UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Discovery and Learning of Navigation Goals from Pixels in Minecraft

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Juan José Nieto Salas

In partial fulfillment
of the requirements for the master in
Advanced Telecommunications Technologies **ENGINEERING**

Advisors:
Xavier Giró Nieto
Víctor Campos
Barcelona, Date 20/05/2021

Contents

List of Figures

Listings

List of Tables

Abstract

Pre-training Reinforcement Learning (RL) agents in a task-agnostic manner has shown promising results. However, previous works still struggle to learn and discover meaningful skills in high-dimensional state-spaces. We approach the problem by leveraging unsupervised skill discovery and self-supervised learning of state representations. In our work, we learn a compact latent representation by making use of variational or contrastive techniques. We demonstrate that both allow learning a set of basic navigation skills by maximizing an information theoretic objective. We assess our method in Minecraft 3D maps with different complexities. Our results show that representations and conditioned policies learned from pixels are enough for toy examples, but do not scale to realistic and complex maps. We also explore alternative rewards and input observations to overcome these limitations. The code is publicly available at `https://github.com/juanjo3ns/mineRL`.

1 Introduction

Reinforcement Learning (RL) [51] has witnessed a wave of outstanding works in the last decade, the ones that received more interest being in games (Schrittwieser et al. [48], Vinyals et al. [55], Berner et al. [9]), but also in robotics (Akkaya et al. [4], Hwangbo et al. [33]). In general, these works follow the classic RL paradigm where an agent interacts with an environment performing some action, and in response it receives a reward. These agents are optimized to maximize the expected sum of future rewards. Rewards are usually handcrafted or overparametrized. This results in a bottleneck that prevents RL to scale. This is why during the last few years there has been an increasing interest in training agents in a task-agnostic manner, making use of intrinsic motivations and unsupervised techniques.

In this paradigm we find plenty of recent works (Campos et al. [14], Gregor et al. [27], Eysenbach et al. [22], Warde-Farley et al. [56], Burda et al. [13], Pathak et al. [41], etc). However, we are still far from the remarkable results obtained in other domains. For instance, in computer vision, Chen et al. [18] achieves an 81% accuracy on ImageNet training in a self-supervised manner, or Caron et al. [16] achieves state-of-the-art results in image and video object segmentation using Visual Transformers [21] and no labels at all. Also in Natural Language Processing where the popular language model GPT-3 [12] was pre-trained in a task-agnostic way.

Humans and animals are sometimes guided through the process of learning. Even though, most of the time and specially at the beginning of our lives, we learn by exploration and intrinsic curiosity. We have good priors that allow us to properly explore our surroundings, which leads to discovering new skills. Learning skills in a task-agnostic manner has proved to be challenging for machines, specially from pixel-based observations [22, 56, 37]. As these works state, it is not efficient to train a pixel-based RL agents end-to-end. The reason behind is that the problem of learning a good state representation is exacerbated by the high dimensionality of the observations. Moreover, most of the successes in RL come from training agents for thousands of simulated years (Berner et al. [9]) or millions of games (Vinyals et al. [55]). This way of learning is very inefficient and from a more practical viewpoint, it constraints who can research on these topics depending on their budget. Due to these limitations, competitions have arisen, such as MineRL (Guss et al. [28]) or ProcGen Benchmark (Cobbe et al. [20]) where they promote the development of algorithms that can reduce the number of samples needed to solve complex tasks. In addition, they also encourage generalization in hierarchical and sparse environments using human priors and demonstrations. This also relates to scaling current methods that work well in structured domains, such as board games, to unstructured domains like biology, robotics or finance. In these fields, the aim is not just to optimize a single objective, rather learn a set of skills that allows these agents to survive, help in different tasks, or, broadly speaking, optimize multiple objectives at once and quickly adapt to new tasks.

Our work is inspired from Campos et al. [14] with their *Explore, Discover and Learn* (EDL) paradigm. EDL relies on *empowerment* [44] for motivating an agent intrinsically. Empowerment aims to maximize the influence in the environment while discovering novel skills. As stated by Salge et al. [44], this can be achieved my maximizing the mutual information

between sequences of actions and final states. Gregor et al. [27] introduces a novel approach that instead of blindly committing to a sequence of actions, each action depends on the observation from the environment. This is achieved by maximizing the mutual information between inputs and some latent variables. Campos et al. [14] embraces this approach as we do. The implementation by Campos et al. [14] makes some assumptions that are not realistic for pixel observations. Due to the Gaussian assumption at the output of variational approaches, we rely on Mean Square Error (MSE) for computing a distance in the state space. This metric does not necessarily match the distance in the environment space. Therefore, we look for alternatives that suit our requirements. One by deriving a different reward from the mutual information, or by studying alternatives to the variational approach.

One of the key challenges for unsupervised skill discovery methods is exploration (as showed by Campos et al. [14]). Moreover, information-theoretic methods have difficulties capturing human priors (Achiam et al. [2]). By using a dataset of human play, one can address both issues at the same time, and it has been used in e.g. robotics (Lynch et al. [38], Learning from Play Data).

Motivated by the mentioned problems, we focus our research on learning meaningful representations, discovering skills and training latent-conditioned policies. In all cases, with no supervision and directly from pixel observations. To develop our idea we adopt the MineRL [28] environment, which is based on the popular Minecraft videogame. Although the game proposes a final goal, Minecraft is well known by the freedom it gives to the players to do whatever they want. The players control an embodied agent that interacts and builds structures in a procedurally[1] generated map. We decided to assess our method in this environment for the following reasons: a) It is an embodied domain with an egocentric view of the 3D environment, similar to the one that real-world robots observe, b) contains several datasets of human-generated trajectories available for different tasks, and c) Has garnered great research interest among the community.

Moreover, these human-generated trajectories provide extra information about the environment. One can leverage the temporal information given by the sequence of observations in order to learn more useful latent representations. For this reason, we also study contrastive alternatives to maximize the mutual information between inputs and latents. The intuition behind this approach is that we want to learn an embedding space where observations close in time are also close in the embedding space. A similar result can be achieved by leveraging the agents' relative position in the form of coordinates. The aim is to infer skills that do not fully rely on pixel resemblance, but also takes into account temporal or spatial relationships. In the literature, the common practice is to use variational techniques for learning the mappings from inputs to latents and backwards. We perform a comparison between the contrastive methods and the standard variational objectives.

Our final goal is to discover and learn skills that can be potentially used in more broad and complex tasks. Either by transferring the policy knowledge or by using hierarchical approaches. Some works have already assessed this idea specially in robotics [23] or 2D games [15]. Once the pre-training stage is completed and the agent has learned some

[1]The map is generated by an algorithm.

basic behaviours or skills, the agent is exposed to an extrinsic reward. These works show how the agents leverage the skill knowledge to train much faster and encourage proper exploration of the environment. However, transferring the policy knowledge it is not as straightforward as in other deep learning tasks. If you want to transfer behaviours, the change in the task might lead to catastrophic forgetting.

Our contributions are the following:

- We empirically demonstrate that expert trajectories are sufficient for discovering skills that generalize across simple handcrafted maps.
- We provide alternatives for discovering and learning skills in procedurally generated maps by leveraging the agents' coordinate information.
- We demonstrate that variational techniques are not the only ones capable of maximizing the mutual information between inputs and latent variables by leveraging contrastive techniques.
- We succesfully implemented the reverse form of the mutual information for optimizing pixel-based agents in a complex 3D environment.

This master thesis is organized as follows. First, in Section 2 we review the state-of-the-art of the topics of our concern. Section 3 introduces the main topics of this thesis, such as, skill discovery, mutual information or contrastive learning. Once the main concepts are explained, we proceed to explain our methodology in Section 4. It consists of three subsections, each of them covering one phase of our method. Then, in Section 5 we show our experiments from the toy examples up to the deployment of our agents in realistic maps. Within this section, we also explain how we evaluate some of the results and we also explore some alternatives to the difficulties we find. Finally, we conclude with a final discussion in Section 6. The appendices contain additional results and two extended abstracts submitted to the Embodied AI Workshop of the CVF/IEEE Conference in Computer Vision 2021, one as the main author, and another one as second author. This later work presents the concurrent work of Roger Creus, undergraduate student at UPC whose bachelor thesis I co-advised during the development of this master thesis.

2 Related Work

Intrinsic Motivations are a very helpful mechanism to deal with sparse rewards. In some environments the extrinsic rewards are very difficult to obtain and, therefore, the agent does not receive any feedback to progress. In order to drive the learning process, we can derive intrinsic motivations in form of intrinsic rewards that will guide the agent towards the extrinsic reward or just towards better exploration.

Skill Discovery. Within the Intrinsic Motivations we can find the concept of Empowerment [44]. In this paradigm, the agent looks for the states where it has more control over the environment. Mohamed and Rezende [39] derived a variational lower bound on the mutual information that allows to maximize empowerment. Skill discovery extends this idea from the action level to temporally-extended actions. Florensa et al. [23] merges skill discovery and hierarchical architectures. They learn a high-level policy on top of some basic skills learned in a task-agnostic way. They show how this set-up improves the exploration and allows for faster training in downstream tasks. Similarly, Achiam et al. [2] emphasize in learning the skills dynamically using a curriculum learning approach, allowing the method to learn up to a hundred of skills. Instead of maximizing the mutual information between states and skills they use skills and whole trajectories. Eysenbach et al. [22] demonstrates that learned skills can serve as an effective pretraining mechanism for robotics. Our work follows their approach regarding the use of a categorical and uniform prior over the latent variables. Campos et al. [14] exposes the lack of coverage of previous works. They propose *Explore, Discover and Learn* (EDL), a method for skill discovery that breaks the dependency on the distributions induced by the policy. Warde-Farley et al. [56] provides an algorithm for learning goal-conditioned policies using an imitator and a teacher. They demonstrate the effectiveness of their approach in pixel-based environments like Atari, DeepMind Control Suite and DeepMind Lab.

Intrinsic curiosity. In a more broad spectrum we find methods that leverage intrinsic rewards that encourage exploratory behaviours. Pathak et al. [41] present an *Intrinsic Curiosity Module* (ICM) that defines the curiosity or reward as the error predicting the consequence of its own actions in a visual feature space. Similarly, Burda et al. [13] uses a Siamese network where one of the encoder tries to predict the output of the other. The bonus reward is computed as the error between the prediction and the random one. Savinov et al. [45] introduces the concept of reachability as the number of steps needed to reach the current observation from those in a memory buffer. This buffer is filled with the observations embeddings whose novelty is above a certain threshold. In the end, this threshold introduces a discretization of the embedding space similar to our VQ-VAE codebook. The intrinsic reward in this case is proportional to the reachability of an observation. In all works, the agent is encouraged to explore states that has barely seen before.

Goal-oriented RL. Many of the works dealing with skill-discovery end up parameterizing a policy. This policy is usually conditioned in some goal $g \sim \mathcal{G}$ or latent variable $z \sim Z$. The goal-conditioned policy formulation along with function approximation was introduced by Schaul et al. [47]. Andrychowicz et al. [5] presents *Hindsight Experience Replay*. HER allows sample-efficient learning from environments with sparse rewards. They

assume that any state can be a goal state. Therefore, leveraging this idea, the agent learns from failed trajectories as if the final state achieved was the goal state. Trott et al. [52] proposes to use pairs of trajectories that encourage progress towards a goal while learns to avoid local optima. Among different environments, *Sibling Rivalry* is evaluated in a 3D construction task in Minecraft. The goal of the agent is to build a structure by placing and removing blocks. They use the number of block-wise differences as a naive shaped reward which causes the agent to avoid placing blocks, simply by adding Sibling Rivalry they manage to improve the degree of construction accuracy. Goal-oriented works have two main drawbacks: First, it is not straightforward to implement a hierarchical policy as it is with skill-based strategies. And second, the goal search space is as big as the state-space which can be intractable in some environments with high-dimensionality.

