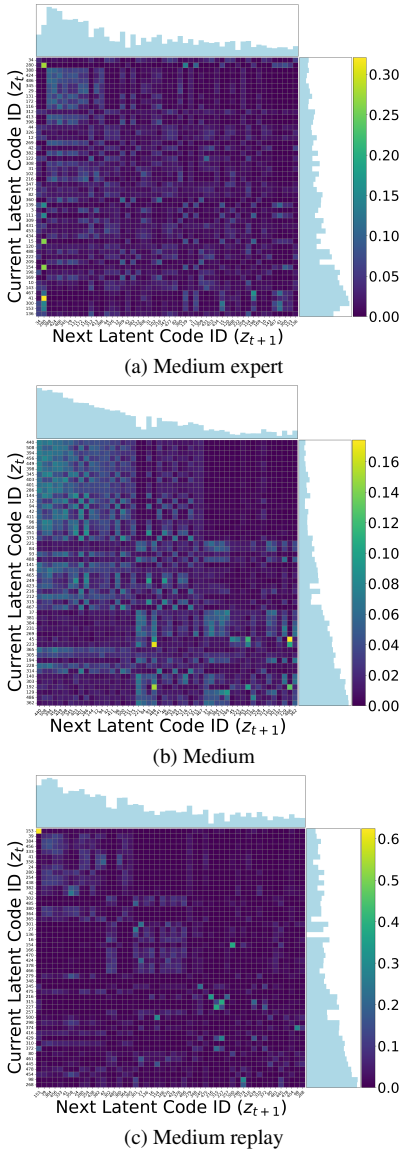
For the HIV Treatment domain, L-MAP and CQL achieve comparable strong performance (59.08 ± 1.96 and 59.74 ± 1.11 respectively), outperforming other baselines. In the Currency Exchange environment, L-MAP substantially outperforms all other approaches, achieving a score of 106.78 ± 5.00 compared to the next best performer CQL at 93.96 ± 1.69. This superior performance demonstrates L-MAP's versatility across different types of stochastic environments.

D. Latent Space Analysis

To empirically demonstrate the uncertainties introduced by non-injective mappings, behavior policy, and environmental stochasticity, we generate heatmaps representing the transition probabilities between latent codes. We focus on the Hopper environment and consider three datasets: `medium-expert`, `medium`, and `medium-replay`, in both deterministic and stochastic settings. The heatmaps are constructed by encoding the state-macro-action pairs into latent codes using our learned representation and visualizing the transition probabilities between these codes.

D.1. Deterministic Environment Heatmaps

In analyzing the heatmaps for deterministic environments as shown in Fig. 6, it becomes evident that transitions from a current latent code z_t to multiple next latent codes z_{t+1} are not strictly deterministic. This observed spread in transitions originates from two primary sources: the **non-injective nature of the learned representation** and the **stochasticity of the behavior policy** employed during data collection.



(a) Medium expert

(b) Medium

(c) Medium replay

Figure 6. Heatmaps for Deterministic Hopper Environment (Top 50 Frequent Latent Codes). In each heatmap, the intensity of the color at position (i, j) represents the probability of transitioning from the current latent code $z_t = i$ to the next latent code $z_{t+1} = j$. The accompanying histograms display the frequency of each latent code occurring across the dataset with the learned encoder as the current (z_t, right histogram) and next (z_{t+1}, top histogram) codes. The observed spread in the heatmaps indicates that, despite the deterministic nature of the environment, transitions from a single z_t lead to multiple z_{t+1}.

