

LEARNING PARSIMONIOUS DYNAMICS FOR GENERALIZATION IN REINFORCEMENT LEARNING

Anonymous authors

Paper under double-blind review

ABSTRACT

Humans are skillful navigators: We aptly maneuver through new places, realize when we are back at a location we have seen before, and can even conceive of shortcuts that go through parts of our environments we have never visited. Current methods in model-based reinforcement learning on the other hand struggle with generalizing about environment dynamics out of the training distribution. We argue that two principles can help bridge this gap: latent learning and parsimonious dynamics. Humans tend to think about environment dynamics in simple terms – we reason about trajectories not in reference to what we expect to see along a path, but rather in an abstract latent space, containing information about the places’ spatial coordinates. Moreover, we assume that moving around in novel parts of our environment works the same way as in parts we are familiar with. These two principles work together in tandem: it is in the latent space that the dynamics show parsimonious characteristics. We develop a model that learns such parsimonious dynamics. Using a variational objective, our model is trained to reconstruct experienced transitions in a latent space using locally linear transformations, while encouraged to invoke as few distinct transformations as possible. Using our framework, we demonstrate the utility of learning parsimonious latent dynamics models in a range of policy learning and planning tasks.

1 INTRODUCTION

Navigation comes easy to humans. We are able to maneuver through novel parts of our environments, self-locate by integrating over convoluted trajectories, and even come up with shortcuts that traverse areas of the environment we have never visited before. Two principles seem to drive these abilities: latent learning and parsimonious dynamics. Latent learning describes the ability to represent paths through our world not as we experience them literally, but in an abstract manner: That is, we reason about trajectories not in reference necessarily to what we expect to see along a path, but rather in an abstract latent space, containing information about the places’ spatial coordinates (Tolman, 1948; Constantinescu et al., 2016). These coordinates are themselves never experienced, but are useful representations constructed to reflect the structure of the environment we inhabit. Parsimonious dynamics describe the fact that the rules governing how state transitions work should be simple. We assume that moving around in novel parts of our environment works the same way as in parts we are familiar with. These two principles work together in tandem: it is in the latent space that the dynamics show parsimonious characteristics.

We extend these ideas to the more general framework of learning latent dynamics models for reinforcement learning (RL). Recent advances in model-based RL have showcased the potential improvements in the performance and sample complexity that can be gained by learning accurate latent dynamics models (Deisenroth & Rasmussen, 2011; Hafner et al., 2019a; Schrittwieser et al., 2020). These models summarize the transitions that the agent experiences by interacting with its environment in a low-dimensional latent code that is learnt alongside their dynamics. By learning such models, agents may for instance perform control by planning ahead in this low-dimensional state space (Hafner et al., 2019b), or, if the latent states contain useful information for policy learning, simply learn a policy over latent states rather than the original states (Ha & Schmidhuber, 2018; Lee et al., 2020). We employ the principle of parsimony to learn a latent dynamics model that is able to generalize about novel transitions, and whose latent states contain information about the topology of the environment. These latent state representations prove to be useful for policy learning, planning

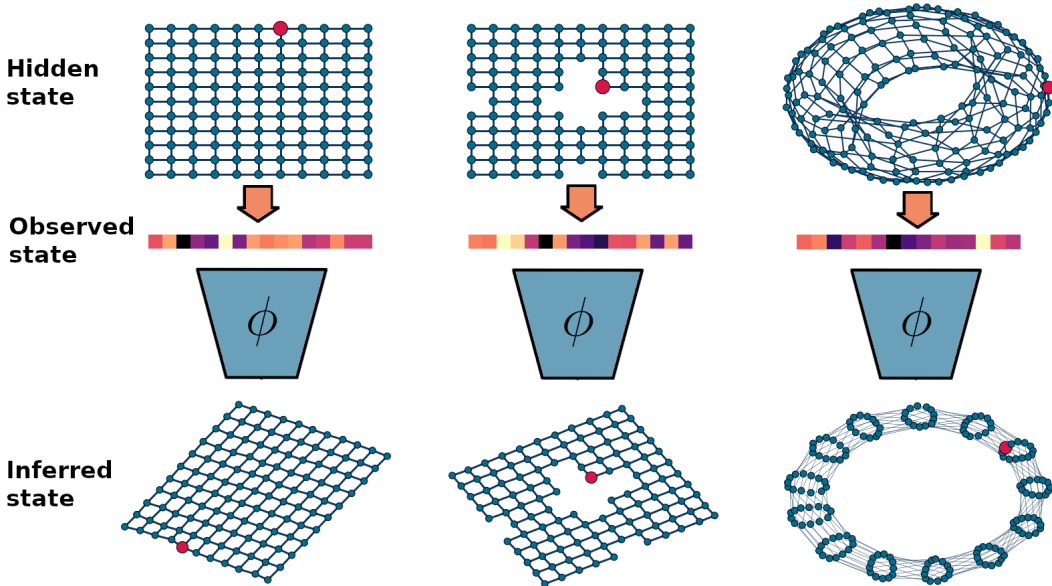


Figure 1: Illustration of the learning problem. Environments can be embedded in an underlying low-dimensional space, and transitions can be described by a small set of linear transformations, like rotations and translations, information that is unavailable to the agent. The observation space \mathcal{S} is not informative of the underlying topological structure of the environments, and must be learnt. Our model discovers a latent space $\tilde{\mathcal{Z}}$ where state-transitions can be described by a small number of learnt transformations.

and future state prediction. Moreover, we show that one can learn such latent representations of the environment simply from encouraging the dynamics to be parsimonious, without supervision about the underlying latent structure.

We draw inspiration from group theory, the study of transformations that preserve symmetries (Kondor, 2008), to learn such latent spaces. Particularly, we adopt the framework of Quessard et al. (2020) and Caselles-Dupré et al. (2019) where the interventions an agent can perform on its environment are treated as transformations belonging to a group, and transitioning between states through selecting actions is equivalent to transforming the source state with the action’s corresponding group transformation. We extend this approach by summarizing a data set of experienced transitions invoking only a small set of different types of learned transformations. We hypothesize that a model that infers a small set of locally linear transformations to explain global transition dynamics should be able to generalize effectively about the transition dynamics of novel parts of the agent’s environment, and that the latent state representations that result from embedding states onto such discovered manifolds are beneficial for policy learning. In the end, we show that our approach outperforms alternative dynamics and representation learning models in planning and policy learning tasks, as well as in an open-loop pixel prediction tasks based on the Deepmind Lab environment (Beattie et al., 2016).