Representation Learning in RL. In RL we seek for low-dimensional states that contain all the information needed in order to make decisions and maximize reward. This becomes crucial when dealing with pixel-based environments, where self-supervised methods are receiving more attention. All the following works learn representations from pixel-based observations, except for the first one. Ghosh et al. [26] aims to capture those factors of variation that are important for decision making and are aware of the dynamics of the environment without the need for explicit reconstruction. The method consists of a goal-conditioned policy that learns the shortest path between arbitrary pairs of states allowing a deep understanding of the environment dynamics. Lee et al. [37] also makes use of variational inference techniques for estimating the posterior distribution, but breaks the Markovian assumption by conditioning the probability of the latent variables with past observations. Oord et al. [40] leverages powerful autoregressive models and negative sampling to learn a latent space from high-dimensional data. This work introduces the InfoNCE loss based on Noise Contrastive Estimation. The intuition behind is that we learn a latent space that allows to classify correctly positive pairs while discriminating from negative samples. We make use of this loss in our contrastive experiments, as well as the following works explained. Laskin et al. [36] trains an end-to-end model by performing off-policy learning on top of extracted features. This features are computed using a contrastive approach based on pairs of augmented observations, while Stooke et al. [50] picks pairs of delayed observations.

3 Information-theoretic skill discovery

Intrinsic Motivations (IM) drive the agents learning process in the absence of extrinsic reward. The agent does not receive any feedback from the environment, but must autonomously learn the possibilities available in it. To achieve that, the agent will seek to gain resources and influence on what can be done in the environment. In the *empowerment* [44] framework, an agent will look for the state in which has the most control over the environment. This concept usually deals with simple actions. In contrast, *skill-discovery* temporally abstracts these simple actions to create high-level actions that are dubbed *skills* or *options*. While empowerment can be formulated as maximizing the mutual information between inputs and actions, skill-discovery instead, is formulated as maximizing the mutual information between inputs and skills. This encourages the agent to learn skills that derive as many different input sequences as possible, while avoiding overlap between sequences guided by different skills. Therefore, it can ease the exploration in complex downstream tasks. Instead of executing random actions, the agent can take advantage of the learned behaviours in order to perform smarter moves towards states with potential extrinsic rewards.

In the next section we formulate the mathematical framework that is used through this work. First we define the classic Markov decision process that typically provide a mathematical formulation for reinforcement learning tasks. Later on, we introduce the concepts from information-theory that allows us to define the maximization of the mutual information.

3.1 Preliminaries

Let us consider a Markov decision process (MDP) as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$. Where $\mathcal{S}$ is the high-dimensional state space (pixel images), $\mathcal{A}$ refers to the set of actions available in the environment and $\mathcal{P}$ defines the transition probability $p(s_{t+1}|s_t, a)$. We learn latent-conditioned policies $\pi(a|s, z)$, where the latent $z \in \mathcal{Z}$ is a random variable.

Given the property of symmetry, the mutual information ($\mathcal{I}$) can be defined using the Shannon Entropy ($\mathcal{H}$) in two ways (Gregor et al. [27]):

$$\begin{aligned} \mathcal{I}(S, Z) &= \mathcal{H}(Z) - \mathcal{H}(Z|S) \quad \rightarrow \quad \text{reverse} \\ &= \mathcal{H}(S) - \mathcal{H}(S|Z) \quad \rightarrow \quad \text{forward} \end{aligned} \tag{1}$$

For instance, we derive the reverse form of the mutual information (MI):

$$\mathcal{I}(S, Z) = \mathbb{E}_{s,z\sim p(z,s)}[\log p(z|s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)] \tag{2}$$

The posterior $p(z|s)$ is unknown due to the complexity to marginalize the evidence $p(s) = \int p(s|z)p(z)dz$. Hence, we approximate it with $q_\phi(z|s)$ by using variational inference or contrastive techniques.

3.1.1 Variational Inference

Variational inference is a well-known method for modelling posterior distributions. Its main advantage is that avoids computing the marginal $p(s)$, which is usually impossible. Instead, one selects some tractable families of distributions q as an approximation of p.

$$p(z|s) \approx q(z|\theta) \tag{3}$$

We fit q with sample data to learn the distribution parameters θ. In particular, we make use of Variational Auto-Encoders (VAE) (Kingma and Welling [34]) for modelling the mappings from inputs to latents and backwards. The encoder $q_\phi(z|s)$ models $p(z|s)$ and the decoder $q_\psi(s|z)$ models $p(s|z)$. As a result, we aim at maximizing a lower bound based on the KL divergence. For a detailed derivation we refer the reader to the original work from Mohamed and Rezende [39].

Therefore, the MI lower bound becomes:

$$\mathcal{I}(S, Z) \geq \mathbb{E}_{s,z\sim p(z,s)}[\log q_\phi(z|s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)] \tag{4}$$

The prior $p(z)$, can either be learned through the optimization process [27], or fixed uniformly beforehand [22]. In our case we fix it to a uniform distribution and hence, we omit it from the equation.

3.1.2 Contrastive Learning

Contrastive learning is a subtype of self-supervised learning that consists in learning representations from the data by comparison. This field has quickly evolved in the recent years, especially in computer vision. In that domain, these comparisons are between pairs of images. Examples of positive pairs may be augmented versions of a given image like crops, rotations, color transformations, etc; and the negative pairs maye be the other images from the dataset. In this case, representations are learned by training deep neural networks to distinguish between positive or negative pairs of observations. The main diference between contrastive self-supervised (or unsupervised) learning and metric learning is that the former does not require any human annotation, while the later does. Learning representations with no need of labeling data allows scaling up the process and overcoming the main bottleneck when training deep neural networks: the scarcity of supervisory signals.

Similar to what is done in variational techniques, we can compute the mutual information between inputs and latents. In this case, instead of learning the latents by the reconstruction error, they are learned by modeling global features between positive and negative pairs of images. Intuitively, two distinct augmentations (positive pair) should be closer in the embedding space than two distinct images (negative pair) from the dataset. Therefore, we need two encoders in paralel instead of an encoder-decoder architecture. The second encoder is usually called momentum encoder (Chen et al. [18]) and its weights are updated using exponential moving averages of the main encoder.

In each training step, we forward a batch of different original images through the main encoder while we forward the positive pairs of each of those images through the momentum encoder. In a batch of N samples, we have for each positive pair, $N-1$ negative pairs. We define z as the output of the encoder and z' as the output of the momentum encoder. Then, e is a convolutional neural encoder and h is a projection head (small multi-layer perceptron (MLP)) that returns a latent z.

$$z = h(e_\theta(s))$$
$$z' = h'(e_{\theta'}(s'))$$

At the output of the network we can perform a categorical cross-entropy loss where the correct classes are the positive pairs in the batch. The correct classes are in the diagonal of the resulting matrix $z^T W z'$, where W is a projection matrix learned during training. In the other positions, we have the similarity between negative pairs in the embedding space. Minimizing this loss, whose name is InfoNCE [40], will encourage the model to find global features that can be found in augmented versions from an image. Moreover, as stated by their authors Oord et al. [40], minimizing the InfoNCE Eq. 5 loss maximizes the mutual information between inputs and latents. As a result, we learn the desired mapping from input states s to latent vectors z.

$$\mathcal{L}_{InfoNCE} = \log \frac{\exp(z^T W z')}{\exp(z^T W z') + \sum_{i=0}^{K-1} \exp(z^T W z'_i)} \tag{5}$$

3.2 Self-supervised perspective

Both approaches explained in the previous section can be interpreted from a self-supervised perspective. In this work, we encourage the agent to learn behaviours in a task-agnostic way. This can be done from the raw pixel states given by the environment, but this approach is very inefficient. As Böhmer et al. [10] states, a good state representation should be markovian, should represent the true value of the policy and should be low-dimensional. In general, we cannot infer meaningful states representations from rewards, specially if the agent is deployed in a sparse environment. In this case, the agent would not receive enough feedback to derive generic features that encode the dynamics of the environment. Moreover, since the feedback would be task-dependent, the learned features might not generalize to other tasks. For these reasons, the representation learning and the agent learning process are usually decoupled (Lee et al. [37]). As mentioned at the beginning, we should also aim for low-dimensional states. Therefore, learning from high-dimensional images is not very efficient. In this work we study different self-supervised methods for learning representations from pixels without the need of any label at all.

4 Methodology

In this section we describe how we tackle the considered problems in this thesis. In particular, we show how we manage to induce human-priors to our state distribution and we also explore alternatives to the forward form of the MI which is not suitable for pixel-based environments. Moreover, we show the implementation details adopted for each of the stages of our method.

4.1 Exploration

EDL (Explore, Discover and Learn) [14] provides empirical and theoretical analysis of the lack of coverage of some methods leveraging the mutual information for discovering skills. Either by using the forward or the reverse form of the MI, the reward given to novel states is always smaller than the one given to known states. The main problem dwells in the induced state distributions used to maximize the mutual information. Most works reinforce already discovered behaviours since they induce the state distribution from a random policy, $p(s) \approx \rho_\pi(s) = \mathbb{E}_z[\rho_\pi(s|z)]$. EDL is agnostic to how $p(s)$ is obtained, so we can induce it from the dataset of expert trajectories provided by Guss et al. [28]. As Achiam et al. [2] states, information-theoretic objectives might do a poor job at capturing human priors in complex environments. Having a dataset of expert trajectories allows us to induce a much richer state distribution that encodes the states that are more commonly visited by real players. But this does not solve the exploratory issues we might find. Therefore, we limit the possible states available for the agent by setting some boundaries in the maps.

4.2 Skill-Discovery

In Section 3, we proposed two distinct approaches for modelling the mapping from the observations s to the latent variables z: variational inference and contrastinng learning. In this section we present the implementations and the pipelines we followed to train these models.

4.2.1 Variational Inference

Vector-Quantized VAE (VQ-VAE) [53] is a variational model that allows us to have a categorical distribution $p(z)$ as a bottleneck, an encoder q_ψ that estimates the posterior $p(z|s)$, and the decoder q_ϕ that estimates $p(s|z)$.

The loss function in the VQ-VAE [53] is defined as:

$$\mathcal{L} = \log p(s|z_q(s)) + \beta||z_e(s) - sg[e]||_2^2 \tag{6}$$

Where e is the codebook of embeddings, z_e is the encoded input observation, z_q is the quantized z_e, sg means *stop gradient* and β is a hyperparameter that weights the second term contribution to the loss. The first term of the loss function belongs to the reconstruction error and it is optimized through the decoder. And, the second, ensures that the

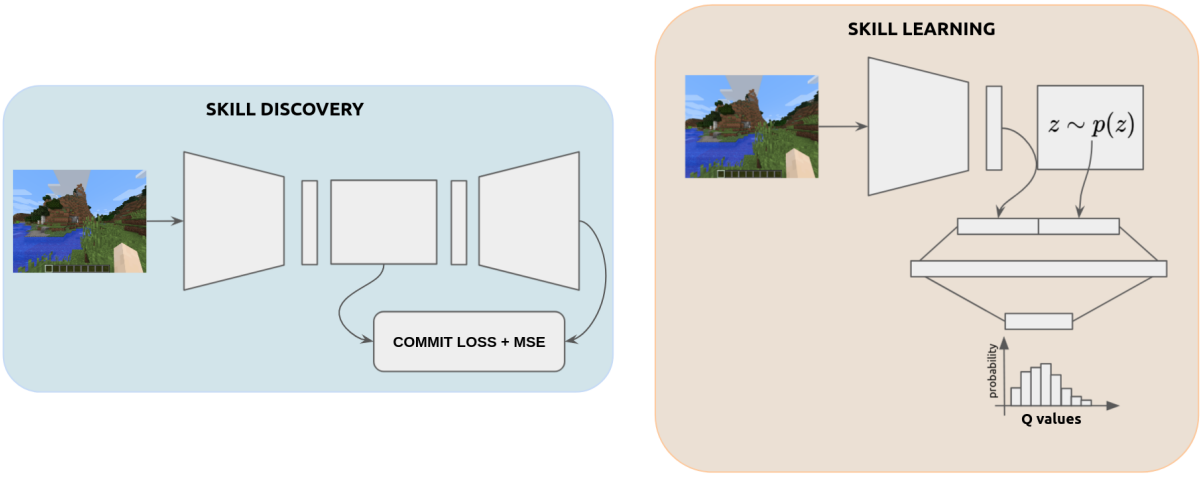


Figure 1: **Variational pipeline:** 1) We discover the categorical latents by reconstructing observations from the trajectories using a VQ-VAE model. 2) We train the RL agent by concatenating the VQ-VAE encoder output with the latent sampled from $p(z)$. Then, we forward it through the linear layers to get a distribution over the q values.

encoder output does not grow and commits to one of the embeddings. Note that in our case, we do not update the codebook through the gradients but with exponential moving averages of z_e due to faster convergence.

