Interpretable (Un)Controllable Features in MDP's

Anonymous authors
Paper under double-blind review

Abstract

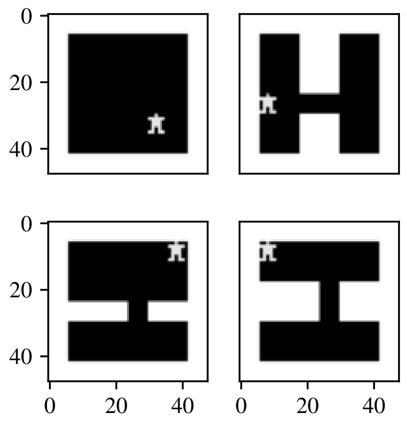
In the context of MDPs with high-dimensional states, downstream tasks are predominantly applied on a compressed, low-dimensional representation of the original input space. A variety of learning objectives have therefore been used to attain useful representations. However, these representations usually lack interpretability of the different features. We present a novel approach that is able to disentangle latent features into a controllable and an uncontrollable partition. We illustrate that the resulting partitioned representations are easily interpretable on three types of environments and show that, in a distribution of procedurally generated maze environments, it is feasible to interpretably employ a planning algorithm in the isolated controllable latent partition.

1 Introduction

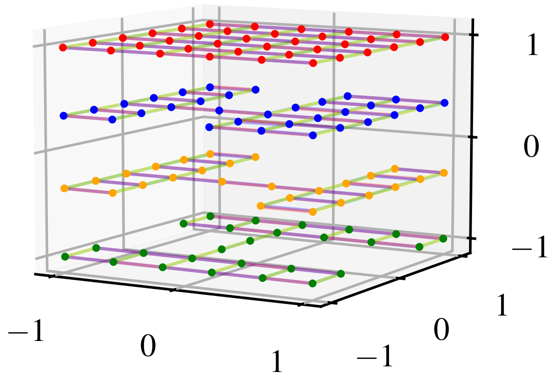
Learning from high-dimensional data remains a challenging task. Particularly for reinforcement learning (RL), the complexity and high dimensionality of the Markov Decision Process (MDP) (Bellman, 1957) states often leads to complex or intractable solutions. In order to facilitate learning from high-dimensional input data, an encoder architecture can be used to compress the inputs into a lower-dimensional latent representation. To this extent, a plethora of work has successfully focused on discovering a compressed encoded representation that accommodates the underlying features for the task at hand (Jonschkowski & Brock, 2015; Jaderberg et al., 2017; Laskin et al., 2020; Lee et al., 2020; Yarats et al., 2021; Schwarzer et al., 2021; Kostrikov et al., 2021).

The resulting low-dimensional representations however seldom contain specific disentangled features, which leads to disorganized latent information. This means that the individual latent states can represent the information from the state in any arbitrary way. The result is a representation with poor interpretability, as the latent states cannot be connected to certain attributes of the original observation space (e.g, the x-y coordinates of the agent). Prior work in structuring a latent representation has shown notions and use of interpretability in MDP representations (Francois-Lavet et al., 2019). When expanding this notion of interpretability to be compatible with RL, it has been argued that the controllable features should be an important element of a latent representation, since it generally represents what is directly influenced by the policy. In this light, Thomas et al. (2017) have introduced the concept of isolating and disentangling controllable features in a low-dimensional maze environment, by means of a selectivity loss. Furthermore, Kipf et al. (2020) took an object-centric approach to isolate distinct objects in MDPs and Ahuja et al. (2022) showed theoretical foundations for this isolation in a weakly-supervised controllable setting. Controllable features however only represent a fragment of an environment, where in many cases the uncontrollable features are of equal importance. For example, in the context of a distribution of mazes, for the prediction of the next controllable (agent) state following an action, the information about the wall structure is crucial (see Fig. 1). We therefore hypothesize that a thorough representation should incorporate controllable and uncontrollable features, ideally in a disentangled, interpretable arrangement; Intepretability is crucial for future real-world deployment (Glanois et al., 2021), while an additional benefit would be that the separation of the controllable and uncontrollable features can be exploited in downstream algorithms such as planning.

Our contribution consists of an algorithm that, showcased in three different MDP settings, explicitly disentangles the latent representation into a controllable and an uncontrollable latent partition. This is highlighted on three types of environments, each with a varying class of controllable and uncontrollable elements. This



(a) Random observations in each of the 4 mazes



(b) Encoded representations $\forall s \in \mathcal{S}$

Figure 1: Visualization in a maze environment of (a) four random states $\in \mathbb{R}^{48\times48}$ and (b) the disentanglement of the controllable latent $z^c \in \mathbb{R}^2$ on the horizontal axes, and the uncontrollable latent $z^u \in \mathbb{R}^1$ on the vertical axis, given for all states in the four maze environments shown in four different colors. The representation is trained on high-dimensional tuples (s_t, a_t, r_t, s_{t+1}), sampled from a replay buffer $\mathcal{B}$, gathered from random trajectories in the four maze environments shown in (a). All possible states are encoded with $z_t = f(s_t; \theta_{enc})$ and plotted in (b) with the transition prediction for each possible action, revealing a clear disentanglement between the controllable agent's position and the uncontrollable wall architecture. Note that all samples are taken from the same buffer, filled with samples from all four mazes.

allows for a precise and visible separation of the latent features, improving interpretability, representation quality and possibly moving towards a basis for building causal relationships between an agent and its environment. The unsupervised learning algorithm consists of both an action-conditioned and a state-only forward predictor, along with a contrastive and an adversarial loss, which isolate and disentangle the controllable versus the non-controllable features. Furthermore, we show an application of learning and planning on the human-interpretable disentangled latent representation, where the properties of disentanglement allow the planning algorithm to operate solely in the controllable partition of the latent representation.

2 Related Work

General Representation Learning. Many works have focused on converting high-dimensional inputs to a compact, abstract latent representation. Learning this representation can make use of auxiliary, unsupervised tasks in addition to the pure RL objectives (Jaderberg et al., 2017). One way to ensure a meaningful latent space is to implement architectures that require a pixel reconstruction loss such as a variational (Kingma & Welling, 2014; Higgins et al., 2017) or a deterministic (Yarats et al., 2021) autoencoder. Other approaches combined reward reconstruction with latent prediction (Gelada et al., 2019), pixel reconstruction with planning (Hafner et al., 2019; 2021) or used latent predictive losses without pixel reconstruction (Lee et al., 2020; Schwarzer et al., 2021).

Representing controllable features. In representation learning for RL, a focus on controllable features can be beneficial as these features are strongly influenced by the policy (Thomas et al., 2017). This can be done using generative methods (Laversanne-Finot et al., 2018), but is most commonly pursued using an auxiliary inverse-prediction loss; predicting the action that was taken in the MDP (Jonschkowski & Brock, 2015). The work in Pathak et al. (2017); Badia et al. (2020) builds a latent representation with an emphasis on the controllable features of an environment with inverse-prediction losses, and uses these features to guide exploratory behavior. Furthermore, Efroni et al. (2021) and concurrent work by Lamb et al. (2022) employ multi-step inverse prediction to successfully encompass controllable features in their representation. However, these works have not expressed a focus on also retaining the uncontrollable features in their representation, which is a key aspect in our work.

Partitioning a latent representation. Sharing similarity in terms of the separation of the latent representation, Bertoin & Rachelson (2022) disentangle the latent representation in the domain adaptation setting into a task-relevant and a context partition, by means of adversarial predictions with gradient reversals and cyclic reconstruction. Fu et al. (2021) use a reconstruction-based adversarial architecture that divides their latent representation into reward-relevant and irrelevant features. Related work by Wang et al. (2022) further divides the latent representation of Dreamer (Hafner et al., 2019), using action-conditioned and state-only forward predictors, into controllable, uncontrollable and their respective reward relevant and irrelevant features. As compared to Wang et al. (2022), who focus on distraction-efficient RL, we purely focus on the representational learning aspect of these predictors, and show notions of separation in low-dimensional, structured representations of MDPs, leaning towards enhanced interpretability. Furthermore, we use an adversarial loss to enforce disentanglement between z^c and z^u, and apply a contrastive loss instead of pixel reconstruction to avoid representation collapse due to latent forward prediction.

Interpretable representations in MDPs. More closely related to our research is the work by Thomas et al. (2017), which connects individual latent dimensions to independently controllable states in a maze using a reconstruction loss and a selectivity loss. The work by Francois-Lavet et al. (2019) visualizes the representation of an agent and its transitions in a maze environment, but does not disentangle the agent state in its controllable and uncontrollable parts, which limits the interpretability analysis and does not allow simplifications during planning. The work by Kipf et al. (2020) uses an object-oriented approach to isolate different (controllable) features, using graph neural networks (GNN's) and a contrastive forward prediction loss, but does not discriminate between controllable and uncontrollable features. Further work in this direction by Ahuja et al. (2022) focuses on theoretical foundations for an encoder to structurally represent a distinct controllable object. We aim to progress the aforementioned lines of research by using a representation learning architecture that disentangles an MDP's latent representation into interpretable, disentangled controllable and uncontrollable features. Finally, we show that having separate partitions of controllable and uncontrollable features can be exploited in a planning algorithm. Exploitations like these are done in combination with prior knowledge of a certain MDP, as in van der Pol et al. (2020).

