Learning Abstract World Models for Value-preserving Planning with Options

Rafael Rodriguez-Sanchez
Department of Computer Science
Brown University
Providence, RI
rrs@brown.edu

George Konidaris
Department of Computer Science
Brown University
Providence, RI
gdk@cs.brown.edu

Abstract

General-purpose agents require fine-grained controls and rich sensory inputs to perform a wide range of tasks. However, this complexity often leads to intractable decision-making. Traditionally, agents are provided with task-specific action and observation spaces to mitigate this challenge, but this reduces autonomy. Instead, agents must be capable of building state-action spaces at the correct abstraction level from their sensorimotor experiences. We leverage the structure of a given set of temporally-extended actions to learn abstract Markov decision processes (MDPs) that operate at a higher level of temporal and state granularity. We characterize state abstractions necessary to ensure that planning with these skills, by simulating trajectories in the abstract MDP, results in policies with bounded value loss in the original MDP. We evaluate our approach in goal-based navigation environments that require continuous abstract states to plan successfully and show that abstract model learning improves the sample efficiency of planning and learning.

1 Introduction

Reinforcement learning (RL) is a promising framework for embodied intelligence because of its flexibility, generality, and online nature. Recently, RL agents have learned to control complex control systems: stratospheric balloons (Bellemare et al., 2020), nuclear fusion reactors (Degrave et al., 2022) and drones (Kaufmann et al., 2023). They have also mastered long-horizon decision-making problems such as the game of Go and chess (Silver et al., 2016; 2018). To achieve these results, each agent's state representation and action spaces were engineered to make learning tractable: the state space was designed to contain only relevant information for decision-making and the actions were restricted to task-relevant decisions to be made at every time step. This is in conflict with the state-action space required for versatile, general-purpose agents (e.g., robots), which must possess broad sensory data and precise control capabilities to handle a wide variety of tasks, such as playing chess, folding clothes or navigating a maze. Abstractions alleviate this tension: action abstractions enable agents to plan at larger temporal scales and state abstractions reduce the complexity of learning and planning; a combination of action and state abstraction results in a new task model that can capture the natural complexity of the task, instead of the complexity of the agent (Konidaris, 2019).

For instance, in model-based RL (MBRL; Sutton (1991); Deisenroth and Rasmussen (2011)), there is a long line of research that focuses on learning transition and reward models to plan by simulating trajectories. Many modern methods learn abstract state spaces (Ha and Schmidhuber, 2018; Zhang et al., 2019; Silver et al., 2018; Hafner et al., 2019; 2021; 2023) to handle complex observation spaces. However, they learn models for the primitive action spaces and work within the single-task setting. Recently, there has been interest in using MBRL for skill discovery: Hafner et al. (2022) learn a model in an abstract state space and learn a further abstraction over it to discover goals in a Feudal RL manner (Dayan and Hinton, 1992). Bagaria and Konidaris (2020) and Bagaria et al. (2021a;b),

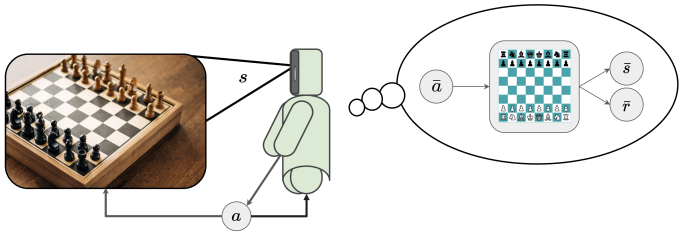


Figure 1: An agent needs to solve a task using its actuators and sensors (on the right). However, it requires an abstract model of the task (on the left) to reason at long time scales. This can be constructed by combining temporally-extended actions $\bar{a}$ with a compatible abstract state representation $\bar{s}$ that contains the minimal information necessary for planning with those actions.

instead, assume that the abstract state space is a graph and learn skills that connect nodes in that graph, effectively building a model that is both abstract in state and in actions. These approaches are ultimately limited because they assume a discrete abstract state space.

On the other hand, in robotics, high-level planning searches for sequences of *temporally-extended* actions (motor skills) to achieve a task. However, the agents needs a model to compute plans composed of their motor skills and this is typically given to the agent. To enable the agent to learn a model compatible with its motor skills from sensor data, Konidaris et al. (2018) propose novel semantics to automatically learn logical predicates from the agent observation space that support task planning with PDDL (Planning Domain Definition Language; Fox and Long (2003); Younes and Littman (2004)). Moreover, they provide theoretical guarantees for learning predicates that support sound task planning. In a similar vein, Ugur and Piater (2015a;b) and Ahmetoglu et al. (2022) propose to cluster the effects of motor skills to build discrete symbols for planning. Similarly, Asai et al. (2022) introduce a discrete VAE (Variational Auto-encoder; Kingma and Welling (2013)) approach to leverage modern deep networks for grounding PDDL predicates and action operators from complex observations. While these approaches consider temporally-extended actions and are promising for planning problems where the appropriate state abstractions are discrete, they are not applicable when planning with the available high-level actions requires a continuous state representation.

Instead, we are interested in learning state abstractions that are continuous, compatible with modern deep learning methods, and that guarantee value-preserving planning with a set of given skills. Specifically, we focus on building abstract world models in the form of Markov decision processes (MDPs) that have abstract state and action spaces and, in contrast to previous approaches, provide a principled approach to characterize the abstract state space that ensures that planning in simulation with this abstract model produces a policy with expected value equal to that we would get by planning if we had access to the real MDP. In summary, we (1) introduce the necessary and sufficient conditions for constructing an abstract Markov decision process sufficient for value-preserving planning for a given set of skills; (2) introduce an information maximization approach compatible with contemporary deep learning techniques, ensuring a bounded value loss when planning using the abstract model; and finally, (3) provide empirical evidence that these abstract models support effective planning with off-the-shelf deep RL algorithms in goal-based tasks (Mujoco Ant mazes (Fu et al., 2020) and Pinball (Konidaris and Barto, 2009) from pixels).

2 Background and Notation

Markov Decision Processes A continuous state, continuous action Markov decision process (MDP) (Puterman, 2014) is defined as the tuple $M = (\mathcal{S}, \mathcal{A}, T, R, p_0, \gamma)$ where $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ is the state space and $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ is the action space ($d_s, d_a \in \mathbb{N}$), $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$[1] denotes is the transition kernel

[1] $\Delta(\cdot)$ indicates the set of probability measures over a given set.

that represent the dynamics of the environment, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function bounded by $R_{Max} \in \mathbb{R}$, $\gamma \in [0, 1)$ is the discount factor and $p_0 \in \Delta(\mathcal{S})$ is the initial state distribution.

Planning and Bellman Equation A solution to an MDP is a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected return $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) | \, S_0 \sim p_0, \pi, T\right]$. An important family of solution methods for MDPs are based on the Bellman optimality principle and the Bellman equation. For a given policy π, the state-value function $v^\pi : \mathcal{S} \to \mathbb{R}$ is defined as $v^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) | S_0 = s, \pi\right]$. The state-value function represents the expected discounted return when following the policy π from state s. Importantly, the value function satisfy the following recursion, known as the Bellman equation, which is used in many current planning and learning methods for MDPs: $v^\pi(s) = \mathbb{E}\left[R(s, a) + \gamma \int_{\mathcal{S}} T(s'|s, a) v^\pi(s') ds'\right]$.

Action Abstractions Options (Sutton et al., 1999) are a formalization of temporally-extended actions, or *skills*, that are used by the agent to plan with a longer temporal scope than that allowed by primitive actions. An option o is defined by the tuple (I_o, π_o, β_o) where $I_o : \mathcal{S} \to \{0, 1\}$ is the initiation set, that is, the set of states in which the option can start execution; π_o is the policy function, and $\beta_o : \mathcal{S} \to [0, 1]$ is the termination probability function that indicates the probability of terminating the option execution at state s.

Expected-length Model of Options Generally, options are used to plan in Semi-Markov decision processes (SMDP; Sutton et al. (1999)), in which modelling jointly the option's dynamics T and duration τ as $T_\gamma(s'|s, o) = \sum_{\tau=0}^{\infty} \gamma^\tau \Pr(S_\tau = s', \beta(s_\tau)|S_0 = s, o)$ and its reward as $R(s, o) = \mathbb{E}_\tau \left[\sum_{t=0}^{\tau-1} \gamma^t R(S_t, A_t)|s, o\right]$, result in the Multi-time model of options. However, we will use a simpler and more practical model of option's dynamics: the expected-length model of options (Abel et al., 2019). In this case, the option's duration is modeled independently from the next-state distribution. More precisely, let $\tilde{\tau}_o$ be the average number of timesteps taken to execute the option o, then $T_\gamma(s'|s, o) = \gamma^{\tilde{\tau}_o} p(s'|s, o)$ where $p(s'|s, o)$ is the probability density function over the next-state observed when the option is executed as a black-box skill.

State Abstractions and Probabilistic Groundings State abstractions (or state aggregation) have commonly been defined in the form of non-injective functions $f : \mathcal{S} \to \bar{\mathcal{S}}$ where $\bar{\mathcal{S}}$ is an abstract state space. Recently, Konidaris et al. (2018) propose probabilistic groundings to define a new class of state abstractions. These groundings are defined by $G : \bar{\mathcal{S}} \to \Delta(\mathcal{S})$ and, contrary to state aggregation approaches, these can have overlapping support. That is, for a state s and abstract states $\bar{s}^1$ and $\bar{s}^2$, we can have that $G_{\bar{s}^1}(s) > 0$ and $G_{\bar{s}^2}(s) > 0$. In state aggregation methods, one state has just one abstract state to map to. Therefore, this provides a more expressive framework to build abstractions.

3 Value-preserving Abstract MDPs

To plan with a set of options, we must build a model of their effects. In this section, we formalize this model as an MDP with the following characteristics: (1) **Action Abstraction**, the action space is the set of task-relevant temporally-extended skills (i.e., the ground actions are not used for planning); (2) **State Abstraction**, because the set of skills operate at a higher-level of abstraction, the observation space will contains more information than required to plan with the skills; (3) **Sufficient for Planning**, the model must support computing a plan with the option set for task-specific rewards. In the case of abstract MDPs, the abstract model must guarantee *accurate* trajectory simulations to leverage the planning and RL algorithms developed for MDPs.

3.1 Ground and Abstract MDPs

We start by defining the ground MDP M, the environment that the agent observes by only executing the options.