First, the **non-injective mapping** of the encoder function f_{enc} may result in multiple distinct high-dimensional state-macro-action pairs being mapped to the same latent

Table 4. Normalized results for D4RL MuJoCo-v2 following the protocol of Fu et al. [11]

		Model-Based				Model-Free	
Dataset Type	**Env**	**L-MAP**	**TAP**	**TT**	**1R2R**	**CQL**	**IQL**
Medium-Expert	HalfCheetah	92.14 ± 0.26	86.40 ± 2.22	**95.0 ± 0.2**	93.99 ± 1.40	91.6	86.7
Medium-Expert	Hopper	105.74 ± 2.24	85.55 ± 3.83	**110.0 ± 2.7**	57.40 ± 6.06	105.4	91.5
Medium-Expert	Walker2D	109.35 ± 0.08	105.32 ± 2.03	101.9 ± 6.8	73.18 ± 6.29	108.8	**109.6**
Medium	HalfCheetah	45.50 ± 0.10	44.73 ± 0.39	46.9 ± 0.4	**73.45 ± 0.15**	44.4	47.4
Medium	Hopper	**73.90 ± 1.91**	69.14 ± 2.33	61.1 ± 3.6	55.49 ± 3.99	58.0	66.3
Medium	Walker2D	**80.31 ± 1.20**	51.75 ± 3.30	79.0 ± 2.8	55.69 ± 4.97	72.5	78.3
Medium-Replay	HalfCheetah	38.45 ± 0.80	40.83 ± 0.72	41.9 ± 2.5	**63.85 ± 0.19**	45.5	44.2
Medium-Replay	Hopper	91.18 ± 0.56	80.92 ± 3.79	91.5 ± 3.6	89.67 ± 1.92	**95.0**	94.7
Medium-Replay	Walker2D	81.04 ± 2.62	72.32 ± 3.26	82.6 ± 6.9	**90.67 ± 1.98**	77.2	77.2
Mean		**79.73**	70.77	78.88	72.60	77.60	77.32

Table 5. Results for HIV Treatment and Currency Exchange.

	Model-Based Approaches				Model-Free Approaches	
Env	**L-MAP**	**TAP**	**TT**	**1R2R**	**CQL**	**IQL**
HIV	59.08 ± 1.96	54.95 ± 1.98	54.46 ± 3.30	56.45 ± 2.17	$\mathbf{59.74 \pm 1.11}$	34.1 ± 1.2
Currency	$\mathbf{106.78 \pm 5.00}$	89.72 ± 3.90	79.28 ± 2.61	78.52 ± 2.08	93.96 ± 1.69	89.41 ± 2.83

code as shown in the histograms of Fig.6. Specifically, for different state-macro-action pairs $x_t^{(1)} = (s_t^{(1)}, m_t^{(1)})$ and $x_t^{(2)} = (s_t^{(2)}, m_t^{(2)})$, it is possible that:

$$f_{\text{enc}}(x_t^{(1)}) = f_{\text{enc}}(x_t^{(2)}) = z_t,$$

even though $x_t^{(1)} \neq x_t^{(2)}$. Consequently, their corresponding next state-macro-action pairs $x_{t+1}^{(1)}$ and $x_{t+1}^{(2)}$ may differ, potentially leading to different next latent codes upon encoding:

$$z_{t+1}^{(1)} = f_{\text{enc}}(x_{t+1}^{(1)}), \quad z_{t+1}^{(2)} = f_{\text{enc}}(x_{t+1}^{(2)}), \tag{4}$$

$$\text{with} \quad z_{t+1}^{(1)} \neq z_{t+1}^{(2)}. \tag{5}$$

Second, because the **behavior policy** π_{b} used for data collection may be stochastic, it introduces variability in the selection of macro-actions at both the current and subsequent time steps. Given a state s_t, the behavior policy determines the macro-action m_t as follows:

$$m_t \sim \pi_{\text{b}}(m \mid s_t).$$

This stochastic selection can result in different macro-actions $m_t^{(1)}$ and $m_t^{(2)}$ being chosen from the same state s_t, which naturally introduces stochasticity. Note that even if the encoder maps both $x_t^{(1)} = (s_t, m_t^{(1)})$ and $x_t^{(2)} = (s_t, m_t^{(2)})$ to the same latent code z_t:

$$f_{\text{enc}}(x_t^{(1)}) = f_{\text{enc}}(x_t^{(2)}) = z_t.$$

the next states $s_{t+1}^{(1)}$ and $s_{t+1}^{(2)}$ might differ, even though the environment dynamics T_{env} are deterministic, i.e.,