Before training, the model requires to fix the length of the codebook. This will determine the granularity of our latent variables. If we choose a large number of codes, we will end up with latent variables that encode very similar states. Instead, if we choose a small number, we encourage the model to find latents that generalize across diverse scenarios. The perplexity metric, measures the number of codebooks needed to encode our whole state distribution. In practice, this metric is computed per batch during training. Since it is not possible to know beforehand the number of useful latent variables that our model will discover, one can iterate over different codebook lengths until they find a good trade-off between generalization and granularity.

In EDL [14], the purpose of training a variational auto-encoder was to discover the latent variables that condition the policies in the Learning stage. But the learned representations were discarded. Instead, in our case, we also leverage the representations learned in the encoder and decoder for training the RL agent. This allows for faster and more efficient trainings of the RL agents.

4.2.2 Contrastive Learning

In the contrastive case, we follow the idea of Stooke et al. [50] where the positive pairs are delayed observations from an expert trajectory. Once the latents are learned, in order to provide easier comparisons with the variational set-up, we cluster the embedding space to have categorical latents. This step is perform using K-Means with K clusters equal to the length of the VQ-VAE codebook.

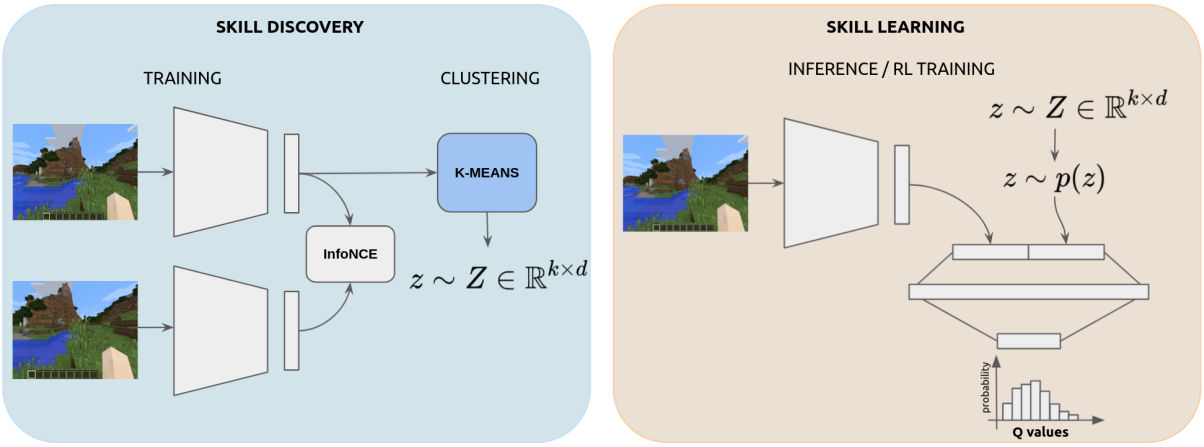


Figure 2: **Contrastive pipeline:** In this approach we have three stages. 1) Training. We learn the latents by minimizing the InfoNCE loss. 2) Clustering. At the end of the training we perform a clustering of the latent space to obtain K categorical latents. 3) Infernce. We get rid of the momentum encoder and forward the observations through the freezed parameters in the encoder. Finally, we perform the concatenation with the conditioning latent which allows to train our policy.

At this point, we have the same set-up we have in the variational case. The K-Means centroids are equivalent to the VQ-VAE codebook embeddings, so we can maximize the mutual information between the categorical latents and the inputs with the same Equation 8.

4.3 Skill-Learning

In the last stage of the process, we aim to train a policy $\pi(a|s, z)$ that maximizes the mutual information between inputs and the discovered latent variables. At each episode of the training, we sample a latent variable $z \sim p(z)$. Then, at each step, this embedding is concatenated with the embedding of the encoded observation at timestep t and forwarded through the network. This process is depicted in Figure 1 and 2, although once we have the categorical latents the learning phase is identical in both frameworks. At this stage, the latent variables are interpreted as navigation goals. Hence, the agent is encouraged to visit those states that brings it closer to the navigation goal or latent variable. Since each of the latent variables encode different parts of the state distribution, this results in covering different regions for each z.

Our method adopts the reverse form of the mutual information. Since we fix $p(z)$ as a uniform distribution we can remove the $\log p(z)$ constant term from the Equation 4, which results in the reward in Equation 8. The agent receives a reward of 1 only if the embedding that is conditioning the policy is the closest to the encoded observation.

$$r(s, z) = q_\phi(z|s) \tag{7}$$

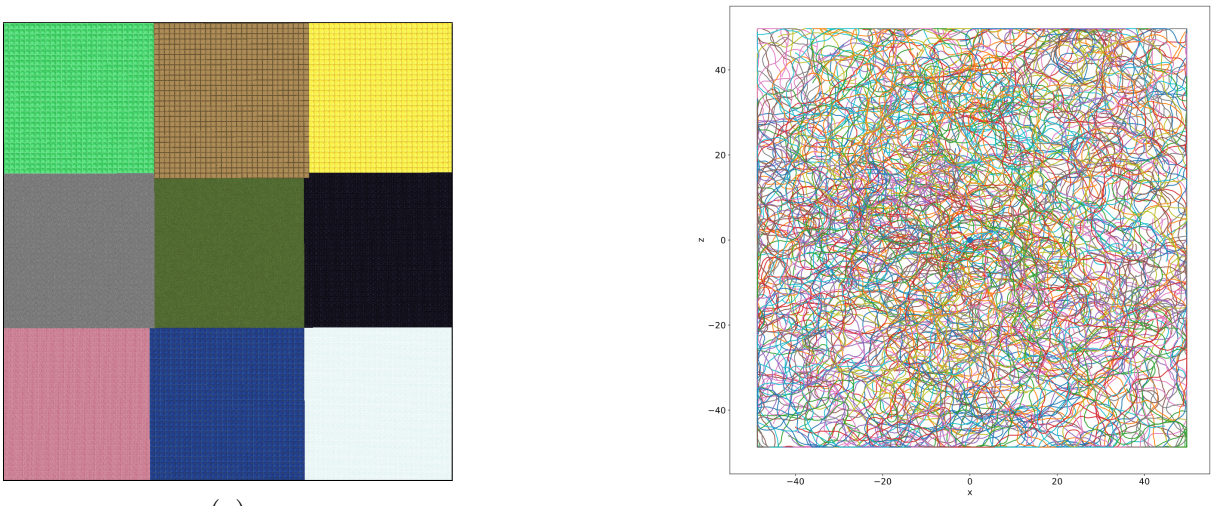
$$q_\phi(z = k|s) = \begin{cases} 1, & \text{if } k = \text{argmin}_j ||z_e(s) - e_j|| \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

5 Experiments

In this section we asses our method in different Minecraft maps and set-ups. In each of the trainings we follow the pipeline described in Section 4, where we first discover the skills and then we train a policy to learn them. We refer the reader to the Appendix A to check the hyperparameters used in the different methods.

5.1 Discovered skills from random exploration

Due to the inherent complexity of a 3D environment, we first test the discovered skills from random trajectories generated in a 2D toy map. Be aware that this is not our main goal, since this agent discovers skills induced by a random policy. Therefore the $p(s)$ will be fitted to the state-subspace that is covered with a random policy. In Figure 3a we show the toy map, which consists of 9 different regions, where the only difference among them is the floor type.

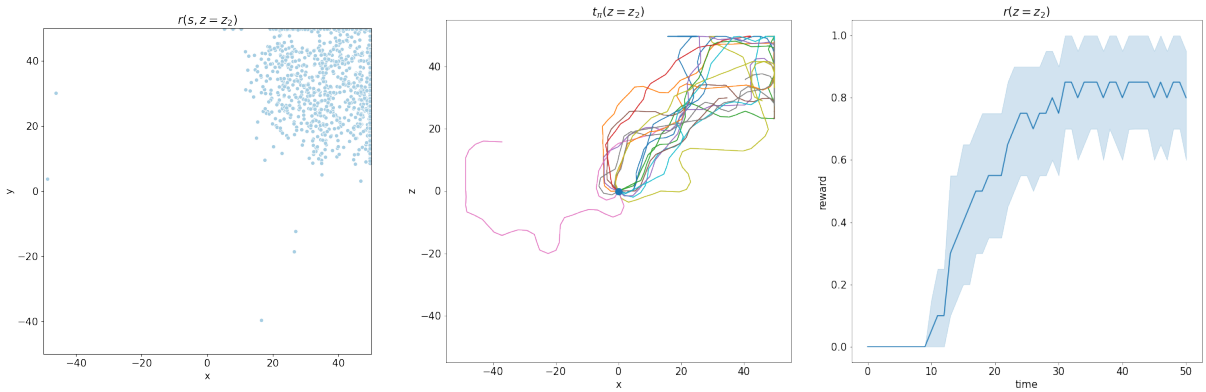


(a) (b)

Figure 3: a) Top-view of the toy map used for asses our method. The dimensions of this map are 100 square meters (As a reference, we count square meters as if the Minecraft coordinate system was measured in meters.), despite that, the agent has no exploratory issues even with a random policy. b) Trajectories generated by a random agent deployed in any region of the map.

Learning representations from these observations is quite straightforward, since the dynamics are not very complex. Despite that, either training with a contrastive or a variational method, we usually find latent centroids that encode observations in between plots. For this reason, if we want to have a categorical latent vector for each of the regions we should overclusterize the latent embeddings (in the contrastive case) or select a larger codebook (in the variational case). Regardless of the method, we achieve very similar results. In this case, in Figure 4 we show the performance of an agent trained on latents discovered by a contrastive approach. The learned latents are not trivial to assess, hence

we show some examples in Section 5.3 of how we do it. Once the encoders are trained. Since we train the agent in a map where we already collected observations from random policies, we can plot the reward that those observations would be given when conditioned in each of the categorical latents. For instance, in Figure 4a we can see that the observations recorded in the top-right corner are encoded as the second discovered latent. Therefore if we are conditioning the policy with it, the agent will receive a reward of 1, following Equation 8.



(a) (b) (c)

Figure 4: a) Observations that are encoded as the latent vector 2.
b) shows the trajectories performed by the agent and c) the average reward received in those trajectories. Both plots conditioning the agent on the latent vector 2.

In Figure 4a, we see that the navigation goal that encodes the second latent vector is on the top-right corner. In Figure 4b, we show how the performance of the agent when conditioned in this skill. All the trajectories head to the navigation goal except for one. Moreover, regardless of the initial direction, we can see that the agent is capable of orientating and go to the right direction. Lastly in Figure 4c, we can see how it takes 10 steps to start triggering the intrinsic reward until reaching a stable 0.8 reward in average. For more examples on other latent codes we refer the reader to the Appendix B.

5.2 Discovered skills from expert trajectories

We show another example in a different toy map (Figure 6) that leverages the skills discovered from expert trajectories. In this case, the map contains biomes that are commonly found navigating Minecraft maps. This time, we make use of a VQ-VAE model for discovering the skills, which means that we can reconstruct the latent codes learned. We visualize them in Figure 5, where we can see that despite being from expert plays, the latent codes inferred lack diversity. In fact, some of them are quite similar and lead to similar scenarios.

