ROBOTICS AND MECHATRONICS

UNIFYING STATE REPRESENTATION LEARNING WITH INTRINSIC MOTIVATIONS IN REINFORCEMENT LEARNING

B.K. (Beck) Wittenstrom

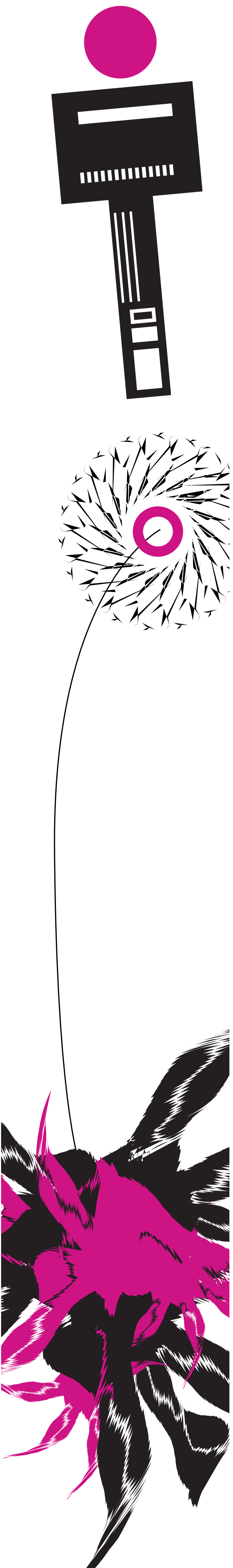
MSC ASSIGNMENT

Committee:
dr. ir. J.F. Broenink
N. Botteghi, MSc
dr. M. Poel

September, 2021

063RaM2021
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE   UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE



Unifying State-Representation Learning with Intrinsic Motivations in Reinforcement Learning

Beck Wittenstrom

September 23, 2021

Contents

Acronyms

MDP Markov Decision Process. 6–9, 14

RL Reinforcement Learning. 2–6, 8, 12–20, 27, 34, 35, 37

SRL State Representation Learning. 3–5, 7–9, 11–16, 18, 19, 21, 25, 34–37

SSR Synthetic State Representation. 3, 4, 8, 12–25, 27, 29–32, 34, 35

TSR True State Representation. 19–21

1 Summary

Robots are in demand to work in unknown or unpredictable environments such as navigating roads or picking objects from a random pile. Model based control methods cannot work in these environments. Learning based control methods, such as reinforcement learning, do not need accurate models to find good control policies in unknown environments. Reinforcement learning however, suffers from the curse of dimensionality. The computation power needed increases exponentionaly with the size of the robots observation. This problem can be mitigated by filtering out important features in the observation into a low dimensional synthetic state representations. Currently, synthetic state representations are trained using a history of observations and actions gathered using a random action policy. A random action policy does not use the information gathered adjust what states its samples. There is a possibility to improve the training of synthetic state representation by improving the choice of actions used to collect samples to train the synthetic state representation.

We trained state representations for an environment with simple consistent visual features, and one with complex distractor features. For each environment, we tested four different training policies random action policy, entropy maximization, prediction error maximization, and uniform sampling.

We found different sampling methods can lead to different sampling distributions, depending on the training parameters and environment. The uniformity of coverage is important for complex environments where distractor features in one part of the environment do not generalize to other areas. The uniformity is not important to learn a good structure when the environments features are consistent enough that the SRL can generalize from one area of the environment to another. Finally, The relation between structure of the state representation and the performance of RL policy is complex. In the simple environment better structural scores seems to improve RL performance. In the complex environment this is the opposite case. In the complex environment, clustering of states caused by the distractor features may be disruptive to policy learning.

Sampling methods that lead to a more uniform sampling distribution may improve the state representation learning structural performance performance. However, this is only the case of complex environments where generalization is impossible. Finally, what a good structure is for a synthetic state representation is still unknown. This is because, the "improved" structure of the state representation does not necessarily lead to higher RL performance. Therefore more research needs to be done into how the structure of state representations can facilitate RL performance, and how sampling methods can support the learning of those structures.

2 Introduction

2.1 Context

There is a demand for robots to work where they do not have complete knowledge of the environment. Robots excel at tasks where the environment is accurately known. For example, in a car assembly line, a component is always placed in a precise position. The robot always knows where to pick up the component and the robot can compensate for small variations in that position. The robot does not have this level of knowledge in a task where the robot must pick up various packages of food from a pile. To do this, the robot locate objects and choose different gripping methods for each object. The robot will have to deal with significant variation in size, shape, and mass that the robot does not know ahead of time. These unknown variations are too difficult for model based control methods, such as impedance control methods.

It is too difficult for model based control methods such as the control methods such as using impedance control. In the pick and place task the position and weight of the object is not known. Without means that the ideal gripper position is not known so the inverse kinematics models cannot determine the between the gripper position and the joint positions. Finally, without ideal joint positions or the weight of the object the impedance control cannot the force on the robot leads to a desired position of the robot. Therefore, new control methods are needed to deal with low-knowledge environments.

2.1.1 Reinforcement learning in Robotics

A learning robot can overcome the lack of knowledge by learning a control method regardless of what environment it is placed in. A robot can learn from experience, which is the history of observations and actions it has seen. The field that studies algorithms that learn control policy from experience is called RL (Sutton et al., 1998). Reinforcement learning can overcome low knowledge environment, but this is at the computational cost that increases exponentially with the dimensional size of the observations and actions. This is a common issue when the robot uses camera observations.

People use experience of the observation-action cycle to learn, for example to throw a ball. The precise weight and shape of the ball may be unknown, but a person can still throw a good pitch with enough trial and error. The person tries different throws, then observes how far the ball got on each throw. The person then learns which types of throws lead to the most distance. This learning method can be applied to any problem where there is some reward to be optimized such as a robot learning maximize how far it can throw a ball, or the completion of a maze. The learning strategy of optimizing the reward by trail and error is called RL in Computer Science (Sutton et al., 1998).

RL works by optimizing the actions taken by a robotic agent given the observation of the environment. For example, a ball-throwing robot may have cameras to observe the environment, and an arm to interact with it. The goal is to throw the ball as far as possible. Using trial and error, the robot gets a history of observations, actions, and distances the ball moved. These histories can be generalized for any environment and task as three signals the observation, action, and reward. The reward is given to the RL algorithm when the goal is achieved. The RL algorithm attempts to learn a control policy that maximizes the total reward gained over time. The control policy is the function that chooses action given the observation. The reward does not have to be continuous like throwing distance, the reward can be discrete such as when the ball is tossed into a bucket. Even if the reward is given when the ball is tossed into a bucket, the Reinforcement Learning (RL) algorithm will learn to complete the task. This means that the engineer does not need to design complicated rewards to teach the robot to complete tasks.

Learning-based control methods are more flexible than standard control methods such as inverse kinematics and impedance control. In the latter control methods, if the prior knowledge is wrong, then the system fails. Reinforcement learning does not need any prior knowledge to learn to complete a task, there fore is robust to false prior knowledge (Sutton et al., 1998) This ability makes the robot that can use RL more robust to the unknown environments than traditional control methods. Research developing better learning-based control methods will help robots ability to complete tasks with unknown knowledge (Schulman et al., 2017; van Hasselt et al., 2015).

One problem with a RL agent is the computation time needed increases exponentially as the dimensions of the observation increases (Sutton et al., 1998). It is easier for a RL algorithm to find good policies if it uses the position of the joints of a robot arm rather than a camera image of the robot arm as the observation. If given the observation, the RL algorithm initially considers all elements of the observation equally important. However, many features of the observation are not critical to the task. For example, background lighting, the color of the robot, etc. The RL algorithm must first determine which features of the image to ignore. The difficulty of deciding what to ignore becomes exponentially harder the more dimensions there are. RL performance can be increased with methods to filter out unimportant information from the image and reducing its dimensionality.

2.1.2 Introduce State Representation Learning

Researchers have compensated for the problem of high dimensional observations by reducing the dimensional of observations using neural networks. The neural networks find important features in the image for the task and discard the rest of the image. These compressed observations are called Synthetic State Representation (SSR) in this paper. The techniques to learn them are called State Representation Learning (SRL). The techniques use assumption about the physical world, such as local linearity, to design loss function that are minimized by learning good features. These neural networks are usually trained on samples gathered using a random action policy. There is room for improvement in SRL algorithms by guiding the robot to take actions that allowing it to learn the best SSR.

A SSR of the observation is a function of an observation that contains only the important features. For example, in an apple picking task, the SSR of the observation of the apple might contain only the diameter and position of the apple. The SSR would discard information such as the color of the apple, the number of surrounding leaves, etc. A SSR of an observation can be extracted using neural networks.

Neural networks can learn features by rewarding SSR with good properties like predictability and generalization. For example, if the robot takes actions that move toward the apple, then the apple will get bigger in the observation. This is a predictable consequence of that action, and the neural network will learn to encode it. The neural network method of learning a SSR of an observation is a state representation learning algorithm.

The purpose of SRL is to learn features that the robot needs to solve the task. For example, the position of the apple and robot. An SRL finds these features by learning from the cycle of actions and observations. SRL is not just compressing the image as much as possible but should be focused on important features for the robot. Physical features such as size, position, and velocity are important for robotic tasks. Physical features have some common characteristics that an SRL can use to find them. Some characteristics are that feature is controllable by the robot, is predictable, and changes slowly (Jonschkowski and Brock, 2015). For example, the apples position is controllable when the robot makes contract with the apple. An SRL finds these features from experience of actions and observations. State of the art SRL algorithms gather experience to train by choosing a random action (Jonschkowski and Brock, 2015; Stadie et al., 2015; Watter et al., 2015).

2.1.3 Exploration and Sample Collection for SRL

A random action policy chooses action uniformly from the set of actions available to the robot (Jonschkowski and Brock, 2015; Stadie et al., 2015; Watter et al., 2015). For example, a robot may choose between moving left, right, up, or down in a navigation task. Sampling random actions would simply select each action with a uniform random distribution. There are three shortcomings to this policy. The first is that it is biased to the starting location leading to repetitive sampling because it does not actively avoid actions that loop back to the initial location. Another problem is obstacles may make a particular sequence of actions required to reach an area (Pathak et al., 2017). Random action sampling may never reach these areas. Finally, some dynamics are easier to learn if done in a specific order. For example, the SSR should learn the dynamics of the robot arm before adding the dynamics of the apple (Sharma et al., 2020).

The SRL process could be improved by using a more intelligent sampling method. For instance, people do not use random actions to learn new things. They explore areas likely to provide new information and avoid unnecessary repetitions. This behavior is called curiosity. Curious behavior can be integrated into the RL framework by having the agent generate a reward when it experiences something new or makes learning progress (Oudeyer et al., 2007). Curiosity rewards are a reward generated from the history of observations and actions of the robot to motivate it to learn efficiently without guidance. A successful internal reward motivates a learning agent to learn efficiently. The goal of a curiosity reward is to learn meaningful behavior without external rewards. External rewards are rewards designed by an expert to teach a robot how to do a particular task. Meanwhile, curiosity rewards are a part of the learner and guide its learning regardless of the task. A curiosity reward could help a SRL algorithm sample more efficiently by mitigating the issues of a random action policy (Aubret et al., 2019). The effect of SRL on RL performance, and the effect of curiosity rewards on RL performance have been studied extensively. The effect of curiosity reward driven sampling on SRL has not.

2.2 Contribution

Our contribution to the literature is investigating how intelligent exploration can improve SRL performance compared to random action policy. In this research we explore the relation between curiosity reward and SRL by comparing the effect of four difference sampling methods on the quality of a the SSR that is learned by the SRL algorithm. The effect of the four difference sampling methods is tested on a simple and complex environment with distractors. The effect of sampling method is split into the effect of the sampling method on the sample distribution. The effect of the coverage of the sample distribution on the structural quality of the SSR. Finally, the effect of structural quality on the RL performance.

In the simple environment we show how uniform sampling from the state action space, random action sampling, and two different curiosity rewards lead to different sampling distributions in the simple environment. The data suggests that the temporal and final characteristics of the sampling distribution do not lead to significant differences in the structural quality of the SSR in a simple feature grid world environment. The improved structural quality of the trained SSR appears to improve the ceiling of RL performance.

On the complex environment the random and two curiosity rewards lead to similar coverage distributions possibly due to the max steps per episode during training. Uniform sampling however increased the likely-hood of good structural scores compared with the other sampling methods. In the complex environment improved structural scores are detrimental to reinforcement learning performance. This may be due to that in some environment adjacent states may require different actions, and the SRL places adjacent states close together making it more difficult to differentiate them for the policy.

2.3 Research Questions

- *Do curiosity rewards improve the, generalization, quality, or sample efficiency of a state representation learning algorithm?*
    - ◇ *To what extent does the uniformity of coverage of observations/states affect the quality of the resulting synthetic state representation?*
    - ◇ *To what extent does the order that observation/states are visited affect the quality of the resulting synthetic state representation?*
    - ◇ *How does the changing state space representation effect the stability of the reinforcement learning algorithm?*

2.4 Report Outline

The report is organized as follows: Chapter 2 presents the background information related to RL and SRL. Chapter 3 presents the state of state representation learning and curiosity learning research. Chapter 4 presents the framework that we used to analyze the connection between curiosity rewards and state representation learning. Chapter 5 presents precise experimental methodology covering all the details and hyperparameters of the experiments. Chapter 6 presents the results of the experiments. Chapter 7 discusses the meaning of the results and concludes the report by answering the research questions.

3 Backround

3.1 Markov Decision Process

A Markov Decision Process (MDP) is a mathematical framework for modelling a decision making process in presence of uncertainties. A MDP is defined by a tuple $\langle S, A, T, R\rangle$. S is a set of states, and A is a set of action that can be chosen. T defines the conditional transition probability from one state to another given the initial state and the chosen action. T is defined mathematically as $p(s_{t+1}|s_t, a_t) = T(s_t, a_t)$ where s_t is the state $s \in S$ at time step t and a_t is the action $a \in A$ at time step t.s_{t+1} is the next state at time step t+1 that results from taking action a at state s. Finally, R is a reward given to the agent as a function of the initial state the action and the final state. The reward is always a real number. The causal dynamics are represented graphically in figure 3.1. A policy is the controller for a MDP. It is defined as $\pi(a|s)$ is the probability of action a given the current state s.

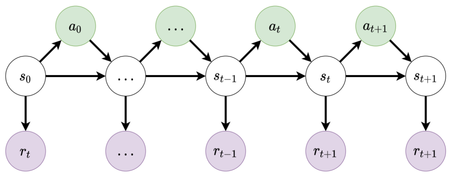


Figure 3.1: MDP

The goal is to find an optimal policy $\pi^*(a|s)$ that maximizes the expected cumulative reward obtained over time. The cumulative reward over time G can be over finite N time steps represented by Equation 3.2. G can also be calculated over a discounted infinite horizon where γ is a decay constant, $0 \le \gamma < 1$ to ensure G cannot be infinity. The γ is a weight on the importance of future rewards. This is shown in Equation 3.2.

$$G_{finite} = \sum_{t=0}^{N} R(s_t) \tag{3.1}$$

$$G_{inf} = \sum_{t=0}^{\infty} \gamma^t R(s_t) \tag{3.2}$$

3.2 Value Functions

The objective of RL agents is to find a policy for a MDP that maximizes the expected reward discounted over all time in the MDP. The expected discounted reward over time given a certain initial state is called the value function. The value function is defined by Equation 3.3

$$V_\pi(s) = \mathbb{E}(G_{inf}|s, \pi)) \tag{3.3}$$

The optimal policy in a MDP is a policy that has a greater or equal value than any other policy. An optimal policy $(\pi^*(a|S))$ is defined in equation 3.4.

$$V_{\pi^*}(s) \ge V_\pi(s), \forall \pi \tag{3.4}$$

The value function can be computed recursively using the Equation 3.5.

$$V_\pi(s) = \sum_a \pi(a|s)\Big(r(s) + \gamma \sum_{s'} p(s'|s, a)\Big) \tag{3.5}$$

3.3 Q-Value

In order to choose the optimal action at a specific state, it is important to relate value to action. The expected value of an action is defined at a state when is defined as the Q value. The Q value is defined by the equation 3.6. For simple environments, the Q value can be calculated using a lookup table for every state action pair. If the state action space is large the memory required to store every for this becomes unfeasible. For example, a continuous environment has infinite states.

$$Q^{\pi}(s,a) = r + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^{\pi}(s) \tag{3.6}$$

If the optimal Q value is known the optimal policy chooses the action that maximizes the Q value.

$$\pi^*(a \mid s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q^*(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The optimal Q value can be found by converging to the correct value using temporal difference iterative updates. The update function is shown in Equation 3.8.

$$Q(s,a) \leftarrow Q(s,a) + a[r\gamma \max_{a'} Q(\tilde{s}', a') - Q(\tilde{s}, a))] \tag{3.8}$$

Neural networks (Ostrovski et al., 2017). A Q-Learning neural network(NN) uses the state as an input to predict the Q value for each action. To ensure sufficient exploration the agent chooses a random action a percent of the time and otherwise chooses the action that leads to the maximum Q value. This policy is called ϵ-greedy where ϵ represents the percent of the time a random action is chosen. The $\epsilon-$greedy policy collects state transitions in its memory. State transitions are a set containing the initial state, action, next state, and reward (s,a,s',r). The state transitions in memory can be used to calculate the Q-update. The NN can use the Q update as a target to train the NN by minimizing the loss in Equation 3.10 for a batch of transitions from the memory.

$$Q_{target}(s,a) = r + \gamma \max Q_{NN}(s',a') \tag{3.9}$$

$$Loss = (Q_{target}(s,a) - Q_{NN}(s,a))^2 \tag{3.10}$$

3.4 Reinforcement Learning

Reinforcement learning framework has two parts a goal directed agent and the enironment that agent interacts with. The agent has to explore to gain knowledge and exploit that knowledge to achieve a goal. The agent can use a variety of tools to do this such a reward maximizing policy or interpreting the environment such as SRL. The environment is the fixed dynamics of the world including the robot model, sensor model, and physical effects. The signals flowing between the agent and the enironment can be visualized in figure 3.2.