$$s_{t+1}^{(1)} = T_{\text{env}}(s_t, m_t^{(1)}), \quad s_{t+1}^{(2)} = T_{\text{env}}(s_t, m_t^{(2)}), \tag{6}$$

$$\text{with} \quad s_{t+1}^{(1)} \neq s_{t+1}^{(2)}. \tag{7}$$

These different next states lead to different next state-macro-action pairs:

$$x_{t+1}^{(1)} = (s_{t+1}^{(1)}, m_{t+1}^{(1)}), \quad x_{t+1}^{(2)} = (s_{t+1}^{(2)}, m_{t+1}^{(2)}).$$

Upon encoding, they may yield different next latent codes:

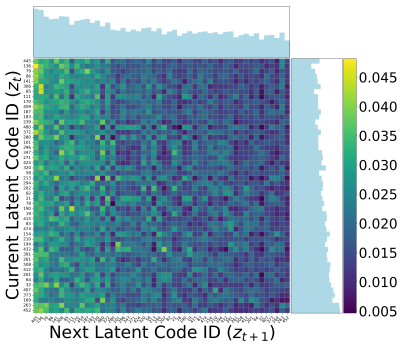
$$z_{t+1}^{(1)} = f_{\text{enc}}(x_{t+1}^{(1)}), \quad z_{t+1}^{(2)} = f_{\text{enc}}(x_{t+1}^{(2)}), \tag{8}$$

$$\text{with} \quad z_{t+1}^{(1)} \neq z_{t+1}^{(2)}. \tag{9}$$

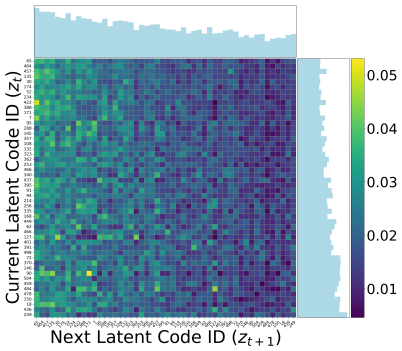
Therefore, even in a deterministic environment, the combination of a non-injective encoder and a stochastic behavior policy introduces variability in the latent transitions. The heatmaps for deterministic environments empirically demonstrate this spread, showing that each z_t does not map deterministically to a single z_{t+1} but rather to a distribution of possible next latent codes.

D.2. Stochastic Environment Heatmaps

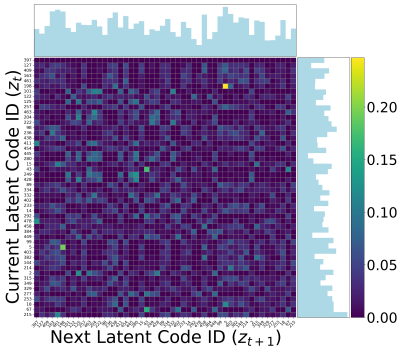
The heatmaps for stochastic environments as shown in Fig. 7 exhibit a more pronounced spread in transition probabilities. This inherent environmental stochasticity means that for a given s_t and m_t, there are multiple possible next