2 MODEL

2.1 PRELIMINARIES

We assume that the environment is a Markov Decision Process (MDP) defined as the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, where \mathcal{S} is the state space, \mathcal{A} is the set of actions, \mathcal{T} is the transition function describing the probability of successor states given the current state-action tuple $s_{t+1} \sim \mathcal{T}(s_t, \mathbf{a}_t)$, \mathcal{R} is the reward function and γ is the discount factor. In every state, the agent selects an action according to the policy $\mathbf{a}_t \sim \pi(s_t)$. The agent’s goal is to learn a policy $\pi(\mathbf{a}_t | s_t)$ or plan using knowledge of environment dynamics to maximize future discounted rewards $\mathbb{E}_{\mathcal{T}} [\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)]$.

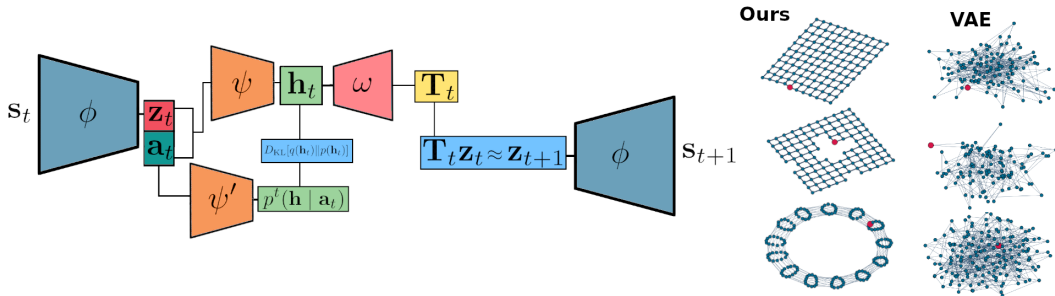


Figure 2: Left: Illustration of model components. Right: Embeddings of environment states produced by our model, compared to those produced by a β -VAE. The VAE embeddings are learnt without parsimony and with a reconstruction objective, making the latent space reflect the topology of the observation space, rather than that induced by the MDP’s transition function.

The original state space of the MDP may be high-dimensional and difficult to perform RL on. We hypothesize that constructing a low-dimensional latent space that exhibits parsimonious dynamics is beneficial for RL in at least two ways: i) Learning a policy on the latent space $\pi(a | z_t)$ should be easier since the latent states are organized such that they match the underlying, hidden topology of the environment. ii) Planning should be possible with less experience: the gains we seek to make here lie in exploiting the knowledge of the transformations that describe how the state variable changes with our actions. For instance, if the agent learns that all state-transitions can be described by a small set of transformations of the source state depending on the action, we can correctly generalize about what the next latent state will be simply if we can predict what the appropriate transformation of our current latent state is, given the action the agent selected.

2.2 MODEL COMPONENTS

Our model learns an encoding function f_ϕ that maps states s_t to latent states z_t . For the subsequent experiments we assume a deterministic function, but it is straightforward to formulate our model as a stochastic latent dynamics model too (see Appendix A.1). In fact, in section 5 we perform pixel prediction with a stochastic variant of our model. Given an action a_t and the current latent state z_t , we seek to predict the next latent state \tilde{z}_{t+1} . We represent \tilde{z}_{t+1} as the product of the current latent state with a linear transformation matrix $z_{t+1} \approx T_t z_t$ which we predict from the latent state-action tuple (z_t, a_t) .

$$\begin{aligned}
 z_t &= f_\phi(s_t) \\
 h_t &= g_\psi(z_t, a_t) \\
 T_t &= j_\omega(h_t, a_t) \\
 \tilde{z}_{t+1} &= T_t z_t
 \end{aligned}
 \tag{1}$$

We use a probabilistic approach to learning T_t : We seek to infer the posterior distribution of a discrete latent code $h_t \sim q_\psi^t(z_t, a_t)$, given a prior $p_{\psi'}^t(h | a_t)$ from which we can decode the appropriate transformation T_t that describes the transition.

$$\begin{aligned}
 \text{Posterior: } & q_\psi^t(h | z_t, a_t) \\
 \text{Prior: } & p_{\psi'}^t(h | a_t)
 \end{aligned}
 \tag{2}$$

We want our model to be able to recapitulate observed transitions as accurately as possible, while maximizing the predictability of the transformations describing individual transitions in latent space from the chosen actions alone. This is what we refer to as the principle of parsimony. We construct $q_\psi^t(h | z_t, a_t)$ as a multivariate Bernoulli distribution of dimensionality n with probability vector p_t , which is predicted by a neural network function $g_\psi(z_t, a_t)$. The latent code h_t is a binary vector which we produce by rounding p_t .

$$h_i = \begin{cases} 1, & \text{if } p_i \geq 0.5 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The latent transition code \mathbf{h}_t is decoded with a learnt decoding function j_ω into the parameters of the linear transformation matrix \mathbf{T}_t with which we predict the next latent state. We then train the transition encoder and decoder networks g_ψ and j_ω using a variational objective (Kingma & Welling, 2013; Hafner et al., 2019b). To backpropagate through the discrete transition representation, we make use of the straight-through estimator (Bengio et al., 2013). Due to the discretization, this transition representation already acts as a bottleneck on the number of transformation matrices we can associate with each action (Van Den Oord et al., 2017). However, a chief aim of our model is to represent the transitions of the environment with as few transformations as possible. We incorporate this desideratum in the way we construct the variational objective for posterior inference of $q_\psi^t(\mathbf{h} | \mathbf{z}_t, \mathbf{a}_t)$. We leverage a second learnt neural network function $g'_{\psi'}(\mathbf{z}_t, \mathbf{a}_t)$ to learn a prior distribution $p_{\psi'}^t(\mathbf{h} | \mathbf{a}_t)$ where \mathbf{h}_t does not depend on the current latent state \mathbf{z}_t .