3.5 Block Markov Decision Process

A robot can gather observations with sensors that may not be the exact true state of the environment. The framework for this is the Block MDP (Sanders et al., 2020). Block MDP is an extension to the MDP framework where there is a set of observations corresponding to every state. Each state may correspond to many observations, but each observation has only one corresponding state. The formal property of an observation is if $p(o|s_1) > 0$ then $p(o|s_j) == 0 \forall s$. The property leads to a cluster structure of observation transitions that are visualized in Figure 3.3. The Block MDP is made up of an observation signal, action space, unobservable state, and rewards (o, a, s, r). A Block MDP can be solved using Q-learning by using the observation as input instead of the state, but the higher dimensionality of the observations makes it more difficult.

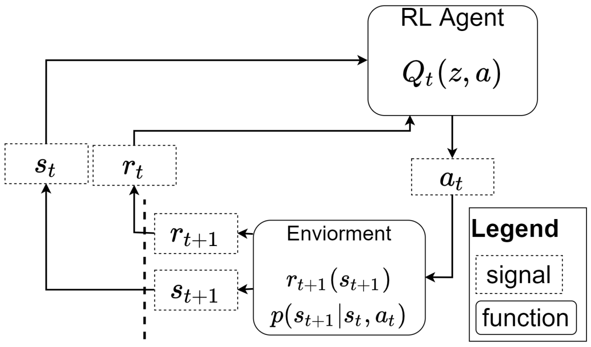


Figure 3.2: Reinforcement learning(RL)

3.6 State representation learning

The objective of SRL is to find the features in an observation that are important for the agent to learn an optimal policy. The features should act as a low dimensional synthetic state space that makes it easier to learn the optimal policy for the underlying MDP. SRL algorithms find features that follow common properties of robotic MDP's. For example, physical states like position usually change slowly and causally from actions (Jonschkowski and Brock, 2015). This property should be present in a learned SSR. The synthetic state space made by made by the compressed features of the observation does not need to have the same equal to the true state space. Instead, the synthetic state space should have certain properties.

A synthetic state space should have four qualities to improve the performance of an RL algorithm. Be low-dimensional, the value function should generalize to states not seen before, be able to represent the true value of the optimal policy, and the state space should be markovian (Böhmer et al., 2015). The low dimensionality and the ability to generalize to new states make it easier for anRL to learn the optimal policy. Low dimensionality makes it easier for the RL to find patterns in the data. The ability to generalize makes learning faster because the policy will not have to change the policy as new data comes in. The ability to represent the true value of the optimal policy and the markovian property are important to ensure anRL algorithm can find the optimal policy. The problem occurs if two observations from two different states map to the same position in synthetic state space. The RL algorithm cannot differentiate the two states and cannot change the policy accordingly. One way to ensure synthetic state space is markovian and can represent the optimal policy is to make the synthetic state space a homomorphism of the MDP.

3.6.1 MDP Homomorphism

If a synthetic state space is a homomorphism of the MDP, then the synthetic state space can represent the optimal value function and is markovian. A synthetic state homomorphism if two properties are satisfied. The properties are shown in Equation 3.11, 3.12. If a synthetic state space is a homomorphism of an MDP, then the optimal policies are the same (Ravindran and Barto, 2004). This property of a homomorphism ensures the homomorphism can represent the optimal value function of the original MDP. The homomorphic synthetic state space continues to be markovian as it is a linear combination of a markovian transition space (Ravindran and Barto, 2004). Generalizability, however, is not guaranteed. If $M = \langle S, A, T, R\rangle$ is an MDP with set of states$S$, set of action A, transition probability distribution T, and set of rewards R'. $\hat{M} = \langle \hat{S}, \hat{A}, \hat{T}, \hat{R}\rangle$ is a homomorphic image of M if transformation state mapping $f : S \rightarrow \hat{S}$ such that.

$$p'(f(s')|f(s),a) = \sum_{s' \in f^{-1}(f(s'))} p(s'|a,s) \tag{3.11}$$

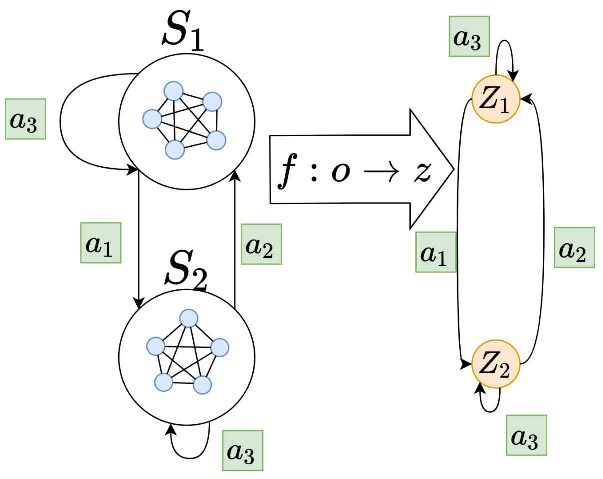


Figure 3.3: Blue circles represent different observations associated with a state. The lines connecting the observations represent the transitions possible between observations within the state

Equation 3.11 is interpreted as the probability transition in the homomorphic image are equal to the sum of transition probabilities

$$R'(f(s)) = R(s) \tag{3.12}$$

For deterministic MDP homomorphism then the equations above simplify to Equation 3.13 and 3.14. Where $T_1()$ is the deterministic transition function of the homomorphism $f(s)$, and $\hat{R}$ is the reward function of the homomorphism.

$$\forall_{s \in S, a \in A} T_1(f(s), a) = f(s') \tag{3.13}$$

$$\forall_{s \in S, a \in A} \hat{R}(f(s)) = R(s) \tag{3.14}$$

Equation 3.12 is interpreted as the equivalent state in the homomorphic image has an equal expected reward. A critical property of homomorphic images is the optimal Q value on the original MDP and the homomorphic image is the same $Q^*(s,a) = Q^*(f(s),a)$. This means that if an optimal policy is learned on the homomorphic image it will also be the optimal policy for the original MDP. A SRL algorithm that generates a homomorphism of the state space can guarantee a good policy can be learned. It does not guarantee that the synthetic state space will generalize the value function well. A visualization of a homomorphic mapping is shown in Figure 3.3.

4 Related Work

Research into state representation learning and curiosity rewards are two separate lines of research. First, I summarize the state of the art approaches used for state representation learning. Then, I summarize state of the art approaches in curiosity reward driven learning. Finally, the possible effect of curiosity reward on different state representation learning methods is discussed.

4.1 Approaches for Learning State Representations

There are six common state representation learning methods. Each uses a neural network to minimize a different losses which try and compress the observation while preserving important information. The six approaches are the auto encoder, forward model, inverse model, robotic priors, and contrastive loss.

The autoencoder minimizes the difference between the observation and a reconstructed observation from the state space. The idea of the autoencoder is to compress all information in the observation into a synthetic state space (Alvernaz and Togelius, 2017). This should ensure that the synthetic state space has all the information to be able to represent the true value of the optimal policy. In practice, autoencoders give too much attention to the background at the expense of features that are important to the agent's task (Alvernaz and Togelius, 2017). For example, in the atari game asteroid, an autoencoder may ignore the small asteroids as they influence the total image less than larger objects and the backround (Anand et al., 2019). Autoencoders also do not encode the dynamics features of the system as they are only rewarded for reproducing their input. The dynamic features can be encoded using by combining the auto encoder with a forward model reconstructing an image at the following time step (Lesort et al., 2017; Zhang et al., 2018).

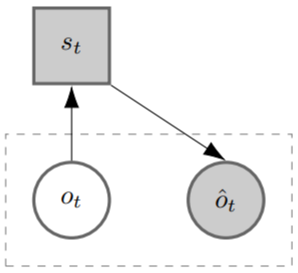


Figure 4.1: Diagram of autoencoder. White components are input data and gray ones are output data. variables with a hat is the estimate of the variable. The dashed box is the error loss (Lesort et al., 2018)

4.1.1 Forward Model

A forward model learns to predict the next state using the current state and action. This method will encode information that facilitates the prediction of the next state (Pathak et al., 2017). The forward model can be forced to be locally linear (Watter et al., 2015). This may help with the generalizability of the representation. A forward model may learn a trivial model such as all observations map to zero in state space. This maximizes state predictability. However, it removes information needed to differentiate rewards. The state space will then not have enough information to represent the true value of the optimal policy. Any forward model is usually paired with some other loss to avoid this outcome (Zhang et al., 2018).

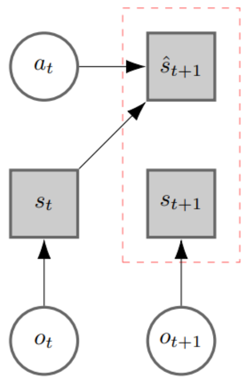


Figure 4.2: Diagram of forward model. (Lesort et al., 2018)

4.1.2 Inverse Model

An inverse model will encode enough information to reconstruct the action. If the state drifts, the drifting features may be ignored because it does not correlate with the action. The trivial solution of the forward model can be prevented by having the loss of the SRL be a sum of an inverse and forward model (Zhang et al., 2018). If the forward model encoding all states to zero, then there will not be enough information to reconstruct the action leading to high loss.

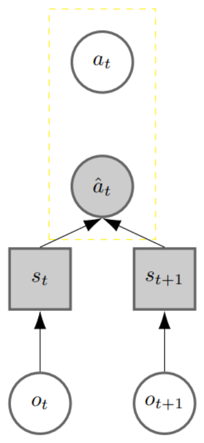


Figure 4.3: Diagram of inverse. loss is the error of the estimate of action and true action (Lesort et al., 2018)

4.1.3 Robot Priors Methods

Robotic prior methods optimize the synthetic state space to conform to certain assumptions about how a robot usually behaves. The assumptions are slowness, causality, proportionality, and repeatability (Jonschkowski and Brock, 2015). The Slowness assumption is that important features change slowly over time. For example, the mass of the robot keeps the velocity from fluctuating erratically. The causality assumption is similar state action pairs lead to similar rewards. The proportionality prior says that the change of the state is proportional to the size of the action. The repeatability assumption that similar state action pairs lead to similar states at the next time step. These assumptions are implemented as a set of loss functions shown in 4.1-4.4. The robotic priors are integrated as a single sum in literature (Jonschkowski and Brock, 2015).

$$\mathscr{L}_{slowness} = \mathbb{E}\left[|\Delta s|^2\right] \tag{4.1}$$

$$\mathscr{L}_{variability} = \mathbb{E}\left[e^{-|\Delta s_{t_1} - \Delta s_{t_2}|}\right] \tag{4.2}$$

$$\mathscr{L}_{Proportionality} = \mathbb{E}\left[(|\Delta s_{t_1}| - |\Delta s_{t_2}|)^2 | a_{t_1} = a_{t_2}\right] \tag{4.3}$$

$$\mathscr{L}_{Repeatability} = \mathbb{E}\left[e^{-|\Delta s_{t_1} - \Delta s_{t_2}|^2} |\Delta s_{t_1} - \Delta s_{t_2}|)^2 | a_{t_1} = a_{t_2}\right] \tag{4.4}$$

4.1.4 Contrastive Loss

The contrastive loss maximizes the distance between the next synthetic state in a trajectory and synthetic states from other time steps in the trajectory. This method rests on the assumption that the trajectory does not backtrack often, and different time steps usually represent unique states. The contrastive loss rewards the encoding if it maps observations from different states to different positions in synthetic state space. A contrastive loss combined with a forward loss and a reward loss that predicts the reward associated with each synthetic state will converge to a homomorphism of the true state. This is proven in Theorem 3.1 of (van der Pol et al., 2020). In practice, it can be difficult to tell if observations are from different states. Thus the contrastive examples are sampled uniformly from memory. Uniform sampling sometimes misclassifies two observations from the same state as from different states. Then, the system converges to not exactly but almost a homomorphism (van der Pol et al., 2020). If an exploration policy over-samples one state, then the contrastive loss might have reduced performance. This is because the number of misclassified negative samples will increase.

4.2 Evaluation of State representation learning

State representations can be evaluated by checking if RL algorithms perform better on the state representation compared to performance on observations (Linke et al., 2020). This process is time-consuming and does not give information about why there is an improvement. For example, a randomly initialized convolutional neural network SSR leads to significant improvement over observations (Burda et al., 2018b). Another method to evaluate a SSR is to check the local topography of the synthetic state relative to the ground truth state. These methods measure local topography by checking if the K nearest neighbors in the SSR and the true state space are similar (Zhang et al., 2012). Finally, a SSR can be tested to determine that it contains all the information in the true state space. This is tested by checking the reconstruction error of the ground truth using regression techniques to relate the SSR and true state space. The reconstruction can be done with linear regression or with a simple neural network. If the true state can be reconstructed from the synthetic state, then the synthetic states should contain all the information of the true states (Jonschkowski et al., 2017; Morik et al., 2019).

4.3 Training Synthetic State Representations

Synthetic state representations are usually trained using random walk (Alvernaz and Togelius, 2017) or during an RL e-greedy policy. Training a SSR during an RL run, harms theRL policy performance as it is training on a nonstationary state space (Burda et al., 2018b). Also, The e-greedy policy is equivalent to the random policy, if only a couple states in the enironment have an associated external reward (Stadie et al., 2015). This is because all sampled states give a reward of zero, and this causes no policy that the algorithm has seen is better than any other.

4.4 Related Curiosity work

Random action is not the most efficient exploration policy. The agent can be given some curiosity reward to improve the sampling policy for the SRL without external rewards or guidance. There is extensive research into how intrinsic rewards be generated from the observation action cycle, and can be used to improveRL performance. There is little research on how curiosity reward affect the sampling policy of an SRL (Aubret et al., 2019). There are three groups of curiosity rewards, count-based, prediction error, and skill learning.

4.4.1 Count-based Exploration

Count-based methods reward anRL agent for visiting states that it has not visited before, using the reward in equation 4.5. $n(s,a)$ is the number of times the state action pair (s,a) has been visited.

$$r_i = \frac{\beta}{\sqrt{n(s,a)}} \tag{4.5}$$

The count method is difficult to calculate large or continuous spaces. The behavior of count-based rewards can be mimicked by using a reward equal to the knn-entropy, shown in equation 4.6. The knn-entropy is the sum of the log of the distance of a point in the synthetics state to the K nearest neighbors. The K nearest neighbors are found from the set of states that the robot has previously visited. This should push the robot to visit each state uniformly (Seo et al., 2021).

$$r^i_{Hmax} = log(||z_i - z_i^{knn}|| + 1) \tag{4.6}$$

4.4.2 Prediction Error Exploration

Prediction error methods use the error in the forward model of a learned SSR as an intrinsic reward for anRL algorithm (Pathak et al., 2017). They work because the robot occupies an area of space it will begin to learn the forward model. The error will then decrease, and the lack of error will make the robot "bored". Then, it will seek out new experiences. This method can get stuck on a white noise generator in the environment as prediction error will never decrease if the environment is not predictable. Maximizing prediction error reward also acts as an adversarial reward to a SSR which may improve learning as it pushes the attention of the SRL algorithm to observations that are difficult to encode.

4.5 Unifying state representation learning

The research on SRL uses random action to train a SSR. Then, the SSR is used as the input for reinforcement learning. The research into curiosity rewards uses SSR to generate curiosity rewards. The rewards guide a reinforcement learning algorithm that uses the raw observations as input. Our research uses the SSR as input for reinforcement learning and curiosity rewards to guide sampling. The purpose is to capitalize on the reduced dimensions of a SSR and the smart sampling of curiosity rewards.

5 Methodology

5.1 Overall Framework

The curiosity reward and SRL are united in the agent in Figure 5.1. The environment and sensor are formulated as a block MDP(o, a, s, r), see Section 3.5. This paper explores the interaction of the curiosity reward and SRL in this new framework. How curiosity rewards can help choose actions that gather samples. Then, how the coverage can improve the SRL algorithm.