3 Preliminaries

We consider an agent acting within an environment, where the environment is modeled as a discrete Markov Decision Process (MDP) defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the environment's transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the environment's reward mapping and γ is the discount factor. We consider the setting where we have access to a replay buffer ($\mathcal{B}$) of visited states $s_t \in \mathcal{S}$ that were followed by actions $a_t \in \mathcal{A}$ and resulted in the rewards $r_t \in \mathcal{R}$ and the next states s_{t+1}. One entry in B contains a tuple of past experience (s_t, a_t, r_t, s_{t+1}). The agent's goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expectation of the discounted return $V^{\pi}(s) = \mathbb{E}_{\tau}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_t = s]$, where τ is a trajectory following the policy π.

Furthermore, we examine the setting where a high-dimensional state ($s_t \in \mathbb{R}^v$) is compressed into a lower-dimensional latent state $z_t \in \mathcal{Z} = \mathbb{R}^w$ where $\mathcal{Z}$ represents the latent space with $w \leq v$. This is done by means of a neural network encoding $f : \mathcal{S} \to \mathcal{Z}$ where f represents the encoder.

4 Algorithm

We aim for an interpretable and disentangled representation of the controllable and uncontrollable latent features. We define controllable features as the characteristics of the MDP that are predominantly affected by any action $a \in \mathcal{A}$, such as the position of the agent in the context of a maze environment. The uncontrollable features are those attributes that are not or only marginally affected by the actions. We show that the proposed disentanglement is possible by designing losses and gradient propagation through two separate parts of the latent representation. Specifically, to assign controllable information to the controllable latent partition, the gradient from an action-conditioned forward predictor is propagated through it. To assign uncontrollable information to the uncontrollable latent partition, the gradient from a state-only forward predictor is propagated through it. The remaining details will be provided in the rest of this Section.

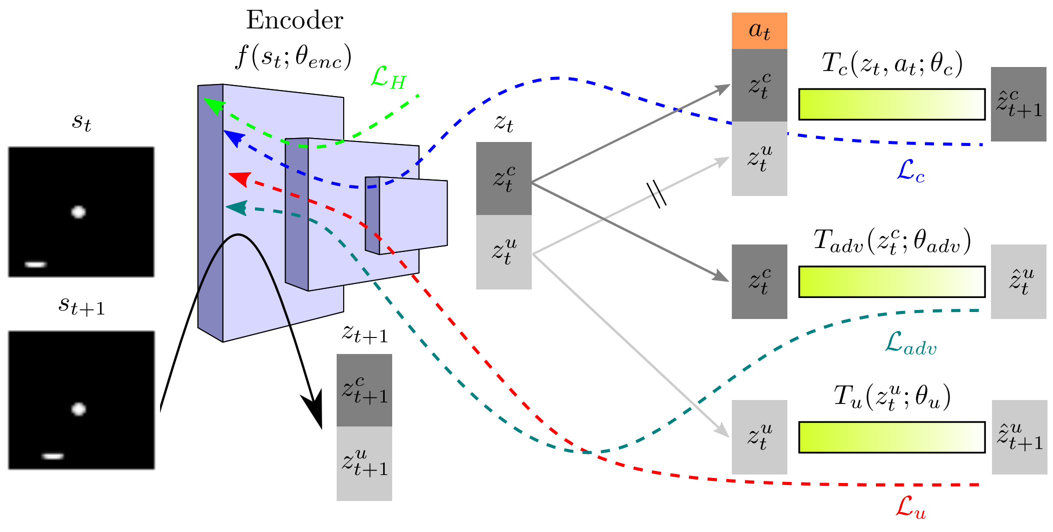


Figure 2: Overview of the disentangling architecture, with dashed lines representing gradient propagation and green rectangles representing parameterized prediction functions. An observation s_t is encoded into a latent representation consisting of two parts; z_t^c and z_t^u, which represent controllable and uncontrollable features respectively. These separated representations are then independently used to make action-conditioned, state-only and adversarial predictions in order to provide gradients to the encoder that disentangle the latent representation z_t into controllable (z_t^c) and uncontrollable (z_t^u) partitions.

We consider environments with high-dimensional states, represented as pixel inputs. These pixel inputs are subsequently encoded into a latent representation $z_t = (z^c, z^u) \in \mathcal{Z} \in \mathbb{R}^{n_c} + \mathbb{R}^{n_u}$, with the superscripts c and u representing the controllable and uncontrollable features, and the superscripts n_c and n_u representing their respective dimensions. The compression into a latent representation $\mathcal{S} \rightarrow \mathcal{Z}$ is done by means of a convolutional encoder, parameterized by a set of learnable parameters θ_{enc} according to:

$$z_t = (z_t^c, z_t^u) = f(s_t; \theta_{enc}). \tag{1}$$

An overview of the proposed algorithm is illustrated in Fig. 2 and the details are provided hereafter. In this section, all losses and transitions are given under the assumption of a continuous abstract representation and a deterministic transition function. The algorithm could be adapted by replacing the losses related to the internal transitions with generative approaches (in the context of continuous and stochastic transitions) or a log-likelihood loss (in the context of stochastic but discrete representations).

4.1 Controllable Features

To isolate controllable features in the latent representation, z_t^c is used to make an action-conditioned forward prediction in latent space. In the context of a continuous latent space and deterministic transitions, z^c is updated using a mean squared error (MSE) forward prediction loss $\mathcal{L}_c = \left|\hat{z}_{t+1}^c - z_{t+1}^c\right|^2$, where $\hat{z}_{t+1}^c$ is the action-conditioned residual forward prediction of the parameterized function $T_c(z, a; \theta_c) : \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{Z}$:

$$\hat{z}_{t+1}^c = T_c(z_t, a_t; \theta_c) + z_t^c \tag{2}$$

and the prediction target z_{t+1}^c is part of the encoder output $f(s_{t+1}; \theta_{enc})$. Note that the full latent state z_t is necessary in order to predict $\hat{z}_{t+1}^c$ (e.g. the uncontrollable features could represent a wall or other static structure that is necessary for the prediction of the controllable features). Furthermore, the uncontrollable latent partition input z_t^u is accompanied by a stop gradient to discourage the presence of controllable features in z^u. When minimizing $\mathcal{L}_c$, both the encoder (θ_{enc}) as well as the predictor (θ_c) are updated, which allows shaping the representation z^c as well as learning the internal dynamics.

4.2 Uncontrollable Features

To express uncontrollable features in the latent space, z_t^u is used to make a state-only (not conditioned on the action a_t) forward prediction in latent space. This enforces uncontrollable features within the uncontrollable latent partition z^u, since features that are action-dependent cannot be accurately predicted with the preceding state only. Following a residual prediction, z^u is then updated using a MSE forward prediction loss $\mathcal{L}_u = \left|\hat{z}_{t+1}^u - z_{t+1}^u\right|^2$, with $\hat{z}_{t+1}^u$ defined as:

$$\hat{z}_{t+1}^u = T_u(z_t^u; \theta_u) + z_t^u \tag{3}$$

and $T_u(z^u; \theta_u) : \mathcal{Z} \rightarrow \mathcal{Z}$ representing the parameterized prediction function. The target z_{t+1}^u is part of the output of the encoder $f(s_{t+1}; \theta_{enc})$. When minimizing $\mathcal{L}_u$, both θ_{enc} and θ_u are updated. In this way the loss $\mathcal{L}_u$ drives the latent representation z^u, which is conditioned on θ_{enc} according to $(z_t^c, z_t^u) = f(s_t; \theta_{enc})$, to only represent the features of s_t that are not conditioned on the action a_t.

4.3 Avoiding Predictive Representation Collapse

Minimizing a forward prediction loss in latent space $\mathcal{Z}$ is prone to collapse (Francois-Lavet et al., 2019; Gelada et al., 2019), due to the convergence of $\mathcal{L}_c$ and $\mathcal{L}_u$ when $f(s_t; \theta_{enc})$ is a constant $\quad \forall \; s_t \in \mathcal{S}$. To avoid representation collapse when using forward predictors, a contrastive loss is used to enforce sufficient diversity in the latent representation:

$$\mathcal{L}_{H_1} = exp\big(- C_d \big\| z_t - \bar{z}_t \big\|_2 \big) \tag{4}$$

where C_d represents a constant hyperparameter and $\bar{z}_t$ is a 'negative' batch of latent states z_t, which is obtained by shifting each position of latent states in the batch by a random number between 0 and the batch size. In the random maze environment, an additional contrastive loss is added to further diversify the controllable representation:

$$\mathcal{L}_{H_2} = exp\big(- C_d \big\| z_t^c - \bar{z}_t^c \big\|_2 \big) \tag{5}$$

where z_t^c is obtained from randomly sampled trajectories. This additional regularizer proved neccessary to avoid collapse of z^c when moving to a near infinite number of possible mazes. More information on this subject can be found in Appendix A.4. The resulting contrastive loss $\mathcal{L}_H$ for the random maze environment then consists of $0.5\mathcal{L}_{H_1} + 0.5\mathcal{L}_{H_2}$. The total loss used to update the encoder's parameters now consists of $\mathcal{L}_{enc} = \mathcal{L}_c + \mathcal{L}_u + \mathcal{L}_H$.