(a) Medium Expert

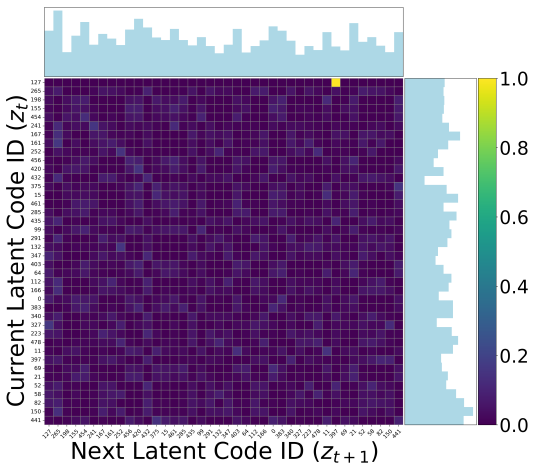


(b) Medium

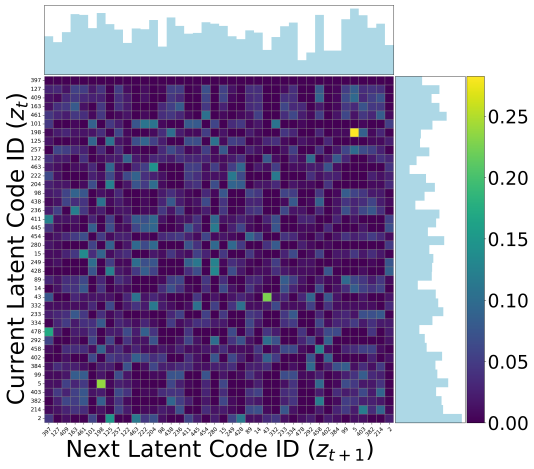


(c) Medium Replay

Figure 7. Heatmaps for Stochastic Hopper Environment (Top 50 Frequent Latent Codes). The observed spread in the heatmaps indicates that inherent environmental stochasticity further contributes to transitions from a single z_t leading to multiple z_{t+1}.



(a) L_1 norm



(b) L_2 norm

Figure 8. Transition Probability Heatmaps for Medium-Replay Datasets from the Stochastic Hopper Environment (Top 50 Frequent Latent Codes). Left: Heatmap depicting transition probabilities when embeddings are regularized using the L1 norm. Right: Heatmap illustrating transition probabilities under L2 norm regularization.

states s_{t+1}, leading to a wider distribution of next latent codes z_{t+1} upon encoding. When combined with the non-injective mapping of the encoder and the stochasticity of the behavior policy, the uncertainties in the latent transitions are further amplified.

D.3. The Impact of L_1 Regularization on Representation Fidelity

The heatmaps shown in Fig.8 reveal distinct patterns between transition probabilities for latent codes encoded by encoders trained with L_1 and L_2 norm regularization in the latent space. The L_2 norm demonstrates more distributed transition probabilities, with multiple moderate-probability transitions (shown as light blue dots) for each current state, indicating the encoder preserves more granular information. In contrast, the L_1 norm exhibits highly deterministic transitions for certain latent codes, shown by the predominantly dark purple background with the bright yellow spot approaching probability 1.0. This suggests that the encoder trained with L_1 regularization tends to collapse dissimilar inputs into the same latent code, leading to less nuanced representations.

Table 6. Hopper Environment Results with Increasing Stochasticity

		Model-Based				Model-Free	
Dataset Type	**Env**	**L-MAP**	**TAP**	**TT**	**1R2R**	**CQL**	**IQL**
Deterministic							
Medium-Expert	Hopper	105.74 ± 2.24	85.55 ± 3.83	**110.0 ± 2.7**	57.40 ± 6.06	105.4	91.5
Medium	Hopper	**73.90 ± 1.91**	69.14 ± 2.33	61.1 ± 3.6	55.49 ± 3.99	58.0	66.3
Medium-Replay	Hopper	91.18 ± 0.56	80.92 ± 3.79	91.5 ± 3.6	89.67 ± 1.92	95.0	94.7
Mean (Deterministic)		**90.27**	78.54	87.53	67.52	86.13	84.17
Moderate Stochasticity							
Medium-Expert-Mod	Hopper	106.11 ± 2.16	40.86 ± 5.42	56.10 ± 3.33	52.19 ± 8.37	**106.17 ± 2.16**	60.61 ± 3.46
Medium-Mod	Hopper	55.07 ± 3.06	43.64 ± 2.25	44.49 ± 2.47	**65.24 ± 3.31**	49.92 ± 3.00	56.00 ± 3.60
Medium-Replay-Mod	Hopper	**52.30 ± 2.65**	38.10 ± 3.22	37.85 ± 1.19	22.82 ± 2.08	40.53 ± 1.52	49.12 ± 3.38
Mean (Moderate Stochasticity)		**71.16**	40.87	46.15	46.75	65.54	55.24
High Stochasticity							
Medium-Expert-High	Hopper	66.93 ± 3.46	37.31 ± 3.66	58.04 ± 3.60	37.99 ± 2.71	**68.03 ± 3.94**	44.83 ± 2.58
Medium-High	Hopper	**55.32 ± 3.56**	43.93 ± 2.66	41.26 ± 5.53	33.99 ± 0.92	45.21 ± 2.97	49.69 ± 2.47
Medium-Replay-High	Hopper	58.05 ± 3.36	48.69 ± 2.97	39.24 ± 2.16	**68.25 ± 3.78**	51.70 ± 3.09	43.27 ± 2.78
Mean (High Stochasticity)		**60.10**	43.31	46.18	46.74	54.98	45.93