Finding the right number of skills is a trade-off between generalization and granularity. We want skills that generalize across different maps, but since we want to treat them

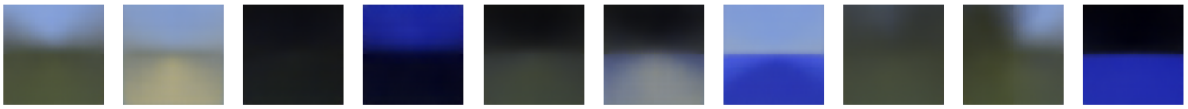


Figure 5: Centroids reconstruction. Each latent codebook is forwarded through the VQ-VAE decoder to generate one image of $3 \times 64 \times 64$ dimensions.

as navigation goals, each skill should be enough specific to encode a region of the state-space. If we have a very large number of skills, most of them will be very similar and the navigation goals will be completely overlapped. We empirically set the number of latents to 10 which gives enough different scenarios while having a great generalization. For instance, the second centroid reconstruction in Figure 5, will encode all observations in a *desert* no matter how different they are. As we see in Section 5.4, this is actually a problem in realistic environments, where a *desert* could be a region of the space too big and therefore ambiguous as a navigation goal. In our controlled environment (Figure 6), the *desert* region is a small plot that is easier to view as a goal.

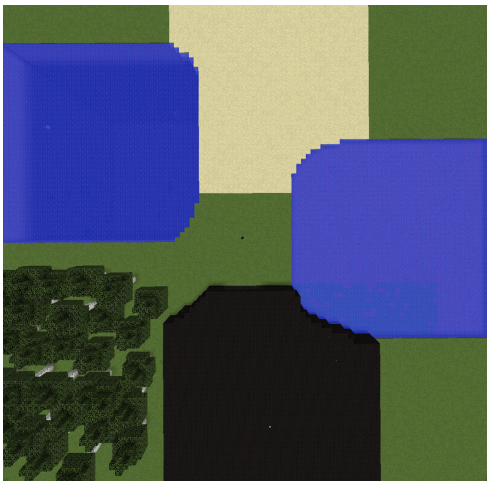
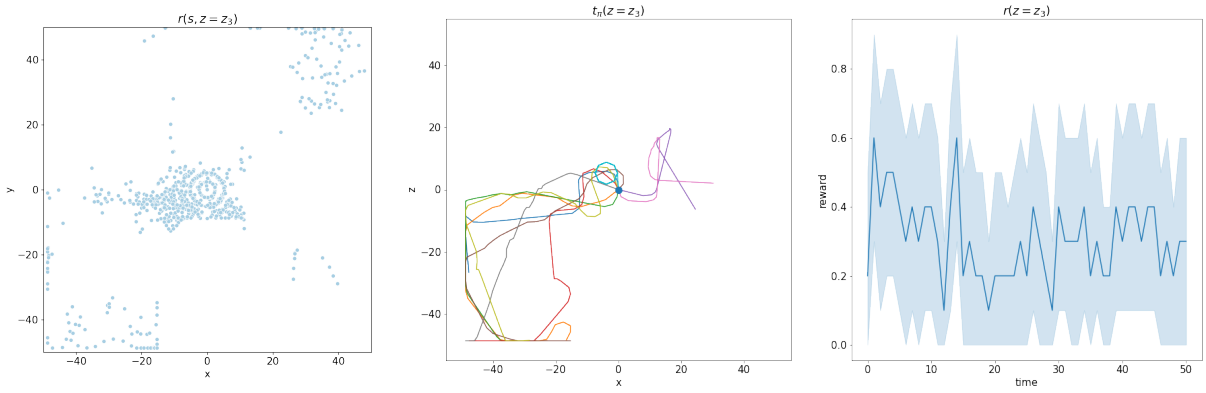


Figure 6: Map for testing the skills discovered through expert trajectories. It is composed by 5 different regions: *desert*, *shallow* and *deep water*, a *dark room* that simulates night time and a *forest*.

In this case, for the skill that conditions the policy to the *forest* direction, we can see in Figure 7b that the trajectories are not as perfectly performed as in the previous case. Now the places where we get a positive reward are more ambiguous and therefore the agent will struggle to determine which are the states that maximizes the future reward. Still, the agent is able to learn conditioned policies that will lead to these goals succesfully. In fact, since we have more latents than planned goals in the map, some policies will lead to the same places. We refer the reader to the Appendix C for a complete view of the results.

The arising problems that we just mentioned are even more accentuated in realistic maps. We tackle this issues in Section 5.4, where we study alternative ways of learning the



(a) (b) (c)

Figure 7: a) Observations that are encoded as the latent vector 3.
b) and c) show the trajectories performed by the agent and the reward received in average when conditioned on skill 3.

conditioning latents.

5.3 Evaluation of the learned representations

Self-supervised methods are not trivial to assess. Since we do not have ground truth labels, we must qualitatively assess the performance of our model. In this section we show two general ways of evaluation and then one specific for the variational and the contrastive case.

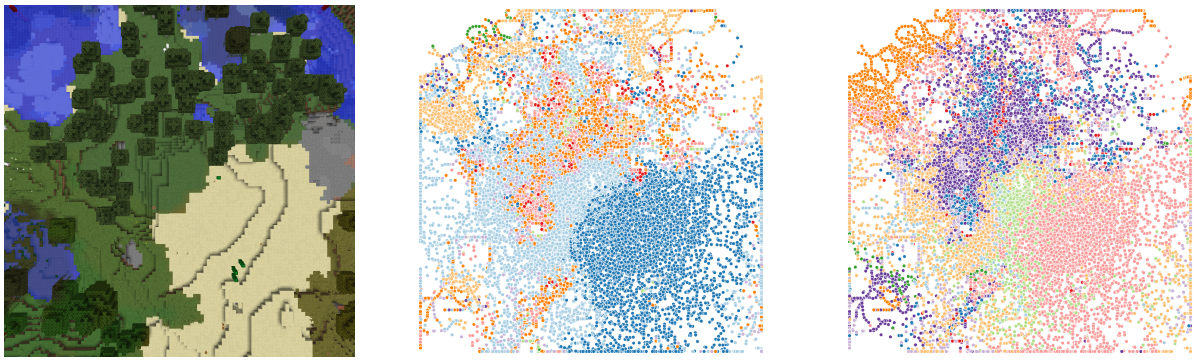
5.3.1 Index maps

First, we can run random trajectories in a given map. Then, encode each observation with the index of the closest embedding in the latent codebook or K-Means centroids. We assign a color to each of the indexes and we get what we refer as an *index map*. Thanks to these maps, it is very easy to visualize whether the learned embeddings are meaningful.

Figure 8 compares the index maps learned with the variational and constrastive models when skills are discovered with expert trajectories, but random trajectories explore the given map.. In general we can see that the variational model discretizes slightly better, although the contrastive model is able to distinguish the *water* biome much better. When studying these visualizations we should be aware that we are not taking into account the direction in which the agent was looking at the time it was recorded. Therefore two points that are next to each other, might be encoding totally different views and hence will result in different latent codes.

5.3.2 Visualization of Principal Component Analysis

We can forward the entire set of trajectories through the trained encoder, and perform a dimensionality reduction over the embedding outputs. In our case, we perform a Principal



(a) (b) (c)

Figure 8: a) Section of a Minecraft realistic map. b) Index map computed with variational model. c) Index map computed with contrastive model.

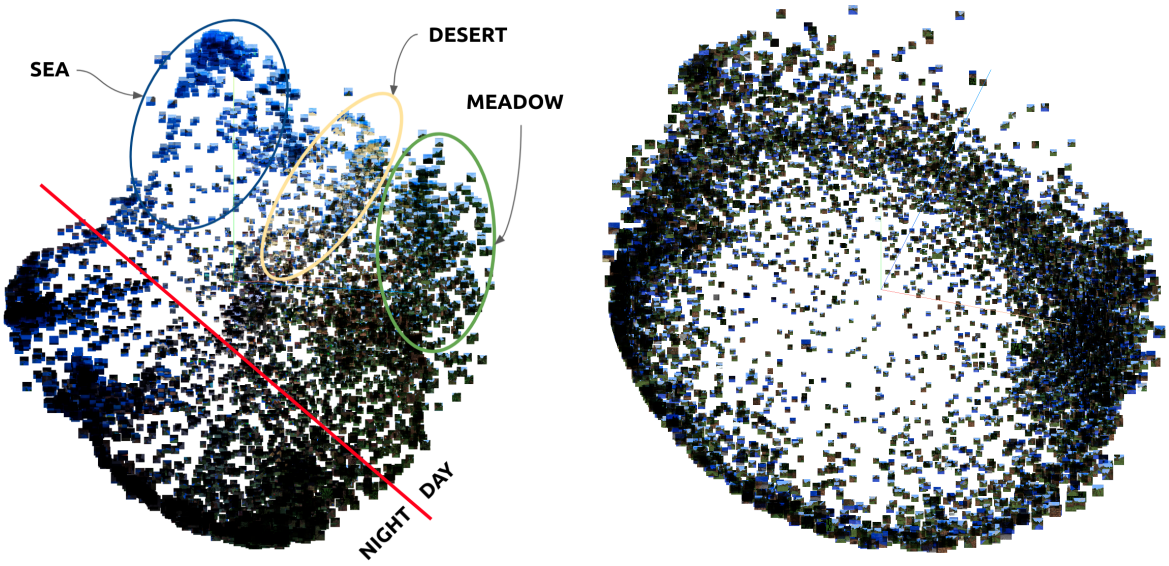
Component Analysis (PCA) implemented in Tensorboard [1] to visualize the dimensions with higher variance in a 2D or 3D map. The PCA latents shown in Figure 9 indicates that the contrastive approach learns better clusters than the variational model. In the contrastive case, we can distinguish a cluster that belongs to sea navigation, desert navigation, and some of them that belong to night time in the bottom of the figure. On the other hand, despite the good and similar results we get with the variational case, we cannot distinguish proper clusters. We hypothesize that the individualism in the reconstruction process might lead to more independent embeddings, where the clusters are difficult to visualize.

5.3.3 Perplexity (variational only)

For the variational case, we have already seen the centroids reconstruction in Figure 5. These reconstructions allow us to assess the performance of our VQ-VAE model. During training, we observed that at the beginning, all observations are encoded into one or two latents (whose reconstructions are very blurry and grayish), and after some iterations, more latents are used to encode the differences in the observations. This is quantified by a metric called perplexity. The perplexity is an information-theoretic metric that it is related to the entropy of a distribution. If the perplexity grows, it means that there is more uncertainty in which latent code will be assigned to each sample. In Figure 10 we plot two curves of different VQ-VAE trainings where they learn to use up to 8 and 6 latent codes from the 10 available.

5.3.4 Similarity with manual goal (contrastive only)

Regarding the contrastive approach, we can not reconstruct from the latents to the pixel-space, and we do not have clusters during the training. For this reason, we come up with a different assessment method. We choose a trajectory and encode one of its observations that will serve as goal state. Then we forward all the observations from the trajectory



(a) (b)

Figure 9: PCA over ouputs latents from a) contrastive and b) variational model.

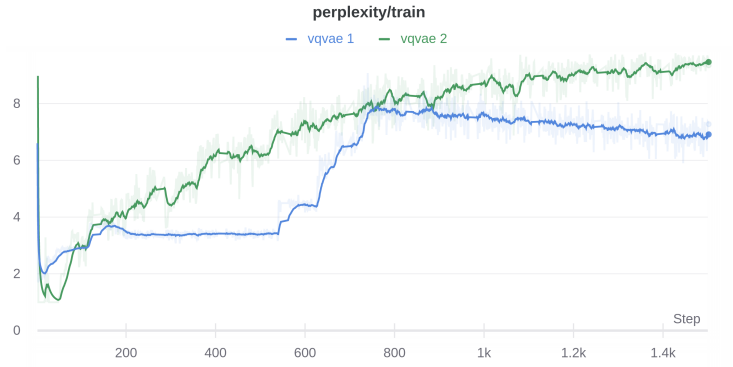


Figure 10: Perplexity of two VQ-VAE trainings. One, trained on pixel observations smoothly increases the number of latents being used. On the other hand, the blue model is trained with pixels and coordinates. In this case, the latents seems to be discovered slower but more steeped. We hypothesize that dealing with more inputs might slow down the training.

and compare it against the goal state embedding. This comparison provides a similarity measure that encodes the likelihood of two images being from the same trajectory.

$$\text{sim} = x_t W x_{t+k} \tag{9}$$

Where x_t and x_{t+k} are the positive pairs and W is a matrix learned during training that projects x_t to an intermediate dimension.