Importantly, doing so makes our prior over transformation matrices, given an action, state-invariant. By enforcing closeness to the prior, we encourage our model to learn a latent state space such that latent state transitions may be predicted accurately even without information about the current latent state \mathbf{z}_t . Finally, we can construct a loss function for learning the posterior distribution as follows:

$$\mathcal{L}_{transition} = \underbrace{\log p(\mathbf{z}_{t+1} | \mathbf{h}_t, \mathbf{a}_t)}_{\text{next state prediction}} + \underbrace{\beta D_{KL} [q_\psi^t(\mathbf{h} | \mathbf{z}_t, \mathbf{a}_t) || p_{\psi'}^t(\mathbf{h} | \mathbf{a}_t)]}_{\text{parsimony}} \quad (4)$$

The first term reflects the accuracy with which our model predicts the next latent state (as encoded by our model) and is a function of the distance between $\tilde{\mathbf{z}}_{t+1}$ and \mathbf{z}_{t+1} (see Appendix A.1). The second term reflects how close our posterior over \mathbf{h}_t is to our state-invariant prior, scaled by the hyperparameter β .

2.3 PARAMETERIZING TRANSFORMATIONS

We consider three types of transformations – rotations, translations and their composition. As such, we assume that all transitions can be represented as an affine transformation of the current latent state. We leverage a decoder network j_ω which takes the current latent transition code and action tuple $(\mathbf{h}_t, \mathbf{a}_t)$ and predicts the parameters of either a rotation matrix \mathbf{R} or a translation matrix. For translations we predict a vector of displacement values \mathbf{v} and predict the next latent state $\tilde{\mathbf{z}}_{t+1} = \mathbf{z}_t + \mathbf{v}_t$, requiring n parameters for a latent state space of dimensionality n . Parameterizing rotations is more involved. Quessard et al. (2020) parameterize rotations with a product of $\frac{n(n-1)}{2}$ 2-dimensional rotations. In our approach we predict the entries of a skew-symmetric matrix. The space of skew-symmetric matrices form the Lie algebra of the special orthogonal group and its elements can thus be viewed as infinitesimal rotations (Sola et al., 2018). By taking the matrix exponential of an $n \times n$ skew-symmetric matrix, we obtain an $n \times n$ -dimensional rotation matrix \mathbf{R} . Since the upper triangle of a skew symmetric matrix is the negative of the lower triangle, we can parameterize the n -dimensional rotation using the same number of parameters as Quessard et al. (2020).

2.4 LEARNING THE ENCODING FUNCTION

Encouraging the latent dynamics to be parsimonious through the KL term in equation 3 is not sufficient, we also need to make sure that the condition of parsimony is not fulfilled vacuously, for instance if the encoder maps all states \mathbf{s}_t to a single latent state. A popular approach for avoiding state collapse is to equip the model with a state decoder that tries to predict the state \mathbf{s}_t from the latent state \mathbf{z}_t (Hafner et al., 2019b; Watter et al., 2015). However, encouraging the model to learn latent states that are easily decodable could conflict with our goal of learning a latent state space governed by parsimonious dynamics, as generative factors of the data distribution could influence the topology of our latent space. Instead, we opt for a contrastive objective to distinguish between states (Oord et al., 2018; Laskin et al., 2020), while giving our transition model the freedom to embed states such that transitions between them can be encoded parsimoniously. Our approach is

inspired by noise contrastive estimation (Gutmann & Hyvärinen, 2010; Oord et al., 2018), which seeks to keep a latent state $\mathbf{z}_t \mid \mathbf{s}_t$ predictable from the observed state, while keeping it diverse from the distinct latent states. For a mini-batch \mathcal{B} we construct target labels for each state $\mathbf{s}_t \in \mathcal{B}$ and corresponding similarity ratings for our encodings of them:

$$\begin{aligned} l(\mathbf{s}, \mathbf{s}') &= e^{-\tau_s \|\mathbf{s} - \mathbf{s}'\|_2} \\ k(\mathbf{z}, \mathbf{z}') &= e^{-\tau_z \|\mathbf{z} - \mathbf{z}'\|_2} \end{aligned} \quad (5)$$

where $l(\mathbf{s}, \mathbf{s}')$ are targets and $k(\mathbf{z}, \mathbf{z}')$ are latent state similarities. Here τ_s and τ_z are scaling parameters quantifying how quickly state similarity decays with distance. To keep latent states diverse we minimize the cross entropy between $k(\mathbf{z}, \mathbf{z}')$ and $l(\mathbf{s}, \mathbf{s}')$:

$$\mathcal{L}_{contrastive} = -\frac{1}{N} \sum_{\mathbf{s}, \mathbf{s}' \in \mathcal{B} \times \mathcal{B}} k(\mathbf{z}, \mathbf{z}') \log l(\mathbf{s}, \mathbf{s}') + (1 - k(\mathbf{z}, \mathbf{z}')) \log(1 - l(\mathbf{s}, \mathbf{s}')) \quad (6)$$

By scaling τ_s sufficiently high, we encourage our model to distinguish between states that are not identical. This facilitates the learning of parsimonious dynamics: as we only require that distinct states are encoded far enough apart, the transition model is afforded freedom to embed states so that the condition of parsimony is satisfied. Our final loss function is then the sum of the transition loss and the contrastive loss.

3 RELATED WORK

World models

Quessard et al. (2020) represent latent state transitions as the product of the current latent state with elements of the special orthogonal group, i.e. a learnt rotation matrix. However, they assume that the rotations describing state transitions are state-invariant. That is, actions can only affect the state in a single way. We make state-invariance a soft constraint by keeping transition representations close to a state-invariant prior, and represent latent state transitions using elements from the affine group. Watter et al. (2015) learn to embed high-dimensional inputs in a low-dimensional space in which the dynamics are locally linear, allowing them to plan with stochastic optimal control. Unlike our approach, they use a reconstruction objective to mitigate state collapse, and do not regularize the state-action representations the agent learns. Hafner et al. (2020), Hafner et al. (2019a), Kaiser et al. (2019) and Ha & Schmidhuber (2018) learn world models with the purpose of learning policies, either by training the agent within the world model entirely, or by extracting useful latent features of the environment.

Planning

Dynamics models are also pervasive in planning tasks. Deisenroth & Rasmussen (2011) use Gaussian process regression to learn environment dynamics for planning in a sample efficient manner. Schrittwieser et al. (2020) learn a latent state dynamics model without a reconstruction objective to play chess, shogi and Go using Monte Carlo Tree Search. Hafner et al. (2019b) learn a recurrent state space model, representing latent states both with a deterministic and stochastic component, and perform planning in pixel environments using the Cross Entropy method. Our approach extends on previous work by building latent state spaces that facilitate planning with incomplete knowledge of the environment. This affordance is due to the latent state space being organized such that transitions can be described with a sparse latent code.