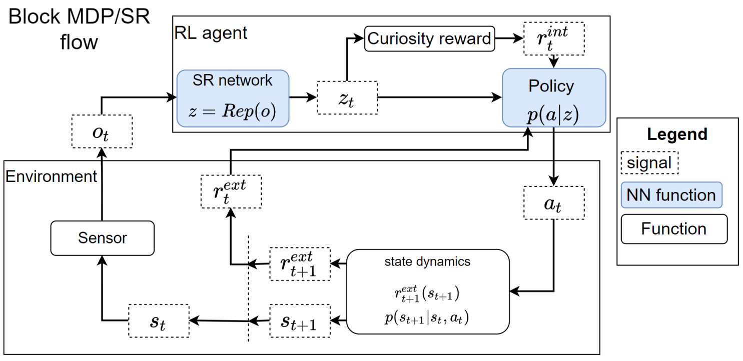


Figure 5.1: The signal flows through neural networks in a trained SRL, andRL policy. During Training the SR network optimizes the loss function in Eq 6.1-6.3 and the Policy Neural network optimizes Eq 3.10 from their memory of the signals

Figure 5.1 shows the loop of signals going from the enironment to the agent and from the agent to the enironment. The agent is executing a the Policy choosing actions for each state. The history of the all transitions $(o_t, a_t, o_{t+1}, r_t^{ext})$ is used to train the SR network. The RL policy is trained on the history of all transitions $z_t, a_t, z_{t+1}, r_t^{ext} + r_t^{int}$, where r^{int} is the curiosity reward generated by the SSR. For our experiments the policy updates once each time step, however this does not need to be the case. The policy could update multiple times each time step, or every 100 steps. This shows that the design space within this framework is vast.

5.2 Analysis of the Framework

In order to analyze the effect of fusing SRL with curiosity on the efficacy of SRL, we focus on four key variables.

1. The sampling method
2. The state action sampling distribution that the sampling method achieves
3. The quality of the learned SSR
4. The RL performance achieved using the learned SSR

They are shown in Figure 5.2.

Bubble one of Figure 5.2 represents the sampling method variable, or method. A method is a sampling policy used to gather transitions to train the SSR. These are simple policies like random action, a RL that optimizes a curiosity reward(ICM, Hmax), or uniform sampling of the state action space. The goal of the method is to add most informative transitions into the memory buffer and add them in an order that facilitates the SRL algorithms ability to learn a good SSR in as little samples and SRL updates as possible. We compare four different methods which are shown under the methods bubble. Random action is uniform selection of all possible actions not taking into account any feedback from the environment. Hmax is a curiosity reward

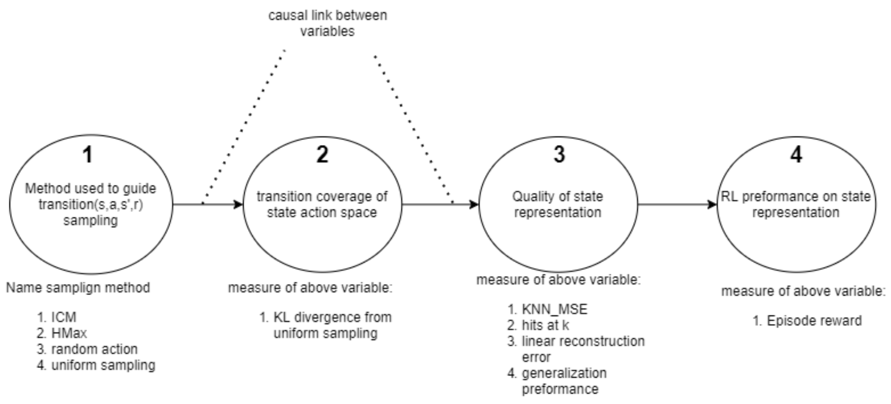

causal link between variables
1
Method used to guide transition(s,a,s',r) sampling
2
transition coverage of state action space
3
Quality of state representation
4
RL preformance on state representation
Name samplign method
1. ICM
2. HMax
3. random action
4. uniform sampling
measure of above variable:
1. KL divergence from uniform sampling
measure of above variable:
1. KNN_MSE
2. hits at k
3. linear reconstruction error
4. generalization preformance
measure of above variable:
1. Episode reward

Figure 5.2: The causal connection between the sampling policy, the gathered transitions, the quality of the state space, and the finalRL performance. Underneath each concept is a number of measures for each variable.

trying to maximize the uniformity of sampling from the synthetic state space, see section 4.4.1. ICM is a curiosity reward that maximizes the cumulative predictive error of the policy, see section 4.4.2. Uniform sampling is a uniform sampling of state and action space, it is a purely theoretical policy as it cannot be implemented on a real robot. Each sampling method affect the probability that a transition $(o_t, a_t, o_{t+1}, r_t^{ext})$ is added into the memory buffer for training.

Bubble two of Figure 5.2 represents the coverage. Coverage is the order and the number of times that a transition is added to the memory buffer. It is causally affected by the sampling method. The coverage can change over the course of training as the curiosity rewards Hmax, and ICM are incorporate feedback from previous samples. Below the bubble is the KL divergence from uniform of the coverage. This will be known as the KL divergence measure. This is a measure of how different the sampled coverage is from a perfectly uniform sampling of the state action space. KL divergence is a measure of one characteristic of the coverage that we think is important to learn a good SSR. The coverage distribution of transitions in memory over time affects the quality of trained SSR. The SRL neural network trains by taking a random set of samples from the memory buffer. The neural network update will be different depending on which transitions are in memory, causes the coverage to influence the quality of the learned SSR.

Bubble three of the Figure 5.2 represents the quality of the Synthetic State Representation (SSR) and the list underneath are various measures of a good SSR. The quality of the SSR is causally influenced by the coverage, and not the sampling method. It is influenced by the coverage as the SSR is trained on the transitions in the memory buffer. The SSR quality is not causally connected to the method because two different sampling methods that lead to the same coverage cannot be differentiated when the SSR is trained. This is because the transition does not include the curiosity reward signal, and the transition will always be the same regardless. Below the bubble are a set of measures of the quality of the SSR. KNN-MSE, and Hits at k measure the similarity of the SSR to the True state space. Linear reconstruction measures if there is enough information in the SSR to reconstruct the true state. Finally, generalization performance measures how robust the SSR is to slight changes unseen before in the observation, such as white noise or new distracting features.

Bubble four of the Figure 5.2 represents the performance of an RL algorithm trained on the SSR. The quality of the SSR influences the speed and quality of the policy that can be learned on the SSR as well as the rate at which the policy is learned. This can be seen when a higher episode reward is achieved and how quickly it is achieved. The purpose of the research question and experiments is to investigate nature of the connections between each of these variables.

6 Experimental design

The analysis framework shows the connection between four key variables to analyze how sampling method affects the quality of the SRL algorithm. The experiment is designed to find situations where there is a strong connection between the variables.

The experiment is made up of a default test environment, a training cycle, an SRL algorithm, and various measures of structural quality. The SRL algorithm and environment are chosen to maximize the chance that a change in the sampling distribution due to curiosity rewards would lead to a measurable improvement in the SSR quality. The four sampling methods are random exploration, two curiosity rewards, and uniform sampling. These offer a control, two curiosity rewards that represent different possible behaviors, and a possible ideal sampling policy.

6.1 Explanation of experiment enviorment

Random action policy samples the starting location the most and the sampling rate decays as the more actions it takes to reach a state. Therefore the environment must have enough separation between different states for random action to have significantly different sampling rates between different parts of the environment. The other constraint is the environment must be simple enough to be solvable by RL. RL performance is a critical measure of the quality of a SSR and if a RL never gains any reward then the quality of the SSR will be impossible to differentiate.

The environment in Figure 6.1 was made to be simple enough to be solvable but it will have a significant difference in the sampling rates from one room to another when explored by a random action policy. The size of the room makes it so there is a limited number of states making it easier to solve. The precise size is based on the gridworld used in (Seo et al., 2021). The wall in the center was added so that random actions do not reach the second room as often as the first. This leads to a difference in sampling rate between the left and right rooms when sampled using random action. The difference in sampling rate between the left and right room may lead to worse quality of the SSR with random action that may be overcome when using curiosity rewards.

We compare synthetic state representations of the MiniGrid environment (Chevalier-Boisvert et al., 2018). Specifically, a grid with two rooms and a narrow passage between them. The agent, represented by a purple square, must navigate from a starting room through a door to another room to reach a goal. The goal is represented as a green square. The agent can choose from four actions move one position up, down, left or right. The agent is given a reward of of 1 when it reaches the goal. The entire grid is observable by the agent and a observation appears as fig 6.1. The enviorment has an wall was chosen to enforce a a large difference in the sampling rate between the initial condition and the goal using a random action sampling policy.

The SSR is trained in cycles with three phases shown in Figure 6.2. The first phase uses a random action policy to collect enough transitions for the RL to train on a batch. The second phase runs for a set number of episodes. An episode begins with the environment resetting to the initial state and runs till the goal is achieved or a maximum number of environment steps is reached. In the second phase, transitions are sampled according to the sampling policy being tested. The tested sampling policies are uniform sampling from state action space, random action, or a curiosity reward maximization. The transitions are added to the RL and SRL memory. In the third phase of the cycle, the SSR is trained using batch gradient descent for a set number of iterations. When the cycle ends the RL memory buffer is cleared as the synthetic states in the buffer are no longer meaningful after the SSR NN was updated. The state representation learning memory buffer containing observations is never cleared.

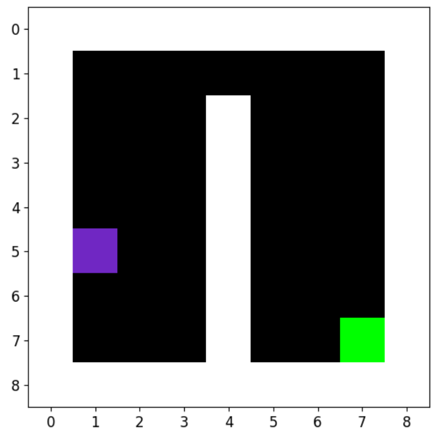


Figure 6.1: Navigation task on 9x9 grid, agent is purple square, the goal green square, The walls are white squares

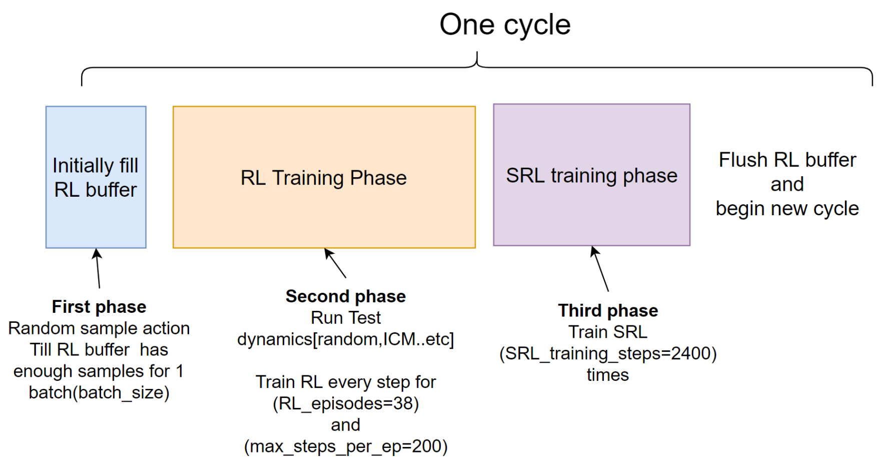

One cycle
Initially fill RL buffer
RL Training Phase
SRL training phase
Flush RL buffer and begin new cycle
First phase
Random sample action Till RL buffer has enough samples for 1 batch(batch_size)
Second phase
Run Test dynamics[random,ICM..etc]
Train RL every step for (RL_episodes=38) and (max_steps_per_ep=200)
Third phase
Train SRL (SRL_training_steps=2400) times

Figure 6.2: Three phases of a training cycle, random action to fill RL buffer,

The SSR neural network is formed by two convolution layers and three fully connected layers. The network minimizes the sum of a forward, reward, and contrastive loss. The convolutional layers both have kernels size 3x3 followed by Relu activation functions. The first convolution layer has 32 filters, and the second has 64. The three fully connected layers output vectors sizes 512, 256, and 10, and each layer uses Relu activation functions.

6.2 State Representation Learning via Homomorphism Metric

The experiment will use a combination of a forward, rewards, and contrastive loss as the SRL. It is a state of the art algorithm and will maximize the effect of bad sampling on the SSR by miscategorizing states in the contrastive loss. This is explained in section 4.1.4.

The SSR is trained from the history of transitions (o,a,o',r) that are added to a memory buffer[1]. The SSR is learned using three neural networks shown in Figure 6.3. The total loss of the network is the sum of the three losses: the forward loss, the reward loss, and the contrastive loss. When these three losses reach zero, we obtain an mdp homomorphism. Then, the optimal pol-

[1]The ideal memory buffer of an agent is infinitely large and stores the entire history of observation action and rewards. However, the computational size make this infeasible. Thus memory buffers are added as first in first out memory of specific length in practice.

icy on the SSR is also the optimal policy on the true MDP of the environment. This is described in section 3.6.1. The losses are shown in Equations 6.1, 6.2, 6.3, where $d(,) = ||x - \hat{x}||_2$ is the L2 norm distance, $d_\neg(,) = max(0, \kappa - d(,))$, and κ is a constant. The gradient of the losses is propagated back through the neural networks. This means that the loss gradient from all three loss functions propagates back through the SSR network called encoder. The forward loss gradient only propagates backward through the forward network. This is the same for the reward loss gradient and reward network. These networks are shown in figure 6.3.

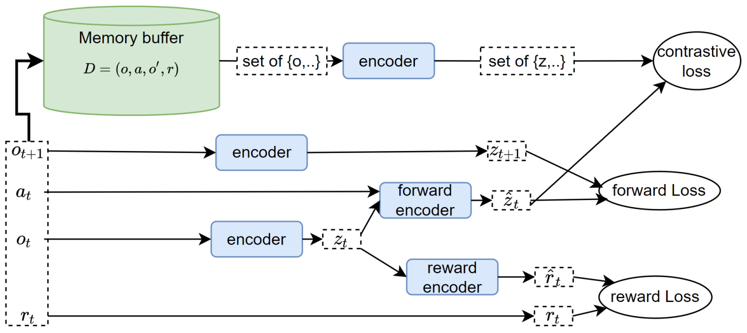


Figure 6.3: The neural networks signal flow diagram for the state representation learning algorithm used for this research. The encoder networks are all the same neural network that is used to generate the synthetic state space

$$L_{reward} = d(\hat{r}, r) \tag{6.1}$$

Equation 6.1 is the reward loss which encourages the synthetic state to map observation so that observations with different rewards are easily differentiated.

$$L_{forward} = d(\hat{z}_{t+1}, z_{t+1}) \tag{6.2}$$

Equation 6.2 is the forward loss which encourages the synthetic state to map observation so that the following state is predictable.

$$L_{contrastive} = d_\neg(z_\neg, z_{t+1}) \tag{6.3}$$

Equation 6.3 is the contrastive loss which encourages the synthetic state to not map observation from different underlying states to a similar area in synthetic state space. Together minimizing these three losses creates a homomorphic state space which allows for an optimal policy to be learned. This is reviewed in more detail in section 3.6.1.

The actions used to gather the transitions for the SRL are chosen by some policy $\pi(a|z)$. The standard policy used in literature is to collect transitions is with a random action policy (Jonschkowski and Brock, 2015). The sampling policy can also be implemented as an RL algorithm to maximize some curiosity reward. The curiosity reward can be designed to encourage different sampling behaviors. For example, uniform sampling of the state action space. The policy influences the pattern of transitions (o,a,o',r) that are generated. Then, the pattern of transitions influences the SSR. If the sampling from the memory batch is uniform then repetitions of (o,a,o',r) adds extra weight to transitions. For example, If one transition was sampled 5 times more than another the effect of the first transition on the synthetic state loss would be 5 times as prominent. The effect is that the SSR neural network perceives the first transition as 5 times more important and would minimize the loss of the first transition at the expense of the second.

6.3 How to evaluate the quality synthetic state representation

A good SSR should generalize the learned value function to unseen states (Böhmer et al., 2015). This can be measured in three ways. First, measuring the RL performance using the SSR as input. Second, the similarity of the synthetic state's structures to the TSR. Finally, the ability to transfer the synthetic state to new environments without a performance in the previous two measures. This can be tested by looking at the reinforcement learning performance of a DQN. The better the synthetic state can generalize the value function the easier it should be easier for the RL to learn the optimal policy. If the Q value generalizes, then the RL does not have to learn the Q value of new states from scratch. Instead, it can just use what it has previously learned leading to a faster learning rate.

Another method to test the generalization is to see if the synthetic state has a similar structure to the true state of the environment. In TSR, the distance two states usually indicates the number actions separating them. State separated by one action have value function within a few percent as the value function decays across actions by γ due to equation 3.3. This means states that are close together in TSR, have similar Value functions. The generalization is usually easier in TSR as similar states usually lead to similar value function this facilitates generalization. There is an exception to this rule when thin walls cut a space shown in Figure 6.4. The similarity of the structure of the synthetic state and the true state measures generalization if the assumption holds that the true state generalizes the value function to new states. The quality of a SSR can be measured in two ways. The RL performance and structural measures, which measure the similarity of the SSR to the TSR.