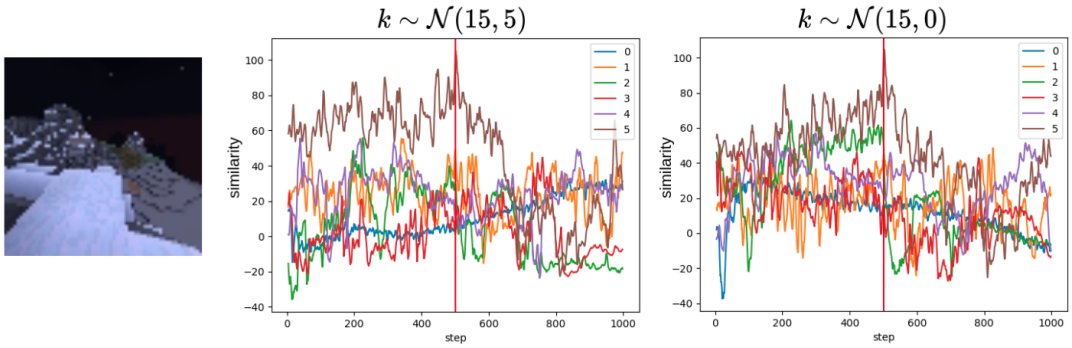


Figure 11: We pick the observation from the middle (red line) of the 5th trajectory and extract its embedding. Then we forward 5 different trajectories and we compare them against this goal state. This results in the curves shown in the middle and right plots. The middle one belongs to a training with Gaussian delays between samples and the right one with a fixed step.

Figure 11 shows how the similarity with the observations from the same trajectory stands out with the Gaussian step. We hypothesize that given the dynamics of our 3D environment, having a delay sampled from a gaussian distribution encourages the model to learn even more global features that encodes the surroundings of the agent. This can be interpreted as data augmentation in the temporal domain, since we sample different delays for each positive pair and for each epoch.

5.4 Scaling to realistic maps

Once we have discovered generic skills leveraging a $p(s)$ induced by expert trajectories, we can evaluate it against realistic procedurally generated maps. We generate a set of rollouts using a random policy in the map shown in Figure 12a. Then, we encode each of the observations with each index k, ($k = \text{argmin}_j ||z_e(s) - e_j||$). As we already mentioned at the beginning of this Section and stated by Gregor et al. [27], we suffer from the dominance of some discovered options and the ambiguity and sparsity of the derived reward. Since we are treating each latent variable as navigation goals, the ones that are spread out over the whole map will not guide but confuse the agent as it is receiving high reward in multiple places.

One option to overcome this issue is to exploit the spatial information provided by the environment in the form of coordinates. The idea is to learn an embedding space that contains a relationship between visual features and relative coordinates with respect to the initial state, always the same one. In this way, our agent might be able to distinguish between two visually identical mountains located at two different positions in the map.

Although, the major drawback is that we are not able to induce $p(s)$ from expert trajectories anymore. The reason is because we do not have access to the coordinates from expert plays and even if we had them, they would belong to different maps. Instead, we have to infer it from rollouts of a random policy in a single map. Although we will not encourage better exploration than a random policy, the skills learned from this exercise

can be very useful for training a hierarchical policy on top.

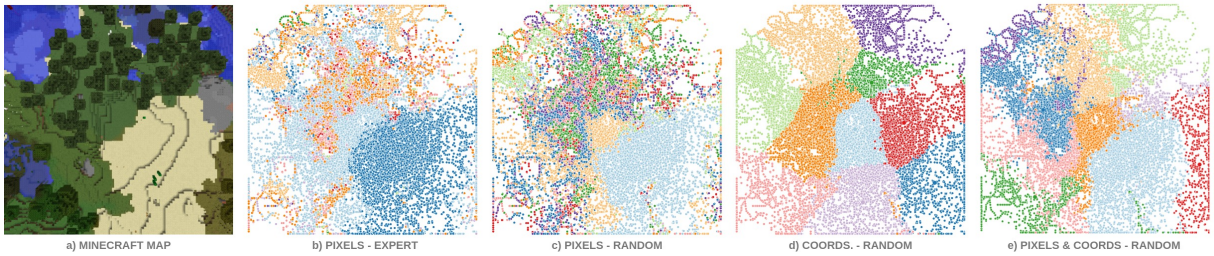


Figure 12: a) Top-view of our Minecraft map. The caption below b) c) d) e) refers to the nature of the states (pixels, coordinates or both) and the type of trajectories (expert or random). Each coloured point indicates the closest centroid to the encoded embedding. More examples can be found in this report.

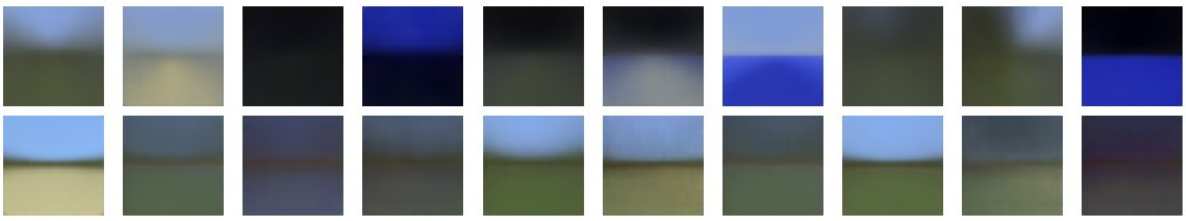


Figure 13: **Top:** images reconstructed from learned centroids using expert trajectories. **Bottom:** images reconstructed from learned centroids using pixels and coordinates from random trajectories.

Figure 12b shows that expert trajectories from pixels discover fewer and sparser skills than the those discovered by random exploration in our map, depicted in Figure 12c. This suggests that while the skills discovered by experts may be more generic as they were collected in different worlds, they are not as useful for our particular map as the skills discovered by a random policy. This hypothesis is supported by Figure 13, where we show the reconstructed images for each of the VQ-VAE codebook centroids. The reconstructions belonging to the expert trajectories contain scenarios that cannot be found in our Minecraft map. In our study case, we aim to learn policies that treat the latent variables as navigation-goals. For this purpose, Figure 12 shows complementary skills discovered from pixels (Figure 12c) or coordinates (Figure 12d). We would like to discover skills that take into account not only the visual similarity but also the position relative to the initial state, so we adopt a solution that considers the two types of state representations (Figure 12e).

Figure 14 shows a clear example of the pixels and coordinates mixing. If we based our encoder in just pixel observations we would have an almost identical latent encoding for both images. Instead, thanks to the coordinates, they become two separate latents. This may help the agent in being more precise and distinguish different regions of the space despite its visual similarity.

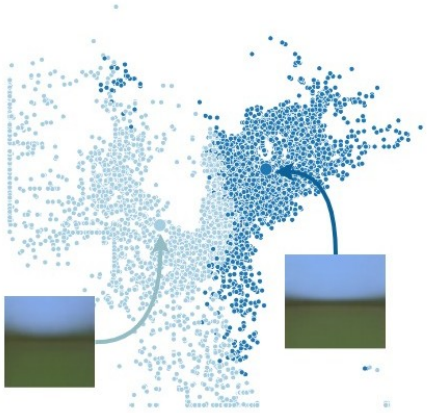
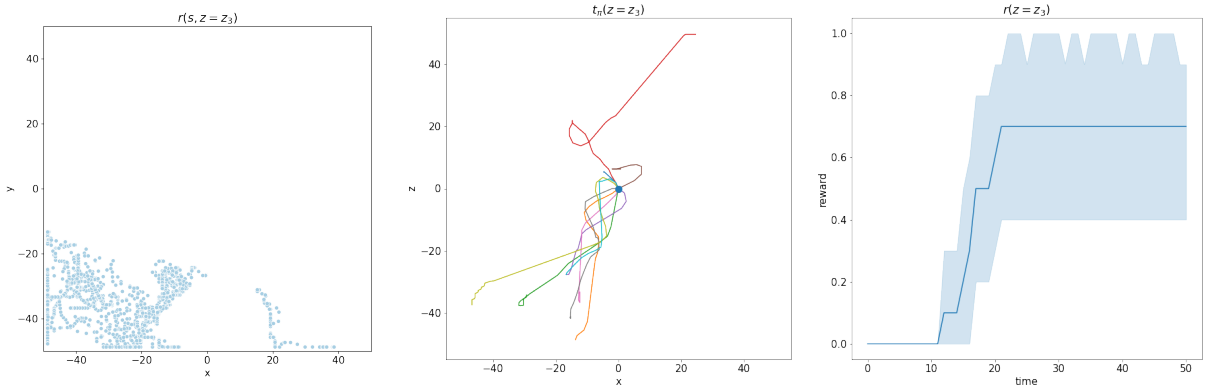


Figure 14: A very homogeneous and big region of the space becomes two different skills when using a combination of pixels and coordinates.

We scale the VQ-VAE [53] to encode both pixels and coordinates independently. We sum up their embeddings and quantize the result to one of the codebook embeddings e. Now, the decoder of the model contains two independent branches, where one reconstructs an image and the other reconstructs a coordinate from the same embedding. Then, the loss defined in Equation 6 becomes:

$$\mathcal{L} = \log p(s_i|z_q) + \lambda \log p(s_c|z_q) + \beta||z_e - sg[e]||_2 \tag{10}$$

$$z_q(s_c, s_i) = e_k, \quad k = \text{argmin}_j||z_{e_c}(s_c) + z_{e_i}(s_i) - e_j||_2 \tag{11}$$



(a) (b) (c)

Figure 15: a) Observations that are encoded as the latent vector 3.
b) and c) show the trajectories performed by the agent and the reward received in average when conditioned on skill 3.

The only difference with respect to previous loss, is the second added term, which is the reconstruction error in the coordinate space. λ weights its contribution in order to have similar magnitudes with the pixel reconstruction error.

Once we have better latent embeddings for realistic maps, we can train an agent conditioned on them. Qualitative results are presented in Figure 15, where we can see an example of a latent encoding the bottom-left observations and the trajectories and rewards recevied in 10 evaluation episodes. The performance of all the conditioned policies is included in the Appendix D. We observed that at least 6 skills out of 10 could be potentially leveraged in a hierarchical policy.

5.5 Reward alternatives

As we have already seen in previous sections, the derived reward from the reverse form of the mutual information is very sparse. Not only that, if we use the 1 or 0 reward, the feedback is not very informative of how similar are two states, and it might lead to ambiguous policies. To overcome this issues we studied the impact of giving either smoother and/or denser rewards and present the results in Figure 16:

- The first alternative is to simply compute the L2 distance in the embedding space each time our observation is encoded with the latent that is conditioning the policy. This reward does not solve the exploratory issues but, at least is more informative than the vanilla one.
- The second alternative is a dense reward that returns the distance to the conditioning latent regardless of the current observation. Although the agent is guided in all places, the reward becomes more ambiguous and the agent will struggle to learn the path to the navigation goal.
- Finally, the last option is only possible when leveraging the coordinates information along the pixels. It consists in using both the reverse and the forward form of the mutual information. If our observation does not match the conditioning latent, we use the forward form by reconstructing the coordinate of the latent vector and comparing with our current coordinate. But, if it matches, the reward is given based on the embedding distance. Intuitively, this reward estimates the distance to the navigation goal, until the agent is close enough to the navigation goal to base your actions in the pixel resemblance. The main inconvenience of this last reward is that when is based only on the coordinates, you might fall in a local optima. This means that the shortest path in the coordinate system might not be possible to follow in the real environment.

In our experiments, since we do not have exploratory issues, we observed better results using the sparser versions of the reward.