Disentangled representations

Learning disentangled representations is a popular approach for building latent variable models (Higgins et al., 2016; Burgess et al., 2018). Higgins et al. (2018) propose a group theoretic definition of disentangled representations and Caselles-Dupré et al. (2019) argue that learning symmetry based disentangled representations requires interactions with the environment. Our model can be viewed as learning a latent state space whose dynamics are described by a small number of transformations belonging to the affine group.

Navigation

We argue that the ability to navigate could be supported by learning parsimonious dynamics. Us-

ing an RL framework, Mirowski et al. (2016) learn to navigate to goal locations in 3D mazes with pixel inputs, for which they rely on auxiliary depth prediction and location prediction tasks. Banino et al. (2018) use velocity and head direction information to learn map-like representations of environments, allowing them to generalize about novel shortcuts to goals. However, biological agents do not usually get information about how close they are to boundaries, their direction of travel, or what their current spatial location is. Rather, these variables must be inferred from interactions with stimuli of much higher dimensionality. We propose a method for learning the latent spatial layout of environments and generalize about their transition dynamics from the principle of parsimony alone, without supervision or additional information about the aforementioned latent variables.

4 EXPERIMENTS

We test our model’s ability to learn latent spaces and dynamics that are useful for policy learning and planning. We designed three environments with different topological properties. In all environments the agent is tasked with navigating to a fixed goal location from a fixed starting location, and once it has arrived at the goal location, stay there for the remainder of the episode. The action space consists of the five actions $\mathcal{A} = \{LEFT, RIGHT, UP, DOWN, STAY\}$. The actions are represented as one-hot encoded vectors to not reveal any information about the transition function of the environment. The agent moves around on the grid by selecting a cardinal direction, which moves the agent one unit in the respective, latent direction. The latent coordinate features of the states are unobservable to the agent. With the *STAY* action the agent stays put on the current state.

Environment states are represented as random vectors drawn from a multivariate Gaussian $\mathbf{s} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$ with a diagonal covariance matrix. These are the vectors that the agent ‘observes’ when occupying a state, and *not*, for instance, a top-down view of the environment. The vectors are drawn when environments are initialized, and then remain fixed for the duration of an experiment. Generating the state vectors from an isotropic Gaussian makes the observation space independent of the underlying hidden variable describing the agent’s position. This is a key property of our tasks: We maintain that the ability to learn the group properties of an environment in a way that is disassociated from learning the generative factors of the observations that the states emit is important. Generally, the manifold that the state-observations lie on may be entirely different from the manifold defined by an environment’s transition function. We hypothesize that structuring the latent state space to reflect the topology of the environment is beneficial for solving several RL tasks.

4.1 GRIDWORLDS

We designed two 11×11 state discrete gridworlds (see Figure 3). On the boundary states of the gridworld there were walls. One of the gridworlds was partitioned into four rooms by walls. Information about whether the agent was facing a wall in any of the four cardinal directions was encoded as a binary vector, which we concatenated with the initial random state vector to produce what the agent sees when occupying a state.

4.2 TORUS

We designed a discrete torus world similar to the gridworld by connecting the gridworlds boundary states to the corresponding boundary states on the opposite end (see Figure 3). The torus contained 13×13 states and no boundaries.

4.3 MODEL FREE LEARNING

We designed a model free learning task for each environment. In each task, the agent needs to learn to move from a starting state \mathbf{s}_{start} to an unknown goal state \mathbf{s}_{goal} and stay there for the remainder of an episode which lasted for 250 timesteps. Each state except for the goal state yields a reward of -1 when exited except for the goal state which yields a reward of $+1$. The agent learns a policy which takes it to the goal location using the Soft Actor Critic (SAC) algorithm (Haarnoja et al., 2018) adapted for discrete action spaces (Christodoulou, 2019) (see Appendix A.6 for details). Crucially, the agent learns a policy over its latent state representations $\pi(\mathbf{a} | \mathbf{z})$ as opposed to the observations the environment emits. We train agents for 200 episodes in the gridworld, 500 episodes

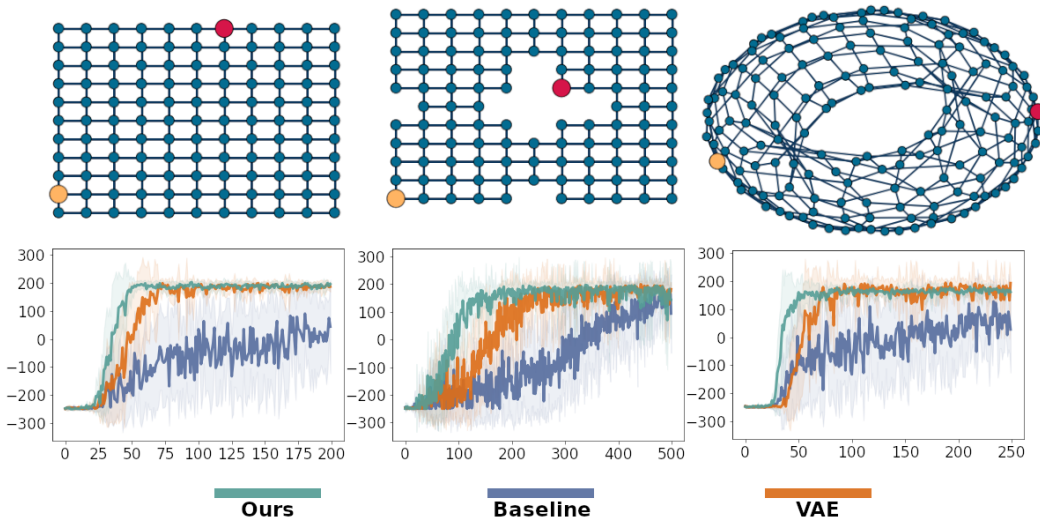


Figure 3: Top: The environments have a starting state (yellow) and a goal state (red). Bottom: The corresponding learning curves of a SAC agent learning policies over three types of latent state spaces, averaged over 10 seeds: *Ours* has a latent state space constrained to be such that dynamics are parsimonious. *Baseline* has no representation learning objective. *VAE* learns a latent state space using the β -VAE model, with next state prediction as an auxiliary task.

in the four rooms environment, and for 250 episodes in the torus world. Each episode lasts for 250 steps. The agents are trained as described in Algorithm A.5. We compare our model to two alternative representation learning approaches: i) The β -Variational Autoencoder (Higgins et al., 2016; Lee et al., 2020) which learns disentangled probabilistic embeddings by reconstructing the states as well as performing next state prediction. Next state prediction is performed by multiplying the current latent state with a predicted affine transformation matrix, however, without regularizing the state-action representation to be parsimonious. ii) A baseline feedforward neural network for which no representation learning objectives influenced latent state representations except the actor and critic losses. Our model achieves the best total score summed over episodes in all environments, averaged over 10 seeds. We fine tuned the regularization coefficient β in equation 4 and the β of the VAE model. Comparisons revealed that regularizing the latent state-action representation \mathbf{h}_t proved beneficial in all environments except the Four Rooms environment in which a β value of 0 proved slightly better than the next best regularized implementation of our model.