4.4 Guiding Feature Disentanglement with Adversarial Loss

When using a controllable latent space $z^c \in \mathbb{R}^x, x \in \mathbb{N}$, where $x > g$, with g representing the number of dimensions needed to portray the controllable features, some information about the uncontrollable features in the controllable latent representation might be present (see Appendix D.2). This is due to the non-enforcing nature of $\mathcal{L}_c$, as the uncontrollable features are equally predictable with or without the action. To ensure that no information about the uncontrollable features is kept in the controllable latent representation, an adversarial component is added to the architecture in Fig. 2. This is done by updating the encoder with an adversarial loss $\mathcal{L}_{adv}$ and reversing the gradient (Ganin et al., 2016). The adversarial loss is defined as

$$\mathcal{L}_{adv} = \left|\hat{z}_t^u - z_t^u\right|^2, \tag{6}$$

with $\hat{z}_t^u = T_{adv}(z_t^c; \theta_{adv})$, where $\hat{z}_t^u$ is the uncontrollable prediction of the parameterized function $T_{adv}(z^c; \theta_{adv}) : \mathcal{Z} \rightarrow \mathcal{Z}$ and z_t^u is the target. Intuitively, since the parameters of $T_{adv}(z^c; \theta_{adv})$ are being updated with $\mathcal{L}_{adv}$ and the parameters of $f(s; \theta_{enc})$ are being updated with $-\mathcal{L}_{adv}$, the prediction function can be seen as the discriminator and the encoder can be seen as the generator (Goodfellow et al., 2014). The discriminator tries to give an accurate prediction of the uncontrollable latent z^u given the controllable latent z^c, while the generator tries to counteract the discriminator by removing any uncontrollable features from the controllable representation. In our case, the predictor is a multi-layer perceptron (MLP), which means that minimizing $\mathcal{L}_{adv}$ enforces that no nonlinear relation between z^c and z^u can be learned. We hypothesize that this is a deterministic approximation of minimizing the Mutual Information (MI) between z^u and z^c. When using the adversarial loss, the combined loss propagating through the encoder consists of $\mathcal{L}_{enc} = \mathcal{L}_c + \mathcal{L}_u + \mathcal{L}_H - \mathcal{L}_{adv}$. Here the minus term in $-\mathcal{L}_{adv}$ represents a gradient reversal to the encoder. Note that the losses are not scaled, as this did not prove to be necessary for the experiments conducted.

Algorithm 1 Interpretable (Un)Controllable Features in MDP's

```
 1: Initialize $\theta_{enc}$, $\theta_c$, $\theta_u$, $\theta_{adv}$
 2: for $iteration = 1, 2, \ldots, N$ do
 3:     Sample batch of tuples $\{s_t, a_t, s_{t+1}\}$
 4:     Encode observations: $f(s; \theta_{enc}) = \{z^c, z^u\}$
 5:     Predict $\hat{z}^c_{t+1} = T_c(z^c_t, z^u_t, a; \theta_c) + z^c_t$          // detach $z^u_t$
 6:     Predict $\hat{z}^u_{t+1} = T_u(z^u_t; \theta_u) + z^u_t$
 7:     Predict $\hat{z}^u_t = T_{adv}(z^c_t; \theta_{adv})$
 8:     Compute losses $\mathcal{L}_c, \mathcal{L}_u, -\mathcal{L}_{adv}, \mathcal{L}_H$
 9:     Update parameters $\theta_{enc}$, $\theta_c$, $\theta_u$, $\theta_{adv}$
10: end for
```

4.5 Downstream Tasks

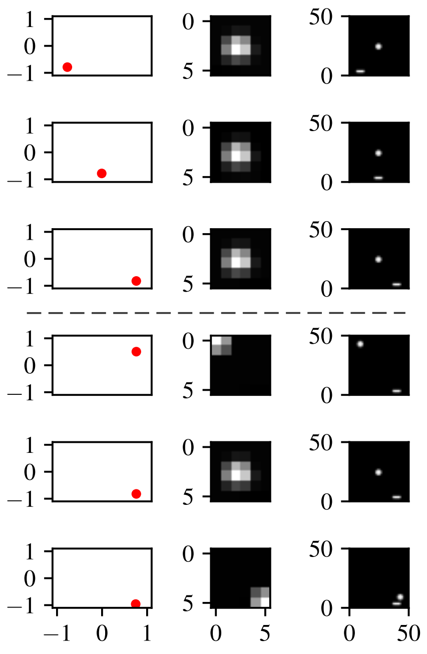
By disentangling a latent representation in a controllable and an uncontrollable part, one can more readily obtain human-interpretable features. While interpretability is generally an important aspect, it is also important to test how a notion of human interpretability affects downstream performance, as it is generally desired to strike a good balance between interpretability and performance. This is examined by training an RL agent on the learned and subsequently frozen latent representation. The action a_t is chosen following an ϵ-greedy policy, where a random action is taken with a probability ϵ, and with $(1-\epsilon)$ probability the policy $\pi(z) = \arg\max_{a \in \mathcal{A}} Q(z, a; \theta)$ is evaluated, where $Q(z, a; \theta)$ is the Q-network trained by Deep Double Q-Learning (DDQN) (Mnih et al., 2015; van Hasselt et al., 2016). The Q-network is trained with respect to a target Y_t:

$$Y_t = r_t + \gamma Q(z_{t+1}, \arg\max_{a \in \mathcal{A}} Q(z_{t+1}, a; \theta); \theta^-) \,. \tag{7}$$

With γ representing the environment's discount factor and θ^- the target Q-network's parameters. The target Q-network's parameters are updated as an exponential moving average of the original parameters θ according to: $\theta^-_{k+1} = (1-\tau)\theta^-_k + \tau\theta_k$, where subscript k represents a training iteration and τ represents a hyperparameter controlling the speed of the parameter update. The resulting DDQN loss is defined as $\mathcal{L}_Q = \big|Y_t - Q(z_t, a; \theta)\big|^2$. The full computation of all losses is shown in pseudocode in Algorithm 1.

5 Experiments

In this section, we showcase the disentanglement of controllable and uncontrollable features on three different environments, the complexity of which is in line with prior work on structured representations (Thomas et al., 2017; Higgins et al., 2018; Francois-Lavet et al., 2019; Kipf et al., 2020; Ahuja et al., 2022): (i) a quadruple maze environment, (ii) the catcher environment and (iii) a randomly generated maze environment. The first environment yields a state space of 119 different observations, and is used to showcase the algorithm's ability to disentangle a low-dimensional latent representation. The catcher environment examines a setting where the uncontrollable features are not static, and the random maze environment is used to showcase disentanglement in a more complex distribution of over 25 million possible environments, followed by the application of downstream tasks by applying reinforcement learning (DDQN) and a latent planning algorithm running in the controllable latent partition . The base of the encoder is derived from Tassa et al. (2018) and consists of two convolutional layers, followed by a fully connected layer for low-dimensional latent representations or an additional CNN for a higher-dimensional latent representation such as a feature map. For the full network architectures, we refer the reader to Appendix C. In all environments, the encoder $f(s; \theta_{enc})$ is trained from a buffer $\mathcal{B}$ filled with transition tuples (s_t, a_t, r_t, s_{t+1}) from random trajectories. Note that, in interpretability, there is generally not a specific metric to optimize for. In order to produce


(a) Without $\mathcal{L}_{adv}$

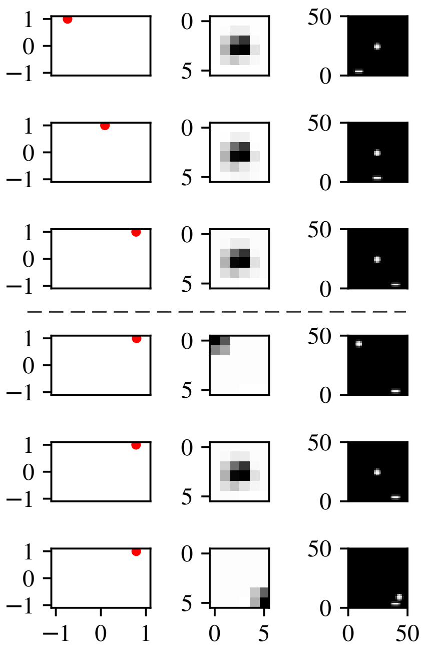

(b) With $\mathcal{L}_{adv}$

Figure 3: Visualization of the latent feature disentanglement in the catcher environment after 200k training iterations, with $z_t = f(s_t; \theta_{enc}) \in \mathbb{R}^2 + \mathbb{R}^{6\times 6}$. In (a) and (b), the left column shows z_t^c, the middle column is a feature map representing z_t^u and the right column is the pixel state s_t. The dashed lines separate observations where the ball position or the paddle position is kept fixed for illustration purposes. z^c tracks the agent position while z^u tracks the falling ball. In b), note that even when having a two-dimensional controllable state (only 1 is needed, see Appendix D.2), the adversarial loss in b) makes sure that distinct ball positions have a negligible effect on z^c (left column), even when the high-level features of the agent and the ball might be hard to distinguish.