5.6 The impact of expert trajectories for discovering skills

In this section we discuss the importance of having expert trajectories for learning skills. Are random trajectories enough to discover meaningful skills? To answer this question we

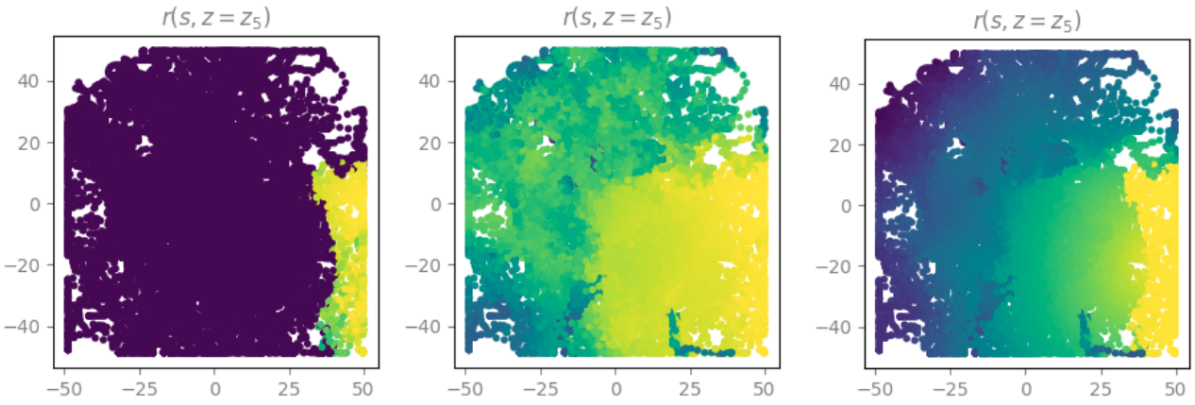


Figure 16: Reward alternatives, from left to right: sparse with distance to embedding. Distance to embedding. Distance in coordinates and distance to embedding depending on the encoded observation.

compare two variational models. One trained with expert trajectories and another one trained on random trajectories from the map in Figure 12. If we pick the observation of these trajectories and count the number of times each categorical latent is selected, we get the plots in Figure 17. As we can see, the distribution in the model trained from expert plays is much more condensed in fewer skills. In contrast, when we discover these skills from random trajectories of this same map, we see that the distribution is more homogeneous. This might seem very obvious, since in the latter case we assess the model with the same data it was trained on. But what if we keep adding more random trajectories from different maps?

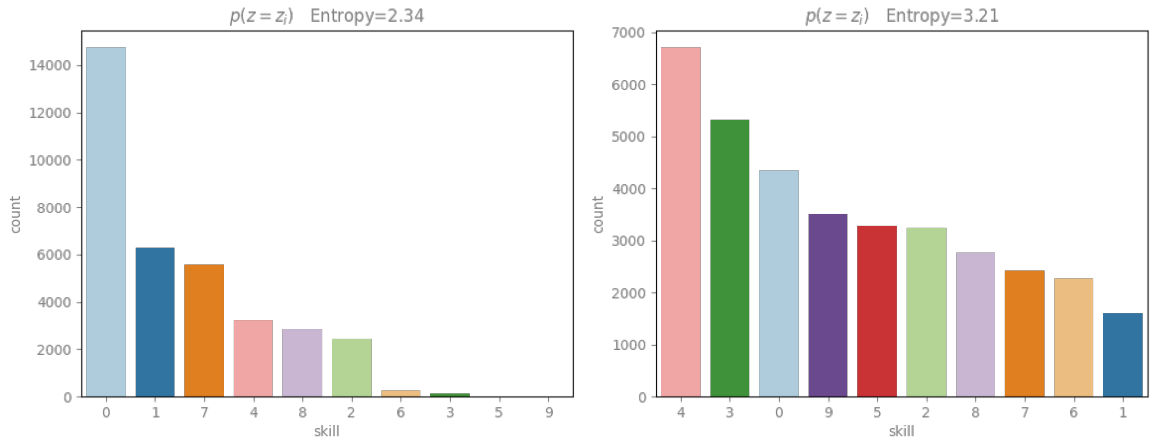


Figure 17: Distribution of skills discovered in a realistic map from expert trajectories (left) and random trajectories (right).

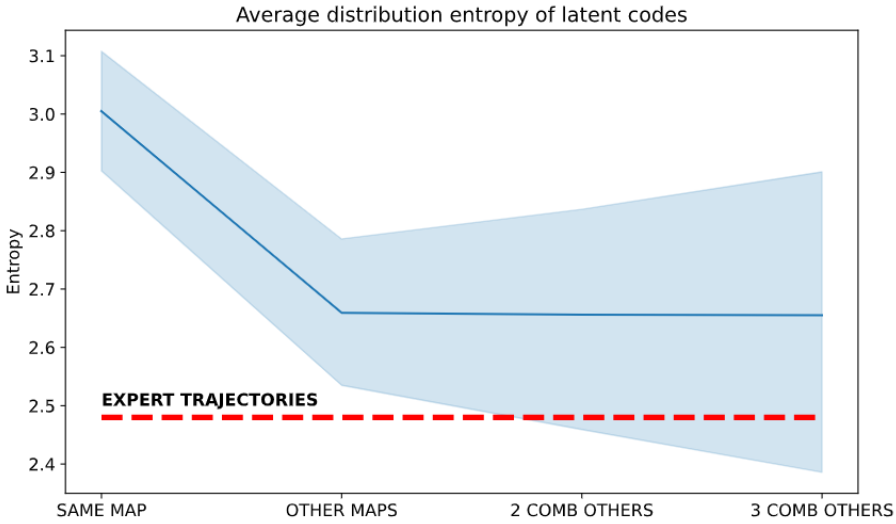


Figure 18: Average distribution entropy across 4 different realistic maps as we keep adding more trajectories from different maps.

In Figure 18, we demonstrate that we can discover skills from random trajectories that generalize as well as models trained with expert ones. To do so, we trained 16 different models and run 40 evaluations in expert and random trajectories from 4 different maps. The different trainings correspond to:

- Expert trajectories.
- Random trajectories.
  - Trajectories from one single map (x4 maps)
  - Trajectories from 2 maps (x6 combinations)
  - Trajectories from 3 maps (x4 combinations)
  - Trajectories from 4 maps (x1 combination)

The evaluations are run in the random trajectories of these 4 maps. As we can see, the uncertainty grows as we add more trajectories from different maps. We surpass the experts entropy distribution with some models trained in trajectories that do not belong to the map being assessed. This means that with enough random trajectories we can generalize as well as training from expert trajectories. This may work for navigation and easy exploration, but it will definitely fail if we try to scale to more complex tasks. Even if we want an agent to explore narrow caves, it will not succeed unless its policy encourages visiting novel states. Another drawback, is that the aggregate of random trajectories in different maps requires an oracle that is able to spawn the agent in different maps or regions of a map.

Therefore, as a conclusion we consider that random trajectories may work for discovering very basic visual features and regions in a bounded space but it will not scale for discovering complex skills.

6 Conclusions and future development

In this thesis we present a framework to extend the EDL paradigm to pixels. This requirement presents some challenges. First, dealing with a pixel-based 3D map increases the exploration difficulty. Moreover, information-theoretic objectives have difficulties on learning human priors. We leverage the MineRL expert trajectories that encode human priors useful to learn state-covering skills. Besides that, we are able to learn good state representations from these trajectories which allows for faster and more efficient trainings. Second, following the EDL derivation of the mutual information, we optimized our agents based on the MSE, which does not have any notion of space in the environment. To overcome that, we propose to derive the reverse form of the mutual information between inputs and latents. In this manner we compute the intrinsic reward in the embedding space which encodes the dynamics learned during the skill-discovery phase. We observed that contrastive and variational approaches are capable of learning the mapping from inputs to latents. In the former case, we leverage the use of positive pairs based on delayed observations, and the K-Means clustering algorithm for determining the categorical latents. In the latter, we make use of a VQ-VAE model that directly allows to infer categorical latents from the input data. We qualitatively show that some skills are correctly learned, specially in the toy maps where the exploration and the environment dynamics are much easier. But some other skills fail or are poorly learned. This is more common in realistic maps where the dynamics are more complex and unstructured.

We comment on some research paths that can be followed with the intention of further improve our method. In our work, we fixed the prior over the latent space to be uniformly. Instead, $p(z)$ could be learned or dynamic, although this might lead to collapsing to fewer skills. Also, the reverse form of the mutual information derives a sparse reward. Moreover, the fact that multiple states can receive a positive reward might lead to unpredictable skills. Despite the proposed alternatives, we believe that adding a method for encouraging state novelty would boost the exploration issues we might suffer.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms, 2018.

[3] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *International Confreencce on Learning Representations*, 2021.

[4] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[5] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5055–5065, 2017.

[6] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2019.

[7] A. G. Barto and O. Simsek. Intrinsic motivation for reinforcement learning systems. In *Proceedings of the Thirteenth Yale Workshop on Adaptive and Learning Systems*, pages 113–118. Citeseer, 2005.

[8] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation, 2016.

[9] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[10] W. Böhmer, J. T. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer. Autonomous learning of state representations for control. *KI-Künstliche Intelligenz*, pages 1–10, 2015.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2020.

[13] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2018.

[14] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giro-i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

[15] V. Campos, P. Sprechmann, S. Hansen, A. Barreto, S. Kapturowski, A. Vitvitskyi, A. P. Badia, and C. Blundell. Coverage as a principle for discovering transferable behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021.

[16] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

[17] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu. Boltzmann exploration done right. *Advances in Neural Information Processing Systems*, 30:6284–6293, 2017.

[18] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised visual transformers. *arXiv preprint arXiv:2104.02057*, 2021.

[19] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4302–4310, 2017.

[20] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[22] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

[23] C. Florensa, Y. Duan, and P. Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2017.

[24] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, et al. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

[25] A. Garivier and E. Moulines. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pages 174–188. Springer, 2011.

[26] D. Ghosh, A. Gupta, and S. Levine. Learning actionable representations with goal conditioned policies. In *International Conference on Learning Representations*, 2018.

[27] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. In *International Conference on Learning Representations (workshop)*, 2017.

[28] W. H. Guss, M. Y. Castro, S. Devlin, B. Houghton, N. S. Kuno, C. Loomis, S. Milani, S. Mohanty, K. Nakata, R. Salakhutdinov, et al. The minerl 2020 competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:2101.11071*, 2021.

[29] D. Hafner, P. A. Ortega, J. Ba, T. Parr, K. Friston, and N. Heess. Action and perception as divergence minimization. *arXiv preprint arXiv:2009.01791*, 2020.

[30] E. Hazan, S. Kakade, K. Singh, and A. Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

[31] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[32] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International Conference on Machine Learning*, pages 1558–1567. PMLR, 2017.

[33] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, 2017.

[34] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2014.

[35] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[36] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[37] A. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020.

[39] S. Mohamed and D. J. Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2125–2133, 2015.

[40] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[41] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.

[42] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

[43] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[44] C. Salge, C. Glackin, and D. Polani. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.

[45] N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T. Lillicrap, and S. Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2018.

[46] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[47] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators.

[48] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[50] A. Stooke, K. Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

[51] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[52] A. Trott, S. Zheng, C. Xiong, and R. Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2019.

[53] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6309–6318, 2017.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[55] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[56] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[57] D. Warde-Farley, T. V. de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2019.