A good SSR should also be robust to noise and distractions. A distraction is a feature in the image that is not important for the underlying markov decision process. The robustness can be measured by transferring the SSR to a similar enviorment with distractor features, or white noise. If the SSR keeps its quality then it is a good SSR.

Reinforcement learning performance is the primary method to measure the quality of an SSR. The purpose of a SSR is to improve RL performance. If a SSR cannot improve RL, then it has failed. RL performance of a SSR is measured as the amount of reward accumulated over an episode by an RL algorithm trained on the SSR. The higher the episode reward the better the RL performance. The classic SRL test is training an SRL using a random exploration policy, then training an RL on the SSR (Lesort et al., 2018). The RL performance of the synthetic state is compared to the baseline of the RL performance of an RL trained on the raw observations. The improvement in the accumulated reward of the RL trained on the SRL over the observations is the improvement of a SSR ove observations. RL performance is a critical test to compare synthetic state representations, but it is not a perfect measure.

The problems with RL performance are it is costly to test, it is a stochastic measure, and there are a large variety of RL algorithms. An RL algorithm, especially on difficult tasks, may require tens of thousands of NN updates (Pathak et al., 2017). This is feasible for simulated tests, but the cost to test on a real robot is high. RL algorithms uses stochastic sampling from the memory bank to learn, and stochastic sampling of the state action space. This randomness can lead to the RL performance being a low accuracy measure and successive measurements of RL performance can significantly vary (Morik et al., 2019). Finally, there are different versions of RL algorithms(DDQN, TD3, PPO). Some may perform better than others on different tasks (Schulman et al., 2017). RL performance is a critical metric, despite the disadvantages. RL performance should be validated with multiple runs or other metrics such as structural measures due to the uncertainty in RL performance results.

If the TSR is a good input to train an RL algorithm, then the SSR can be evaluated by how similar its structure is to the TSR. The structure of two spaces is similar if the mapping between TSR and SSR is locally linear, and points close together in TSR stay close when mapped to SSR

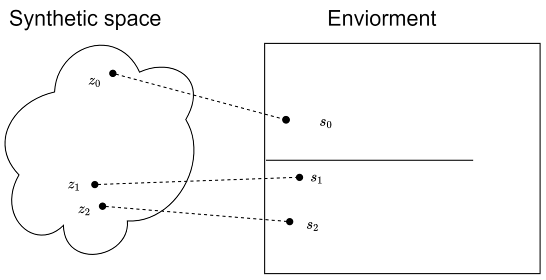


Figure 6.4: z_0 is placed far away in synthetic space because the wall means that many actions will need to be taken to reach it from z_1. So while s_0 and s_1 may be close in Cartesian coordinates they are far away from the perspective of the agent.

(Zhang et al., 2012; Lesort et al., 2019). The structural similarity is measured by checking if corresponding points in true and synthetic space are similarly grouped (Lesort et al., 2017). For example, if three points in TSR are close together, then the three corresponding points in SSR should also be close. Two measures of the structure are KNN-MSE, and Hits@-K, these both measure the similarity of the K nearest neighbors of a pair of corresponding points in true and synthetic representations. The importance of structural similarity is not always necessary or desired. For example, (Ghosh et al., 2019) argues the more actions it takes to move from one state to another, the farther away they should be placed in latent space. For example, two points separated by a wall should be mapped far apart in SSR even though they are close in TSR. This can be visualized in Figure 6.4. Structural similarity between true state and SSR suggests a good SSR, however, is not a sufficient indicator of quality. RL performance and structural metrics can validate each other if correlated.

The K nearest neighbor mean squared error(KNN-MSE) metric measures the structural similarity between the true and synthetic representations. It measures the distance between the set of points in TSR that correspond to the K nearest neighbors of a point in SSR. The score of the SSR is the mean of the KNN-MSE of every point in the SSR. The KNN-MSE score for one synthetic state point(z_i) is calculated by finding the corresponding point in TSR s_i using mapping $\phi(z) = s$. Then, find the K nearest neighbors of z_i in SSR $KNN(z_i) = z_i^{KNN}$. Next, map the z_i^{KNN} back to TSR s_i^{KNN}. The score for one point is the mean of the distance from each $s_j \ ins_i^{KNN}$ to s_i.

$$KNN-MSE(z_i) == \frac{1}{k} \sum_{z_j \in KNN(z_i,k)} ||\phi(z_i) - \phi(z_j) \tag{6.4}$$

If one of the K nearest neighbors of z_i maps to a state far away from s_i, then the structures are not similar. This simple metric structure is shown to correlate with more complicated measures of structure such as NIEQA (Jonschkowski and Brock, 2015).

Hits at K is a measure of structure similar to KNN-MSE. It is the number of common nearest neighbors between two corresponding points in true and synthetic representations. Hits at k is calculated for each point in TSR. To calculate hits at k, first, take point s_i, then find its corresponding point in synthetic space z_i via the mapping $z_i = f(s_i)$. Then, find the K nearest neighbors for s_i and z_i, $KNN(s_i) = s_i^{KNN}$ and $KNN(z_i) = z_i^{KNN}$. Finally, map $s_j \in s_i^{KNN}$ to $z_j = f(s_j)$ and count how often z_j is one of the k nearest neighbors of z_i($z_j \in z_i^{KNN}$).

$$\{z'_{knn}\} = KNN(z_i) \mapsto \phi(\{z'_{knn}\})) = \{s'_{knn}\}). \tag{6.5}$$

Hits at k has the advantage, over KNN-MSE, of being independent of the structure of true state. For example, if states in an enviorment are far away from each other this leads to a higher KNN-MSE.

The final measure of structure is the reconstruction error of the true state from the synthetic state. The most simple version of this measure is the linear regression error between true and synthetic states. Linear reconstruction can be too restrictive, and (Jonschkowski et al., 2017; Anand et al., 2019) used simple fully connected neural networks to reconstruct the true state from the synthetic state. If the reconstruction function is complex, then low reconstruction error is guaranteed. This is because the SSR has more dimensions than the TSR to guarantee the synthetic state represent the ideal value function. The extra dimensions can allow the reconstruction function to overfit on noise, leading to low reconstruction error.

The ability of the SSR to generalize to new environments or tasks is important for the quality of the SSR and its utility. Generalization is the ability of the SSR to transfer its quality to new environments or tasks. For example, a SSR trained on a pendulum should work on a pendulum of different lengths. A generalizable SSR is important so the SSR does not have to be relearned for every slight change in the task or environment. The ability to generalize also demonstrates that the SSR is not overfitting on unimportant details in the observation. A SSR is generalizable if the evaluation metrics of SSR stay consistent when used on slightly different tasks and environments (Jonschkowski and Brock, 2015).

6.4 Comparing current state of art representation learning to curiosity based methods

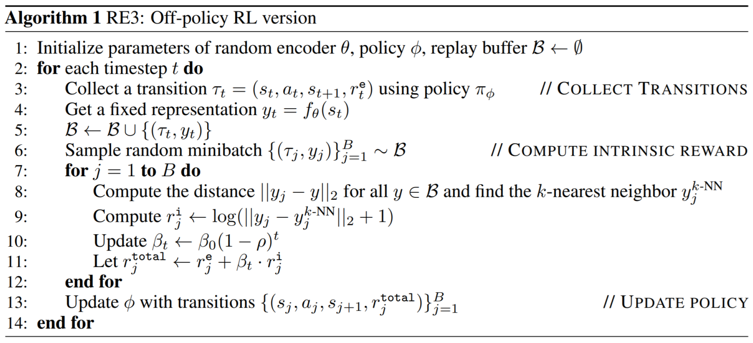
The current state of the art sampling method for training SSR is random exploration. Random exploration is where each action is selected with the same probability as any other action at any state. The performance of a SSR trained using random action exploration will serve as a control group to evaluate the effect of curiosity rewards. The SRL is trained on the history of observations and actions recorded during the random exploration. The evaluation metrics of an SRL trained using random exploration will be compared to a SSR trained using two curiosity rewards.

I tested two different curiosity rewards. The curiosity rewards are transition prediction error and latent space entropy maximization. The prediction error is an intrinsic motivation that rewards the agent for going to states where it does not know the dynamics. Entropy maximization rewards the agent for going to states that are far from states it has been before. The entropy reward is maximized when states are sampled uniformly. If there is an improvement in the quality of the SSR when trained using the curiosity rewards instead of random exploration, then the curiosity rewards improve state representation learning. The choice of focusing on unknown dynamics vs. maximizing the uniformity of state sampling. Then, a curiosity reward that maximizes uniformity vs one that maximizes prediction error should be useful for testing whether a more uniform distribution is always better, research question 2.

Predictive error is an intrinsic reward that is greater when the agent cannot predict the outcome of a state transition. The more an agent visits one state, the more accurately it will learn the transitions from that state. The state will then give less reward causing the agent to get bored. The lack of reward encourages the agent to stop visiting states it has already learned and explore new ones. The equation for predictive error reward equation 6.6, where f() is a neural network trained on the loss of the squared predictive error.

$$r_i = ||s_{t+1} - f(s_t, a_t)||_2 \tag{6.6}$$

Latent entropy maximization rewards the agent for action maximizing the entropy of visited latent space. This drives he robot to visit latent space uniformly. The true entropy of a distribution is given by the expected log of the probability distribution, $H(z) = \mathbf{E}[\log p(z)]$. $1/N\sum_i^N \log||x_i - x_i^{k-NN}||_x$ (Singh et al., 2003) easy to compute value that is approximately proportional to the true value of the entropy.



Algorithm 1 RE3: Off-policy RL version

```
1:  Initialize parameters of random encoder $\theta$, policy $\phi$, replay buffer $\mathcal{B} \leftarrow \emptyset$
2:  for each timestep $t$ do
3:      Collect a transition $\tau_t = (s_t, a_t, s_{t+1}, r_t^{\mathrm{e}})$ using policy $\pi_\phi$        // COLLECT TRANSITIONS
4:      Get a fixed representation $y_t = f_\theta(s_t)$
5:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\tau_t, y_t)\}$
6:      Sample random minibatch $\{(\tau_j, y_j)\}_{j=1}^{B} \sim \mathcal{B}$        // COMPUTE INTRINSIC REWARD
7:      for $j = 1$ to $B$ do
8:          Compute the distance $||y_j - y||_2$ for all $y \in \mathcal{B}$ and find the $k$-nearest neighbor $y_j^{k\text{-NN}}$
9:          Compute $r_j^{\mathrm{i}} \leftarrow \log(||y_j - y_j^{k\text{-NN}}||_2 + 1)$
10:         Update $\beta_t \leftarrow \beta_0(1-\rho)^t$
11:         Let $r_j^{\mathtt{total}} \leftarrow r_j^{\mathrm{e}} + \beta_t \cdot r_j^{\mathrm{i}}$
12:     end for
13:     Update $\phi$ with transitions $\{(s_j, a_j, s_{j+1}, r_j^{\mathtt{total}})\}_{j=1}^{B}$        // UPDATE POLICY
14: end for
```

Figure 6.5: latent entropy maximization algorithm (Seo et al., 2021)

6.5 Complex environment

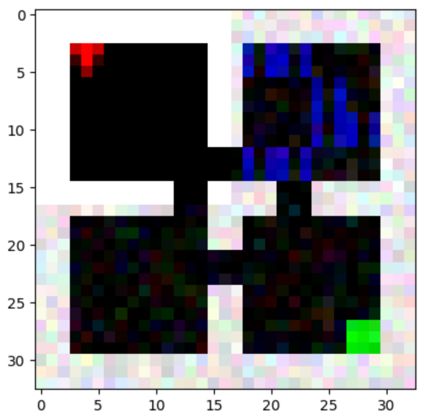
The simple default environment is shown in Figure 6.1. In the simple default environment, different sampling distributions achieved similar structural scores. This is shown in Section 7.4. The result could be attributed to the fact that the features of the environment are consistent no matter the location of the agent in the environment. The consistent features allow the SSR to generalize well even without a uniform distribution. The generalization could allow a SSR trained with repetitive samples to generalize to the whole environment. We created a complex environment with features that prevent generalization. The complex environment should prevent the generalization of samples from one part of the environment to another. This should increase the strength of the relation between uniform sampling and the quality of the structure of the SSR.

The complex environment is a four-room environment with added distractors features. The distractor features are designed so features of the observation learned in one room may be detrimental in another. The environment is an 11 by 11 grid split into four rooms. The upper left room is just like the simple environment. The lower-left room has added stationary white noise. The upper right room has added white noise and blue distractors features that do not move or interact with the agent. Finally, the lower right room has purple distractors features are observerable when the agent is in the lower right room. The starting observation of the environment and the purple distractors can be seen in Figure 6.6a and Figure 6.6b respectively.

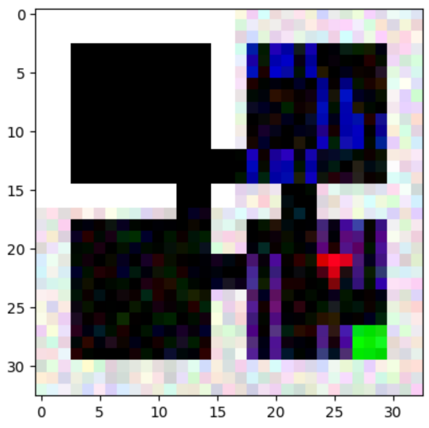
The color purple was chosen because it is a mix of blue and red colors. The signal of purple overlaps with the agent's red color and blue distractors of the upper right room. The overlap with the blue should give the SRL a hint that the purple distractors are not important. The red will force the SRL to differentiate from the purple floor and the red agent. This prevents generalizability because the SRL must learn that the location of a red signal is the agent, but not always. The distractors only appearing when the agent enters the room ensures there will be no generalization from the top left to bottom right rooms.

The complex environment is trained differently from the simple environment. First, random action is used to gather enough samples for one batch(64). Then, the tests sampling method is used to gather samples for one episode with a maximum step count of 50. Then, the SSR is trained for 16 update steps. The one episode of RL and 16 SRL updates are repeated 250 times. For a total of 4000 SRL update steps. Finally, the memory buffer of the RL is never cleared and is instead is updated with the new SSR before the start of each episode.

The reduced max step count per episode purpose is reduce the chance that a random action policy reaches the bottom right room. Then, curiosity rewards that drive the agent towards the last room should see higher structural scores. The reduced number of SSR updates reduces the change between different version of the SSR. This should improve the transference of a curiosity driven policy to a new versions of the SSR while training. The transference should improve the sampling distribution.



(a) Initial observation of the environment. Agent in Red



(b) Observation when agent enters bottom right room. Agent in Red

7 Results

The results demonstrate the links between the sampling method, coverage, quality of the SSR, and RL performance variables in Figure 5.2. The links are shown for the simple and complex environment. The relationship between the sampling method and RL performance is shown by comparing the episode reward achieved by each method. The relationship between the method and the coverage is shown with an example state visitation count and the KL divergence between the state action sampling coverage and a uniform distribution. The relationship between the coverage and synthetic state structure is plotted as a KL divergence and the mean structure scores of each test run. Finally, the relationship between the structure scores and RL performance measured as the mean total reward achieved over training is plotted.

7.1 Simple Environemnt

7.2 Effect Method on RL performance

Figure 7.1 shows reinforcement learning performance achieved by the four sampling methods and randomly initialized networks; random sampling(Random), latent space entropy maximization(Hmax), prediction error maximization(ICM), Uniform sampling state action(Uniform), random network initialization(RandInit). The reinforcement learning performance is measured as the per episode reward, $r_{ep} = 1 - \frac{steps}{200}$, where 200 is the maximum steps possible during an episode. The environment reward is 1 or 0 if the agent reaches the goal. The episode reward plot shows the step modulated reward to increase the resolution of the RL performance metric. For example, a policy that reaches the goal in fewer steps is better and should get a better score. Figure 7.1 plots the modulated RL performance of each sampling method across training epochs averaged across 4 tests with a moving average of 10. Figure 7.1 shows that the RL performance of all methods is similar at episode 100. The most prominent difference between the methods is that the randomly initialized network takes much longer to reach that peak performance than the trained networks. The raw data for each run can is shown in Appendix A.8.