interpretable representations, finding the right hyperparameters required manual (human) inspection of the plotted latent representations. An ablation of the hyperparameters used can be found in Appendices A1-A3

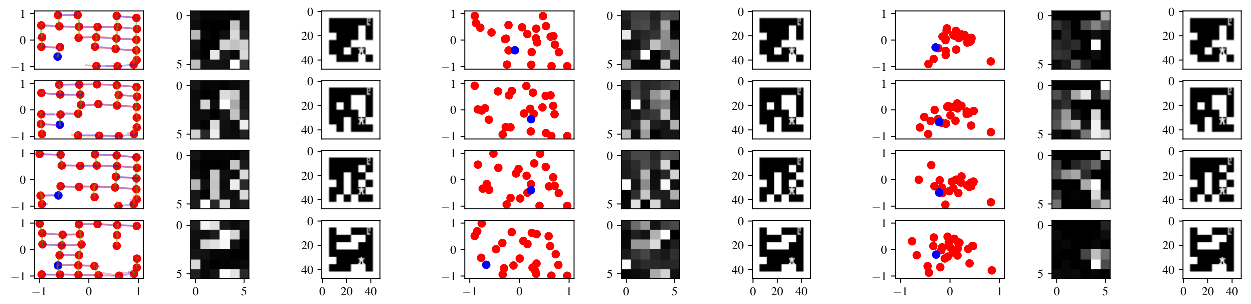
5.1 Quadruple Maze Environment

The maze environment consists of an agent and a selection of four distinct, handpicked wall architectures. The environment's state is provided as pixel observations $s_t \in \mathbb{R}^{48\times 48}$, where an action moves the agent by 6 pixels in each direction (up, down, left, right) except if this direction is obstructed by a wall. We consider the context where there is no reward ($r_t = 0 \;\; \forall \;\; (s_t, a_t) \in (\mathcal{S}, \mathcal{A})$) and there is no terminal state.

We select a two-dimensional controllable representation ($z^c \in \mathbb{R}^2$) and a one-dimensional uncontrollable representation ($z^u \in \mathbb{R}^1$). The remaining hyperparameters and details can be found in Appendix B. The experiments are conducted using a buffer $\mathcal{B}$ filled with random trajectories from the four different basic maze architectures. The encoder's parameters are updated using $\mathcal{L}_{enc}$ in Section 4.3 with $\mathcal{L}_H = \mathcal{L}_{H_1}$. After 50k training iterations, a clear disentanglement between the controllable (z^c) and uncontrollable (z^u) latent representation can be seen in Fig. 1. One can observe that the encoder is updated so that the one-dimensional latent representation z^u learns different values that define the type of wall architecture. A progression to this representation is provided in Appendix D.1.

5.2 Catcher Environment

As opposed to the maze environment, the catcher environment encompasses uncontrollable features that are non-stationary. The ball is dropped randomly at the top of the environment and is falling irrespective of the actions, while the paddle position is directly modified by the actions. The environment's states are defined as pixel observations s_t of size $\mathbb{R}^{51\times 51}$. At each time step, the paddle moves left or right by 3



(a) $\mathcal{L}_c = \mathcal{L}_c$ (b) $\mathcal{L}_c = \mathcal{L}_{inv}$ (c) $\mathcal{L}_{enc} = \mathcal{L}_Q$

Figure 4: A plot of the latent representation for all observations in a single randomly sampled maze when training with the aforementioned losses (a), substituting the action-conditioned forward-prediction loss $\mathcal{L}_c$ for an inverse-prediction loss $\mathcal{L}_{inv}$ (b) and when end-to-end updating the encoder with only the Q-loss $\mathcal{L}_Q$ from DDQN for 500k iterations (c). The left column shows the controllable latent $z_t^c \in \mathbb{R}^2$ with the current state in blue, the remaining states in red, and the predicted movement due to actions as different colored bars for each individual action. The middle column shows the uncontrollable latent $z_t^u \in \mathbb{R}^{6\times6}$ and the right column shows the original state $s_t \in \mathbb{R}^{48\times48}$. Evidently, the controllable representations in (b) and (c) lack disentanglement and interpretability. Furthermore, the representation in (c) seems to have very little structure at all, showing that a representation that is optimized without prior structural incentives will often represent a black box.

pixels. Since we are only doing unsupervised learning, we consider the context where there is no reward ($r_t = 0 \ \forall \ (s_t, a_t) \in (\mathcal{S}, \mathcal{A})$) and an episode ends whenever the ball reaches the paddle or the bottom.

We take $z^c \in \mathbb{R}^2$ and $z^u \in \mathbb{R}^{6\times6}$. To test disentanglement, z^c is of a higher dimension than needed since the paddle (agent) only moves on the x-axis and would therefore require only one feature (see Appendix D.2 for the simpler setting with $z^c \in \mathbb{R}^1$). To show disentanglement, the redundant dimension of z^c should not or negligibly have information about z^u. The encoder's parameters are updated using $\mathcal{L}_{enc}$ in Section 4.4 with $\mathcal{L}_H = \mathcal{L}_{H_1}$. After training the encoder for 200k iterations, a selection of state observations s_t and their encoding into the latent representation $z = (z^c, z^u)$ can be seen in Fig. 3. A clear distinction between the ball and paddle representations can be observed, with the former residing in z^u and the latter in z^c.

5.3 Random Maze Environment

The random maze environment is similar to the maze environment from Section 5.1, but consists of a large distribution of randomly generated mazes with complex wall structures. The environment's state is provided as pixel observations $s_t \in \mathbb{R}^{48\times48}$, where an action moves the agent by 6 pixels in each direction. We consider $z^c \in \mathbb{R}^2$ and $z^u \in \mathbb{R}^{6\times6}$. This environment tests the generalization properties of a disentangled latent representation, as there are over 25 million possible maze architectures, corresponding to a probability of less than $4 \cdot 10^{-8}$ to sample the same maze twice. Note that because z^c is 2-dimensional, results with and without adversarial loss are in practice extremely close. After 50k training iterations, the latent representation $z = (z^c, z^u)$ shows an interpretable disentanglement between the controllable and the uncontrollable features (see Fig. 4a). A clear distinction between the agent and the wall structure can be found inside z^c and z^u. Note that Instead of using a single dimension to 'describe' the uncontrollable features z^u (see Fig. 1), using a feature map for z^u allows training an encoding that provides a more interpretable representation of the actual wall architecture.

Using an Inverse Predictor. An alternative to the state-action forward prediction method used throughout the paper is the inverse (action) prediction loss. An inverse prediction loss is often referred to in previous work that focuses on controllable features (Jonschkowski & Brock, 2015; Pathak et al., 2017; Badia et al., 2020). A single-step inverse prediction loss is defined as:

$$\hat{a}_t = I(z_t^c, z_{t+1}^c, z_t^u; \theta_{inv}). \tag{8}$$

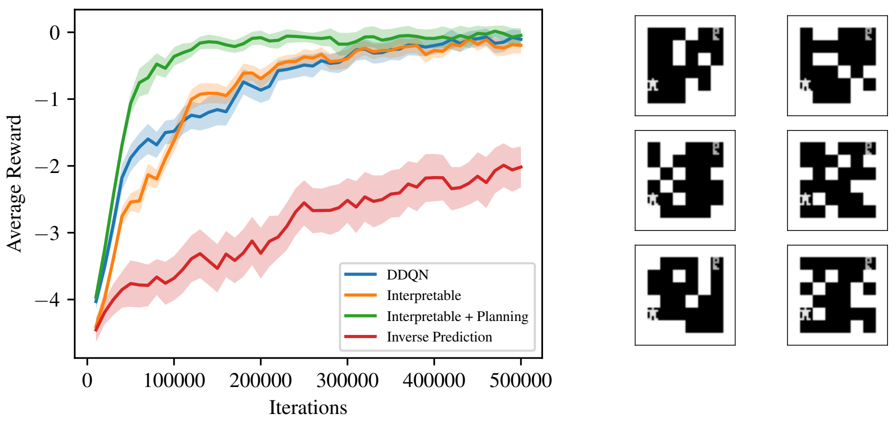


Figure 5: Performance of different (pre)trained representations on the random maze environment, measured as a mean (full line) and standard error (shaded area) over 5 seeds. The 'Interpretable' setting uses an encoder pre-trained with 50k iterations to acquire a representation as in Fig. 4a, after which the encoder is frozen and a Q-network is trained on top with DDQN for 500k iterations. The 'Interpretable + Planning' curve is similar to the 'Interpretable' setting but uses DDQN with a planning algorithm in the controllable partition of the latent space with a depth of 3. The 'DDQN' setting uses an encoder trained end-to-end with only DDQN for 500k iterations and the 'Inverse Prediction' setting is equal to the 'Interpretable' setting but has an encoder pre-trained with $\mathcal{L}_{inv}$ instead of $\mathcal{L}_c$. On the right, a random subset of the vast amount of procedurally generated mazes used in the reward evaluation is shown.