Appendices

A Hyperparameters

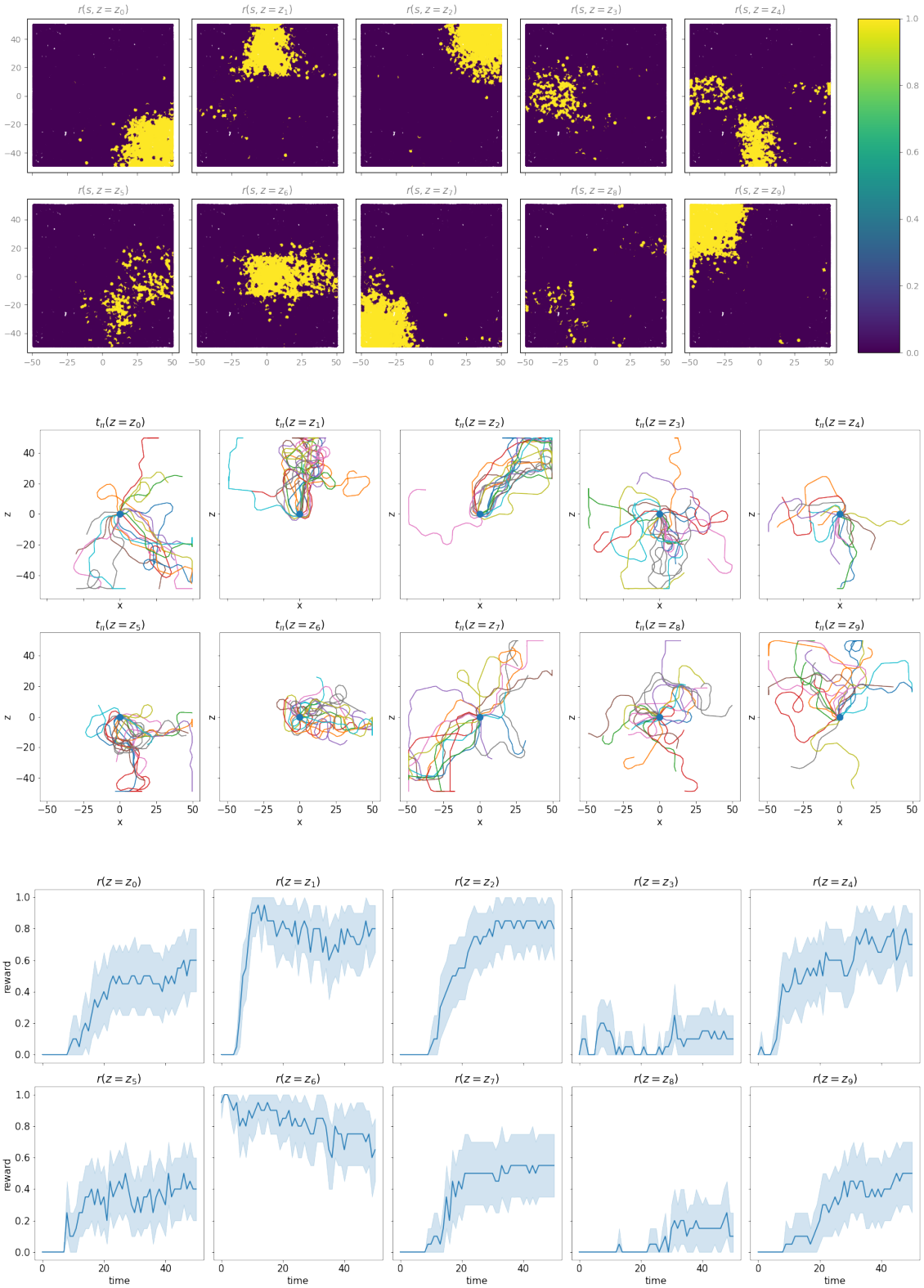
VQ-VAE	
learning rate	1e-3
batch size	256
num hiddens	64
num residual hiddens	32
num residual layers	2
embedding dim	256
num embeddings	10
β	0.25
decay	0.99

Contrastive	
learning rate	1e-3
batch size	128
τ	5e-3
soft update	2
embedding dim	128
k_μ	15
k_σ	5

Rainbow	
learning rate	2.5e-4
batch size	64
sampling	uniform
max episode steps	500
update interval	4
frame skip	10
gamma	0.99
clip delta	yes
num step return	1
adam eps	1.e-8
gray scale	no
frame stack	no
final expl frames	3.5e5
final epsilon	0.01
eval epsilon	0.001
replay capacity	10.e6
replay start size	1.e4
prioritized	yes
target upd interval	2.e4

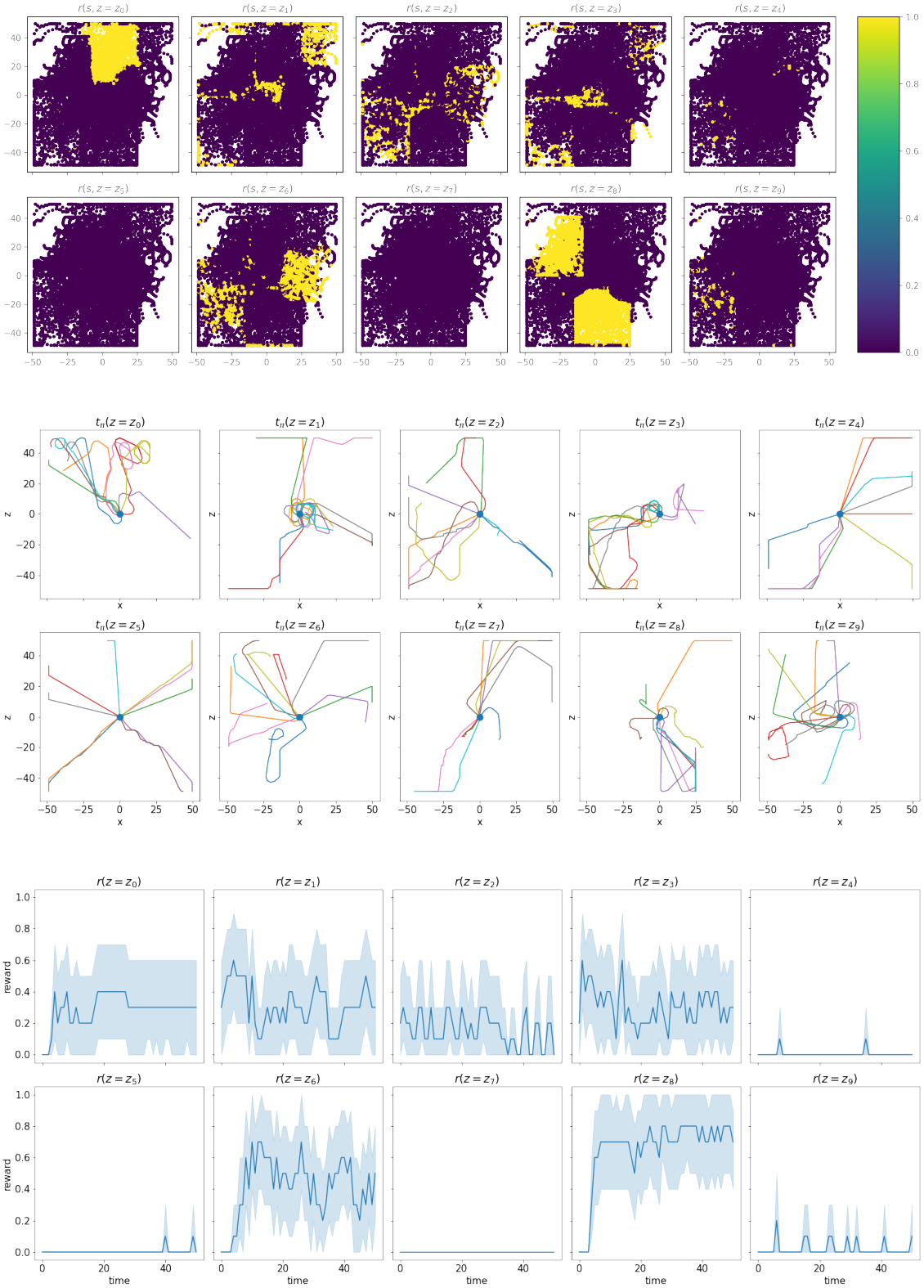
B Learning from Pixels with Contrastive approach

(From random trajectories in toy map)



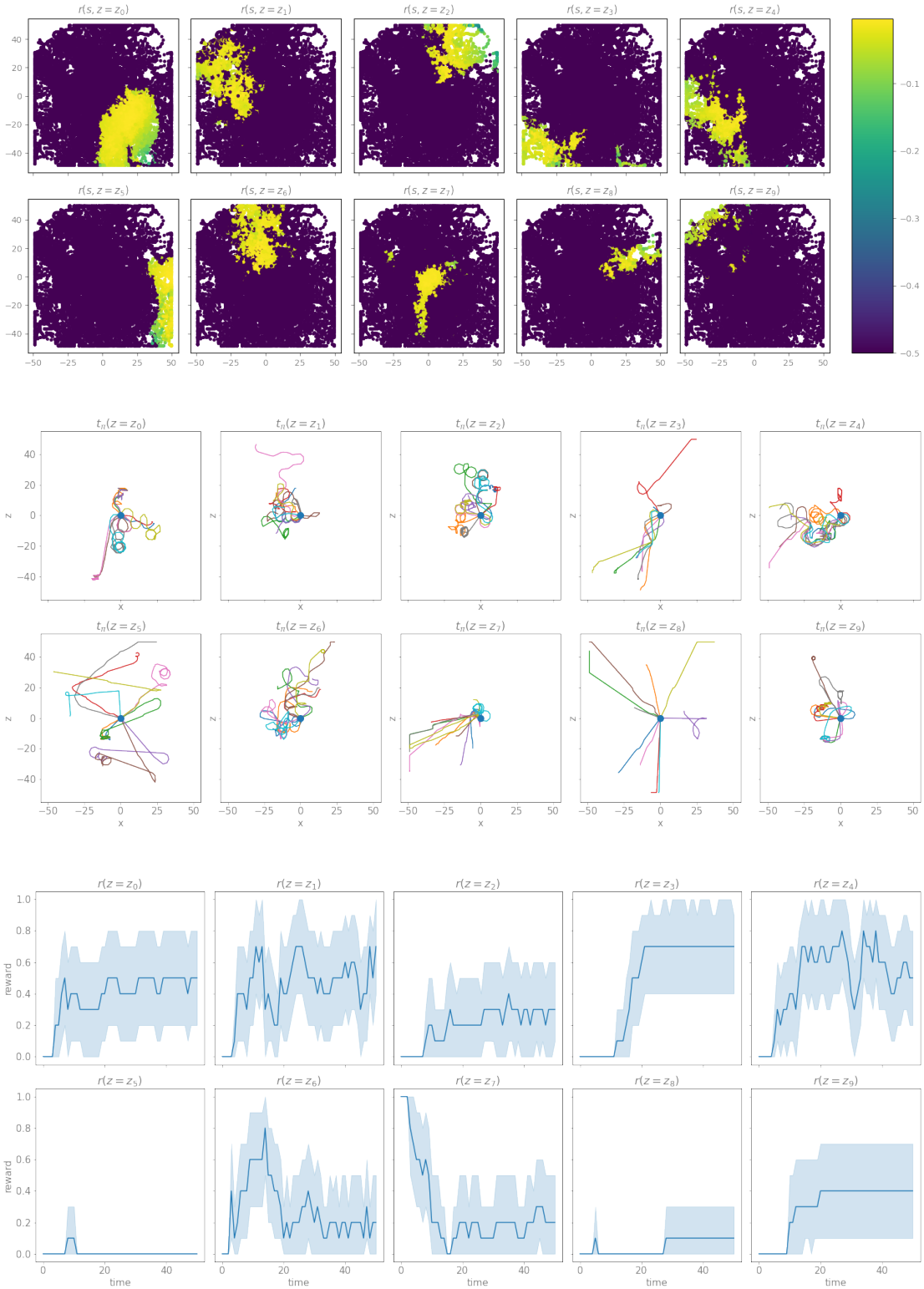
C Learning from Pixels with Variational approach

(From expert trajectories)



D Learning from Pixels and Coordinates with Variational approach and realistic maps

(From random trajectories in realistic map)



PiCoEDL: Discovery and Learning of Minecraft Navigation Goals from Pixels and Coordinates

Juan José Nieto[1] Roger Creus Castanyer[1] Xavier Giro-i-Nieto[1,2,3]
[1]*Universitat Politècnica de Catalunya* [2]*Barcelona Supercomputing Center* [3]*Institut de Robòtica i Informàtica Industrial, CSIC-UPC*
juanjo.3ns@gmail.com creus99@protonmail.com xavier.giro@upc.edu

1. Introduction

Defining a reward function in Reinforcement Learning (RL) is not always possible or very costly. For this reason, there is a great interest in training agents in a task-agnostic manner making use of intrinsic motivations and unsupervised techniques [7, 6, 15, 2, 14, 3]. Due to the complexity to learn useful behaviours in pixel-based domains, the results obtained in RL are still far from the remarkable results obtained in domains such as computer vision [5, 4] or natural language processing [1, 12]. We hypothesize that RL agents will also benefit from unsupervised pre-trainings with no extrinsic rewards, analogously to how humans mostly learn, especially in the early stages of life.