4.4 PLANNING

For each environment we generated a set of planning problems where the agent starts in a random state, and needs to plan a sequence of actions to reach a goal state and stay there for the remainder of an episode which lasted for 50 timesteps. The agent has no knowledge of environment dynamics initially except for what the goal state is, and needs to learn a viable dynamics model as it engages with the task. The agent attempts to solve the planning problems by encoding the goal state into its learnt latent state space \mathbf{z}_{goal} and by simulating trajectories that take it to the goal. The agent estimates the return of a trajectory as the sum of latent state occupancies weighted by the exponential of their negative distance to the latent goal state $G = \sum_{t=0}^H e^{-\|\mathbf{z}_t - \mathbf{z}_{goal}\|_2}$. After an episode, the agent fits its dynamics model to the observations it gathered through executing its plan. Each planning task consists of 30 such planning problems, varying with difficulty, as some goal locations are further away from the agents’ starting location. Following Hafner et al. (2019b), we use the Cross Entropy Method (CEM) (Rubinstein, 1999) as our planning algorithm (see A.7). We verified that it was able to solve most tasks when using the true environment dynamics with a moderate planning budget.

We compare our model to alternative latent dynamics models: A deterministic RNN and a stochastic latent state model (Hafner et al., 2019b) trained with a β -VAE objective (Higgins et al., 2016).

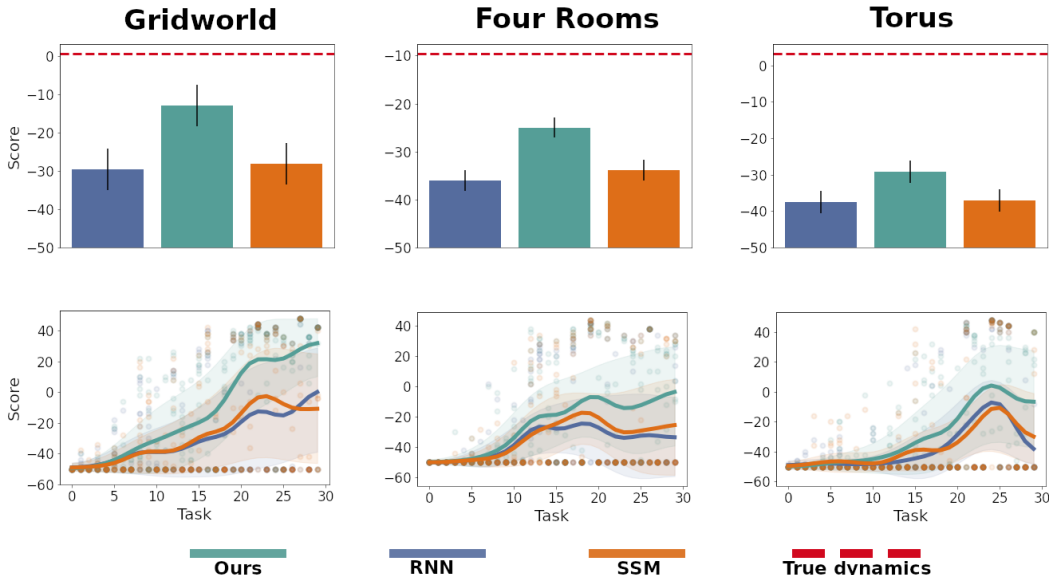


Figure 4: Top row: The score achieved by the planning agent using different dynamics models, averaged over the 30 tasks and 10 seeds: Our approach beats the deterministic recurrent world model (RNN) and the stochastic world model (SSM) trained with a disentanglement objective. Error bars show standard deviation computed across seeds. Red lines show score of planning agent using the true environment dynamics. Bottom row: The score achieved for each task, dots indicate individual samples across 10 seeds, and curves are smoothed using a Gaussian filter with standard error computed across seed, with lengthscale $\sigma = 2$.

All models represent state transitions as the product of the current latent state with a learnt affine transformation matrix, but lack the parsimony constraint at the core of our model. The models were trained as described by Algorithm A.7. As with the policy learning task, we found that our model achieved the best score pooled over the 30 planning tasks and 10 seeds. Moreover, regularizing \mathbf{h}_t proved beneficial in all environments, providing further evidence that parsimonious dynamics are beneficial for planning.

5 LEARNING PARSIMONIOUS DYNAMICS FROM PIXELS

We sought to evaluate our model’s ability to perform long-term future state prediction in an environment with pixel inputs. For this task, we relied on the Deepmind Lab environment, a challenging partially observable environment with image observations (Beattie et al., 2016). To make our model suitable for pixel prediction, to mitigate the partially observability, and to make it comparable to other models in the literature, we used the stochastic variant of our model (see Appendix A.1) and image reconstruction loss rather than a contrastive objective to avoid latent state collapse. Furthermore, we endowed it with a convolutional neural network image encoder, a transpose convolutional neural network decoder, and recurrent neural network whose outputs were concatenated with the inferred latent state for pixel prediction (see Appendix A.3).