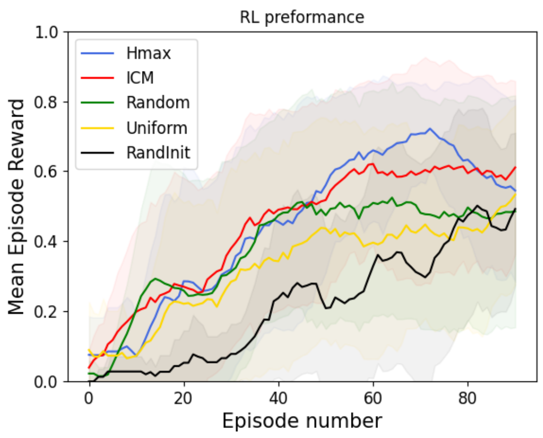
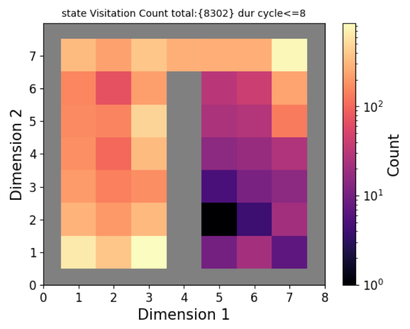


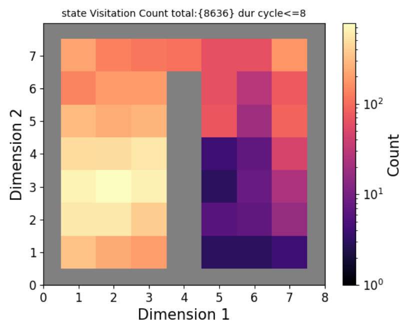
Figure 7.1: The mean RL performance across 4 runs and standard deviation. The mean episode reward is also smoothed using a moving average of 10. The shaded area is the standard deviation of the RL performance across 4 runs smoothed with a moving average of 10. It represents the uncertainty of the mean RL performance.

7.3 The Effect of Method on Coverage

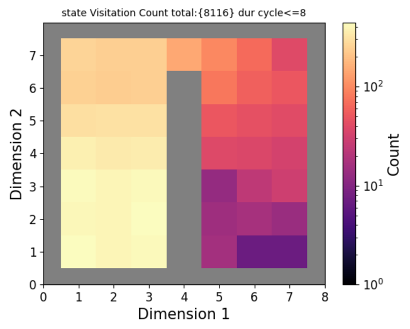
The maps in Figure 7.2 show the logarithmic count of the number of times each state in the environment was visited during the training of the SSR. The data is from a single test run from each sampling method. The purpose is to show each sampling method leads significant differences in coverage distributions. The log count is the log(state visitation count +1). If a state is never visited, then its log visitation count is 10^0. The Hmax sampling method had two states at the bottom center(3,1) and top right(7,7) that each have about 800 visits. The state (5,2) was not sampled once. Random action sampling has a more even sampling distribution with each of the states around the initial position (1,3) getting around 400 visits, and every state is visited at least once. Finally, ICM has a peak of 700 at the initial state(2,3).



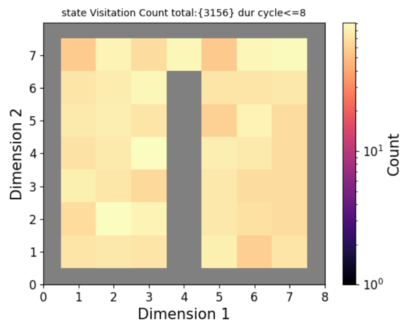
(a) Entropy Max(Hmax)



(b) ICM



(c) Random



(d) Uniform sampling

Figure 7.2: Map of state visitation count for each method from 1 test. The count variable is the number of times the state was visited during training.

Figure 7.3 shows KL divergence of coverage distribution of each method from a uniform distribution. This metric represents how different the coverage distribution is from a uniform distribution during training of the SRL. Each point is the KL divergence for a single test run of SRL training. The KL divergence metric shows a simplified picture of how the different methods lead to different coverage distributions. The curious sampling methods Hmax and ICM are less uniform than the random or uniform sampling. ICM also has a wider distribution of KL-divergences than Hmax. This also shows that the the coverage distribution is significantly different between each of the sampling methods in this test. The uniform sampling is not exactly zero because of variance in the sampling. This variance can be seen in Figure 7.2.

7.4 Effect Coverage on Structural Performance

Figure 7.4, shows the relationship between the KL divergence of the coverage from a uniform distribution and the structural scores of the SSR. The more uniform coverage, the more consistently the state representation learning reached a good KNN-MSE score. The uniform sampling has a consistently low KNN-MSE score between 1 and 1.05. Sampling methods with high kl di-

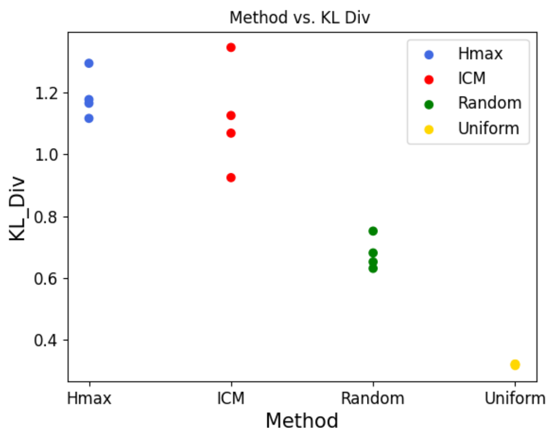
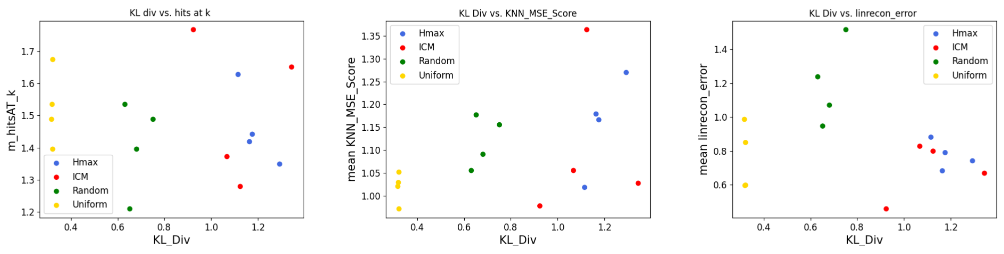


Figure 7.3: The KL divergence of the state action sampling distribution achieved by each of the methods from a uniform distribution.

vergence achieved these low KNN-MSE score but also got scores as high as 1.35. This relation between coverage uniformity does not extend to the other structural quality metrics. Random exploration leads to the worst hits at k and linear reconstruction performance, and it has a KL divergence between uniform and the curiosity rewards ICM and Hmax.



(a) **(b)** **(c)**

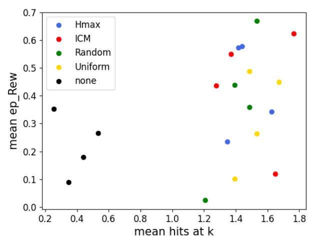
Figure 7.4: The relationship between the KL divergence of the coverage vs a uniform distribution over states and action

7.5 Effect Structural Score on RL Performance

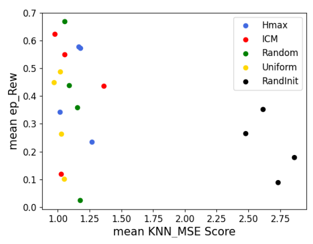
Figures 7.5a, 7.5b, and 7.5c show the relationship between the structural score achieved by each test and RL performance. The RL performance is measured as the mean episode reward averaged over all training episodes of the RL. The per-episode reward usually grows over time then flattens out as a locally optimal policy is achieved. The mean episode reward over all episodes is affected by the rate at which the optimal policy is achieved and that policy's peak performance. All methods show higher structural scores than random initialization regardless of the sampling method. The higher structural scores increase the ceiling of the total mean episode reward. The lower total mean episode reward reduction appears to be driven by a reduced learning rate when the RL is trained on the randomly initialized network, shown in Figure 7.1.

7.6 Effect Method on Generalization Performance

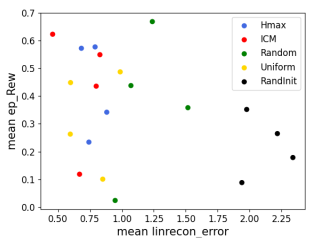
Figure 7.6 shows the robustness of the SRL,RL to added noise.The original environment observation was an RGB image. The noise was added by selecting a random integer between positive or negative 30 that was added to each color channel of each pixel. If the sum of the channel is above 255 or below 0 then the sum is clipped to 255 or 0. The noise is stationary across all ob-



(**a**) Mean Hits at K:4 vs RL performance for each individual test.



(**b**) Mean KNN-MSE score vs RL performance for each individual test.



(**c**) Mean linear reconstruction error vs RL performance for each individual test.

Figure 7.5: The RL performance is measured as the total mean reward over all training episodes reward achieved during the test.

servations. This means that each channel always has the same distortion. The channel will not have a different added noise at different time steps. The initial image of the environment can be seen in Figure A.5. The SRL encoder is kept stationary and the observations with added noise are used as input. The RL is initialized as the last policy after training from the RL performance test on the base environment. The change in the mean total reward stays is between 0.22 and -0.1. This is a significant portion of max reward of 0.9. The RL performance sometimes improves on the new environment. However considering the range in total mean reward of trained policies in Figure 7.5a are from 0.7 to 0 it is unlikely that these are significant differences. Figure

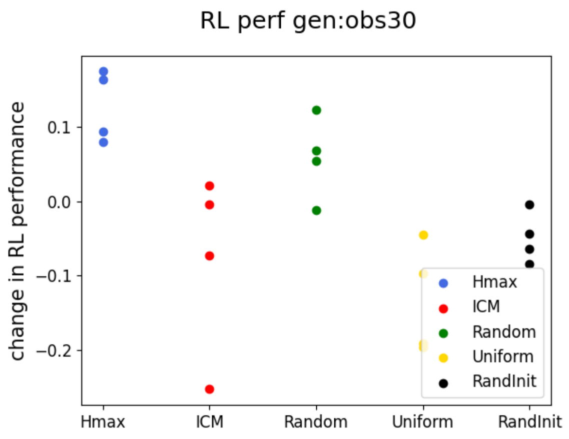
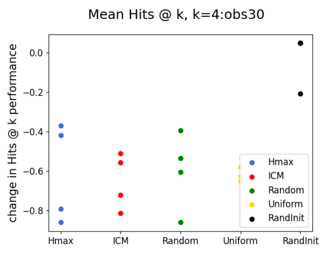


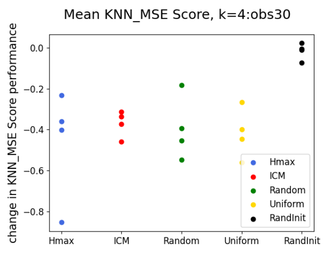
Figure 7.6: The difference between the mean total episode reward on environment and the mean total episode reward after transfer to environment with added stationary white noise.

7.7 shows that the structural scores are reduced when noise is added to the environment.Each point in Figure represents the change in the quality of the score with a positive value being an improvement in the score. The change in the structural score of the randomly initialized network is around zero. This is expected as the randomly initialized network are not biased to any environment. There is reduction in structural scores for all the sampling methods, however this did not lead to significant changes in the RL performance transfer.

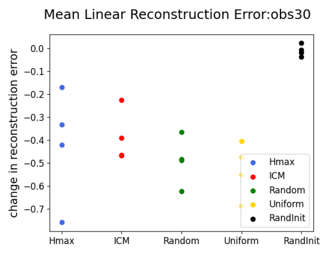
Figure 7.8 shows the performance of the trained RL policy after transfer to an environment with a different MDP structure. In this case the goal is on the left side of the environment and the agent starts on the right. This is the opposite of the original environment. Figure shows that all performance is lost by all methods. This means that the SSR or the RL policy did not learn a SSR or a policy that could not adapt to changes in the goal position. Since the environment stayed the same during training it is not unsurprising that the SSR and policy did not pick up this.



(a) Hits at K



(b) KNN-MSE



(c) Linear Reconstruction

Figure 7.7: The reduction in the structural score after transfer to environment with added noise

Figure 7.9 shows that the structural loss is significant, but the transferred structure still has superior performance than random initialization. This suggests that the drop in RL performance after transfer to the inverted environment may be primarily do to the RL not transferring.

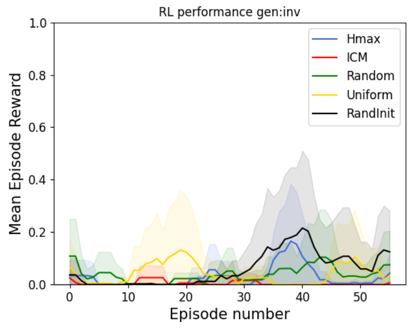
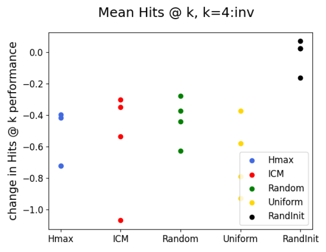
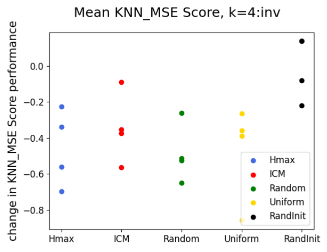


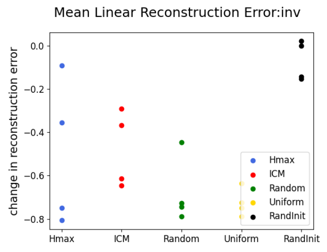
Figure 7.8: The mean episode reward after transfer to inverted environment A.6.



(a) Hits at K



(b) KNN-MSE



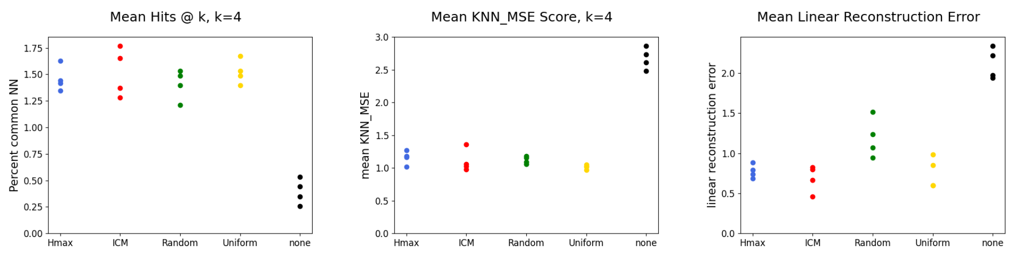
(c) Linear Reconstruction

Figure 7.9: The change in the structural score performance after transfer to inverted environment shown in figure A.6

7.7 Effect of Sampling order on Structural performance

Figures 7.10a, 7.10b, and 7.10c show how the method affects the synthetic state structural score. They compare the structural scores achieved by different sampling methods to the scores achieved by a randomly initialized neural network with no training. The data shows that the structural scores achieved by the sampling methods are much better than a randomly initialized network. The effect of the sampling method on the structural performance is limited compared to the presence of training. The distributions of the structural scores overlap significantly for mean hits at k and KNN MSE score, suggesting that there may be no statistically significant relationship between the sampling method and the structural scores.

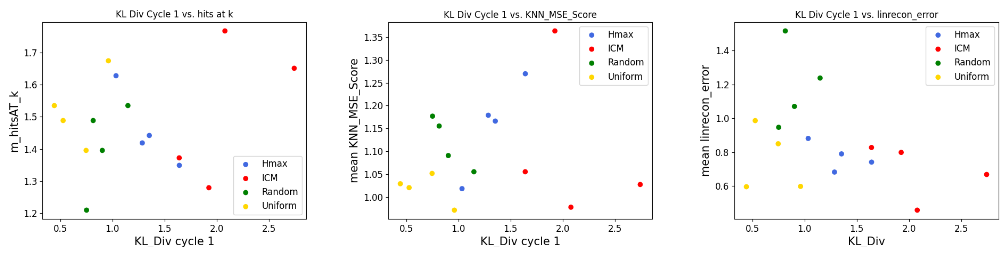
Figure 7.11 explores the relation between the KL divergence from the uniform of the coverage of the first SRL training cycle vs. the structural score achieved at the end of training. If the



(a) **(b)** **(c)**

Figure 7.10: The Hits at k score , KNN-MSE, and linear reconstruction for each of the 4 test runs each test achieved from each of the 4 sampling methods method. The Hits at k is the average number of common nearest neighbors between the synthetic and true state space. A perfect score would be 4, the worse score is 0. The lower THE KNN-MSE the better, a perfect score for the Grid world environment is close to 1. The lower the linear reconstruction error the higher the quality of the SSR. A perfect score of 0 would mean that the synthetic state space is a linear transform of the true state space.

coverage at the beginning and end of training affect the structural score differently, then the temporal behavior of the coverage may be important relative to the final coverage. Figure 7.11 shows that the coverage of the first cycle has similar relation to structure as the coverage after eight cycles. The KNN-MSE score has a lower ceiling as the kl divergence from uniform decreases, and random sampling leads to the highest linear reconstruction error. There is one major difference in the coverage between the first and last cycle. After one cycle the Hmax reward has a lower mean KL divergence from uniform than ICM. After 8 cycles, Hmax has a similar mean KL divergence from uniform.