Definition 3.1 (Ground MDP)**.** Let $\mathcal{O}$ be a set of options defined over the agent's state-action space. The ground MDP is $M = (\mathcal{S}, \mathcal{O}, T, R, \gamma, \tau, p_0)$. $T(s'|s, o)$ is the next-state probability density function seen by the agent when executing option o at s and its accumulated discounted reward is

$R(s,o) = \mathbb{E}_\tau \left[\sum_{t=1}^{\tau} \gamma^t R(S_t, A_t)|s,o\right]$, and $\tau : \mathcal{S} \times \mathcal{O} \to [0,\infty)$ is the expected option's execution time of option o when initiated at state s.[2]

Definition 3.2 (Abstract MDP)**.** The abstract MDP is $\bar{M} = (\bar{\mathcal{S}}, \mathcal{O}, \bar{T}, \bar{R}, \gamma, \bar{\tau}, \bar{p_0})$ where $\bar{\mathcal{S}}$ is the abstract state space, $\bar{T} : \bar{\mathcal{S}} \times O \to \Delta(\bar{S})$ is the abstract transition kernel, $\bar{R} : \bar{\mathcal{S}} \times \mathcal{O} \to \mathbb{R}$ is the abstract reward function, γ is the discount factor, $\bar{\tau} : \bar{\mathcal{S}} \times \mathcal{O} \to [0,\infty)$ is the option's duration model and $\bar{p}_0$ is the initial abstract state distribution.

Given that the objective is to compute plans in the abstract model, we will only consider policies of the form $\pi : \bar{\mathcal{S}} \to \mathcal{O}$ in the rest of the paper. Moreover, to connect the abstract MDP to the ground MDP, we use a grounding function defined in terms of probability density functions, as introduced by Konidaris et al. (2018). The grounding of an abstract state $\bar{s}$ is defined by the probability of the agent being in a state s.

Definition 3.3 (Grounding function)**.** Let M be a ground MDP and $\bar{M}$ be an abstract MDP. A grounding function $G : \bar{\mathcal{S}} \to \Delta(\mathcal{S})$ maps $\bar{s}$ to probability measures over $\mathcal{S}$ of M. Given an abstract state $\bar{s}$, we denote by $G_{\bar{s}}$ its grounding probability density. We will denote the tuple $(M, \bar{M}, G)$ as a grounded abstract model.

3.2 The Dynamics Preserving Abstraction

Our goal is to build an abstract model that enables the agent to simulate trajectories as though it had access to a simulator of the ground model. To achieve this, we establish two key distributions: the future state distribution and the grounded future state distribution.

Definition 3.4 (Future State Distribution)**.** Let the tuple $(M, \bar{M}, G)$ be a grounded abstract model. Let the future state distribution be B_t, and defined recursively as follows,

$$B_0(s_0) = p_0(s_0);$$
$$B_t(s_t, ..., s_0|o_0, ..., o_{t-1}) = T(s_t|s_{t-1}, o_{t-1}) B_{t-1}(s_{t-1}, ..., s_0|o_0, ..., o_{t-2});$$

and the grounded future state distribution $\bar{B}_t$ is the estimate obtained by grounding the estimate obtained by simulating trajectories in the abstract model $\bar{M}$

$$P(s_t, \bar{s}_t, ..., s_0, \bar{s}_0|o_0, ..., o_{t-1}) = G_{\bar{s}_t}(s_t)\bar{T}(\bar{s}_t|\bar{s}_{t-1}, o_{t-1}) P_{t-1}(s_{t-1}, \bar{s}_{t-1}, ..., s_0, \bar{s}_0|o_0, ..., o_{t-2});$$
$$\bar{B}_t(s_t, ..., s_0|o_0, ..., o_{t-1}) = \int P(s_t, \bar{s}_t, ..., s_0, \bar{s}_0|o_0, ..., o_{t-1}) d\bar{s}_0 ... \bar{s}_t;$$

Hence, we say that when $B_t(s_t, \ldots, s_0|o_0, ..., o_{t-1}) = \bar{B}_t(s_t, \ldots, s_0|o_0, ..., o_{t-1})$, then simulating a trajectory in the abstract model is the same as in the ground model. To satisfy this, we can build an abstract model based on dynamics-preserving abstractions.[3]

Definition 3.5 (Dynamics Preserving Abstraction)**.** Let ϕ be a mapping $\phi : \mathcal{S} \to \mathcal{Z} \subseteq \mathbb{R}^{d_z}$ for some dimension $d_z \in \mathbb{N}$, typically with $d_z \ll d_s$. If for all $o \in \mathcal{O}$ and all $s \in \mathcal{S}$ that are reachable with probability greater than 0, the following holds,

$$T(s'|s,o) = T(s'|\phi(s), o); \tag{1}$$
$$\Pr(I_o = 1|s) = \Pr(I_o = 1|\phi(s)); \tag{2}$$

where, I_o is an indicator variable corresponding to the option's initiation set. Then, we say that ϕ is dynamics-preserving. That is, the information in $\phi(s)$ is sufficient to predict the option's effect and determine if an option is executable.

This is similar to model-preserving abstractions (Li et al., 2006) and bisimulation (Givan et al., 2003; Ferns et al., 2004). However, 1) it is stronger in the sense that z must be a sufficient statistic for next-state prediction, and more importantly, 2) this does not impose a condition over the ground

[2]The ground MDP would be an SMDP if we used the multi-time model of options (Sutton et al., 1999).
[3]We defer all proofs to Appendix A.1.

reward function. Because we want to build an abstract model to be re-used for task-specific rewards (as we will see in Section 4.3), the ground reward function is considered as a way to measure the cost (negative reward) of executing a skill—retaining Markovianity with respect to the ground reward function would limit how much information can potentially be abstracted away.

We will now build a sensible abstract MDP $\bar{M}$, as follows. Let $\phi : \mathcal{S} \to \mathcal{Z}$ be a dynamics-preserving abstraction. Given that $T(s'|s,o) = T(s'|z,o)$, where $\phi(s) = z$, then we can build a transition function in $\mathcal{Z}$-space, $T(z'|z,o)$, and a grounding function G, that can let us reconstruct $T(s'|z,o)$.

$$
\begin{aligned}
p_0(z) &= \int p_0(s)\mathbb{1}[\phi(s) = z]ds; \\
T(z'|z,o) &= \int T(s'|z,o)\mathbb{1}[\phi(s') = z']ds'; \\
G(s'|z,o,z') &= \begin{cases} \frac{p_0(s')\mathbb{1}[\phi(s')=z']}{p_0(z')} & \text{if } z' \text{is an initial state (there is not previous } (z,o)) \\ \frac{T(s'|z,o)\mathbb{1}[\phi(s')=z']}{T(z'|z,o)} & \text{otherwise} \end{cases};
\end{aligned}
$$

Given that just knowing z is not enough to determine its grounding distribution, we can build an abstract state space $\bar{\mathcal{S}} \triangleq \mathcal{Z} \times \mathcal{O} \times \mathcal{Z}$ of transition tuples—with special values $z_\perp$ and $o_\perp$ to form $\bar{s}_0 = (z_\perp, o_\perp, z_0)$ for initial abstract states. Let $\bar{s} = (\hat{z}, \hat{o}, \hat{z}')$ and $\bar{s}' = (z, o, z')$ be two abstract states in $\bar{\mathcal{S}}$, we define the abstract MDP functions in this new $\bar{\mathcal{S}}$, as follows.

$$
\begin{aligned}
G_{\bar{s}}(\cdot) &= G(\cdot|\hat{z}, \hat{o}, \hat{z}'); \\
\bar{T}(\bar{s}'|\bar{s}, o) &= \begin{cases} T(z'|z,o) & \text{if } \hat{z}' = z \\ 0 & \text{otherwise} \end{cases}; \\
\bar{R}(\bar{s}, o) &= \mathbb{E}_{s\sim G_{\bar{s}}}\left[R(s,o)\right]; \qquad \bar{\tau}(\bar{s}, o) = \mathbb{E}_{s\sim G_{\bar{s}}}\left[\tau(s,o)\right];
\end{aligned}
$$

That is, if the tuples corresponding to $\bar{s}$ and $\bar{s}'$ are not compatible, we define its transition probability as 0, and we define the abstract reward and abstract option's execution length as their corresponding expected values under the grounding function. Finally, the following theorem formally states that this construction is sound.

Theorem 3.6. *Let the tuple $(M, \bar{M}, G)$ be a grounded abstract model and a function $\phi : \mathcal{S} \to \mathcal{Z} \subseteq \mathbb{R}^{d_z}$. The model satisfies that $B_t(\cdot \mid o_0, ..., o_{t-1}) = \bar{B}_t(\cdot \mid o_0, ..., o_{t-1})$ if and only if ϕ is dynamics-preserving.*

This theorem states that if we learn a dynamics-preserving abstraction, we can simulate accurate trajectories in the abstract model. Therefore, planning in the abstract model is accurate, in the sense, that the value of an abstract state $v^\pi(\bar{s})$ computed using the abstract model is the same as the one would get by generating trajectories in the ground MDP and computing the expected value under grounding G, $\mathbb{E}_{s\sim G_{\bar{s}}}[v^\pi(s)]]$.

Corollary 3.7. *Let the tuple $(M, \bar{M}, G)$ be a grounded abstract model. If the dynamics preserving property holds then the value of policy π computed in abstract model $\bar{M}$ satisfies that $v^\pi(\bar{s}) = \mathbb{E}[v^\pi(s)|s \sim G_{\bar{s}}]$. That is, the grounded abstract model preserves the expected value under the grounding G.*

Proof. Given that we have that, by definition, $T(s'|s,o) = T(s'|\bar{s},o) = \mathbb{E}_{\bar{s}'\sim\bar{T}(\cdot|\bar{s},o)}[G_{\bar{s}'}(s)]$. It follows that

$$
\begin{aligned}
\mathbb{E}_{s\sim G_{\bar{s}}}[v^\pi(s)] &= \mathbb{E}_{s\sim G_{\bar{s}}}\left[\mathbb{E}_{o\sim\pi}\left[R(s,o) + \mathbb{E}_{s'\sim T(s'|s,o)}\left[\gamma^\tau v^\pi(s')\right]\right]\right] \\
&= \mathbb{E}_{o\sim\pi}\left[\mathbb{E}_{s\sim G_{\bar{s}}}\left[R(s,o)\right] + \mathbb{E}_{s\sim G_{\bar{s}}, s'\sim T(s'|s,o)}\left[\gamma^\tau v^\pi(s')\right]\right] \\
&= \mathbb{E}_{o\sim\pi}\left[\bar{R}(\bar{s},o) + \mathbb{E}_{\bar{s}'\sim\bar{T}(\cdot|\bar{s},o)}\mathbb{E}_{s'\sim G_{\bar{s}}}\left[\bar{\gamma} v^\pi(s')\right]\right] \\
&= \mathbb{E}_{o\sim\pi}\left[\bar{R}(\bar{s},o) + \mathbb{E}_{\bar{s}'\sim\bar{T}(\cdot|\bar{s},o)}\left[\bar{\gamma} v^\pi(\bar{s}')\right]\right] = v^\pi(\bar{s}).
\end{aligned}
$$

□

The Skills to Symbols framework (Konidaris et al., 2018) introduces the strong subgoal property to build grounded discrete symbols for sound classical planning. The next corollary proves that the strong subgoal is a special case of the dynamics preserving property when the appropriate abstraction function has finite co-domain. Therefore, we can build discrete dynamics preserving models if and only if the strong subgoal property holds.