As a comparison model we chose the Recurrent State Space Model (RSSM) from Hafner et al. (2019b) (see Appendix A.4). When applicable, we also used the hyperparameters they provide for our model. We trained both models to reconstruct sequences of images, conditioned on previous image and action observations, collected from an agent executing a random policy for 250 episodes in the `seekavoid_arena_01` environment. No velocity or location information was provided to the agent. We then made the models perform open-loop prediction of 30 test sequences of 149 environment steps, that were not in the training set. We evaluated open-loop reconstruction errors and the KL divergence between predicted future latent states and closed-loop inferred latent states from observations $KL_D [q(\tilde{\mathbf{z}}_{t+1} | \mathbf{z}_{\leq t}, \mathbf{u}_{\leq t}, \mathbf{a}_{\leq t}) || p(\mathbf{z}_{t+1} | \mathbf{s}_{t+1}, \mathbf{u}_t)]$, where \mathbf{u}_t is a deterministic la-

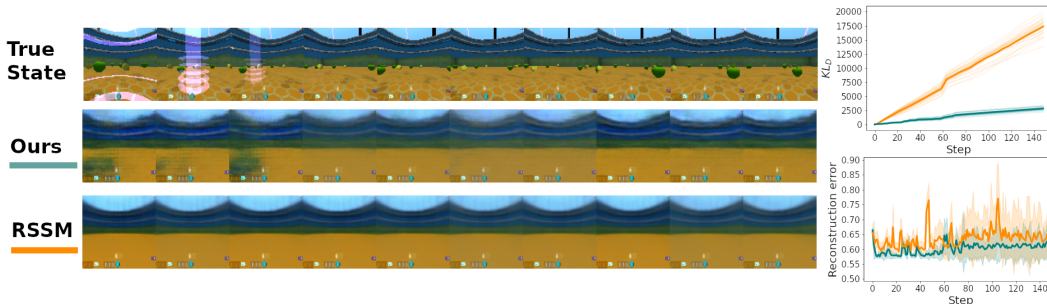


Figure 5: Open loop prediction of future states in the Deepmind Lab environment. Our model is better at predicting transitions in latent space, and reconstructing images from imagined latent states.

tent variable provided by the recurrent neural network. Our model generalized better to the test sequences, achieving both lower KL divergences between predicted and encoded states, and lower average reconstruction error in the open-loop prediction task. This provides evidence for the utility of learning parsimonious dynamics in more challenging pixel environments as well.

6 DISCUSSION

In the current paper, we introduced a model that learns a parsimonious set of transformation matrices to describe the dynamics of MDPs in a latent state space. Learning a world model whose states are organized according to this principle proved beneficial in a policy learning task, a planning task and a video prediction task. With the objective of carving the environment at its joints rather than the observations its states emit, the learnt latent states contained information that was valuable for policy learning. Moreover, planning in the learnt latent space became feasible having observed fewer state transitions: This is because the agent could systematically generalize about the dynamics of parts of the environment that were not yet explored exhaustively. We have shown that simply endowing the dynamics model’s objective with a term encouraging parsimony was sufficient to produce latent spaces that display useful characteristics.

We investigated the utility of parsimonious dynamics in simpler environments whose transitions could indeed be characterized by a small set of linear transformations. A limitation of our approach is that the environments we investigated were rather simplistic compared to the rich environments humans and other state-of-the-art models have been shown to be able to navigate through. To remedy this we provided promising initial evidence that parsimonious dynamics can facilitate future state prediction in a Deepmind Lab environment. In future work we intend to scale up our approach to also perform policy search in similarly rich environments. A further limitation is that the degree of sparseness with which the model tries to recapitulate transitions is controlled by the KL scaling term β . Though we avoid assuming that dynamics are completely state-invariant, we still have to tune β . In future work, we seek to address this issue by adaptively regulating the complexity of our learnt dynamics to reflect the complexity of the environment.

The principle of parsimony was initially motivated from the viewpoint of cognitive science. Tolman (1948) showed that rats preferred a novel shortcut over a repeatedly reinforced longer route to a goal location, hypothesizing that animals learn and use latent representations of their environment that must contain assumptions about how they are structured. Seminal work in neuroscience demonstrated the existence of neurons selectively tuned to specific spatial positions (place cells), and others that represent global geometrical information about the environment (grid cells), representations that were also found in artificial agents trained to navigate in Euclidean spaces (Banino et al., 2018; Cueva & Wei, 2018). Recent cognitive neuroscience studies revealed that humans rely on cognitive maps to navigate complex environments (Epstein et al., 2017), abstract spaces (Constantinescu et al., 2016; Garvert et al., 2017), generalize about rewards (Garvert et al., 2021), and draw inferences about transition dynamics in novel environments (Mark et al., 2020). Future work could also investigate the role of parsimony in the mental maps that humans and other animals build of the worlds they inhabit.

REFERENCES

- Andrea Banino, Caswell Barry, Benigno Uribe, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018.
- Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.
- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599*, 2018.
- Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. *Advances in Neural Information Processing Systems*, 32, 2019.
- Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- Alexandra O Constantinescu, Jill X O’Reilly, and Timothy EJ Behrens. Organizing conceptual knowledge in humans with a gridlike code. *Science*, 352(6292):1464–1468, 2016.
- Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.
- 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)*, pp. 465–472. Citeseer, 2011.
- Russell A Epstein, Eva Zita Patai, Joshua B Julian, and Hugo J Spiers. The cognitive map in humans: spatial navigation and beyond. *Nature neuroscience*, 20(11):1504–1513, 2017.
- Mona M Garvert, Raymond J Dolan, and Timothy EJ Behrens. A map of abstract relational knowledge in the human hippocampal–entorhinal cortex. *elife*, 6, 2017.
- Mona M Garvert, Tankred Saanum, Eric Schulz, Nicolas W Schuck, and Christian F Doeller. Hippocampal spatio-temporal cognitive maps adaptively guide reward generalization. *bioRxiv*, 2021.
- Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.
- David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.
- 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*, pp. 2555–2565. PMLR, 2019b.

- Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.
- Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Open Review*, 2016.
- Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.
- Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.
- 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.
- Imre Risi Kondor. *Group theoretical methods in machine learning*. Columbia University, 2008.
- Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.
- Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.
- Shirley Mark, Rani Moran, Thomas Parr, Steve W Kennerley, and Timothy EJ Behrens. Transferring structural knowledge across cognitive maps in humans and models. *Nature communications*, 11(1):1–12, 2020.
- Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.
- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- Robin Quessard, Thomas Barrett, and William Clements. Learning disentangled representations and group structure of dynamical environments. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19727–19737. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/e449b9317dad920c0dd5ad0a2a2d5e49-Paper.pdf>.
- Reuven Rubinfeld. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.
- Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.
- Edward C Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189, 1948.
- Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

A APPENDIX

A.1 TRANSITION LOSS

Our model can be formulated as both a deterministic and stochastic state space model. The loss function for deterministic transitions is described in Equation 4. In the stochastic case, we produce probabilistic state embeddings $\mathbf{z}_t \sim p(\mathbf{z}_t | \mathbf{s}_t)$, where p is a Gaussian with mean and standard deviation $(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t) = f_\phi(\mathbf{s}_t)$. We construct a lower-bound on the data-log-likelihood by minimizing either a reconstruction error term or a contrastive term, in addition to a KL term constraining the posterior latent state to be close to the prior.

$$\mathcal{L} = \log p(\mathbf{s}_t | \mathbf{z}_t) + \beta D_{KL} [q(\mathbf{z}_t | \mathbf{s}_t) \| p(\tilde{\mathbf{z}}_t | \mathbf{z}_{t-1}, \mathbf{a}_{t-1})] \quad (7)$$

We allow gradients to pass through the stochastic component using the reparameterization trick (Kingma & Welling, 2013).