Here, $\hat{a}_t$ is the predicted action and $I(z_t^c, z_{t+1}^c, z_t^u; \theta_{inv}) : \mathcal{Z} \rightarrow \mathcal{A}$ is the inverse prediction network. To see whether an inverse predictor can generate structured, controllable representations in the random maze environment, we replace the action-conditioned forward predictor with an inverse predictor, so that z^c is no longer updated with $\mathcal{L}_c$ but with $\mathcal{L}_{inv}$ (see Appendix A.6 for details on $\mathcal{L}_{inv}$).

The resulting representation can be seen in Fig. 4b. It seems that using $\mathcal{L}_{inv}$, causes an absence of interpretable structure in the controllable latent representation z_t^c. Furthermore, there is a less precise disentanglement between the controllable and uncontrollable features, as differences can be observed in z_t^c when encoding equal agent positions as pixel states s_t. In addition, an inverse predictor does not allow forward prediction in latent space, which can be used for planning as shown hereafter. It thus seems that in some environments, an inverse prediction loss might be insufficient to isolate the controllable features. Take for example the maze agent in the top-right maze of Fig. 4, where the agent can only move in the left direction. Even when using the wall information (z_t^u), an inverse predictor will not be able to predict the action taken when the agent does not go left. However, an action-conditioned forward predictor is able to predict the next state correctly regardless of which action was taken.

Reinforcement Learning. In order to verify whether a human-interpretable disentangled latent encoding is informative enough for downstream tasks, we formalize the random maze environment into an MDP with rewards. The agent acquires a reward r_t of -0.1 at every time step, except when it finds the key in the top right part in which case it acquires a positive reward of 1. The episode ends whenever a positive reward is obtained or a total of 50 environment steps have been taken. For each new episode, a random wall structure is generated, and the agent starts over in the bottom left section of the maze (see Fig. 5). To see whether an interpretable disentangled latent representation is useful for RL, we compare different scenarios of (pre)training; (i) An encoder pretrained for 50k iterations to attain the representation in Fig. 4a and subsequently trained with DDQN for 500k iterations (ii) an encoder identical to the aforementioned but trained with DDQN and a planning algorithm (iii) an encoder pretrained for 50k iterations with $\mathcal{L}_{inv}$ instead of $\mathcal{L}_c$ and subsequently trained with DDQN for 500k iterations (iv) an encoder purely trained with DDQN gradients for 500k iterations. The resulting performances are compared in Fig. 5. We find that a disentangled structured representation is suitable for downstream tasks, as it achieves comparable performance to training an encoder end-to-end with DDQN for 500k iterations. Although performance is similar, Fig. 4c shows that an encoder updated solely with the DDQN gradient can lose any form of interpretability. Moreover, we show in Fig. 5 that a representation trained with an inverse prediction loss instead of a state-action forward prediction loss leads to poor downstream performance in the random maze environment.

Planning. As seen in Fig. 4a, after pre-training with the unsupervised losses, an interpretable disentangled representation with the corresponding agent transitions is obtained. Due to this disentanglement of the controllable and uncontrollable features, we can for instance employ prior knowledge that the uncontrollable features in the maze environment are static, and employ latent planning in the controllable latent space only (see Fig. 6). The planning algorithm used is derived from Oh et al. (2017), and is used to successfully plan only in the controllable partition of the latent representation z^c, while freezing the input for z^u regardless of planning depth. More details on the planning algorithm can be found in Appendix A.5. It can be observed that even when planning with a relatively small depth of 3, we achieve better performance than the pre-trained representation with an ϵ-greedy policy and than the purely DDQN-updated encoder.

6 Limitations

While the work presented here provides a step towards a better understanding of disentangling controllable and uncontrollable features within an encoder architecture, there remain some limitations that we must acknowledge, and which can provide a basis for future research.

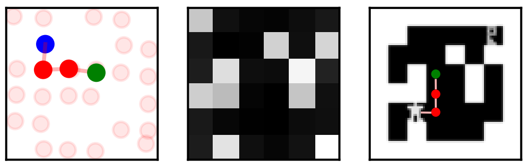
First, our method's effectiveness was predominantly demonstrated on environments with relatively simple underlying dynamics. In these environments, the disentanglement process was easier to achieve due to the limited complexity of internal dynamics present. As we begin to transfer our approach to more complex environments characterized by more extensive internal dynamics, there can arise two problems; The first being that the separation of controllable from uncontrollable features may not be as clear-cut in more complex MDPs, but can be more on a spectrum, complicating the fundamental differences between a state-only and a state-action forward predictor. The second being that interpretability will be harder to enforce when there are a large number of underlying factors of variation. As distinct seeds can give different orderings and signs of the neurons in the final layer of the encoder, identifying a factor of variation can become exponentially harder for more complex environments.

Lastly, while our work showed that an action-conditioned forward predictor could be preferred over an inverse predictor in some environments for isolating controllable features, it may not hold for all scenarios. The inherent properties of different environments might show a necessity of using different predictors. Consequently, there could very well be MDPs where our current approach might not provide the same level of disentanglement showed in the MDPs used in this paper.

Despite these limitations, we believe our work provides a strong foundation upon which future research can build and further extend the possibilities of achieving a highly interpretable latent representation through disentanglement of controllable and uncontrollable features.

7 Conclusion and Future Work

We have shown the possibility of disentangling controllable and uncontrollable features in an encoder architecture, strongly increasing the interpretability of the latent representation while also showing the potential use of this for downstream learning and planning, even in a single latent partition. This disentanglement of controllable and uncontrollable features in the latent representation of high-dimensional MDPs was achieved by propagating an action-conditioned forward prediction loss and a state-only forward prediction loss through distinct sections of the latent representation. Additionally, a contrastive loss and an adversarial loss were used to respectively avoid collapse and further disentangle the latent representation. Furthermore, we showed that an action-conditioned forward predictor can, in some environments, be preferred as compared to an inverse predictor in terms of isolating controllable features in the representation. Finally, by employing forward prediction in latent space, we were able to successfully run a planning algorithm while leveraging the properties of the environment. In particular, the disentanglement of controllable and uncontrollable features allowed us to keep z^u frozen regardless of planning depth in the context of a distribution of randomly generated mazes, i.e. we only do forward prediction in z^c.


(a) Planning depth 3

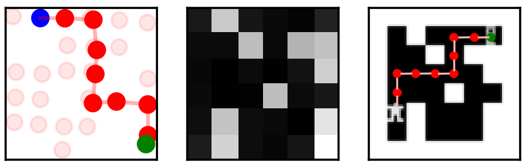

(b) Planning depth 9

Figure 6: Visualization of the latent representation through an actual planning iteration utilizing a planning depth of 3 (a) and a planning depth of 9 (b), with the controllable representation $z^c \in \mathbb{R}^2$ (left), the uncontrollable representation $z^u \in \mathbb{R}^{6\times 6}$ (middle) that is kept static throughout planning depth and the original pixel input $s_t \in \mathbb{R}^{48\times 48}$ (right). The translucent red dots represent every possible encoded state in the random maze environment, the full blue dot represents the current encoded state, the red dots represent intermediate encoded states estimated by planning and the green dot represents the final predicted state as chosen by the planning algorithm, consistent with its depth.

Future work could focus on gradually transferring our notion of disentanglement and interpretability to environments with more extensive underlying internal dynamics. Further work could also look at the ordering of the latent dimensions, as a latent representation is often arbitrarily ordered. This means that distinct seeds will lead to a different ordering and sign of the neurons in the final layer of the encoder. For example, if seed one would give agent position +x and +y for neurons 1 and 2 respectively, then seed two could give agent position -y and +x to the same neurons. As we are additionally using a contrastive loss while learning our representation, these results are compliant with the theory that a contrastive loss can recover the original latent information up to an orthogonal linear transformation (Zimmermann et al., 2021).

Certain benefits can be obtained as well with a particular design of the encoder architecture, as we have done in this paper using estimates of the necessary dimensions of z^c and z^u for the different MDP environments. This can be seen as an inductive bias to aid disentanglement, as mentioned by Locatello et al. (2019). Succeeding work could also focus on finding more algorithmic benefits of this disentanglement of controllable/uncontrollable features in more complex environments. For example, in the context of safety, a disentangled interpretable representation could allow incorporating latent state constraints in a planning algorithm. Lastly, as discussed by Glanois et al. (2021); Locatello et al. (2019), an interesting venue could be to further investigate the trade-off between interpretability and downstream performance. This is due to the fact that black-box representations such as Figure 4c still seem to have excellent downstream performance with DDQN, where for the task of maze navigation, a human would perform substantially better using the representation portrayed in Figure 4a as compared to using the representation in Figure 4c.