Corollary 3.8. *Let the tuple $(M, \bar{M}, G)$ be a grounded abstract model. Let the strong subgoal property (Konidaris et al., 2018) for an option o be defined as, $Pr(s'|s,o) = Pr(s'|o)$. The dynamics preserving property holds with a finite abstract state space $\mathcal{Z} = [N]$ for some $N \in \mathbb{N}$ if and only if the strong subgoal property holds.*

4 Learning the Abstract Model

4.1 Information Maximization to Learn a Dynamics-Preserving ϕ

The mutual information (MI) between random variables X and Y, $MI(X;Y)$, measures the information that each variable holds about the other. We are interested in finding a function ϕ that is dynamics-preserving such that we can build our abstract MDP. By Definition 3.5, we want to learn $\phi(s)$ that is maximally predictive of the effect of o when executed in s and to predict if option o is executable. That is, we want to maximize the following:

$$\max_{\phi \in \Phi} MI(S', I; \phi(S), O) \equiv \max_{\phi \in \Phi} MI(S'; \phi(S), O) + MI(I; \phi(S)), \tag{3}$$

where Φ is a class of functions that map the high-dimensional ground states to lower-dimensional space. I is binary random variable for the initiation set prediction. S', S, O are random variables over the ground states $\mathcal{S}$ and the options set $\mathcal{O}$.

In general, by the data processing inequality, $MI(S'; \phi(S), O)$ is upper-bounded by $MI(S'; S, O)$. Therefore, we can show that optimizing the above objective results in a bounded value loss when using the abstract model to plan. To see this, we first note that by compressing through ϕ, we lose information $\Delta MI \triangleq MI(S'; S, O) - MI(S'; Z, O)$, where $Z = \phi(S)$, in the transition dynamics simulation. We show that,

$$\Delta MI \overset{(a)}{=} \mathbb{E}_{p(s)} \left[D_{KL} \left(T(s'|s,o) || \tilde{T}(s'|z,o) \right) \right] \overset{(b)}{\geq} 2 \ln 2 \cdot \mathbb{E}_{p(s)} \left[\| T(s'|s,o) - \tilde{T}(s'|z,o) \|_1^2 \right].$$

where $p(s)$ is a distribution over s that will depend on the data collection policy and (a) follows from the definition of the KL divergence and (b) from the well-known bound relating the KL divergence and L1 norm[4]. Therefore, the error in the learned transition dynamics is minimized by our objective and this implies, by the following theorem, that this objective also minimizes the value loss resulting from the approximation.

Theorem 4.1 (Value Loss Bound)**.** *Let $(M, \bar{M}, G)$ be a grounded abstract model and $\tilde{T}(s'|\bar{s}, o) = \int G_{\bar{s}'}(s') \bar{T}(\bar{s}'|\bar{s}, o) d\bar{s}'$ be the approximate transition dynamics from the grounded model. If the following conditions hold for all $o \in \mathcal{O}$ and all $s \in \mathcal{S}$ with $G_{\bar{s}}(s) > 0$: (1) $\| T(s'|s,o) - \tilde{T}(s'|\bar{s},o) \|_1^2 \leq \epsilon_T$, and (2)$|R(s,o) - \bar{R}(\bar{s},o)|^2 \leq \epsilon_R$; then, for any policy π,*

$$|Q^\pi(s,o) - Q^\pi(\bar{s},o)| \leq \frac{\sqrt{\epsilon_R} + \gamma V_{Max} \sqrt{\epsilon_T}}{1-\gamma}.$$

4.2 Contrastive Abstract Model Learning

We maximize the previous Infomax objective (3) as follows. The term $MI(I;Z)$ reduces to a cross entropy loss, so we will focus on estimating the term $MI(S'; Z, O)$: we can prove that maximizing both sides of the identity $MI(Z'; Z, O) = (MI(S'; Z') - MI(S'; Z'|Z, O))$ implicitly

[4] $D_{KL}(P,Q) \geq 2 \ln 2 \cdot \|P - Q\|_1^2$

maximizes $MI(S'; Z, O)$ (see extended derivation details in Appendix A.2). Intuitively, the first term $MI(Z'; Z, O)$ makes z' predictable from knowing the option executed and the previous z. The second term avoids collapsing ϕ to a trivial solution: maximizing $MI(S'; Z') - MI(S'; Z'|Z, O)$ makes ϕ retain information about the ground state s (avoiding collapse of the representation) that is maximally predicted by the previous (z, o).

We choose to maximize these mutual information terms contrastively using InfoNCE (Oord et al., 2018) to avoid making assumptions about tractable density models (other MI estimators (Poole et al., 2019; Alemi et al.; Belghazi et al., 2018) can be used). Using these estimators allows the model to implicitly learn complex grounding functions that improve the quality of the abstract state space. Note that by using InfoNCE for the terms above, this algorithm corresponds to Temporal Predictive Coding (TPC; Nguyen et al. (2021)) which proposes abstract states without reconstruction objectives. Therefore, our formulation corresponds to the TPC algorithm in the degenerate case of options being the primitive actions.[5]

Algorithm 1 Planning and Learning with an Abstract Model

Require: Agent π, Ground Environment M, Abstract Model $\bar{M}$, Goal $\mathcal{G}$

1: Initialize dataset $\mathcal{D}$ by rolling out N trajectories
2: $\bar{M} \leftarrow$ PretrainAbstractMDP($\mathcal{D}$)
3: $\bar{M} \leftarrow$ MakeTaskMDP($\bar{M}$, $\mathcal{G}$)
4: **while** true **do**
5: $\quad\mathcal{D} \leftarrow$ Roll out for L steps.
6: $\quad$**if** H steps have passed **then**
7: $\quad\quad\bar{M} \leftarrow$ TrainModel($\bar{M}$,$\mathcal{D}$)
8: $\quad\quad\pi \leftarrow$ TrainAgentImagination($\bar{M}$, π)
9: $\quad$**end if**
10: **end while**

In practice, we assume that we have access to a dataset of transition samples $\mathcal{D} = \{(s_i, o_i, r_i^\gamma, s_i', \tau_i, I_i)\}_{i=1}^N$ that correspond to the execution of option o_i from state s_i, terminating in s_i' with a duration of τ_i and accumulated return $r_i^\gamma = \sum_{t=0}^{\tau_i - 1} \gamma^t r_t$. I_i corresponds to the initiation sets of all options in state s_i. This dataset might be initialized by rolling out trajectories with a random agent and further enhanced during the agent's learning (see Algorithm 1).

We propose to learn the abstract model $M_\theta^\phi = (T_\theta^\phi, R_\theta, I_\theta^\phi, \tau_\theta)$ based on the abstraction ϕ parameterized by a function approximator f_ϕ. Notice, that because we need to guarantee good initiation sets by $MI(I; \phi(S))$, the initiation set loss also affects the learning of f_ϕ:

$$\begin{aligned}
\mathcal{L}_\phi &= -MI_\phi(Z'; Z, O) - MI_\phi(S'; Z'); \\
\mathcal{L}_{\theta,\phi}^I &= -\log I_\theta^\phi(I_i | f_\phi(s_i)); \\
\mathcal{L}_{\theta,\phi}^T &= -\log \bar{T}_\theta(f_\phi(s_i') | f_\phi(s_i), o_i);
\end{aligned}$$

Therefore, $\mathcal{L}_\phi$, $\mathcal{L}_{\theta,\phi}^I$ and $\mathcal{L}_{\theta,\phi}^T$ are used to learn the abstraction function f_ϕ. Moreover, to compensate for any imbalances in the data, we use a weighted negative log-likelihood loss for the initiation loss to learn an initiation classifier to be used during planning. To learn the rest of the model, we consider f_ϕ fixed and minimize the following losses and consider samples of the form $(s_{i-1}, o_{i-1}, s_i, o_i, r_i^\gamma, \tau_i)$ which can be obtained by slicing trajectories appropriately. We map them considering f_ϕ and minimize the following,

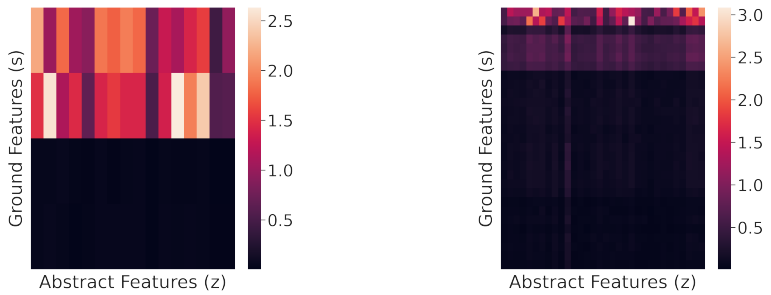
$$\mathcal{L}_\theta^R = (R_\theta(z_{i-1}, o_{i-1}, z_i, o_i) - r_i^\gamma)^2; \qquad \mathcal{L}_\theta^\tau = (\tau_\theta((z_{i-1}, o_{i-1}, z_i, o_i) - \tau_i)^2;$$

Finally, we minimize $\mathcal{L}_{\theta,\phi} = \beta_{\text{info}}\mathcal{L}_\phi + \beta_I \mathcal{L}_{\theta,\phi}^I + \beta_T \mathcal{L}_\theta^T + \beta_R \mathcal{L}_\theta^R + \beta_\tau \mathcal{L}_\theta^\tau$. In our experiments, all constants were $\beta_{\text{info}} = \beta_I = \beta_T = \beta_R = \beta_\tau = 1$.