A.1.1 DETERMINISTIC TRANSITIONS

If our transition model is deterministic, we define the reconstruction term based on the exponential of the norm of the vector $\mathbf{z}_{error} = \tilde{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}$:

$$\mathcal{L}_{\text{deterministic transitions}} = \|\tilde{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2^2 + e^{-\|\tilde{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2} \quad (8)$$

We found that adding the mean squared error term in the transition reconstruction helped the models converge.

A.1.2 STOCHASTIC TRANSITIONS

In the stochastic case, we use as our transition reconstruction term the KL divergence between the two distributions predicted for the next latent state: $\mathcal{N}(\mathbf{T}_t \boldsymbol{\mu}_t, \tilde{\boldsymbol{\sigma}}_{t+1})$ where $\tilde{\boldsymbol{\sigma}}_{t+1} = g_\psi(\boldsymbol{\mu}_t, \mathbf{a}_t)$ and $\mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\sigma}_{t+1}) = f_\phi(\mathbf{s}_{t+1})$, e.g.:

$$\mathcal{L}_{\text{stochastic transitions}} = D_{KL} [p(\mathbf{z}_{t+1}) \| q(\tilde{\mathbf{z}}_{t+1})] + D_{KL} [q(\mathbf{h} | \mathbf{z}_t, \mathbf{a}_t) \| p(\mathbf{h} | \mathbf{a}_t)] \quad (9)$$

Our composite loss function therefore becomes the standard variational objective for training the stochastic state space model, plus the KL term describing the divergence between our prior state action representation $p(\mathbf{h}_t)$ and our posterior $q(\mathbf{h}_t)$.

A.2 DYNAMICS MODELS

For the policy learning and planning experiments we parameterized our dynamics models using feedforward neural network encoders and decoders (when applicable), each with two hidden layers and 1200 ReLU units Nair & Hinton (2010). The latent space the encoders projected to had 15 dimensions. The state-action variable \mathbf{h}_t , which also had 15 dimensions, was learnt using an encoder and decoder with the same hyperparameters. When we used the contrastive loss to avoid state collapse, we used similarity scalings $\tau_s = 100$ and $\tau_z = 0.1$

A.3 DEEPMIND LAB

We trained our dynamics models on 80×80 resolution image sequences from Deepmind Lab’s `seekavoid_arena_01`. The sequences were produced by an agent with a random policy. The agent could execute six discrete actions: Move forward, backwards, left or right, and turn in both horizontal directions. We equipped our model with a convolutional neural network for image encodings and a transposed convolutional neural network for image reconstructions, both with the architecture from Ha & Schmidhuber (2018). For our recurrent neural network producing the deterministic latent variable \mathbf{u}_t we used a GRU (Chung et al., 2014) with 200 hidden units. The feedforward neural networks which were used as encoders and decoders had all two hidden layers with 200 ReLU units.

Like in Hafner et al. (2019b), we train our model on sequences of image-action pairs: To perform one gradient step with our dynamics models we draw $B = 50$ sequence chunks of length $N = 50$ and compute the dynamics loss using Equation 10:

$$\begin{aligned} \mathcal{L} = & \underbrace{\log p(\mathbf{s}_t | \mathbf{z}_t, \mathbf{u}_t)}_{\text{reconstruction loss}} \\ & + \beta_1 \underbrace{D_{KL} [q(\tilde{\mathbf{z}}_{t+1} | \mathbf{z}_t, \mathbf{u}_t, \mathbf{a}_t) || p(\mathbf{z}_{t+1} | \mathbf{s}_{t+1}, \mathbf{u}_t)]}_{\text{transition loss}} \\ & + \beta_2 \underbrace{D_{KL} [q(\mathbf{h} | \mathbf{z}_t, \mathbf{u}_t, \mathbf{a}_t) || p(\mathbf{h} | \mathbf{u}_t, \mathbf{a}_t)]}_{\text{parsimony}} \end{aligned} \quad (10)$$

The latent variables \mathbf{z}_t and \mathbf{u}_t had 30 dimensions each. We perform 25 gradient steps per episode, with the transition KL term $\beta_1 = 0.1$ and the parsimony KL term $\beta_2 = 0.5$.

A.4 RSSM

We implemented the RSSM from Hafner et al. (2019b). The RSSM consists of the following components:

$$\begin{aligned} \text{Deterministic state model} \quad & \mathbf{h}_{t+1} = f(\mathbf{z}_t, \mathbf{h}_t, \mathbf{a}_t) \\ \text{Stochastic state model} \quad & \mathbf{z}_{t+1} \sim p(\mathbf{h}_t) \\ \text{Observation model} \quad & \mathbf{s}_t \sim p(\mathbf{z}_t, \mathbf{h}_t) \end{aligned} \quad (11)$$

where $f(\mathbf{z}_t, \mathbf{h}_t, \mathbf{a}_t)$ is a GRU with 200 hidden dims. The RSSM is trained similarly to our model above, performing gradient steps on reconstructions of batches of image sequences. The loss function used to train the RSSM is the following:

$$\begin{aligned} \mathcal{L} = & \underbrace{\log p(\mathbf{s}_t | \mathbf{z}_t, \mathbf{h}_t)}_{\text{reconstruction loss}} \\ & + \beta_1 \underbrace{D_{KL} [q(\mathbf{z}_t | \mathbf{s}_t, \mathbf{h}_t) || p(\mathbf{z}_t | \mathbf{h}_t)]}_{\text{transition loss}} \end{aligned} \quad (12)$$

Identically to our model, the latent variables \mathbf{z}_t and \mathbf{h}_t had 30 dimensions each. Image encoding and decoding was done with the same convolutional neural network architectures described in the previous section. We perform 25 gradient steps per episode, drawing $B = 50$ sequence chunks of length $N = 50$, with the transition KL term $\beta_1 = 0.1$.