References

Kartik Ahuja, Jason Hartford, and Yoshua Bengio. Weakly supervised representation learning with sparse perturbations. In *Advances in Neural Information Processing Systems, NIPS*, 2022.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, and Charles Blundell. Never Give Up: Learning Directed Exploration Strategies. In *International Conference on Learning Representations, ICLR*, 2020.

Richard Bellman. A markovian decision process. In *Journal of Mathematics and Mechanics*, volume 6, 1957.

David Bertoin and Emmanuel Rachelson. Disentanglement by cyclic reconstruction. In *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provable RL with exogenous distractors via multistep inverse dynamics. In *International Conference on Machine Learning, ICML*, 2021.

Vincent Francois-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined Reinforcement Learning via Abstract Representations. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2019.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In *International Conference on Machine Learning, ICML*, 2021.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, Urun Dogan, Marius Kloft, Francesco Orabona, Tatiana Tommasi, and al Ganin. *Domain-Adversarial Training of Neural Networks*. In *Journal of Machine Learning Research*, volume 17, 2016.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning, ICML*, 2019.

Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *arXiv preprint arXiv:2112.13112*, 2021.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems, NIPS*, 2014.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning, ICML*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations, ICLR*, 2021.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. In *International Conference on Machine Learning, ICML*, 2017.

Irina Higgins, David Amos, David Pfau, Sébastien Racanière, Loïc Matthey, Danilo J. Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *International Conference on Learning Representations, ICLR*, 2017.

Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3), 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations, ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations, ICLR*, 2014.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive Learning of Structured World Models. In *International Conference on Learning Representations, ICLR*, 2020.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. In *International Conference on Learning Representations, ICLR*, 2021.

Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Didolkar, Dipendra Misra, Dylan Foster, Lekan Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of controllable latent states with multi-step inverse models. *arXiv preprint arXiv:2207.08229*, 2022.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *37th International Conference on Machine Learning, ICML*, 2020.

Adrien Laversanne-Finot, Alexandre Pere, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. In *Proceedings of The 2nd Conference on Robot Learning*. PMLR, 2018.

Kuang Huei Lee, Ian Fischer, Anthony Z Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in RL. In *Advances in Neural Information Processing Systems, NIPS*, 2020.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th International Conference on Machine Learning, PMLR*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value Prediction Network. In *Advances in Neural Information Processing Systems, NIPS*, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW*, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations. In *International Conference on Learning Representations, ICLR*, 2021.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently Controllable Factors. *arXiv preprint arXiv:1708.01289*, 2017.

Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. In *Advances in Neural Information Processing Systems, NIPS*, 2020.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2016.

Tongzhou Wang, Simon Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised MDPs: Learning world models better than the world itself. In *Proceedings of the 39th International Conference on Machine Learning, PMLR*, 2022.

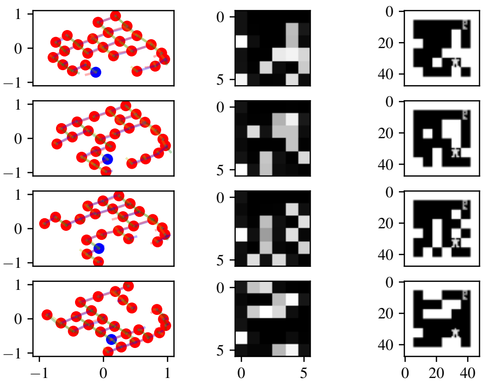
Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving Sample Efficiency in Model-Free Reinforcement Learning from Images. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2021.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12979–12990. PMLR, 18–24 Jul 2021.

A Additional Material

A.1 Ablation of the contrastive scalar

Without using a pixel reconstruction loss, the contrastive loss $\mathcal{L}_H$ is crucial in avoiding the trivial solution for any latent forward predictor (Francois-Lavet et al., 2019; Gelada et al., 2019). The contrastive scalar that regulates the $\mathcal{L}_H$ however remains the most influential hyperparameter. When C_d is chosen too low, the representation remains in a compact cluster. On the other hand, when C_d is chosen too high, unnecessary inter-sample distances are formed to enforce large individual latent distances. Two ablations of the contrastive scalar C_d are shown in Fig. 7.


(a) $C_d = 13$

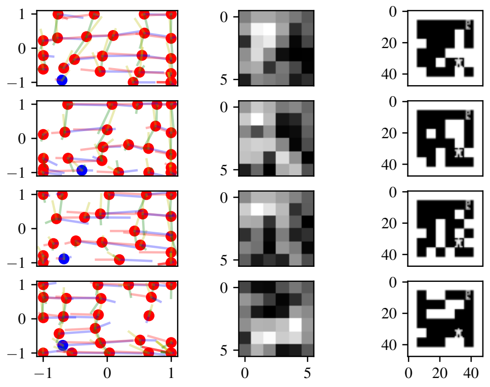
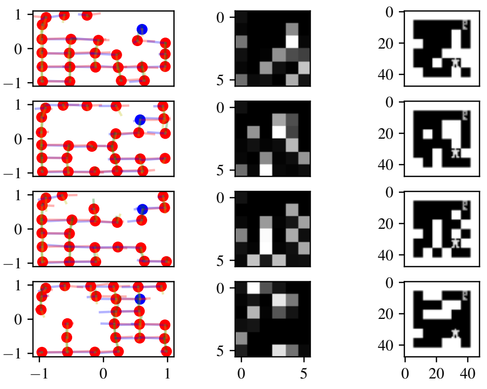

(b) $C_d = 3$

Figure 7: Ablation of the hyperparameter C_d, where a higher value of C_d enforces less entropy in the representation, while a lower value of C_d especially pushes the controllable features z^c towards shapes that ensure large distances between samples. In both figures, the left column is $z^c \in \mathbb{R}^2$, the middle column is $z^u \in \mathbb{R}^{6\times6}$ and the right column is the input state $\in \mathbb{R}^{48\times48}$.

A.2 Ablation of learning rates

We show experiments in Fig. 8 and Fig. 9 where we employ different learning rates for the encoder and the action-conditioned forward predictor, respectively.


(a) Encoder learning rate of 2e-4

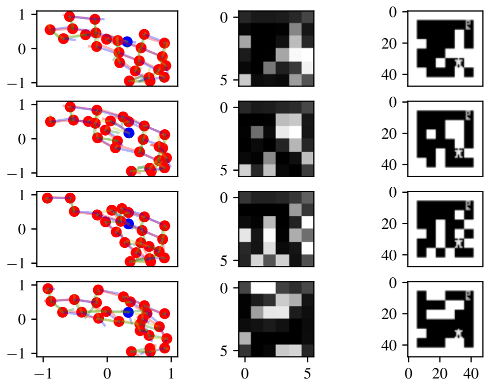
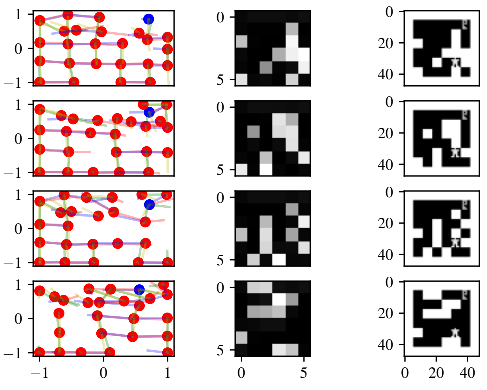
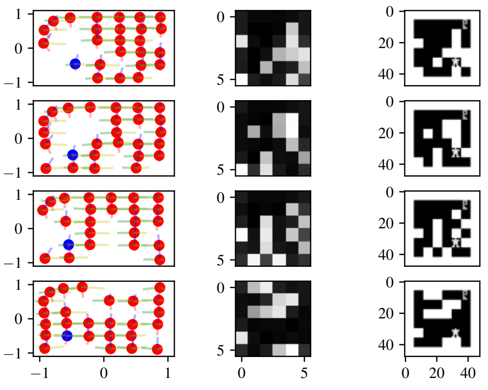

(b) Encoder learning rate of 2e-6

Figure 8: Ablation of the learning rates for the encoder, where a too low learning rate causes collapse of z^c and a too high learning rate causes distortions in the uncontrollable features z^u. In both figures, the left column is $z^c \in \mathbb{R}^2$, the middle column is $z^u \in \mathbb{R}^{6\times6}$ and the right column is the input state $\in \mathbb{R}^{48\times48}$.