4.3 Goal-based Planning with an Abstract Model

Consider a goal set $\mathcal{G} \subset S$ and $\mathcal{G}_\phi \subset \mathcal{Z}$, its mapping to $\mathcal{Z}$. In order to define the task MDP $M_\mathcal{G}$ (Algorithm 1, Line 3) for the agent to plan in, we define the task reward function for abstract state

[5] Extended discussion in Appendix A.2



(a) Pinball (b) Medium Antmaze

Figure 3: MI matrix: ground features s are in the vertical axis and abstract features z are in the horizontal axis. High MI (first two rows) corresponds to the position of the ball or the ant.

$\bar{s} = (\hat{z}, \hat{o}, z)$ as $R_{\mathcal{G}}(\bar{s}, o) = R_\theta(\bar{s}, o) + R_{\text{task}}\mathbb{1}[z \in \mathcal{G}_\phi]$ where R_{task} is the goal reward. The first term can be interpreted as the base cost/reward of executing a skill while the second term indicates to the agent the task-specific rewarding states. Moreover, we augment the transition dynamics and set all $z \in \mathcal{G}_\phi$ as terminating states by setting $\bar{T}_{\mathcal{G}}(z_{\text{done}}|z, o) = \mathbb{1}[z \in \mathcal{G}_\phi]$. The agent uses the task MDP $\bar{M}_{\mathcal{G}}$ to simulate trajectories and improves its policy (Algorithm 1, Line 8) and it can rollout the policy in the environment to collect new data (Algorithm 1, Line 7) that further improves the abstract model.

5 Experiments

Pinball environment (Konidaris and Barto, 2009) This domain has a continuous state space with position vector $(x, y) \in [0, 1]^2$ and velocities $(\dot{x}, \dot{y}) \in [-1, 1]^2$. As opposed to its original formulation, we consider a variant with continuous actions that decrease or increase the velocity by $\Delta(\dot{x}, \dot{y}) \in [-1, 1]^2$. Moreover, we also consider the top view pixel observation of the environment as the agent's observation. As options, we handcrafted position controllers implemented as PID controllers that move the ball in the coordinate directions by a fixed step size.

Antmaze We consider the problem of controlling a Mujoco (Todorov et al., 2012; Fu et al., 2020) Ant to navigate through a maze. The state space is a 29-dimensional vector that contains the position of the ant in the maze and the ant's proprioception. We consider the Medium Play maze as defined by Fu et al. (2020). We use 8 options learned using TD3 (Fujimoto et al., 2018) that move the ant in the coordinate directions (north, south, east, west and the diagonal directions) in the maze by a fixed step size.

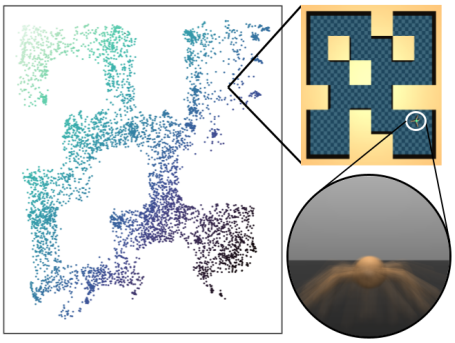


Figure 2: **Medium Antmaze**. 2D MDS projection of the learned ϕ: it learns to represent the position in the maze. The average grounding shows possible configurations of the ant joints when it is in the represented position.

5.1 Abstract State Space Preserves Relevant Information for Planning

Our main hypothesis is that abstract actions drive state abstraction because the information needed to plan with a structured option set will be less than the ground perception space of the agent. To quantify this, we measure the information contained in the abstract state space about the ground features by estimating the MI using non-parametric methods based on k-nearest neighbors (Kozachenko and Leonenko, 1987). We use Scikit-learn implementation (Pedregosa et al., 2011). In Figure 3a, we show the MI matrix between

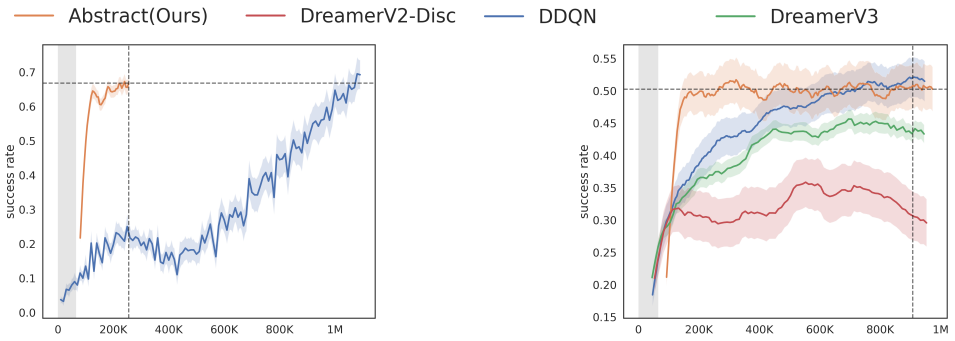
Pinball's ideal features (position and velocities) and the learned features from the pixel observations. For Antmaze (Figure 3b), we purposely over-parameterized the abstract space to give enough capacity to learn the full observation, if necessary. However, we can see that features that are not necessary for planning with the skills are effectively abstracted away. In the case of Pinball only the first two dimensions corresponding to the ball position have high MI. In the Antmazes, similarly, the first 7 dimensions have the highest MI which corresponds to position in the maze (first two dimensions) and orientation of the ant's torso. Qualitatively, we can visualize the learned abstract state space using Multidimensional Scaling (MDS; Borg and Groenen (2005)). Figure 2 shows the abstract state space learned for the Antmaze and it reveals the pattern of the coordinate positions of the ant in the maze. Additionally, we show grounded observations that correspond to an abstract state: the ant at the represented position in the maze with many different configurations of the joints and torso.

5.2 Planning with an Abstract MDP

To evaluate the effectiveness of these models for multiple goal-based tasks, we pretrained abstract models and use them to plan in imagination using Double DQN (Van Hasselt et al., 2016): the DDQN agent rolls out imagined trajectories to improve its policy and then rolls it out in the ground environment to collect new data that is used to learn the task reward function (we keep fix the rest of the model). As our baseline, we use DDQN tuned to learn a policy with the same options but interacting with the ground MDP. In Figure 4, we show learning curves (success rate vs. ground environment steps) averaged over different goals and seeds. The error areas represent one standard deviation.

For the pinball domain we use pixel observations as input. In Figure 4a, we compare learning curves averaged over 8 goals and 5 seeds where the gray area represent the number of samples used for pretraining phase of the model. These curves show that planning in the abstract model achieves similar performance to the same agent learning directly in the ground MDP which showcases the gain obtained in terms of sample efficiency.

Figure 4b shows an analogous plot for Antmaze (9 goals and 5 seeds). In this domain we provide additional results for state-of-the-art model-based RL methods: DreamerV2 and DreamerV3 (Hafner et al., 2021; 2023). These methods have been shown to work in diverse domains by building (discrete) latent states based on reconstruction losses. However, their performance is limited in comparison to our abstract model: (1) notice that after the gray area our abstract model collects data only to improve the goal reward prediction, whereas the baselines continuously collect data that further improves their models which shows the sample efficiency afforded by our skill-driven abstraction, and (2) our simple DDQN agent learns faster in imagination that the more sophisticated planning agents of the baselines.



(a) Pinball

(b) Antmaze

Figure 4: Planning with an abstract model. Success rate v. Environment steps averaged over goals and 5 seeds. The gray area represents the offset for the steps needed to pre-train the model.

6 Related Works

Grounded Classical Planning Konidaris et al. (2018) present a skill-driven method for constructing PDDL predicates (Fox and Long, 2003; Younes and Littman, 2004) for classical planning. This family of work formally bridges the options framework to classical planning, and recent work have extended this framework to work with portable skills (James et al., 2020) and object-centric skills (James et al., 2022), and to ground natural language in robotics (Gopalan et al., 2017). Importantly, this framework offers guarantees that the learned grounded symbols support sound planning. A related body of work bridges deep learning with classical planning. Asai and Fukunaga (2018); Asai (2019); Asai et al. (2022) learn abstract binary representations to ground PDDL predicates and action operators from complex observations. Similarly, Ugur and Piater (2015a;b) approach the grounding problem by clustering action effects to create discrete symbols for planning, and Ahmetoglu et al. (2022) extends this approach to leverage deep learning methods. While these approaches manage to empirically work with complex observations, they do not offer formal guarantees that the symbols learned are sufficient for planning. Our approach, while not applied to classical planning, generalizes abstract state learning to continuous cases, it is compatible with the deep learning toolbox and it is theoretically principled.

Model-based RL and State Abstractions Learning MDP models from experience has been extensively studied (Sutton, 1991; Deisenroth and Rasmussen, 2011) for their benefits in generalization, sample efficiency, and knowledge transfer. Recent successful approaches use deep networks to handle complex observations spaces and long-term reasoning (Krishnan et al., 2015; Ha and Schmidhuber, 2018; Silver et al., 2018; Gregor et al., 2018; Buesing et al., 2018; Zhang et al., 2019; Hansen et al., 2022; 2023). An important challenge of this approach is learning an effective abstract state space and, most of them, have focused in learning abstract representations of complex observations based on reconstruction losses (Gregor et al., 2018; Buesing et al., 2018; Zhang et al., 2019; Hafner et al., 2019; 2021; 2023). In contrast, recent approaches have moved away from this idea and focused in minimal abstract state spaces relevant for acting such as value prediction (Silver et al., 2018; Grimm et al., 2020; Yue et al., 2023), Markov states (Gelada et al., 2019; Zhang et al., 2020; Allen et al., 2021; Nguyen et al., 2021), and controllability (Lamb et al., 2022). In fact, many of these explicitly use information maximization and information bottleneck approaches that are theoretically justified by our work.

From a theoretical point of view, there is extensive research to characterize the types of state abstractions (or state aggregation) (Li et al., 2006; Ferns et al., 2004; Castro and Precup, 2010) that are useful for RL. More recent work characterizes *approximate* state abstractions (Abel et al., 2016; 2018) that guarantee bounded value loss and the type of options that are compatible with a given state abstraction to guarantee value preservation (Abel et al., 2020).

Temporally-extended Models MDP models with skills have been recently considered in skill discovery research. Some work approach the problem assuming that the abstract state space is a graph and options are learned to reach the initiation set of another option (Bagaria and Konidaris, 2020; Bagaria et al., 2021a;b). Hafner et al. (2022) approaches the problem by building on the Dreamer algorithm (Hafner et al., 2019; 2021; 2023) and discover goals by abstracting over the learned abstract state. Similarly, Nair and Finn (2019) use generative models for subgoal generation and skill learning, and plan with a learned model in observation space. Other approaches learn forward dynamics models for skills discovered from an offline set of trajectories but do not abstract the state based on these skills (Freed et al., 2023; Shi et al., 2023; Zhang et al., 2023). While our method assumes that the options are given, it does not impose discrete constraints to the abstract state space, does not need to model the state dynamics at the finest time step, and it builds a principled abstract state space.