A.5 POLICY LEARNING

Agents were trained to maximize future reward using the actor-critic method. Both actors and critics were parameterized by feedforward neural networks with an encoder and decoder, each of which consisted of two hidden layers with 800 ReLU units. The policy was learnt using the following algorithm:

Algorithm 1 Policy learning

```

for each episode do
  for each environment step do
     $\mathbf{z}_t = f_\phi(\mathbf{s}_t)$ 
     $\mathbf{a}_t \sim \pi_{\theta_{\text{actor}}}(\mathbf{z}_t)$ 
     $r_t, \mathbf{s}_{t+1} \sim \text{Env.step}(\mathbf{s}_t, \mathbf{a}_t)$ 
     $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ 
  end for
  for each dynamics gradient step do
    Sample batch of  $N$  tuples  $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \sim \mathcal{D}$ 
    Compute loss  $\mathcal{L}_{\text{dynamics}}$  ▷ Use objective of the relevant dynamics model
     $\theta_{\text{dynamics}} \leftarrow \theta - \alpha \nabla \mathcal{L}_{\text{dynamics}}$ 
  end for
  for each policy gradient step do
    Sample batch of  $M$  tuples  $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \sim \mathcal{D}$ 
    Compute actor and critic losses  $\mathcal{L}_{\text{actor}}, \mathcal{L}_{\text{critic}}$  ▷ Use soft actor critic losses
     $\theta_{\text{actor}} \leftarrow \theta_{\text{actor}} - \alpha \nabla \mathcal{L}_{\text{actor}}$ 
     $\theta_{\text{critic}} \leftarrow \theta_{\text{critic}} - \alpha \nabla \mathcal{L}_{\text{critic}}$ 
  end for
end for

```

A.6 SOFT ACTOR CRITIC

The actor and critic networks were trained using the Soft Actor Critic algorithm (SAC) (Haarnoja et al., 2018). To learn Q-values, we used two source critic networks and two target critic networks, fine-tuned the inverse reward scale parameter $\alpha = 0.5$, the number of policy gradient steps $M = 15$, batch size (150 in the gridworld and torus environments, and 350 in the Four Rooms environment) and the target value smoothing update constant $\tau_{\text{target}} = 0.1$. We trained the networks with the Adam optimizer (Kingma & Ba, 2014), using a learning rate of $1e - 4$. These hyperparameters were fine-tuned to maximize performance of the baseline agent (which had no latent dynamics model). We used these hyperparameter settings with other models and did no further fine-tuning, except for optimizing hyperparameters specific to the dynamics models. We trained the dynamics models using a learning rate of $1e - 3$, also using the Adam optimizer.

A.7 PLANNING

In the planning task we leveraged the Cross Entropy Method (Rubinstein, 1999) in tandem with learnt dynamics models, using hyperparameters similar to Hafner et al. (2019b). The agents planned over a horizon length $H = 15$, with 10 iterations I , 1000 samples J per iteration, and updating plans using the $K = 200$ best samples. We trained the dynamics models using a learning rate of $1e - 3$ with the Adam optimizer. To make sure the agents explored sufficiently, we used the epsilon-greedy heuristic, scheduling ε so that the agents explored a lot early in the experiment and little late. We made ε a function of the number of tasks played $\mathcal{E}(N) = 1 - \left(\frac{N - 1}{T}\right)^V$ where N is the current task number, T the total number of tasks and $V = 2.8$ a scaling term.

Below is a description of the Cross Entropy Method, drawing inspiration from Hafner et al. (2019b). We adapt it for a discrete action setting, by optimizing the logits of a categorical distribution from which actions are sampled.

Algorithm 2 Planning

```

for each task do
   $\mathbf{s}_{\text{start}}, \mathbf{s}_{\text{goal}} \sim \mathcal{S}$  ▷ Draw start and goal states uniformly, s.t.  $\mathbf{s}_{\text{start}} \neq \mathbf{s}_{\text{goal}}$ 
   $\mathbf{z}_{\text{goal}} = f_{\phi}(\mathbf{s}_{\text{goal}})$ 
   $\tilde{r}(\mathbf{z}) = e^{-\|f_{\phi}(\mathbf{s}) - \mathbf{z}_{\text{goal}}\|_2}$  ▷ Define reward function for model
   $\varepsilon_i \leftarrow \mathcal{E}(\text{task})$ 
  for each environment step do
     $\mathbf{z}_t = f_{\phi}(\mathbf{s}_t)$ 
     $\mathbf{a}_t = \begin{cases} \text{CEM}(\mathbf{z}_t) & \text{with probability } 1 - \varepsilon_i \\ \text{Random} & \text{with probability } \varepsilon_i \end{cases}$  ▷ Use Cross Entropy Method
     $r_t, \mathbf{s}_{t+1} \sim \text{Env. step}(\mathbf{s}_t, \mathbf{a}_t)$ 
     $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ 
  end for
  for each dynamics gradient step do
    Sample batch of  $N$  tuples  $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \sim \mathcal{D}$ 
    Compute loss:  $\mathcal{L}_{\text{dynamics}}$  ▷ Use objective of the relevant dynamics model
     $\phi_{\text{dynamics}} \leftarrow \phi - \alpha \nabla \mathcal{L}_{\text{dynamics}}$ 
  end for
end for

```

Algorithm 3 Cross Entropy Method

```

Require: Dynamics model  $p(\mathbf{z}_{t+1} \mid \mathbf{a}_t, \mathbf{z}_t)$ , reward function  $\tilde{r}(\mathbf{z})$ 
Initialize  $J$  logits for categorical action distributions  $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$ , where  $\mathbf{w} \in \mathbb{R}^{|\mathcal{A}| \times H}$ 
for each iteration  $i = 1, \dots, I$  do
  for each action sequence  $j = 1, \dots, J$  do
     $\mathbf{a}_{1:H} \sim \text{Softmax}(\mathbf{w})$  ▷ Sample actions
     $\mathbf{z}_{1:H} \sim \mathbf{p}(\mathbf{z}_{1:H} \mid \mathbf{a}_{1:H-1}, \mathbf{z}_{1:H-1})$  ▷ Simulate future states with dynamics model
     $\mathbf{r} \leftarrow \tilde{r}(\mathbf{z})$  ▷ Compute rewards using reward function
  end for
   $\mathcal{K} = \text{argsort}\{\mathbf{r}_{j=1}^J\}_{1:H}$  ▷ Get logits of  $K$  best performing samples
   $\boldsymbol{\mu}_{1:H} \leftarrow \frac{1}{K} \sum_{k \in \mathcal{K}} w_{1:H}^k$ 
   $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I})$  ▷ Sample new logits based on average logits of  $K$  best samples
end for

```