(a) Hits at K **(b)** KNN-MSE **(c)** Linear Reconstruction

Figure 7.11: The relation between First cycle kl divergence divergence from uniform and structural values

One issue with training a state representation is that the synthetic state space is not stationary. The changing state space makes it difficult to learn a policy while the SSR is training. Figure 7.12 shows the external reward achieved during training of ICM and Hmax the sampling methods that try and maximize the total reward of the environment. The external reward achieved does not improve after multiple cycles. This means that alternating between training the RL and SRL did not allow the DQN to compensate for the changing SSR. If the RL could compensate for the changing SSR, then the policy improvement would be similar to the RL performance on a stationary state-space shown in Figure 7.1.

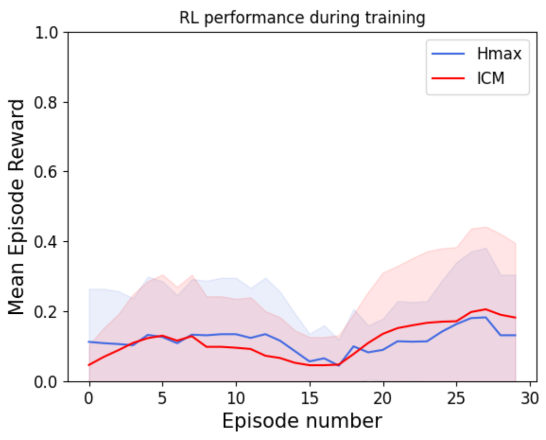


Figure 7.12: RL performance during training of the state representation network on default environment

7.8 Complex Environment

The Figure 7.13 shows the mean reward achieved over episode training a RL policy on each of the SSR trained with different sampling methods. Only the randomly initialized synthetic state representation network were able to learn policies. This suggests that the training of the SSR regardless of sampling distribution was detrimental to the RL performance of the synthetic state representations.

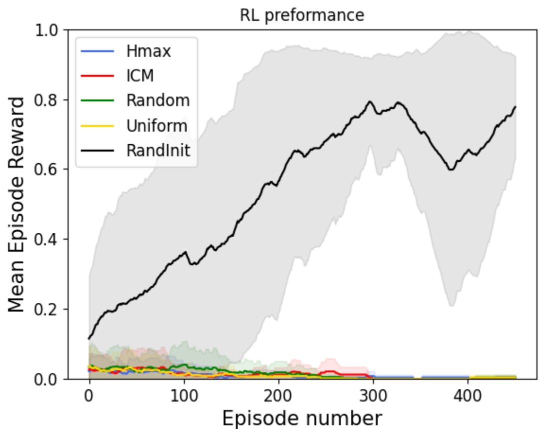


Figure 7.13: For the Complex environment, the mean RL performance across 4 runs and standard deviation. The mean episode reward is also smoothed using a moving average of 10. The shaded area is the standard deviation of the RL performance across 4 runs smoothed with a moving average of 10. It represents the uncertainty of the mean RL performance.

The Figure 7.14 shows the divergence from uniform coverage distribution achieved by various sampling methods. The Hmax, ICM, and Random arrived at very similar KL divergence values. The uniform sampling achieved an almost uniform coverage. This can be due to the fact that the uniform sampling has a little noise due to the sampling being random therefore it is not perfectly uniform.

In Figure 7.15 an example coverage heat map for each sampling method is shown.The data is taken from one test run of the four test runs for each sampling method. This reaffirms the data shown in Figure 7.14, the coverage in the Hmax, ICM, and Random tests are very similar which is reflected in the similar KL divergence from uniform.

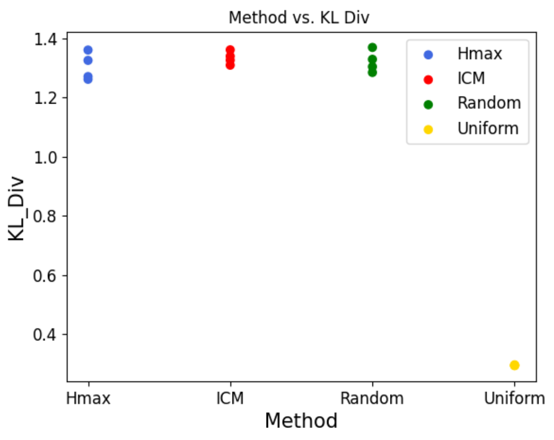
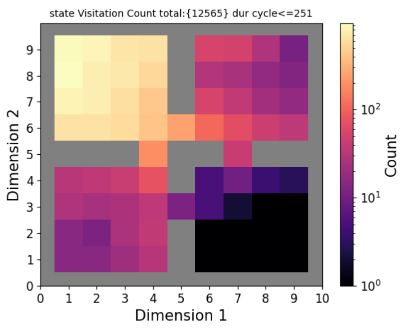
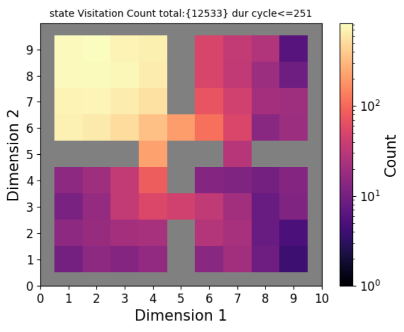


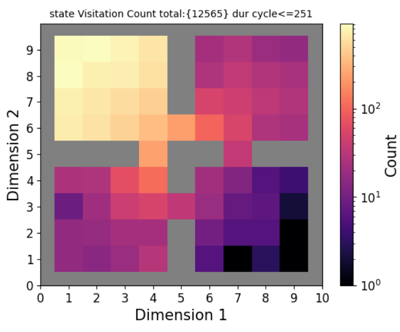
Figure 7.14: For the complex environment, the KL divergence of the state action sampling distribution achieved by each of the methods from a uniform distribution.



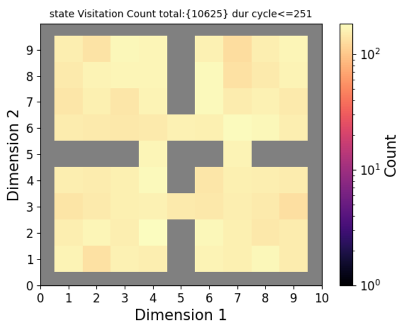
(a) Entropy Max(Hmax)



(b) ICM



(c) Random

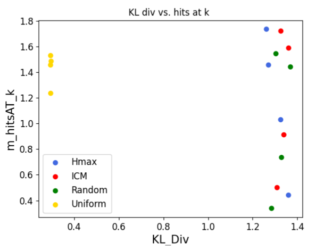


(d) Uniform sampling

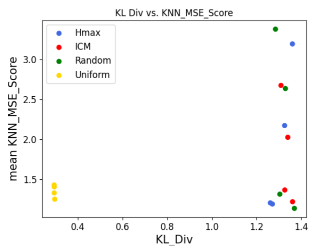
Figure 7.15: The complex environment Map of state visitation count for each method from 1 test. The count variable is the number of times the state was visited during training.

The Figure 7.16 shows the relation between the KL divergence from uniform of each test and the corresponding structural scores. Uniform sampling creates a cluster of good structural scores with low variance. The structural scores of the other sampling methods achieve a lower mean structural score with a larger variance than uniform sampling. The concentration of good scores for the uniform sampling across all three scores suggests that uniform sampling increases the likelihood of good structural scores.

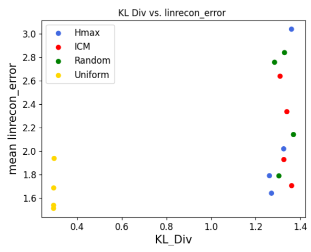
The Figure 7.17 shows the relationship between the structural scores and the RL performance. The random initialized networks have bad structural scores however score the highest RL performance. Some of the trained SSR have similar structural scores to the randomly initialized networks however still do not get any reward. This suggests that something about the untrained



(**a**) Hits at K



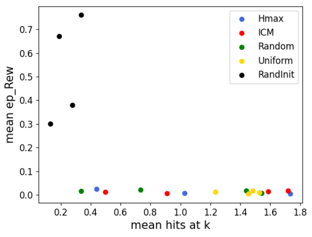
(**b**) KNN-MSE



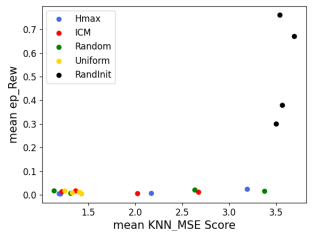
(**c**) Linear Reconstruction

Figure 7.16: In the complex environment, the relationship between the KL divergence from uniform of the coverage and the structural measures Hits at k, KNN-MSE, and linear Reconstruction

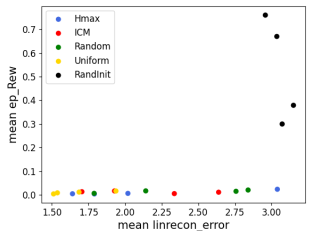
structure is allowing the policy to be learned, while the trained SSR are making it more difficult to train a policy.



(a)



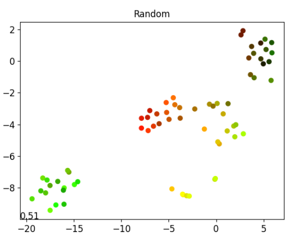
(b)



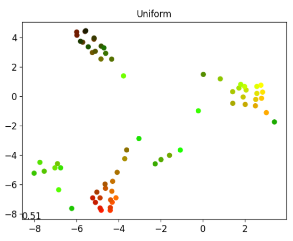
(c)

Figure 7.17: The complex environments relation between structural scores and rl performance

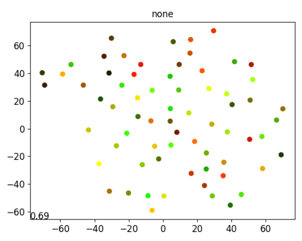
The Figure 7.18 shows some example TNSE mappings of the 10 dimensional synthetic state space for the complex environment into 2 dimensions. Figure 7.19 shows some example the TNSE mapping for the simple environment. The TNSE mapping allows for the visualization of high dimensional structures in two dimensions. The TNSE is a nonlinear mapping and the scale of the TNSE mapping does not have meaning. The TNSE of the synthetic state representations for complex environment, Figure 7.18, shows the states in each room tend to cluster when the SRL is trained. Randomly initialized networks do not have any particular structure. In the TNSE of the synthetic state representations of the simple environment, Figure 7.19, you can see that SRL maps the left and right rooms away from each other with adjacent states close together. This difference in how the environments are structured could be why the trained SRL gives good RL performance in the simple environment and not in the complicated environment.



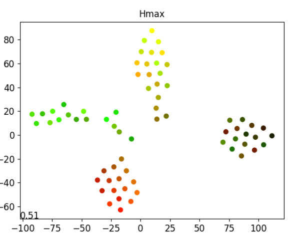
(a)



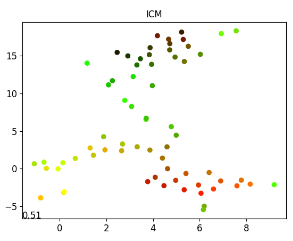
(b)



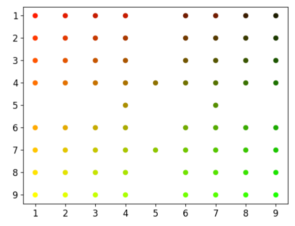
(c)



(d)

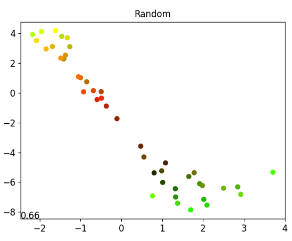


(e)

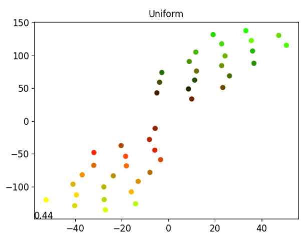


(f) Color Legend in true state space

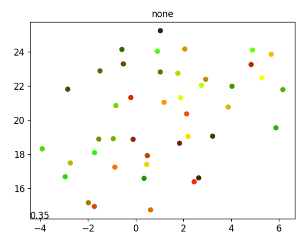
Figure 7.18: Example TNSE of complex environment for perplexity 15 for each sampling method



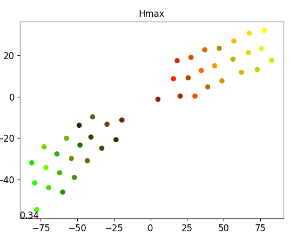
(a)



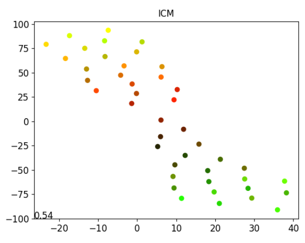
(b)



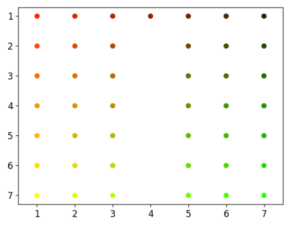
(c)



(d)



(e)



(f) Color Legend in true state space

Figure 7.19: Example TNSE of simple environment for perplexity 15 for each sampling method

8 Discussion

The results of the experiment are that different sampling method do not significantly improve the RL performance of the trained SSR. Training the SRL improves the learning rate of an RL when trained on the SSR for only a simple environment. For the simple environment the difference in coverage does not significantly affect the structural scores of the SSR. uniform coverage improves the likelihood of good strctural scores. Improved structural scores can improve RL performance however this may be heavily dependent on the environment. The results are not conclusive enough evidence to claim that curiosity rewards improve the SSR on a simple environment or complex environments.

8.1 Effect Method on RL performance

The results shown in Figure 7.1 show that the mean episode reward of all sampling policies(random action, uniform, ICM, and Hmax) are within a standard deviation of one another. The small change in mean compared with the standard deviation makes it difficult to conclude one sampling policy leads to a significantly higher RL performance than another. Therefore, we conclude the sampling policy does not affect the RL performance metric of the synthetic state representation. The relation between RL performance and the sampling method is determined by two intermediate variables coverage and synthetic state structure. These variables are linked by three causal connections, the effect of the sampling method on the coverage, the effect of coverage on the structure of the synthetic state representation, and the effect of structure on RL performance. The lack of relation between the beginning and end of the chain means that at least one of these causal connections is not strong. The lack of relationship between the method and the RL performance could be caused by the simplicity of the environment. The environments simplicity may allow for good generalization could allow performance metrics to be very similar even with sub-optimal sampling coverage.

The results in Figure 7.13 show the effect of the SRL sampling method on the RL performance on the complex environment. In the simple environment the trained synthetic state representations improved the RL performance even if the improvement was slight. In the complex environment the trained SSR significantly hurt the RL performance. This could be because the complicated environment needs different actions for each state in each room. As shown in Figure 7.18, the trained synthetic state representations for the complex environment cluster all the states in each room together. This could make it difficult to apply different actions to each state within the room. This property is mentioned as detrimental to a SSR in section 3.6. The simple environment can get a reasonably successful policy with only the left room action being up and the right room action being down. This might mean that if adjacent states do not require the same action state representation learning could be detrimental to policy learning.

8.2 The Effect of Method on Coverage

There is a significant relationship between method and coverage distribution as shown in Figure 7.3. The different methods achieve different KL distributions. The difference in coverage can also be seen in the example state visitation map in Figure 7.2. Each of the methods is significantly different distribution considering the log scale. Therefore, there is a relation between the method and the coverage.

The results in figure 7.14 show that different curiosity driven sampling methods can get the same coverage. This could be caused by the episode length being to short to allow an unique distribution.

8.3 Effect Coverage on Structural Performance

The relation between the coverage and the structural scores does not appear to be significant. There could be a relation between KL divergence and KNN-MSE where less uniform coverage leads to worse scores on average, see Figure 7.4b. However, the connection does not appear in the other two structural quality metrics. Therefore, it difficult to conclude that there is a significant relationship in this environment.

In the complex environment the uniformity of the coverage leads to a significant improvement in the the consistency of good structural scores is improved. This can be seen in Figure 7.16. This shows that uniformity of sampling is important to structural performance when the environment has features that do not generalize from one part of the environment to another.

The KL divergence metric is the KL divergence between the state action sampling distribution of the method and uniform distribution. KL divergence metric is not a perfect measure to use as the coverage variable as it has a narrow meaning. The first issue is significantly different distributions can lead to the same KL divergence. For example, if the two distributions are symmetric across the state action space. The KL divergence also does not capture the change in the coverage over training. Since the KL divergence metric was calculated after the training had been completed. The metric ignores the temporal component of the sampling distribution, which could be important considering past research on curriculum learning (Bengio et al., 2009). The KL divergence is only a measure of how uniform the coverage is, and some other characteristics of the coverage may be important to learn a good SSR.

8.4 Effect Structural Score on RL Performance

Finally, there appears to be a relationship between structural scores and RL performance. The Figures 7.5a, 7.5b, and 7.5c show two clusters are formed by the trained SRLs and the randomly initialized synthetic state representation. The trained SRLs have a higher performance ceiling than those trained on a randomly initialized SRL. The lack of relationship between the sampling method and RL performance appears to be caused by the fact that all sampling methods achieve similar structural performance despite differences in coverage.