7 Conclusion

We introduce a method for learning abstract world models, designed to have agents with effective planning capabilities for goal-oriented tasks. Our core premise is that an agent must be capable of

building a reusable abstract model for planning with a given skill set. We do this in a principled manner by characterizing the state abstraction that guarantees that planning in simulation guarantees bounded value loss. In other words, planning with a learned abstract model is sufficient to compute a policy for the real-world environment.

Acknowledgments

We would like to thank Sam Lobel, Akhil Bagaria, Saket Tiwari, and other members in the IRL at Brown for useful discussions during the development of this project. Moreover, we would like to thank David Abel for his contribution to the Value Loss theorem. This project was funded by NSF grant #1955361, NSF CAREER #1844960 to Konidaris, ONR grant #N00014-22-1-2592. Partial funding for this work provided by The Boston Dynamics AI Institute ("The AI Institute").

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 10–19. PMLR, 2018.

David Abel, John Winder, Marie DesJardins, and Michael Littman. The expected-length model of options. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1951–1958, 2019.

David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In *International Conference on Artificial Intelligence and Statistics*, pages 1639–1650. PMLR, 2020.

A. Ahmetoglu, M.Y. Seker, J. Piater, E. Oztop, and E. Ugur. DeepSym: Deep symbol generation and rule learning for planning from unsupervised robot interaction. *Journal of Artificial Intelligence Research*, 75:709–745, 2022.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*.

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8229–8241, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

M. Asai. Unsupervised grounding of plannable first-order logic representation from images. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 583–591, 2019.

M. Asai and A. Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 6094–6101, 2018.

M. Asai, H. Kajino, A. Fukunaga, and C. Muise. Classical planning in deep latent space. *Journal of Artificial Intelligence Research*, 74:1599–1686, 2022.

A. Bagaria and G.D. Konidaris. Option discovery using deep skill chaining. In *Proceedings of the Eighth International Conference on Learning Representations*, 2020.

A. Bagaria, J. Senthil, and G.D. Konidaris. Skill discovery for exploration and planning using deep skill graphs. In *Proceedings of the Thirty-Eighth International Conference on Machine Learning*, 2021a.

A. Bagaria, J. Senthil, M. Slivinski, and G.D. Konidaris. Robustly learning composable options in deep reinforcement learning. In *IProceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021b.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

Lars Buesing, Theophane Weber, Sébastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Pablo Castro and Doina Precup. Using bisimulation for policy transfer in mdps. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1065–1070, 2010.

P. Dayan and G.E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems V*, 1992.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 162–169, Arlington, Virginia, USA, 2004. AUAI Press. ISBN 0974903906.

Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

Benjamin Freed, Siddarth Venkatraman, Guillaume Adrien Sartoretti, Jeff Schneider, and Howie Choset. Learning temporally AbstractWorld models without online experimentation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10338–10356. PMLR, 23–29 Jul 2023.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

N. Gopalan, M. desJardins, M.L. Littman, J. MacGlashan, S. Squire, S. Tellex, J. Winder, and L.S. Wong. Planning with abstract Markov decision processes. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, pages 480–488, 2017.

Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. In *International Conference on Learning Representations*, 2018.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 5541–5552, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462. Neural information processing systems foundation, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

N Hansen, X Wang, and H Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning, PMLR*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2023.

S. James, B. Rosman, and G.D. Konidaris. Learning portable representations for high-level planning. In *Proceedings of the Thirty-Seventh International Conference on Machine Learning*, pages 4682–4691, 2020.

S. James, B. Rosman, and G.D Konidaris. Autonomous learning of object-centric abstractions for high-level planning. In *Proceedings of the Tenth International Conference on Learning Representations*, 2022.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

G. Konidaris. On the necessity of abstraction. *Current Opinion in Behavioral Sciences*, 29:1–7, 2019.

G.D. Konidaris and A.G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1015–1023, 2009.

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

Lyudmyla F Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J Foster, Lekan P Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2022.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. 2006.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2019.

Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pages 8130–8139. PMLR, 2021.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Conference on Robot Learning*, pages 2262–2272. PMLR, 2023.

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

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

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.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

E. Ugur and J. Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2627–2633. IEEE, 2015a.

E. Ugur and J. Piater. Refining discovered symbols with multi-step interaction experience. In *Proceedings of the 15th IEEE-RAS International Conference on Humanoid Robots*, pages 1007–1012. IEEE, 2015b.

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.

Håkan LS Younes and Michael L Littman. Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects. *Techn. Rep. CMU-CS-04-162*, 2:99, 2004.

Yang Yue, Bingyi Kang, Zhongwen Xu, Gao Huang, and Shuicheng Yan. Value-consistent representation learning for data-efficient reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11069–11077, 2023.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2020.

Jingwei Zhang, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, Abbas Abdolmaleki, Dushyant Rao, Nicolas Heess, and Martin Riedmiller. Leveraging jumpy models for planning and fast learning in robotic domains. *arXiv preprint arXiv:2302.12617*, 2023.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pages 7444–7453. PMLR, 2019.

A Appendix

A.1 Proofs

Theorem A.1. *Let the tuple $(M, \bar{M}, G)$ be a grounded abstract model and a function $\phi : \mathcal{S} \to \mathcal{Z} \subseteq \mathbb{R}^{d_z}$. The model satisfies that $B_t(\cdot \mid o_0, ..., o_{t-1}) = \bar{B}_t(\cdot \mid o_0, ..., o_{t-1})$ if and only if ϕ is dynamics preserving.*

Proof. Let $\phi^{-1}(z) = \{s \in \mathcal{S} \mid \phi(s) = z\}$. We construct $\bar{T}$ and G such that it satisfies that,

$$\bar{T}(z'|z, o) = \int_{s' \in \phi^{-1}(z')} T(s'|z, o) ds';$$
$$G(s'|z, o, z') = \frac{T(s'|z, o)\mathbb{1}[\phi(s') = z']}{\bar{T}(z'|z, o)}$$

If the dynamics preserving property holds, we have that there exists a mapping ϕ such that $T(s'|s, o) = T(s'|\phi(s), o)$. Hence, by defining that abstract state as $\bar{s} = (z, o, z')$, we can build the grounded abstract model such that it follows that $B_t = \bar{B}_t$, by construction.

To prove the converse, we assume that $B_t = \bar{B}_t$.

Hence, by construction, we have that $P(s_t, ..., s_0|o_0, z_0, ..., o_{t-1}, z_{t-1}) = \prod_t P(s_t|o_0, z_0, ..., o_{t-1}, z_{t-1})$. Therefore, we have that

$$\begin{aligned}
\bar{B}_t(s_t, ..., s_0|o_0, ..., o_{t-1}) &= \int \prod_{i=0}^{t} P(s_i|o_0, z_0, ..., o_{i-1}, z_{t-1}) P(z_i, ..., z_0|o_0, ..., o_{i-1}) dz_0 ... z_t \\
&= \int \prod_{i=0}^{t} P(s_i|z_i, o_{i-1}) P(z_i, ..., z_0|o_0, ..., o_{i-1}) dz_0 ... z_t \\
&= \int \prod_{i=0}^{t} G(s_i|z_{i-1}, o_{i-1}, z_i) P(z_i, ..., z_0|o_0, ..., o_{i-1}) dz_0 ... z_t \\
&= \prod_{i=0}^{t} \int G(s_i|z_{i-1}, o_{i-1}, z_i) P(z_i, z_{i-1}|o_0, ..., o_{i-1}) dz_i z_{i-1} \\
&= \prod_{i=0}^{t} \int G(s_i|z_{i-1}, o_{i-1}, z_i) \bar{T}(z_i|z_{i-1}, o_{i-1}) P(z_{i-1}|o_0, ..., o_{i-2}) dz_i z_{i-1} \\
&= \prod_{i=0}^{t} \int \tilde{T}(s_i|z_{i-1}, o_{i-1}) P(z_{i-1}|o_0, ..., o_{i-2}) dz_{i-1}
\end{aligned}$$

$$\begin{aligned}
B_t(s_t, ..., s_0|o_0, ..., o_{t-1}) &= p_0(s_0) \prod_{i=1}^{t} T(s_i|s_{i-1}, o_{i-1}) \\
&= \prod_{i=1}^{t} T(s_i|s_{i-1}, o_{i-1}) P(s_{i-1}|o_0, ..., o_{t-2})
\end{aligned}$$

Hence, we must have that for all $s_{i-1} \in z_{i-1}$ and all $i \in [t]$ and $t \geq 0$

$$\int T(s_i|s_{i-1}, o_{i-1}) P(s_{i-1}|o_0, ..., o_{t-2}) ds_{i-1} = \int \tilde{T}(s_i|z_{i-1}, o_{i-1}) P(z_{i-1}|o_0, ..., o_{i-2}) dz_{i-1}$$

That is,

$$\begin{cases} P(s_0) = p_0(s_0) = \int G(s|z_0)p_0(z_0)ds & \text{for } t = 0 \\ P(s_1|o_0) = \int T(s_1|s_0, o_0)p_0(s_0)ds_0 = \int \tilde{T}(s_0|z_0, o_0)p_0(z_0)dz_0 & \text{for } t = 1 \end{cases}$$

By definition, $t = 0$ holds. For $t = 1$, we have

$$\begin{aligned} P(s_1|o_0) &= \int T(s_1|s_0, o_0)p_0(s_0)ds_0 \\ &= \int T(s_1|s_0, o_0)G(s_0|z_0)p_0(z_0)dz_0ds_0 \\ &= \int \tilde{T}(s_1|z_0, o_0)p_0(z_0)dz_0 \end{aligned}$$

which follows from the equation at $t = 0$. Hence, it must be true that for any $s_0 \in \phi^{-1}(z_0)$, for any z_0 with $p_0(z_0) > 0$.

$$\tilde{T}(s_1|z_0, o_0) = \int T(s_1|s_0, o_0)G(s_0|z_0)ds_0$$

We can see that for any $s_0 \in \phi^{-1}(z_0)$ such that $T(s_1|s_0, o_0) \neq \tilde{T}(s_1|z_0, o_0)$, the abstract model would commit a non-zero error in its prediction. Hence, it must be that $T(s_1|s_0, o_0) = \tilde{T}(s_1|z_0, o_0)$ for $s_0 \in \phi^{-1}(z_0)$.