(a) T_c learning rate of 4e-4

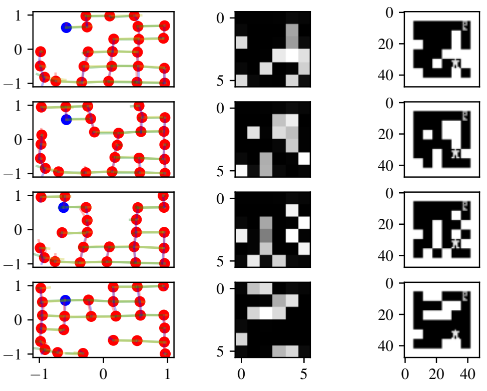


(b) T_c learning rate of 4e-6

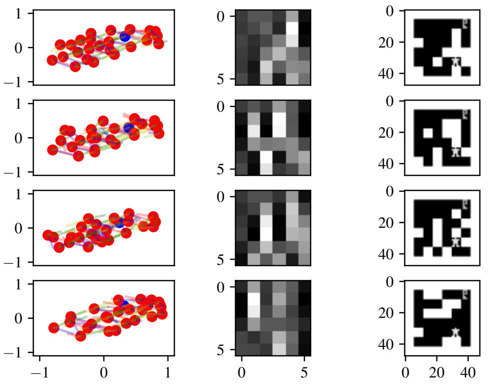
Figure 9: Ablation of the learning rates for the action-conditioned forward predictor. A too high learning rate will cause the controllable representation to lose structure, while a low learning rate retains structure but does not learn strong transition dynamics. In both figures, the left column is $z^c \in \mathbb{R}^2$, the middle column is $z^u \in \mathbb{R}^{6\times 6}$ and the right column is the input state $\in \mathbb{R}^{48\times 48}$.

A.3 Ablation of the detachment of z^u and ablation of the residual prediction

As seen in the main paper in Figure 2, we detach the uncontrollable representation z^c from $\mathcal{L}_c$ as we do not want controllable features to be present in z^u. We can see in Figure 10 that updating z^u with $\mathcal{L}_c$ leads to slightly better transition predictions in z^c, but also results in a less interpretable encoding of z^u. Furthermore, we can also see in Figure 10 that, when using normal forward predictions instead of residual forward predictions, we lose almost all of our interpretable structure in z^u.



(a) No detachment of z^u

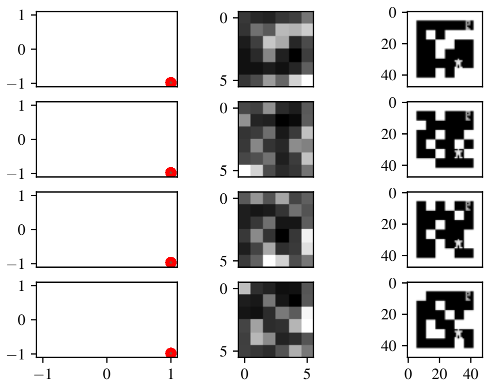


(b) No residual predictions of z^c_{t+1} and z^u_{t+1}

Figure 10: In both figures, the left column is $z^c \in \mathbb{R}^2$, the middle column is $z^u \in \mathbb{R}^{6\times 6}$ and the right column is the input state $\in \mathbb{R}^{48\times 48}$.

A.4 Ablation of the entropy loss $\mathcal{L}_{H2}$

As the amount of possible encoded maze architectures goes to infinity due to the procedural generation, a collapse in the controllable features z^c can be noticed when using only $\mathcal{L}_{H1}$ as the contrastive loss (see Fig. 11). On the other hand, when using only $\mathcal{L}_{H2}$ as the contrastive loss, there is no more clear distinction in the uncontrollable representation z^u. The best results were obtained using a combination of the aforementioned losses.


(a) $\mathcal{L}_H = \mathcal{L}_{H1}$

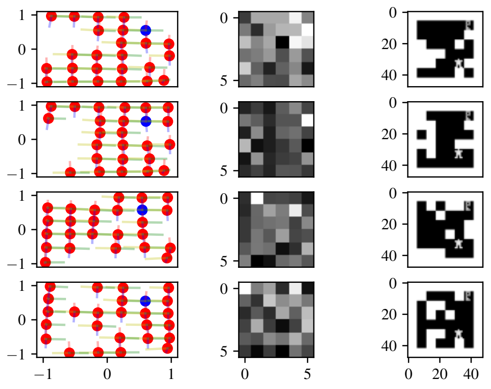

(b) $\mathcal{L}_H = \mathcal{L}_{H2}$

Figure 11: In both figures, the left column is $z^c \in \mathbb{R}^2$, the middle column is $z^u \in \mathbb{R}^{6\times 6}$ and the right column is the input state $\in \mathbb{R}^{48\times 48}$.

A.5 Planning

We use a planning algorithm derived from (Oh et al., 2017; Francois-Lavet et al., 2019), where we employ d-step planning as:

$$\hat{Q}^d((\hat{z}^c_t, z^u), a) = \begin{cases} P((\hat{z}^c_t, z^u), a; \theta_r) + \Gamma((\hat{z}^c_t, z^u), a; \theta_\gamma) \max\limits_{a' \in \mathcal{A}^*} \hat{Q}^{d-1}((\hat{z}^c_{t+1}, z^u), a'), & \text{if } d > 0 \\ Q((\hat{z}^c_t, z^u), a; \theta), & \text{if } d = 0 \end{cases} \tag{9}$$

$$Q^D_{plan}((\hat{z}^c_t, z^u), a) = \sum_{d=0}^{D} \hat{Q}^d((\hat{z}^c_t, z^u), a) \tag{10}$$

Where $P(s_t, a; \theta_r) : \mathcal{Z} \times \mathcal{A} \to \mathcal{R}$ represents the reward predictor and $\Gamma(s, a; \theta_\gamma) : \mathcal{Z} \times \mathcal{A} \to \gamma$ represents the discount value predictor. The action is chosen by taking the argmax of $Q^D_{plan}((\hat{z}^c_t, z^u), a)$. Note in the results from Section 5.3, we are only forward predicting in the controllable latent space z^c, and that z^u remains a fixed value regardless of planning depth. This is possible by making use of the prior knowledge of the maze environments together with a disentangled controllable and uncontrollable latent representation.

A.6 Inverse Prediction

A common single-step inverse prediction is defined as:

$$\hat{a}_t = f(s_t, s_{t+1}) \tag{11}$$

where $\hat{a}_t$ is the predicted action and $f(s_t, s_{t+1})$ represents an arbitrarily structured function. In the random maze environment, we use a parameterized inverse predictor which predicts in latent space:

$$\hat{a}_t = I(z^c_t, z^c_{t+1}, z^u_t, z^u_{t+1}; \theta_{inv}) \tag{12}$$

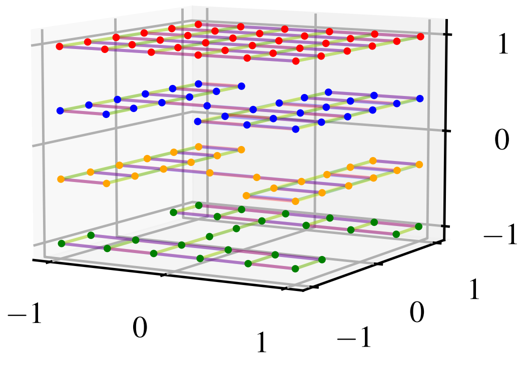
Where $I(\cdot; \theta_{inv}) \in \mathcal{I} : \mathcal{Z} \to \mathcal{A}$ is a parameterized inverse prediction function. Since we have 4 actions, we use the 4-dimensional logit output $\hat{a}_t$ to calculate the inverse prediction loss $\mathcal{L}_{inv}$ as:

$$S(\hat{a}_i) = \frac{\exp(\hat{a}_i)}{\sum_{j=1}^{n_a} \exp(\hat{a}_j)}, \quad \mathcal{L}_{inv} = -\sum_{i=1}^{n_a} a_i \log(S(\hat{a}_i)) \tag{13}$$

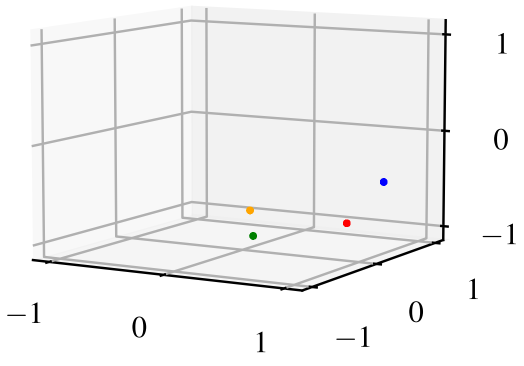
Here, n_a is the number of actions, $S(\hat{a}_i)$ represents the softmax operator and a_i is the actual action, given as a 0 or 1 truth label. This is more commonly known as the Cross-Entropy loss computation.

A.7 Reconstruction

We run an additional ablation on the four mazes environment, where the contrastive loss $\mathcal{L}_H$ is replaced with a pixel reconstruction loss. The resulting representation comparison can be seen in Fig. 12.