In the simple environment the relation between structure and RL performance appears only because of the low score and lower RL performance of the randomly initialized SRL. This suggests that the structure performance affects the RL performance. However, the relationship between different sampling policies and structure performance for this experimental setup is very low. This could be because the test environment has low complexity. The change in the observation is consistent for each action across the entire state space. This allows the state transitions in the initial room may generalize well to the other room. Good generalization means sampling bias toward one location is not as impactful.

This relation does not appear in the complex environment. In the complex environment the improved structural scores do not translate to improved RL performance. This is shown in Figure 7.17, where the randomly initialized SSR have low structural score and high RL performance. The trained synthetic state representations also can have similar structural scores to the randomly initialized representations. These trained representation also have low RL performance this suggests that the structural scores are not closely related to RL performance. Instead a different unknown component of structure may be related to RL performance. The trained SSR regardless of the structural scores show some sort of state clustering where adjacent states are placed close together, when adjacent states require different actions this clustering is detrimental to policy learning. The difference in the relationship between structural scores and RL performance between complex and simple environment suggests that structural scores are not a good indicator of RL performance for all types of environments. This conclusion is supported by other works that have found randomly initialized networks to be superior to trained networks on some environments (Burda et al., 2018a; Raffin et al., 2019).

8.5 Effect Method on Generalization Performance

The sampling method also does not seem to significantly improve generalization performance. All SRLs trained using the four sampling methods experiencing similar loss of structural performance when transferred onto a new enironment. The structural scores are after transfer are still superior to a randomly initialized network showing that the structural performance of the SRLs is at least somewhat robust to noise as shown in Figure A.7.

The RL performance sometimes improved when the policy was tranfered to the noisy enironment which should not occur. This could be caused by the instability of the DQN method, with it producing a very wide range of RL performances as seen in Figure A.8. The DQN performance can shift radically from episode to episode.

The transfer to the inverted enironment caused similar loss in structural score however the performance of the RL policy did not transfer at all. This was expected as the policy was trained on the same enironment it did not have the opportunity to learn that it needed to reach the goal, and instead learned that it needed to go up in the left room right at the top and down in the right room to get the reward.

8.6 Effect of Sampling order on Structural performance

We found that the temporal characteristic of the coverage seems to not affect the structural quality. An effect may appear in a more complex environment, such as an environment with tiered difficulty. An example, being the paper (Oudeyer et al., 2007) in which there are three tasks of varying difficulty and analysis. The easiest task is sampled first and the hardest last to improve learning. The grid environment does not have dynamics of varying difficulty, therefore this test cannot determine if a sampling easiest to hardest improves state representation learning.

This conclusion is supported by the lack of relationship between each method and structural scores shown in Figure 7.10. Assume different methods may lead to different coverage sampling orders and different orders led to changes in structural score. Then, there would be a relationship between method and structural score. The lack of relationship suggests that there is no effect of the order on the.

The evidence of no connection between the sampling order and the reward, relies on the assumption that the test environment will generalize to all other test environment and that different methods generated significantly different sampling orders. This is similar to the idea that the coverage may not significantly affect structural scores, because the generalization from one room to another is to easy. Structural quality gains from a specific sampling order may only visible when the environment has areas contain many of the same features but add complexity in certain areas.

8.7 Effect of non-stationary state space on RL performance

The work (De Bruin et al., 2018) uses alternating training to train an RL while the state space is change. The strategy did not work for the test environment. Likely, the DQN could not transfer performance between updates because there were too many SRL updates per cycle. (De Bruin et al., 2018) used alternating cycle used episode then made 16 updates of the SRL with batch size 16, then 16 updates of the RL with the same batch size. Our experiments use 1000 SRL updates per cycle. The number of updates in a cycle of DeBruin is much smaller than perhaps allowing the RL to adjust to the new states. The other possibility is that a different learning rate caused the change DeBruin also used a learning rate of 0.0003. We use a similar learning rate of 0.0002.

8.8 Further research

Past research has shown that the different environments and different SRL losses can profoundly affect RL performance (Raffin et al., 2019). Different environments and SRL losses may also affect the relationship between the sampling method and SRL performance. A different SRL method might benefit more from a nonrandom sampling policy. For example, an SRL that uses the entire trajectory instead of isolated transitions will be improved significantly by nonrandom action. A different environment that does not generalize well from its initial location may allow the sampling method to improve the SRL more. The environment will make the bias of random action to the initial location more detrimental to learning a good synthetic state space. Another option is an environment with dynamics of varying difficulty that can be learned in an effective order. For example, two rooms on either side of the starting location, the room on the left has simple dynamics(an empty room) meanwhile the room on has moving obstacles. The state representation learning might benefit from learning the simple dynamics first then moving to more complicated ones.

9 Conclusion

Do curiosity rewards improve the, generalization, quality, or sample efficiency of a state representation learning algorithm?

Curiosity rewards do not improve any property of state representation learning algorithms above random action policy. In the simple environments different coverages did not lead to superior structural scores or RL performance. This could be because the feature transitions at the initial location generalize to the entire enironment. On the complex environment, curiosity rewards failed to achieve significantly different coverage from a random action policy. The rl performance was also not significantly affected by the sampling method. The results show that curiosity rewards did not improve state representation learning compared to random action policy. On the complex environment the uniform sampling had a significant effect on the structural scores. This suggests that in environments without generalizable features the sampling method has an effect on the structure on the synthetic state representation. The results also showed that this structure did not lead to improved RL performance. However, the connection between coverage and structure implies that if the connection between structure and RL performance were better understood curiosity driven sampling could be an important component to ensure good structure.

The results show that there is no benefit to curiosity rewards for this environment, but the possibility that it could be beneficial for a more complicated one.

To what extent does the uniformity of coverage of observations/states affect the quality of the state representation?

Uniformity increases the likelihood of good structural scores only on the complex environment. Uniformity ensures that all state transitions are sampled and weighted equally in the state representation learning algorithm. This property contributes to state quality when the environment is complicated enough that features learned in one area of the environment do not easily generalize to other areas. For example, in pick and place robotics task features learned for picking boxes may not generalize to picking fruit.

To what extent does the order that observation/states are visited affect the quality of the state representation?

The order of observations has no effect on the quality of the state representation. The difference in correlation between the first cycle, second cycle and structural quality is not a very strong measure. Since, the measure for this relation is weak a confident conclusion is difficult to make. A environment with a clear curriculum learning path where different areas in the environment offer increasing difficulty may show more of a result. An example may be where the combination of two features in two areas of an enironment allow generalization to a third unexplored area.

How to compensate for a non-stationary state space? Alternating, between RL and SRL updates was not enough for the RL to learn a good policy while the SRL was training. The result may be due to non optimal training hyperparameters. Practically this means that the time spent training the SRL will make training its corresponding rl take longer. Therefore, further research may need to consider when it is better to train an SRL with the expectation that it improves overall performance and when randomly initialized SRL's work better.

A Appendix 1

A.1

A.2 Information theory background

A learning robot has its progress constrained by the minimal directed information rate from the enviorment to the agentTiomkin and Tishby (2017). There for it is worth investigating measures of directed information for SRL. Directed information is the causal mutual information between two signals over time. It also bounds the capacity of a channel with feedback much like mutual information fora memoryless channel. This property also means it bounds possible gains in a reward from causal side information. To calculate directed information it is first import to understand the causal conditioning of probability is different from non-causal conditioning, shown below:

$$p(X_t^T||Y_t^T) = \prod_{i=t}^{T} p(x_i|x_t^{i-1}y_t^i)$$

$$p(X_t^T|Y_t^T) = \prod_{i=t}^{T} p(x_i|x_t^{i-1}y_t^T)$$

The signal at time step t must be synchronized between all sequences. A variable that can be causally conditioned on a sequence or a random variable at a particular time τ leading to the following formula.

$$p(X_t^T||Y_{t+\tau}) = \prod_{i=t+\tau}^{T} p(x_i|X_t^{i-1}, Y_{t+\tau}) \prod_{i=t}^{t+\tau} p(x_i|X_t^{i-1})$$

$$p(X_t^T|Y_{t+\tau}) = \prod_{i=t}^{T} p(X_i|X_t^{i-1}, Y_{t+\tau})$$

The entropy a causal condition variable is defined by the following causally conditioned entropy, where <.> is an expectation operator.

$$H(X_t^T||Y_t^T) = -\left\langle \ln\left[p(X_t^T||Y_t^T)\right]\right\rangle_{p(X_t^T,Y_t^T)}$$

This entropy lets us define the Directed information.

The directed information from one sequence to another is defined as the reduction in entropy of one sequence causally conditioned on the other Kramer (1998).

$$I[X_t^T \to Y_t^T] = H(Y_t^T) - H(Y_t^T||X_t^T)$$

A.3 Directed information reward to agent to reward

In Tiomkin and Tishby (2017) and Binas et al. (2019) they use the directed information from the state to the agent as a measure of how fast state the agent can learn about the enviorment. The directed information over a trajectory is decomposed into a bellman type operator where mutual information is added to a previous estimate of the directed information. This allows for the directed information to be integrated into a classic RL algorithm. The calculation of mutual

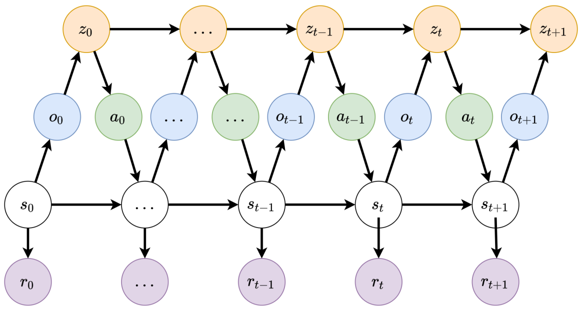


Figure A.1

information is then simplified to the estimation error of the next state given the last state and action and the uncertainty when the next action is also known de Abril and Kanai (2018).

$$r_i = |s_{t+1} - f(s_t, a_t)| - |s_{t+1} - f(s_t, a_t, a_{t+1})|$$

This connects nicely with Pathak et al. (2017) that used forward state prediction error. My hypothesis is the direct information from the reward to the agent will optimize the agents learning of the state representing.
To turn the directed information into a usable reward we first represent it as entropy calculated by the expectation operator <.>.

$$I(a_t^T \to r_t^T || s_{t+1}^{T+1}) = H(r_t^T || s_{t+1}^{T+1}) - H(r_t^T || s_{t+1}^{T+1}, a_t^T)$$

$$= \left\langle \ln \frac{p(r_t^T || a_t^T, s_{t+1}^{T+1})}{p(r_t^T || s_{t+1}^{T+1})} \right\rangle_{p(r_t^T, a_t^T, s_{t+1}^{T+1})}$$

Then using the graph in figure A.1 using d-separation rules it is possible to prove that the probability of R is independent of past values of R, A and S.

$$= \left\langle \ln \prod_{i=t}^{T} \frac{p(r_i | a_i, s_{i+1}, s_i)}{p(r_t^T | s_{i+1}, s_i)} \right\rangle_{p(r_t^T, a_t^T, s_{t+1}^{T+1})}$$

Use log rules and this becomes a sum of

$$= \sum_{i=t}^{T} \left\langle \ln \frac{p(r_i | a_i, s_{i+1}, s_i)}{p(r_t^T | s_{i+1}, s_i)} \right\rangle_{p(r_t^T, a_t^T, s_{t+1}^{T+1})}$$

This can be written as a bellman recursion. Where

$$= \left\langle \ln \frac{p(r_t | a_t, s_{t+1}, s_t)}{p(r_t | s_{t+1}, s_t)} \right\rangle_{p(r_t, a_t^T, s_{t+1}^{T+1})} + \sum_{i=t+1}^{T} \left\langle \ln \frac{p(r_i | a_i, s_{i+1}, s_i)}{p(r_t | s_{i+1}, s_i)} \right\rangle_{p(r_t^T, a_t^T, s_{t+1}^{T+1})}$$

$$= H(r_t | s_{t+1}, s_t) - H(r_t | s_{t+1}, s_t, a_t) + I(a_{t+1}^T \to r_{t+1}^T || s_{t+2}^{T+1})$$

$$= I(r_t; a_t | s_{t+1}, s_t) + I(a_{t+1}^T \to r_{t+1}^T || s_{t+2}^{T+1})$$

and according to de Abril and Kanai (2018) entropy can be approximated by prediction error so the mutual information $I(r_t; a_t | s_{t+1}, s_t)$ can be written as:

$$\approx |r_t - f(s_{t+1}, s_t)| - |r_t - f(s_{t+1}, s_t, a_t)|$$

This can be used as a reward in a standard Q maximization algorithm.

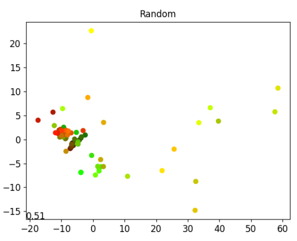
A.4 Related Curiosity works

Random action is not the most efficient exploration policy. The agent can be given some intrinsic reward to improve the exploration policy without external rewards. The intrinsic reward is added to the external reward r_t. The intrinsic reward should reward the agent for visiting true states that have not been previously visited. This leads to some exploration methods based on rewarding an RL agent for visiting states that it has not visited before using the reward in equation 4.5 $n(s,a)$ is the number of times the state action pair (s,a) has been visited.

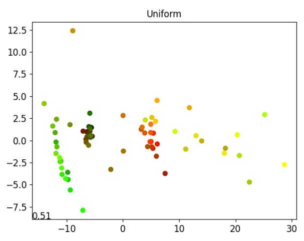
$$r_i = \frac{\beta}{\sqrt{n(s,a)}} \tag{A.1}$$

The count method is ineffective in continuous spaces. Two methods that encourage the similar behavior are prediction error based methods and goal based methods. Prediction error methods the error in the forward model of a learned state representation as a reward for an RL algorithm Pathak et al. (2017). They work off the assumption that as the robot occupies an area of space it will begin to learn the forward model the lack of error will make the robot "bored" and it will seek out new experience. This method can get stuck on a white noise generator in the enviorment as prediction error will never decrease if the enviorment is not predictable. Goal based methods generate goals that if the robot reaches it gets a reward. The goals are generated so that it is not trivially easy for the agent to reach the goal. The more the robot learns the harder the goal generator will make the goalsHeld et al. (2018). The prediction error and goal based methods both work by encouraging the agent to attempt tasks that are difficult but not too difficult. This idea has its roots in learning psychology Oudeyer et al. (2007).

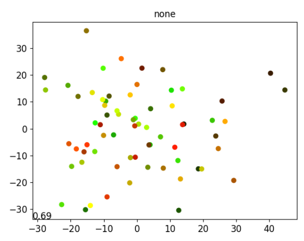
A.5 Figures



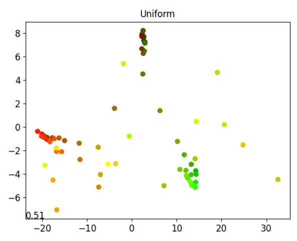
(a)



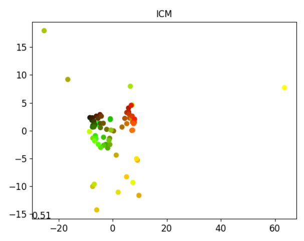
(b)



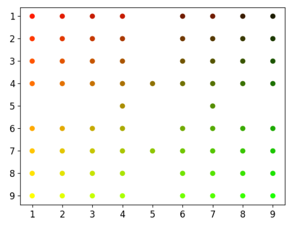
(c)



(d)



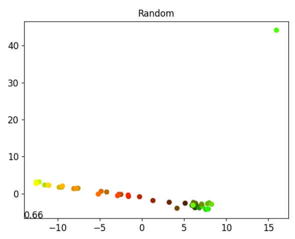
(e)



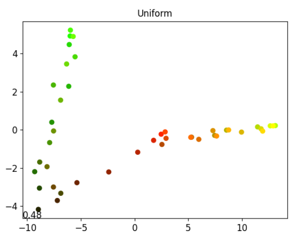
(f) Color Legend in true state space

Figure A.2: Example of 2 dimension PCA of Complex environment

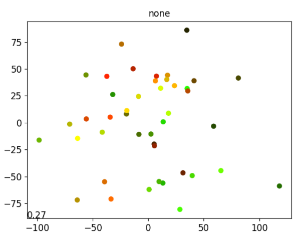
A.6 Figures



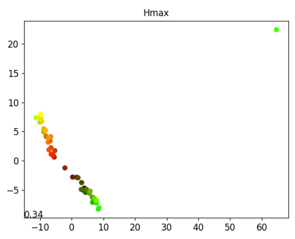
(a)



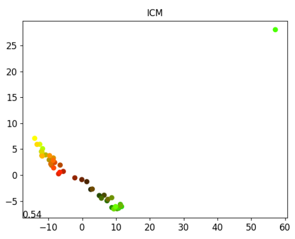
(b)



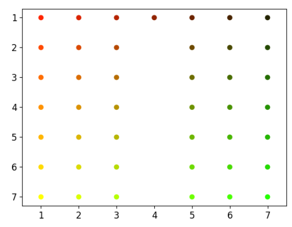
(c)



(d)



(e)



(f) Color Legend in true state space

Figure A.3: Example of 2 dimension PCA of simple environment

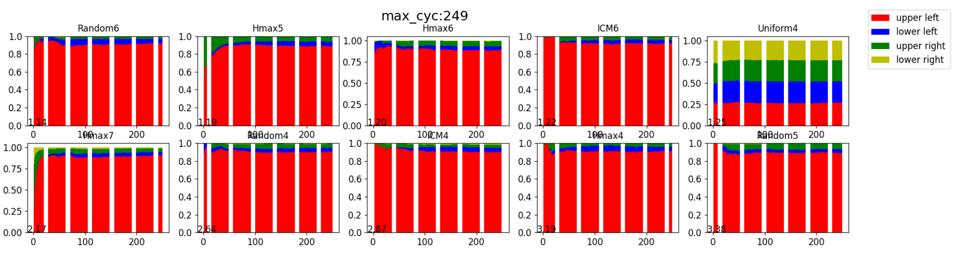


Figure A.4: The distribution of samples between the four rooms of the complex environment over time. The plots are sorted from Best KNN-MSE score in the upper left of the figure to worse in the lower right.