Let the equations at time $t = i - 1$ and $t = i - 2$ hold, then

$$\begin{aligned} &P(s_i|o_0, ..., o_{i-1}) = \int T(s_i|s_{i-1}, o_{i-1})p_{i-1}(s_{i-1}|o_0, ...o_{i-2})ds_{i-1} \\ &= \int T(s_i|s_{i-1}, o_{i-1})\tilde{T}(s_{i-1}|z_{i-2}, o_{i-2})p_{i-2}(z_{i-2}|o_0, ..., o_{i-3})ds_{i-1}dz_{i-1}dz_{i-2} \\ &= \int T(s_i|s_{i-1}, o_{i-1})G(s_{i-1}|z_{i-2}, o_{i-2}, z_{i-1})\bar{T}(z_{i-1}|z_{i-2}, o_{i-2})p_{i-2}(z_{i-2}|o_0, ..., o_{i-3})ds_{i-1}dz_{i-1}dz_{i-2} \\ &= \int \tilde{T}(s_i|z_{i-1}, o_{i-1})p_{i-1}(z_{i-1}|o_0, ..., o_{i-2})dz_{i-1} \end{aligned}$$

Because $p_{i-1}(z_{i-1}|o_0, ..., o_{i-2}) = \int \bar{T}(z_{i-1}|z_{i-2}, o_{i-2})p_{i-2}(z_{i-2}|o_0, ..., o_{i-3})dz_{i-2}$ hold by construction of the abstract MDP, we need the following to hold.

$$\tilde{T}(s_i|z_{i-1}, o_{i-1}) = \int T(s_i|s_{i-1}, o_{i-1})G(s_{i-1}|z_{i-2}, o_{i-2}, z_{i-1})ds_{i-1}. \tag{4}$$

Therefore, as in the base case, we need that $\tilde{T}(s_i|z_{i-1}, o_{i-1}) = T(s_i|s_{i-1}, o_{i-1})$ for all $s_{i-1} \in \phi^{-1}(z_{i-1})$ that have $G(s_{i-1}|z_{i-2}, o_{i-2}, z_{i-1}) > 0$. Then, ϕ must be dynamics preserving.

□

Corollary A.2. *Let the tuple $(M, \bar{M}, G)$ be a grounded abstract model. Let the strong subgoal property (Konidaris et al., 2018) for an option o be defined as, $Pr(s'|s, o) = Pr(s'|o)$. The dynamics preserving property holds with a finite abstract state space $\mathcal{Z} = [N]$ for some $N \in \mathbb{N}$ if and only if the strong subgoal property holds.*

Proof. If the strong subgoal property holds, we have that $Pr(s'|s,o) = Pr(s'|o)$. Then, for any function $\phi : S \to \mathcal{Z}$, it holds that $P(s'|\phi(s),o) = P(s'|s,o)$.

Therefore, it is only important to be able to know if a given option is executable in a given abstract state. Therefore, we can construct the function $I_{\mathcal{O}}(s) = [I_0(s), ..., I_{|\mathcal{O}|}(s)]$ that returns a binary vector that indicates which options are executable in s.

Define the equivalence relation $s_0 \sim_{\mathcal{O}} s_1$ iff $I_{\mathcal{O}}(s_1) = I_{\mathcal{O}}(s_2)$. We can define the abstract state space as $Z \triangleq S/\sim_{\mathcal{O}}$, that is, the set of equivalent classes. Given that there at most $2^{|\mathcal{O}|} \in \mathbb{N}$ classes, then the abstract MDP is finite.

We assume that the dynamics preserving property holds and that the abstract state space Z is finite to prove the converse. Then, there exists $\phi : S \to \mathcal{Z}$ such that $P(s'|\phi(s),o) = P(s'|s,o)$ and $P(I_o = 1|s) = P(I_o = 1|\phi(s))$.

We can construct a factored $\phi(s) = [\phi_D(s), \phi_I(s)]$, such that, $P(s'|\phi(s),o) = P(s'|\phi_D(s),o)$ and $P(I_o = 1|\phi(s)) = P(I_o = 1|\phi_I(s))$.

If we define ϕ_I based on the function $I_{\mathcal{O}}$, as before, then ϕ_I maps to a set of at most $2^{|\mathcal{O}|}$ elements. As $\mathcal{Z} = \mathcal{Z}_D \times \mathcal{Z}_I$ is finite, then $\mathcal{Z}_D$ is also finite. Thus, we construct $Z_D = [M]$ and for each option o and equivalence class $m \in [M]$ options from each option o such that $Pr(s'|o_m) \triangleq Pr(s'|m,o)$. Then, the strong subgoal property holds for every o_m.

□

Proposition A.3. *Let ϕ be a dynamics-preserving abstraction and $\bar{s} = (\hat{z}, \hat{o}, z)$. For $\epsilon > 0$, if $\|G_z(s) - G_{\bar{s}}(s)\|_1^2 \leq \epsilon$, then there exists $\epsilon_T > 0$ and $\epsilon_R > 0$ such that $\|T(s'|s,o) - \tilde{T}(s'|z,o)\|_1^2 \leq \epsilon_T$ and $\|R(s,o) - \tilde{R}(z,o)\|_1^2 \leq \epsilon_R$.*

Proof. First, we prove that the bounded grounding error implies bounded transition distribution error. If ϕ is a dynamics abstraction, then we can learn $\tilde{T}(z'|z,o)$ and we have that $T(s'|s,o) = T(s'|z,o) = \int G_{\bar{s}}(s)\bar{T}(z'|z,o)dz'$ and its corresponding approximation $\tilde{T}(s'|z,o) = \int G_{z'}(s)\bar{T}(z'|z,o)dz'$

$$
\begin{aligned}
\|T(s'|s,o) - \tilde{T}(s'|z,o)\|_1 &= \left| \int \left(G'_{\bar{s}}(s)\bar{T}(z'|z,o) - G_{z'}(s)\bar{T}(z'|z,o) \right) dz' \right| \\
&\leq \int \bar{T}(z'|z,o)|G_{\bar{s}'}(s) - G_{z'}(s)|dz'ds \\
&\leq \sqrt{\epsilon}
\end{aligned}
$$

Analogously, we can bound the error of the reward function.

$$
\begin{aligned}
\|\bar{R}(z',o) - \tilde{R}(z',o)\|_1 &= \left| \int G_{\bar{s}'}(s)R(s,o)ds - \int G_{z'}(s)R(s,o)ds \right| \\
&\leq \int |G_{\bar{s}'}(s) - G_{z'}(s)|\,|R(s,o)|\,ds \\
&\leq RMax \int |G_{\bar{s}'}(s) - G_{z'}(s)|\,ds \\
&\leq RMax\sqrt{\epsilon}
\end{aligned}
$$

Then, it follows from Minkowski's inequality that

$$
\begin{aligned}
\|R(s,o) - \bar{R}(z',o)\|_1 &= \|R(s,o) - \tilde{R}(z',o) + \tilde{R}(z',o) - \bar{R}(z',o)\|_1 \\
&\leq \|R(s,o) - \tilde{R}(z',o)\|_1 + \|\tilde{R}(z',o) - \bar{R}(z',o)\|_1 \\
&\leq \sqrt{\epsilon} + RMax\sqrt{\epsilon} = \sqrt{\epsilon_R}
\end{aligned}
$$

□

Theorem A.4 (Value Loss Bound)**.** *Let $(M, \bar{M}, G)$ be a grounded abstract model and $\tilde{T}(s'|\bar{s}, o) = \int G_{\bar{s}'}(s')\bar{T}(\bar{s}'|\bar{s}, o)d\bar{s}'$ be the approximate transition dynamics from the grounded model. If the following conditions hold for all $o \in \mathcal{O}$ and all $s \in \mathcal{S}$ with $G_{\bar{s}}(s) > 0$: (1) $\|T(s'|s, o) - \tilde{T}(s'|\bar{s}, o)\|_1^2 \leq \epsilon_T$, and (2)$|R(s, o) - \bar{R}(\bar{s}, o)|^2 \leq \epsilon_R$; then, for any policy π,*

$$|Q^\pi(s, o) - Q^\pi(\bar{s}, o)| \leq \frac{\sqrt{\epsilon_R} + \gamma VMax\sqrt{\epsilon_T}}{1 - \gamma}.$$

Proof. We proceed by induction on $Q_n^\pi(\bar{s}, o)$, where

$$v_0^\pi(\bar{s}) = \mathbb{E}_{s\sim\bar{s}}\left[v^\pi(s)\right], \tag{5}$$

$$Q_1^\pi(\bar{s}, o) = \int_{s\in\mathcal{S}} P(s)\left(R(s, o) + \gamma^\tau v_0^\pi(\bar{s}')\right) ds, \tag{6}$$

$$= \int_{s\in\mathcal{S}} P(s)\left(R(s, o) + \gamma^\tau \int_{s'\in\mathcal{S}} T^{s,o,s'} v^\pi(s')ds'\right) ds, \tag{7}$$

$$Q_i^\pi(\bar{s}, o) = \int_{s\in\mathcal{S}} P(s)\left(R(s, o) + \gamma^\tau v_{i-1}^\pi(\bar{s}')\right) ds, \tag{8}$$

with $\bar{s}' = T(\cdot \mid s, o)$. I use $P(s)$ as shorthand for $P(s \sim \bar{s})$ and $T^{s,o,s'}$ for $T(s' \mid s, o)$, and let

$$\epsilon_{Q,n} = \sum_{i=0}^{n} \sqrt{\epsilon_R} + \gamma^i\left(\text{VMax}\sqrt{\epsilon_T}\right). \tag{9}$$

Base Case: $Q^\pi \approx Q_1^\pi$.