E. Analysis of Performance Trends with Increasing Stochasticity

This section examines how L-MAP and baseline methods respond to increasing levels of stochasticity in the Hopper environment. Table 6 presents the performance metrics across deterministic, moderate, and high stochasticity settings.

In the **deterministic** setting, L-MAP achieves a mean score of **90.27**, indicating strong performance and outperforming all other model-based methods. Among the baselines, TT attains a mean of 87.53, TAP achieves 78.54, and 1R2R scores 67.52. The model-free methods CQL and IQL also perform well, with mean scores of 86.13 and 84.17, respectively. The high scores across all methods suggest that the deterministic environment poses minimal challenges, allowing both L-MAP and the baselines to excel.

As the environment introduces **moderate stochasticity**, L-MAP's mean performance decreases to **71.16**, reflecting a reduction of approximately 21% from its deterministic performance. The model-based baselines experience larger declines; TAP's mean drops to 40.87 (a 48% reduction), TT's to 46.15 (a 47% reduction), and 1R2R's to 46.75 (a 31% reduction). The model-free methods also suffer performance losses; CQL's mean decreases to 65.54 (a 24% reduction), and IQL's to 55.24 (a 34% reduction). Despite the reductions, L-MAP maintains a higher mean score than all baselines in this setting, indicating better resilience to moderate stochasticity among both model-based and model-free methods.

In the setting of **high stochasticity**, L-MAP's mean further decreases to **60.10**, representing a total reduction of about 33% from the deterministic case. The model-based baselines continue to show declining trends; TAP's mean falls to 43.31 (a 45% reduction), TT's to 46.18 (a 47% reduction), and 1R2R's to 46.74 (a 31% reduction). The model-free methods also see further decreases; CQL's mean drops to 54.98 (a 36% reduction), and IQL's to 45.93 (a 45% reduction). While all methods experience performance degradation, L-MAP consistently outperforms the model-based baselines TAP and TT, and maintains an edge over the model-free methods CQL and IQL. The performance of L-MAP shows relatively better robustness among the baselines.

The overall trend indicates that increasing stochasticity adversely affects all methods, but L-MAP's performance diminishes at a slower rate compared to the other model-based methods. These results suggest that L-MAP is more robust to stochastic variations in the environment than most of the baseline methods, particularly the model-based ones.

F. Planning Hyperparameters

Table 7. Planning Hyperparameters

Environment	**M**	**N**	**B**	λ	γ
Stochastic MuJoCo	16	4	4	0.5	0.99
D4RL MuJoCo	16	4	4	0.5	0.99
Adroit	10	2	4	0.5	0.99
AntMaze	16	2	4	0.5	0.998
Currency	32	4	4	0.5	0.99
HIV Treatment	5	4	4	1.0	0.99

For all environments, we utilize the following hyperparameters for sampling during the search process: $\alpha = 0.1$ and $\epsilon = 1$, which determine the exploration rate of progressive widening; and set the number of Monte Carlo Tree Search (MCTS) iterations to 100. Detailed parameters for each environment are presented in Table 7.