(a) Contrastive loss

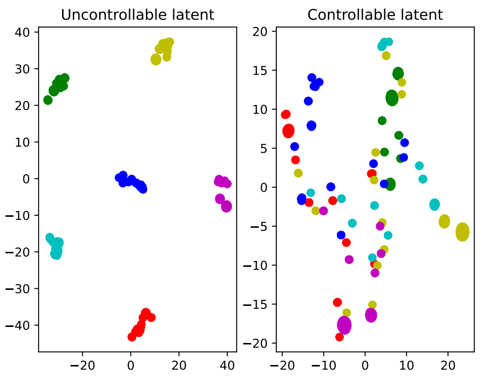


(b) Reconstruction loss

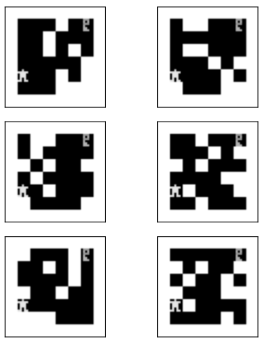
Figure 12: Visualization in a maze environment of the disentanglement of the controllable latent $z^c \in \mathbb{R}^2$ on the horizontal axes, and the uncontrollable latent $z^u \in \mathbb{R}^1$ on the vertical axis, given for all states in the four maze environments shown in four different colors. The representation is trained on high-dimensional tuples (s_t, a_t, r_t, s_{t+1}), sampled from a replay buffer $\mathcal{B}$, gathered from random trajectories in the four maze environments. All possible states are encoded with $z_t = f(s_t; \theta_{enc})$ and plotted in (a) and (b) together with the transition prediction for each possible action. In (a), a clear disentanglement between the controllable agent's position and the uncontrollable wall architecture is portrayed. In (b), it seems that a reconstruction loss groups observations with similar pixel inputs together, and thus allows the forward predictors to 'collapse' to unit matrices, decreasing representation quality.

A.8 T-SNE

We conduct an additional experiment in the random maze environment where we use a latent dimension of 32, partition it in half to form $z^c \in \mathbb{R}^{16}$ and $z^u \in \mathbb{R}^{16}$ and show the a T-SNE visualization of 6 different trajectories in random mazes in Fig. 13. Note that, because the trajectories are random, only a subpart of the possible agent positions in every random maze is present.



(a) T-SNE of z^u and z^c



(b) 6 random mazes

Figure 13: Ablation of a dimensionality increase in our random maze environment. Here, the total latent space is a 32-dimensional MLP output, where z^c and z^u are both 16-dimensional. in (a), 6 random trajectories are plotted using T-SNE (perplexity=20) in different colors for both z^c and z^u, where z^u remains similar across a trajectory (same wall architecture), and z^c differs across the trajectory (different agent positions). In (b), a collection of the random mazes are shown from which the random trajectories have been taken.

B Experiment details

The Pytorch framework was used for all experiments, as well as the Adam optimizer (Kingma & Ba, 2015). We employ a batch size of 32 tuples (s_t, a_t, r_t, s_{t+1}) for every update. In all experiments, we detach z_t^c in the calculation of $\mathcal{L}_c$, as it allowed us to use a larger learning rate for T_c without causing instabilities.

Simple Maze The replay buffer $\mathcal{B}$ is filled with 5k transitions from each of the four wall architectures. The transitions are collected by the agent following a random policy. The learning rate for the encoder is $5 \cdot 10^{-5}$, for the action-conditioned forward predictor $1 \cdot 10^{-3}$ and for the uncontrollable forward predictor $5 \cdot 10^{-5}$. The contrastive scalar C_d is set to 15.

Catcher The replay buffer $\mathcal{B}$ is filled with 25k transitions. The transitions are collected by the agent following a random policy. A new random maze is created after 50 time steps or when the reward is acquired. The learning rate for the encoder is $2 \cdot 10^{-5}$, for the action-conditioned forward predictor $4 \cdot 10^{-5}$ and for the uncontrollable forward predictor $1 \cdot 10^{-5}$. When using the adversarial loss, we use a learning rate of $1 \cdot 10^{-3}$ for the adversarial predictor. The contrastive scalar C_d is set to 5.

Random Maze The replay buffer $\mathcal{B}$ is filled with 50k transitions, representing around 1000 maze architectures. The transitions are collected by the agent following a random policy. The learning rates used are equal to those of the catcher environment; for the encoder $2 \cdot 10^{-5}$, for the action-conditioned forward predictor $4 \cdot 10^{-5}$ and for the uncontrollable forward predictor $1 \cdot 10^{-5}$. After freezing the encoder, we train the action-conditioned forward predictor for an additional 250k iterations on the same 50k transitions in the buffer $\mathcal{B}$. For updating the Q-network with DDQN, we use a learning rate of $1 \cdot 10^{-4}$, and a τ of 0.02. The contrastive scalar C_d is set to 13. When using planning, we employ a learning rate of $5 \cdot 10^{-5}$ for the reward and discount prediction networks.

Contrastive Loss For the catcher and random maze environment, given that z^c is 1 or 2-dimensional, and z^u is a 36-dimensional feature map, we alleviate dimensional mismatch when calculating the contrastive loss in Equation 4 in the main paper. This is done by taking a random subset of 15 out of 36 feature values in z^u for every batch.

C Network Architecture

We use the same base encoder for all experiments, made up of 2 convolutional layers of 32 channels each, with a kernel size of 3 and stride 2, except for the final layer which has stride 1. Both convolutional layers have a Rectified Linear Unit (ReLU) nonlinear activation.

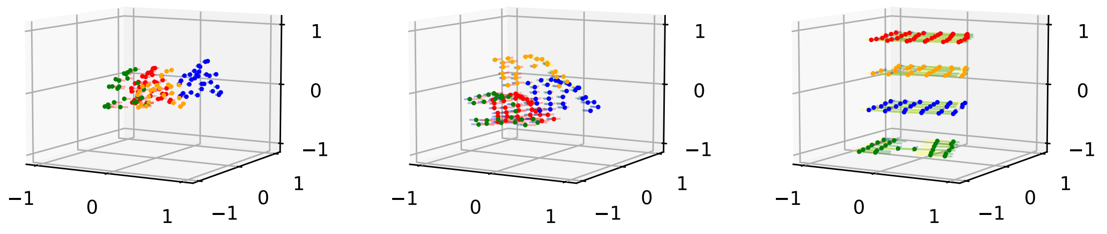
In the quadruple maze environment, the output of the base convolutional encoder is flattened and used as an input to a single linear layer with 3 outputs ($z^c + z^u$) and a hyperbolic tangent (tanh) activation function.

In the catcher and random maze environments, we use the following encoder head to extract the uncontrollable features; the base convolutional layers are followed by a single convolutional layer with 32 channels, a kernel size of 4 and a stride of 1. This layer is followed by a ReLU activation function and an AveragePool layer with an output size of 6. For the controllable features, we flatten the output of the base convolutional encoder and use this as an input to a linear layer with 200 neurons and a tanh activation function. This layer is followed by another linear layer with n_c neurons and a tanh activation function.

The transition and prediction models all have the same structure, with linear layers of 32-128-128-32-x neurons where x is the output dimension in line with the predicted feature's dimension. The linear layers all have tanh activation functions except for the final output. Only the action-conditioned transition predictor of the random maze environment has larger layer sizes, with linear layers of 128-512-512-128-2, to account for slightly more complicated transitions. The DQN network used is of size 128-512-512-128-4, with an output value corresponding to each possible action.

D Additional Figures

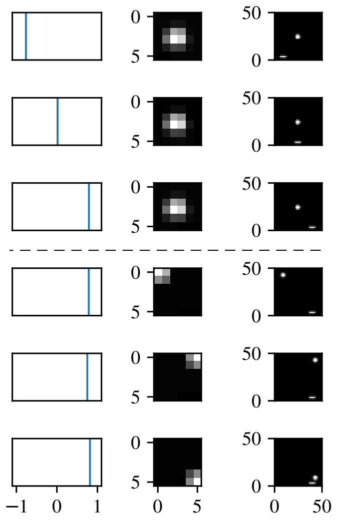
D.1 Quadruple Maze Progression



(a) 1k iterations (b) 2k iterations (c) 5k iterations

Figure 14: Progression of the separation of the controllable z^c (x and y-axis) and uncontrollable z^u (z-axis) features in the maze environment.

D.2 Catcher


(a) $z^c \in \mathbb{R}^1$ without $\mathcal{L}_{adv}$

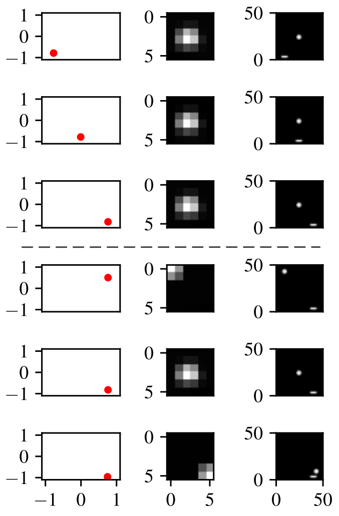

(b) $z^c \in \mathbb{R}^2$ without $\mathcal{L}_{adv}$

Figure 15: Comparison of training the representation for the catcher environment with either 1 or 2-dimensions for the controllable representation z^c. When using more dimensions for z^c than needed, it can be observed that some information of the ball position can be present in z^c.