$$Q^\pi(s, o) - Q_1^\pi(\bar{s}, o) \tag{10}$$

$$= R(s, o) + \gamma^\tau \int_{s'} T^{s,o,s'} v^\pi(s')ds' - \int_s P(s)\left(R(\bar{s}, o) - \gamma^\tau v_0^\pi(\bar{s}')ds\right), \tag{11}$$

$$= \underbrace{R(s, o) - R(\bar{s}, o)}_{\leq\sqrt{\epsilon_R}} + \gamma^\tau \int_{s'} T^{s,o,s'} v^\pi(s')ds' - \int_s P(s)\gamma^\tau v_0^\pi(\bar{s}')ds, \tag{12}$$

$$\leq \sqrt{\epsilon_R} + \gamma^\tau \int_{s'} T^{s,o,s'} v^\pi(s')ds' - \gamma^\tau \int_s P(s)\mathbb{E}_{s'\sim\bar{s}'}[v^\pi(s')]ds \tag{13}$$

$$\leq \sqrt{\epsilon_R} + \gamma^\tau \int_{s'} T^{s,o,s'} v^\pi(s')ds' - \gamma^\tau \int_s P(s) \int_{s'} P(s' \sim \bar{s}')v^\pi(s')ds'\ ds, \tag{14}$$

$$\leq \sqrt{\epsilon_R} + \gamma^\tau \int_{s'} T^{s,o,s'} v^\pi(s')ds' - \gamma^\tau \int_s P(s) \int_{s'} T^{s,o,s'} v^\pi(s')ds'\ ds, \tag{15}$$

$$\leq \sqrt{\epsilon_R} + \gamma^\tau \text{VMax} \underbrace{\int_{s'} T^{s,o,s'} - \int_s P(s)T^{s,o,s'} ds\ ds'}_{\leq\sqrt{\epsilon_T}}, \tag{16}$$

$$\leq \sqrt{\epsilon_R} + \gamma^\tau \text{VMax}\sqrt{\epsilon_T}. \tag{17}$$

This concludes the base case. □

Inductive Case: $Q^\pi \approx Q_n^\pi \implies Q^\pi \approx Q_{n+1}^\pi$. We assume that, for every $s \in \mathcal{S}$ and any o,

$$Q^\pi(s, o) - Q_n^\pi(\bar{s}, o) \leq \epsilon_{Q,n}, \tag{18}$$

and prove that

$$Q^\pi(s, o) - Q_{n+1}^*(\bar{s}, o) \leq \epsilon_{Q,n+1}. \tag{19}$$

By algebra,

$$
\begin{aligned}
& Q^{\pi}(s,o) - Q^{\pi}_{n+1}(\bar{s},o) && (20)\\
&= R(s,o) + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \int_{s} P(s)\left(R(s,o) + \gamma^{\tau} v^{\pi}_{n}(\bar{s}')\right)ds, && (21)\\
&= \underbrace{R(s,o) - R(\bar{s},o)}_{\leq\sqrt{\epsilon_R}} + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \gamma^{\tau}\int_{s} P(s) v^{\pi}_{n}(\bar{s}')ds, && (22)\\
&\leq \sqrt{\epsilon_R} + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \gamma^{\tau}\int_{s} P(s) v^{\pi}_{n}(\bar{s}')ds, && (23)\\
&= \sqrt{\epsilon_R} + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \gamma^{\tau}\int_{s} P(s) \underbrace{v^{\pi}_{n}(\bar{s}')}_{\geq \mathbb{E}_{s'\sim\bar{s}'}[v^{\pi}(s')] - \epsilon_{Q,n}} ds, && (24)
\end{aligned}
$$

$$
\begin{aligned}
&\leq \sqrt{\epsilon_R} + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \gamma^{\tau}\int_{s} P(s)\left(\mathbb{E}_{s'\sim\bar{s}'}[v^{\pi}(s')] - \epsilon_{Q,n}\right)ds, && (25)\\
&= \sqrt{\epsilon_R} + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \gamma^{\tau}\int_{s} P(s)\int_{s'} T^{s,o,s'} v^{\pi}(s')ds'\ ds + \gamma^{\tau}\epsilon_{Q,n}, && (26)\\
&= \sqrt{\epsilon_R} + \gamma^{\tau}\int_{s'} T^{s,o,s'} v^{\pi}(s')ds' - \gamma^{\tau}\int_{s'} \underbrace{\int_{s} P(s) T^{s,o,s'}}_{=T^{\bar{s},o,s'}} v^{\pi}(s')ds\ ds' + \gamma^{\tau}\epsilon_{Q,n}, && (27)\\
&\leq \sqrt{\epsilon_R} + \gamma^{\tau}\textsc{VMax}\underbrace{\int_{s'} T^{s,o,s'} - T^{\bar{s},o,s'} ds'}_{\leq\sqrt{\epsilon_T}} + \gamma^{\tau}\epsilon_{Q,n}, && (28)\\
&\leq \sqrt{\epsilon_R} + \gamma^{\tau}\textsc{VMax}\sqrt{\epsilon_T} + \gamma^{\tau}\epsilon_{Q,n}, && (29)\\
&\leq \sqrt{\epsilon_R} + \gamma\textsc{VMax}\sqrt{\epsilon_T} + \gamma\epsilon_{Q,n}, && (30)\\
&= \epsilon_{Q,n+1}. && (31)
\end{aligned}
$$

This concludes the inductive case. □

Thus, by induction and the convergence of the geometric series, for any s, o, π, we conclude that

$$
Q^{\pi}(s,o) - Q^{\pi}(\bar{s},o) \leq \frac{\sqrt{\epsilon_R} + \gamma\textsc{VMax}\sqrt{\epsilon_T}}{1-\gamma}. \tag{32}
$$

□

A.2 TPC is Dynamics Preserving

We start by considering that by learning an abstract state space such that $MI(S';Z,O)$ is maximized. The following decomposition based on the mutual information chain rule corresponds to the TPC algorithm (Nguyen et al., 2021). In the original paper, they work at the primitive action level and all actions available always, hence, there's no need to consider initiation sets.

$$
\begin{aligned}
MI(S',Z';Z,O) &\overset{(a)}{=} MI(S';Z,O) + \underbrace{MI(Z';Z,O|S')}_{=0};\\
&\overset{(b)}{=} MI(Z';Z,O) + \underbrace{MI(S';Z,O|Z')}_{(1)};\\
&\overset{(c)}{=} MI(Z';Z,O) + MI(S';Z,A) - MI(S';Z') + MI(S';Z'|Z,O);
\end{aligned}
$$

where (a) follows from the fact that give s' we can determine z', (b) follows from decomposing the term on the left-hand size and (c) from decomposing term (1).

The above implies that $MI(Z'; Z, O) = MI(S'; Z') - MI(S'; Z'|Z, O)$. Therefore, if we maximize both sides of this identity, we must have a latent space that preserve *only* the information of the state s' that is predictable from the previous (z, a) pair. $MI(Z'; Z, O)$ ensures that the next abstract state is predictable from the (z, o) tuple. $MI(S; Z)$ ensures that the abstract state has information about the ground state which is measured by $g(s|z)$.

$$MI(S; O) = \int p(s, z) \log \frac{g(s|z)}{p(s)} ds dz \tag{33}$$

The following decomposition shows the two extra terms required by the TPC algorithm to estabilize the optimization. Term (a) is the (differential) entropy of ϕ which tends to infinity for a deterministic function. This is solved by smoothing it with Gaussian noise of 0 mean and fixed standard deviation, as done in TPC. The second term (b) corresponds to the consistency term, that is, the transition function $p(z'|z, a)$ must have low entropy, which ensures that the abstract dynamics are learnt.

$$\begin{aligned}
M(S'; Z'|Z, O) &= \int p(s', z', z, o) \log \frac{p(s', z'|z, o)}{p(s'|z, o)p(z'|z, o)} ds' dz' dz do \\
&= \int p(s', z', z, o) \log \frac{p(z'|s')}{p(z'|z, o)} \\
&= \underbrace{\int p(s', z') \log p(z'|s') ds' dz'}_{(a)} - \underbrace{\int p(z', z, o) \log p(z'|z, o) dz' dz do}_{(b)}
\end{aligned}$$

By maximizing $MI(Z'; Z, O)$ and $MI(S'; Z')$ using InfoNCE (Oord et al., 2018), we obtain the TPC algorithm.

B Experiments

For all our planning experiments we use DDQN (Van Hasselt et al., 2016) modified to consider initiation sets for action selection and target computation to make it compatible with options. We use Adam (Kingma and Ba, 2014) as optimizer. As exploration, we use linearly decaying ϵ-greedy exploration.

B.1 Experiments

B.1.1 Environments

Pinball Domain (Konidaris and Barto, 2009) We use a continuous action variant of the original environment. The state space $s = (x, y, \dot{x}, \dot{y})$ with $(x, y) \in [0, 1]^2$ and $(\dot{x}, \dot{y}) \in [-1, 1]$. The action space is the ball acceleration expressed in the form of $\Delta(\dot{x}, \dot{y} \in [-1, 1]^2$. The layout of the obstacles is as in the original environment, show in Figure 8. The reward function takes -5 per unit of acceleration. The discount factor is $\gamma = 0.9997$.

Pinball Options Pinball options were designed to the agent in the coordinate dimensions by step size 0.04. The initiation set are all the position in which the ball would not hit an obstacle by moving in the desired direction. The termination probability is determined by a Gaussian centered in the goal position with standard deviation as 0.01. For the policy, we handcrafted PI controllers for the position with constants $K_p = 50$ and $K_i = 8$.

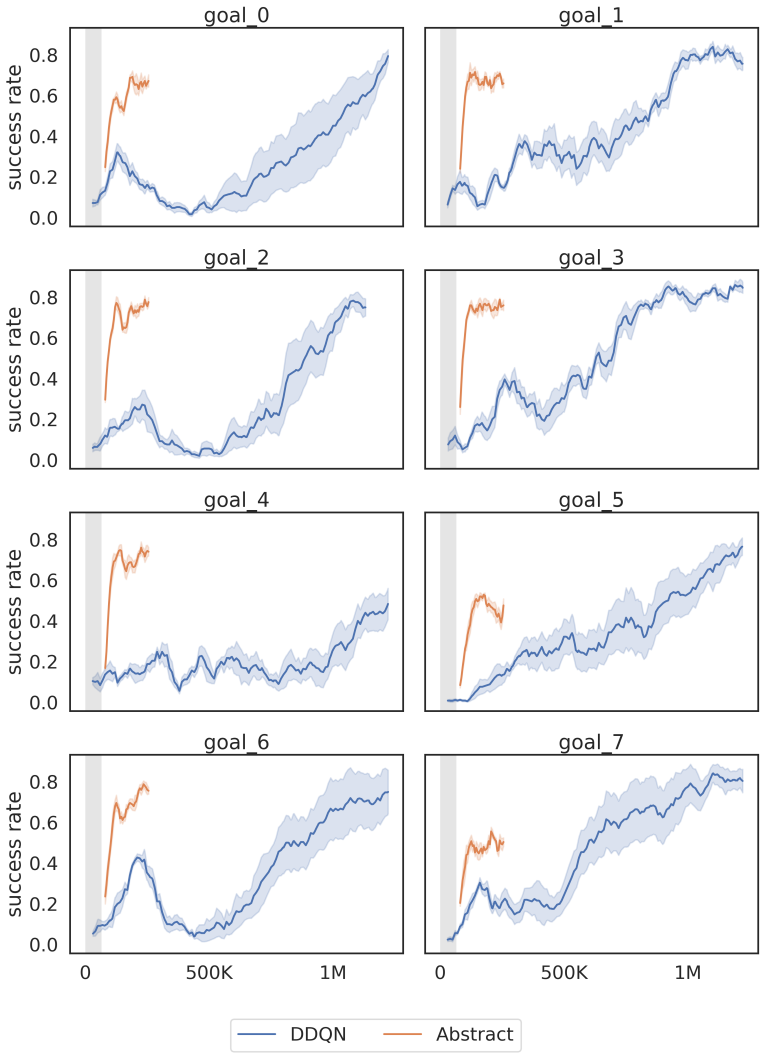


Figure 5: **Pinball from pixels**. Ground baseline vs Abstract planning. Each goal learning curve is averaged over 5 seeds and 1 standard deviation shown in the shaded area of each curve. The gray area corresponds to the offset that corresponds to samples used to pre-train the model. Although is shown in every plot, it is common to all goals.