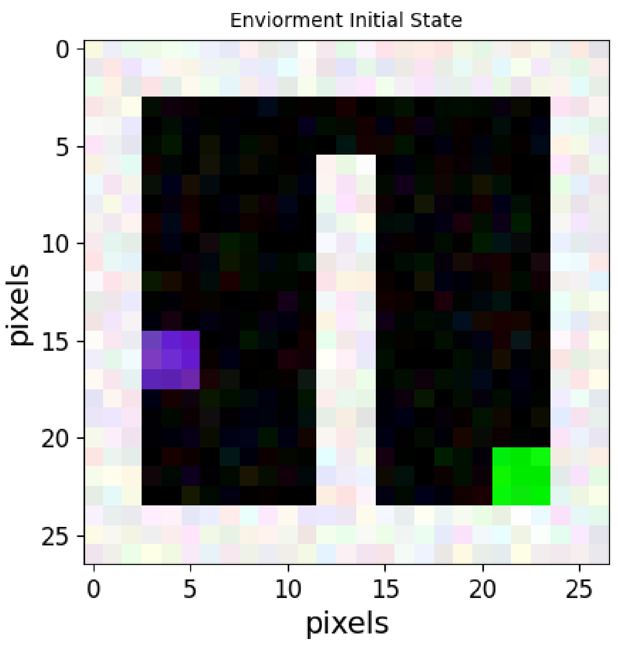


Figure A.5: observation of enviorment with added noise

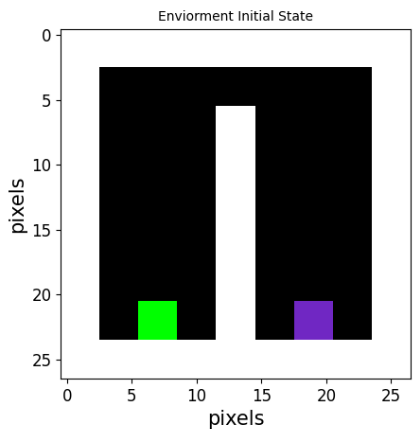
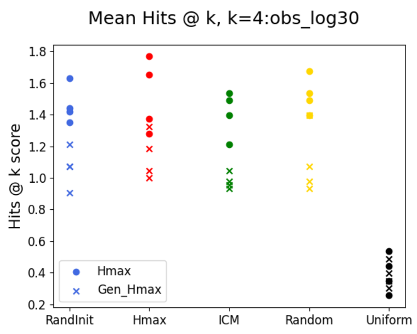
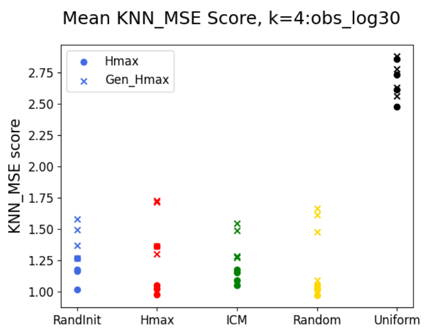


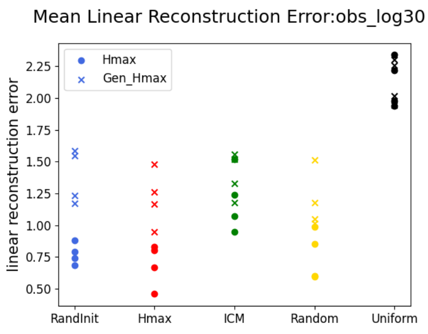
Figure A.6: inverted enviorment



(a) Hits at K

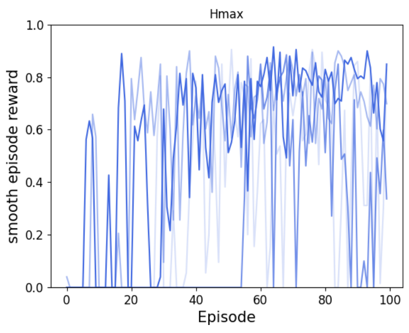


(b) KNN-MSE

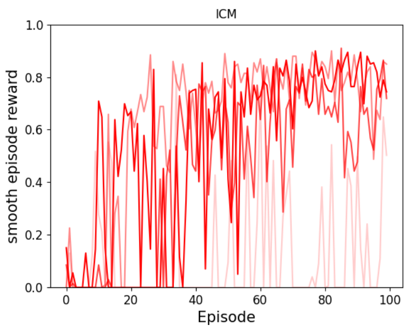


(c) Linear Reconstruction

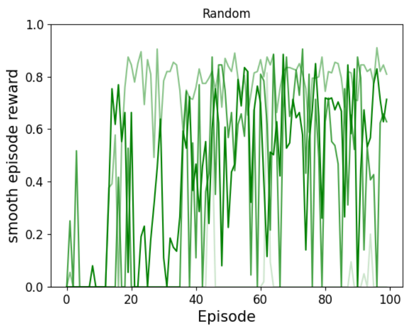
Figure A.7: Effect of transferring state representation to enviorment with added white noise on the structural scores



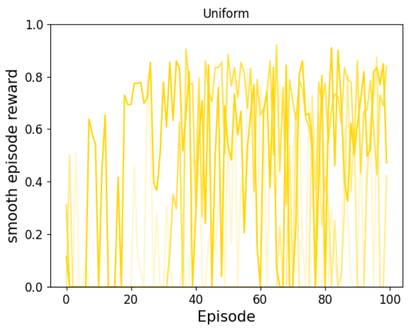
(a) Hits at K



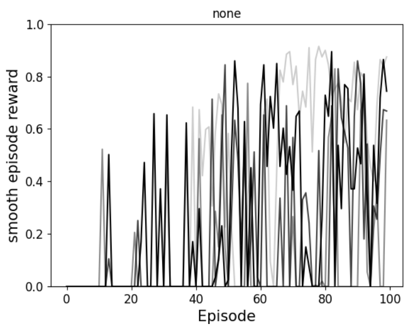
(b) Hits at K



(c) Hits at K



(d) Hits at K



(e) Hits at K

Figure A.8: The RL performance Episode reward of different runs of each sampling method with no moving average.

Bibliography

de Abril, I. M. and R. Kanai (2018), A unified strategy for implementing curiosity and empowerment driven reinforcement learning, *CoRR*, **vol. abs/1806.06505**. http://arxiv.org/abs/1806.06505

Alvernaz, S. and J. Togelius (2017), Autoencoder-augmented Neuroevolution for Visual Doom Playing, *CoRR*, **vol. abs/1707.03902**. http://arxiv.org/abs/1707.03902

Anand, A., E. Racah, S. Ozair, Y. Bengio, M.-A. Côté and R. D. Hjelm (2019), Unsupervised State Representation Learning in Atari, in *Advances in Neural Information Processing Systems*, volume 32, Eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/6fb52e71b837628ac16539c1ff911667-Paper.pdf

Aubret, A., L. Matignon and S. Hassas (2019), A survey on intrinsic motivation in reinforcement learning.

Bengio, Y., J. Louradour, R. Collobert and J. Weston (2009), Curriculum Learning, in *Proceedings of the 26th Annual International Conference on Machine Learning*, Association for Computing Machinery, New York, NY, USA, ICML '09, p. 41–48, ISBN 9781605585161, doi:10.1145/1553374.1553380. https://doi.org/10.1145/1553374.1553380

Binas, J., S. Ozair and Y. Bengio (2019), The Journey is the Reward: Unsupervised Learning of Influential Trajectories, *CoRR*, **vol. abs/1905.09334**. http://arxiv.org/abs/1905.09334

Burda, Y., H. Edwards, D. Pathak, A. Storkey, T. Darrell and A. A. Efros (2018a), Large-scale study of curiosity-driven learning, *arXiv preprint arXiv:1808.04355.*

Burda, Y., H. Edwards, D. Pathak, A. J. Storkey, T. Darrell and A. A. Efros (2018b), Large-Scale Study of Curiosity-Driven Learning, *CoRR*, **vol. abs/1808.04355**. http://arxiv.org/abs/1808.04355

Böhmer, W., J. Springenberg, J. Boedecker, M. Riedmiller and K. Obermayer (2015), Autonomous Learning of State Representations for Control: An Emerging Field Aims to Autonomously Learn State Representations for Reinforcement Learning Agents from Their Real-World Sensor Observations, *KI - Künstliche Intelligenz*, **vol. 29**, doi:10.1007/s13218-015-0356-1.

Chevalier-Boisvert, M., L. Willems and S. Pal (2018), Minimalistic Gridworld Environment for OpenAI Gym, https://github.com/maximecb/gym-minigrid.

De Bruin, T., J. Kober, K. Tuyls and R. Babuska (2018), Integrating State Representation Learning into Deep Reinforcement Learning, **vol. 3**, no.3, pp. 1394–1401, doi:10.1109/LRA.2018.2800101, cited By 32. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85063306351&doi=10.11092fLRA.2018.2800101&partnerID=40&md5=957cc2a34f491a360addef5a1c6c77d3

Ghosh, D., A. Gupta and S. Levine (2019), Learning actionable representations with goal-conditioned policies, in *NA*, cited By 4. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083954008&partnerID=40&md5=6fc2830c82f26106326337784ccd18b8

van Hasselt, H., A. Guez and D. Silver (2015), Deep Reinforcement Learning with Double Q-learning, *CoRR*, **vol. abs/1509.06461**.

http://arxiv.org/abs/1509.06461

Held, D., X. Geng, C. Florensa and P. Abbccl (2018), Automatic Goal Generation for Reinforcement Learning Agents, in *International conference on machine learning*, volume 4, pp. 2458–2471, cited By 13.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85057324384&partnerID=40&md5=1bada8170e30e0e24a4b90b8ea29398c

Jonschkowski, R. and O. Brock (2015), Learning state representations with robotic priors, *Autonomous Robots*, **vol. 39**, pp. 407–428, doi:10.1007/s10514-015-9459-7.

Jonschkowski, R., R. Hafner, J. Scholz and M. A. Riedmiller (2017), PVEs: Position-Velocity Encoders for Unsupervised Learning of Structured State Representations, *CoRR*, **vol. abs/1705.09805**.
http://arxiv.org/abs/1705.09805

Kramer, G. (1998), *Directed information for channels with feedback*, Citeseer.

Lesort, T., N. D. Rodr'iguez, J. Goudou and D. Filliat (2018), State Representation Learning for Control: An Overview, *CoRR*, **vol. abs/1802.04181**.
http://arxiv.org/abs/1802.04181

Lesort, T., M. Seurin, X. Li, N. DIaz-Rodriguez and D. Filliat (2019), Deep unsupervised state representation learning with robotic priors: A robustness analysis, doi:10.1109/IJCNN.2019.8852042, cited By 3.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85073186353&doi=10.11092fIJCNN.2019.8852042&partnerID=40&md5=826246a3e8329f64257156d322b89137

Lesort, T., M. Seurin, X. Li, N. D. Rodr'iguez and D. Filliat (2017), Unsupervised state representation learning with robotic priors: a robustness benchmark, *CoRR*, **vol. abs/1709.05185**.
http://arxiv.org/abs/1709.05185

Linke, C., N. Ady, M. White, T. Degris and A. White (2020), Adapting behavior via intrinsic reward: A survey and empirical study, *Journal of Artificial Intelligence Research*, **vol. 69**, pp. 1287–1332, doi:10.1613/JAIR.1.12087, cited By 0.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099406083&doi=10.16132fJAIR.1.12087&partnerID=40&md5=efe08c807dfea4d885e72bdea0eaf262

Morik, M., D. Rastogi, R. Jonschkowski and O. Brock (2019), State Representation Learning with Robotic Priors for Partially Observable Environments, in *IROS*, pp. 6693–6699, doi:10.1109/IROS40897.2019.8967938, cited By 0.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85081165779&doi=10.11092fIROS40897.2019.8967938&partnerID=40&md5=3c6fd5a1f4b2ffd58f077fed4aec1259

Ostrovski, G., M. Bellemare, A. Van Den Oord and R. Munos (2017), Count-based exploration with neural density models, in *NA*, volume 6, pp. 4161–4175, cited By 25.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85048544793&partnerID=40&md5=28a371e8230e46b8c6622468f6a45034

Oudeyer, P.-Y., F. Kaplan and V. Hafner (2007), Intrinsic motivation systems for autonomous mental development, *IEEE Transactions on Evolutionary Computation*, **vol. 11**, doi:10.1109/TEVC.2006.890271.

Pathak, D., P. Agrawal, A. Efros and T. Darrell (2017), Curiosity-Driven Exploration by Self-Supervised Prediction, in *International Conference on Machine Learning*, volume 2017-July, pp. 488–489, doi:10.1109/CVPRW.2017.70, cited By 155.
https://www.scopus.com/inward/record.uri?eid=2-s2.

0-85030252336&doi=10.11092fCVPRW.2017.70&partnerID=40&md5=015dae6c236758b4a30be52dcb9706be

van der Pol, E., T. Kipf, F. Oliehoek and M. Welling (2020), Plannable approximations to MDP homomorphisms: Equivariance under actions, in *NA*, volume 2020-May, pp. 1431–1439, cited By 0.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-85095254905&partnerID=40&md5=7cc1136c18d855e8d7bbcdbbf9ebada8

Raffin, A., A. Hill, K. R. Traor'e, T. Lesort, N. D. Rodr'iguez and D. Filliat (2019), Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics, *CoRR*, **vol. abs/1901.08651**.
http://arxiv.org/abs/1901.08651

Ravindran, B. and A. G. Barto (2004), Approximate Homomorphisms: A framework for non-exact minimization in Markov Decision Processes, *International Conference on Knowledge Based Computer Systems*.

Sanders, J., A. Proutière and S.-Y. Yun (2020), Clustering in block markov chains, **vol. 48**, no.6, pp. 3488–3512.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov (2017), Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347*.

Seo, Y., L. Chen, J. Shin, H. Lee, P. Abbeel and K. Lee (2021), State Entropy Maximization with Random Encoders for Efficient Exploration, *arXiv preprint arXiv:2102.09430*.

Sharma, A., S. Gu, S. Levine, V. Kumar and K. Hausman (2020), Dynamics-Aware Unsupervised Discovery of Skills.

Singh, H., N. Misra, V. Hnizdo, A. Fedorowicz and E. Demchuk (2003), Nearest Neighbor Estimates of Entropy, **vol. 23**, no.3-4, pp. 301–321, doi:10.1080/01966324.2003.10737616.
https://doi.org/10.1080/01966324.2003.10737616

Stadie, B. C., S. Levine and P. Abbeel (2015), Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models, *CoRR*, **vol. abs/1507.00814**.
http://arxiv.org/abs/1507.00814

Sutton, R. S., A. G. Barto et al. (1998), *Introduction to reinforcement learning*, volume 135, MIT press Cambridge.

Tiomkin, S. and N. Tishby (2017), A Unified Bellman Equation for Causal Information and Value in Markov Decision Processes, *CoRR*, **vol. abs/1703.01585**.
http://arxiv.org/abs/1703.01585

Watter, M., J. Springenberg, J. Boedecker and M. Riedmiller (2015), Embed to control: A locally linear latent dynamics model for control from raw images, in *NA*, volume 2015-January, pp. 2746–2754, cited By 171.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-84965129327&partnerID=40&md5=a628b2ac5bd4d410c8e1493ad1dc83fd

Zhang, A., H. Satija and J. Pineau (2018), Decoupling Dynamics and Reward for Transfer Learning.

Zhang, P., Y. Ren and B. Zhang (2012), A new embedding quality assessment method for manifold learning, *Neurocomputing*, **vol. 97**, pp. 251–266, doi:10.1016/j.neucom.2012.05.013, cited By 17.
https://www.scopus.com/inward/record.uri?eid=2-s2.0-84865328263&doi=10.10162fj.neucom.2012.05.013&partnerID=40&md5=3493c704a66ca98ecb5f289a97c16cf2