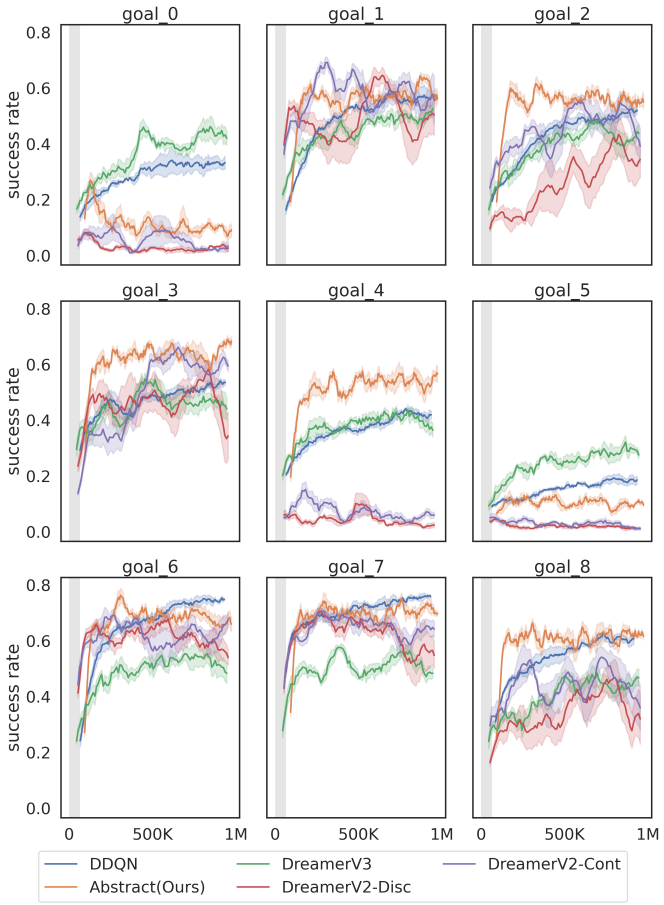


Figure 6: **Medium Play Antmaze**. Ground baseline vs Abstract planning. Each goal learning curve is averaged over 5 seeds and 1 standard deviation shown in the shaded area of each curve. The gray area corresponds to the offset that corresponds to samples used to pre-train the model. Although is shown in every plot, it is common to all goals.

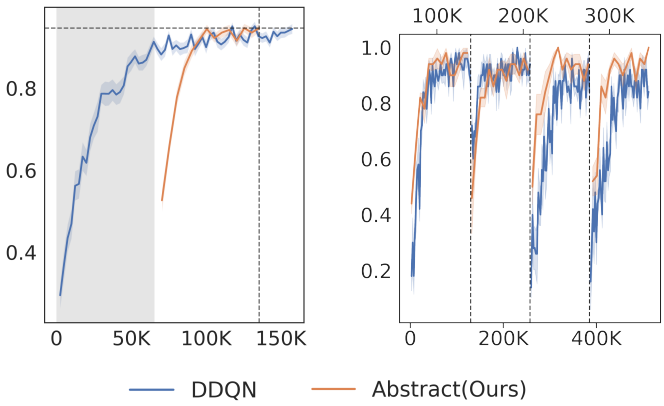


Figure 7: **U-Maze Antmaze**.

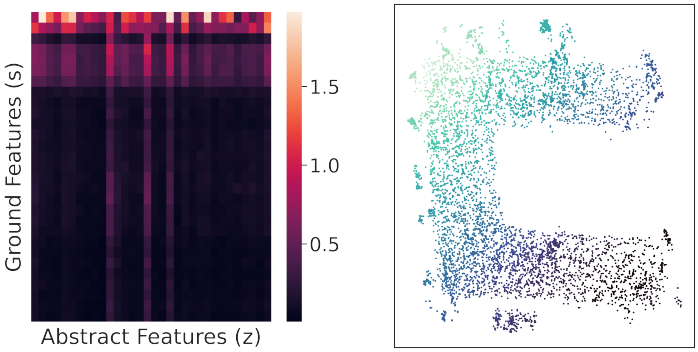
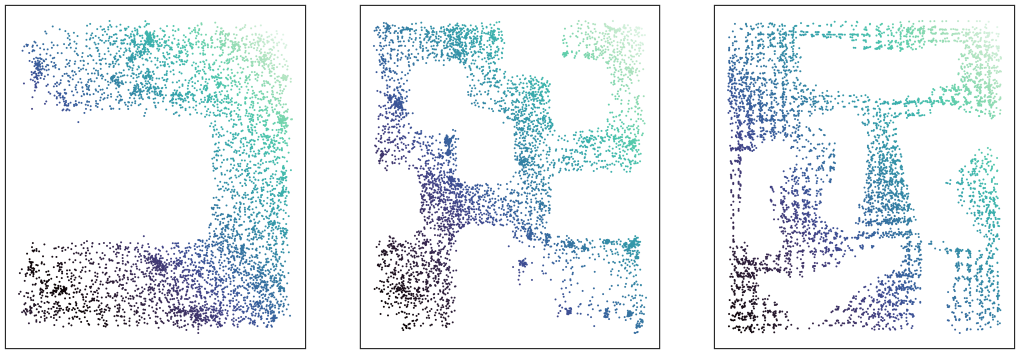


Figure 8: **U-Maze Antmaze**.



(a) U-Maze (b) Medium Play Maze (c) Pinball Domain

Figure 9: Ground truth visualization of possible positions of the agent in the evaluation Environments

Antmazes We consider the U-Maze and Medium-Play mazes implemented by D4RL (Fu et al., 2020) with the Mujoco ant. In Figure 9 we show diagrams of the considered mazes. The state space is $\mathcal{S} \in \mathbb{R}^{29}$, where the first two dimensions corresponds to the position of the ant in the maze and the rest is proprioception for the ant controls. The action space is $\mathcal{A} \subset [-1, 1]^8$ to control the ant joints.

Antmaze Options We consider options that move the ant in the 8 directions (North, South, East, West, North-East, North-West, South-East, South-West) by a distance of 1 unit. For the position controller, we train a goal-conditioned policy using Hindsight Experience Replay (HER; Andrychowicz et al. (2017)) and TD3 (Fujimoto et al., 2018) that would take a goal position in an drive walk the ant to it. This is generally hard for arbitrary goals given the separation between the current position and the goal, however, we only needed the policy to become accurate for short distances, so we sampled initial positions within 1.5 of the desired goal. The goals were sampled uniformly over the possible positions in the maze. Then we learned the initiation sets as classifiers were the option execution would be successful. The termination condition is a threshold of 0.5 distance to the goal.

B.1.2 Network Architectures

Pixel Observations As encoder for pixel observation, we use ResNet Convolutional Networks, as used in Dreamer (Hafner et al., 2021). The ResNet starts with an initial 24 depth and doubles in depth until reaching the minimal resolution. See Table 1.

Table 1: ResNet CNN Configuration

Parameter	Value
in width	50
in height	50
color channels	1
depth	24
cnn blocks	2
min resolution	4
mlp layers	$[256,]$
outdim	4
mlp activation	silu
cnn activation	silu

MLP Architectures For all other models, we use MLPs with the relevant input and output dimensions. This includes encoder, initiation classifiers, transition function, reward function and duration. For the reward function we use the symlog transformation (Hafner et al., 2021) and a log transformation for the option duration network.

Table 2: MLP Configuration

Parameter	Value
hidden dims	$[128, 128]$
activation	relu

Density Estimation We use mixture of Gaussians with 4 components and Gaussians with diagonal covariance matrices. We use the reparameterization trick (Kingma and Welling, 2013) to optimize the mean and variance functions.

B.1.3 Agent Hyperparameters

To train our baseline DDQN agent with the following parameters that we tune by doing grid search for 5 goal positions and 2 seeds, we use all these parameters to learn for all goals.

Pinball Domain For pixel observations we use the same architecture as described before for the world model encoder. For simpler observation, we use an MLP as before.

Dreamer Baselines We use the publicly available implementations for the Dreamer baselines. For the DreamerV2 (Hafner et al., 2021) baseline, we used the hyperparameters recommended for DeepMind Control environments with (only) proprioception inputs. Instead, for the DreamerV3 (Hafner et al., 2023) baseline we used the recommended hyperparameters.

B.1.4 World Model Hyperparameters

Table 3: Pinball ground DDQN parameters

Parameter	**Value**
final exploration steps	500000
final epsilon	0.1
eval epsilon	0.001
replay start size	10000
replay buffer size	500000
target update interval	10000
steps	1250000
update interval	5
num step return	1
learning rate	10^{-5}
γ	0.9997

Table 4: Ground DDQN Parameters for the Antmazes

Parameter	**Value**
final exploration steps	$350,000$
final epsilon	0.1
eval epsilon	0.001
replay start size	$1,000$
replay buffer size	$100,000$
target update interval	$1,000$
steps	$1,000,000$
update interval	5
num step return	1
learning rate	5×10^{-4}
γ	0.995

Table 5: U-Maze Imagination DDQN Parameters

Parameter	**Value**
final exploration steps (proportion)	30% of agent training steps
final epsilon	0.1
eval epsilon	0.001
replay start size	1000
replay buffer size	100000
target update interval	10000
update interval	5
num step return	1
learning rate	1×10^{-4}
rollout length	100

Table 6: World Model Parameters

Parameter	**Value**
buffer size	$100,000$
batch size	16
learning rate	1×10^{-4}
train every	8
max rollout length	64