Our main contribution is the deployment of the *Explore, Discover and Learn* (EDL) [3] paradigm for unsupervised learning to the pixel and coordinate space (PiCoEDL). In particular, our work focuses on the MineRL [9] environment, where the observation of the agent is represented by: (a) its spatial coordinates in the Minecraft virtual world, and (b) an image from an egocentric viewpoint. Following the idea of *empowerment* [10], our goal is to learn latent-conditioned policies by maximizing the mutual information between states and some latent variables, instead of sequences of actions [7]. This allows the agent to influence the environment while discovering available skills.

2. From pixels and coordinates to skills

We formulate a Markov decision process (MDP) as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$. $\mathcal{S}$ is the high-dimensional state space (pixel images and coordinates), $\mathcal{A}$ refers to the set of actions available in the environment and $\mathcal{P}$ defines the transition probability $p(s_{t+1}|s_t, a)$. We learn latent-conditioned policies $\pi(a|s, z)$, where the latent $z \in \mathcal{Z}$ is a random variable.

Given the property of symmetry, the mutual information ($\mathcal{I}$) can be written using the Shannon Entropy ($\mathcal{H}$) in two ways:

$$\begin{aligned} \mathcal{I}(S, Z) &= \mathcal{H}(Z) - \mathcal{H}(Z|S) \quad \rightarrow \quad \text{reverse} \\ &= \mathcal{H}(S) - \mathcal{H}(S|Z) \quad \rightarrow \quad \text{forward} \end{aligned} \tag{1}$$

Maximizing the mutual information (MI) requires knowledge of unknown distributions ($p(s), p(s|z), p(z|s)$). For the former we study two cases: (a) Using expert trajectories, and (b) Using the distribution induced by a random policy. A comparison of both strategies can be found in Section 2.1, for now we assume the latter case of $p(s)$. We rely on variational inference techniques for estimating the mappings from the states to the latent variables and backwards. These are estimated from the rollouts induced by the random policy. Finally, we also need to define a prior $p(z)$ to sample from, which in our case, following EDL [3] is a fixed uniform categorical distribution.

If we proceed with the derivation of the forward form in EDL [3], we can find that in our case the intrinsic objective becomes a distance in the pixel space. Since we cannot assume that it is representative of a meaningful distance in the environment, we discard this approach. Instead, we adopt the reverse form of the MI. We use the VQVAE [13] model, that allows us to estimate the posterior $p(z|s)$ with the encoder $q_\phi(z|s)$ by maximum likelihood on (s, z) tuples and also contains a categorical bottleneck for $p(z)$. Then, our final objective becomes:

$$r(s, z) = q_\phi(z = k|s) = \begin{cases} 1, & \text{if } k = \text{argmin}_j ||z_e(s) - e_j|| \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$z_e(s)$ is the sum of the outputs of two encoders: (a) a 2D convolutional encoder for the images, and (b) a multilayer perceptron for the coordinates. Both have the same output dimension that allows summing up the resulting embeddings. Then in Equation 2, we find the index of the closest embedding in the VQVAE codebook e. Only if this index matches the sampled latent variable z that is conditioning the policy, it will return a reward of 1. Despite the sparsity of rewards, since we use a $p(s)$ that is induced from a random policy, we

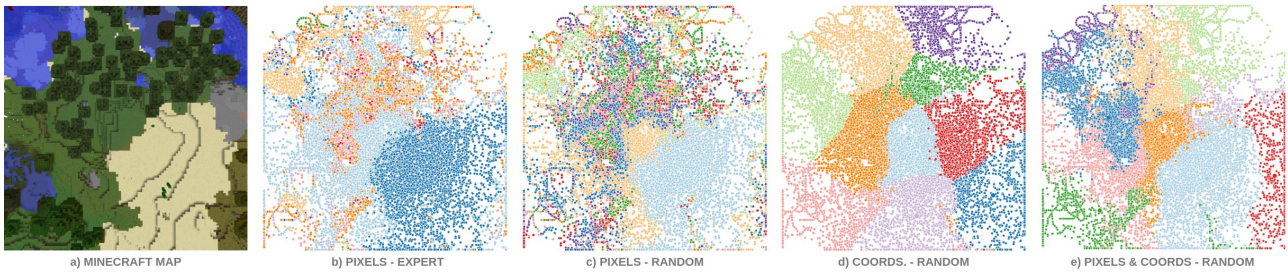


Figure 1. a) Top-view of our Minecraft map. The caption below b) c) d) e) refers to the nature of the states (pixels, coordinates or both) and the type of trajectories (expert or random). Each coloured point indicates the closest centroid to the encoded embedding.

know that these states are reachable and we will not suffer from exploration problems. Also, there are other problems due to multiple states rising positive rewards which could lead to ambiguous objectives. This can be tackled using a smoother reward function such as computing the distance in the embedding state, but in our experiments we did not have any problem by leveraging the previous reward.

In the following subsections we specify the implementation details of our approach.

2.1. Exploration and Skill Discovery

Firstly, we considered using expert trajectories to induce the distribution over the states. We used the MineRL dataset [8], which contains expert trajectories from different Minecraft worlds. Using expert trajectories may seem preferable since they contain human priors that give more weight to those states that are meaningful for discovering useful skills. However, Figure 1b shows that expert trajectories from pixels discover fewer and sparser skills than the those discovered by random exploration in our map, depicted in Figure 1c. This suggests that while the skills discovered by experts may be more generic as they were collected in different worlds, they are not as useful for our particular map as the skills discovered by a random policy. This hypothesis is supported by Figure 2, where we show the reconstructed images for each of the VQ-VAE codebook centroids. The reconstructions belonging to the expert trajectories contain scenarios that cannot be found in our Minecraft map.

In our study case, we aim to learn policies that treat the latent variables as navigation-goals. For this purpose, Figure 1 shows complementary skills discovered from pixels (Figure 1c) or coordinates (Figure 1d). We would like to discover skills that take into account not only the visual similarity but also the position relative to the initial state, so we adopt a solution that considers the two types of state representations (Figure 1e). This way our agent can distinguish between two visually identical mountains located at two different positions in the map.

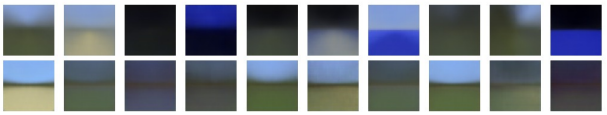


Figure 2. **Top:** images reconstructed from learned centroids using expert trajectories. **Bottom:** images reconstructed from learned centroids using pixels and coordinates from random trajectories.

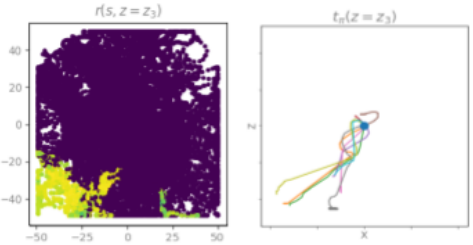


Figure 3. **Left:** Observations in yellow are encoded to the codebook embeddding z_3. **Right:** Trajectories followed by the agent conditioned by z_3.

2.2. Skill-Learning

In the last stage of PiCoEDL, we leverage Equation 2, derived from maximizing the mutual information between states and latent variables to maximize the expected cumulative reward. The latent codes discovered are now treated as goal states in a navigation task. We utilize Rainbow [11] algorithm to train our RL embodied agent. The input to the network is composed of the concatenation of the embedded observation with the latent embedding that is conditioning the policy. For each episode, we sample uniformly from $p(z)$ to determine the conditioning latent.

While there are some policies that are correctly learned, we find some others that do not achieve satisfactory results. We hypothesize that these are the latent codes that encode smaller regions of the state space, and with further tuning may achieve the desirable results. Figure 3 depicts the trained policy conditioned with the third codebook. More examples are available in our project site[1].

[1] https://imatge-upc.github.io/PiCoEDL/

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[2] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. 1

[3] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giro-i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020. 1

[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers, 2021. 1

[5] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised visual transformers. *arXiv preprint arXiv:2104.02057*, 2021. 1

[6] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 1

[7] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. 1

[8] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. In *IJCAI*, 2019. 2

[9] W. H. Guss, M. Y. Castro, S. Devlin, B. Houghton, N. S. Kuno, C. Loomis, S. Milani, S. Mohanty, K. Nakata, R. Salakhutdinov, et al. The minerl 2020 competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:2101.11071*, 2021. 1

[10] D. Hafner, P. A. Ortega, J. Ba, T. Parr, K. Friston, and N. Heess. Action and perception as divergence minimization. *arXiv preprint arXiv:2009.01791*, 2020. 1

[11] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018. 2

[12] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019. 1

[13] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017. 1

[14] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017. 1

[15] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018. 1

PixelEDL: Unsupervised Skill Discovery and Learning from Pixels

Roger Creus Castanyer[1] Juan José Nieto[1] Xavier Giro-i-Nieto[1,2,3]
[1]*Universitat Politècnica de Catalunya* [2]*Barcelona Supercomputing Center* [3]*Institut de Robòtica i Informàtica Industrial, CSIC-UPC*
creus99@protonmail.com juanjo.3ns@gmail.com xavier.giro@upc.edu

1. Introduction

We tackle embodied visual navigation in a task-agnostic set-up by putting the focus on the unsupervised discovery of skills (or options [2]) that provide a good coverage of states. Our approach intersects with empowerment [10]: we address the reward-free skill discovery and learning tasks to discover *what* can be done in an environment and *how*. For this reason, we adopt the existing Explore, Discover and Learn (EDL) [1] paradigm, tested only in toy example mazes, and extend it to pixel-based state representations available for embodied AI agents. The information-theoretic paradigm of EDL [1] aims to learn latent-conditioned policies, namely skills $\pi(a|s,z)$, by maximizing the mutual information (MI) between the inputs s and some latent variables z. Hence, EDL [1] consists of unsupervised skill discovery and training of reinforcement learning (RL) agents without considering the existence of any extrinsic motivation or reward. We present *PixelEDL*, an implementation of the EDL paradigm for the pixel representations provided by the AI Habitat [11] and MineRL [3] environments. In comparison with EDL, PixelEDL involves self-supervised representation learning of image observations for information-theoretic skill discovery. Still, PixelEDL aims to maximize the MI between inputs and some latent variables and for that it consists of the same three stages of EDL (explore, discover and learn). By breaking down the RL end-to-end training pipeline into the three stages, we also simplify the implicit difficulty in learning both representations and policies from a high dimensional input space all at once [7].

Figure 1. Top-down views of the three considered environments: (i) a custom "toy example" Minecraft map, (ii) a Realistic Minecraft map, and (iii) a Habitat apartment.

The results presented in this extended abstract are further extended in our project site[1].

2. Methodology

We assume an underlying Markov Decision Process (MDP) without rewards: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$ where $\mathcal{S}$ is the high-dimensional set of states (defined by image pixels). $\mathcal{A}$ is the action space and $\mathcal{P} = p(s|s,a)$ is the transition function. Moreover, we define the objective of PixelEDL as the maximization of the MI in equation (1), which requires knowledge of the unknown distributions $p(s), p(s|z), p(z|s)$.

$$\begin{aligned} \mathcal{I}(S,Z) &= \mathcal{H}(Z) - \mathcal{H}(Z|S) \quad \rightarrow \quad \text{reverse} \\ &= \mathcal{H}(S) - \mathcal{H}(S|Z) \quad \rightarrow \quad \text{forward} \end{aligned} \tag{1}$$

2.1. Exploration

The first task to tackle in PixelEDL is exploration. Without any prior knowledge, a reasonable choice for discovering state-covering skills is to define the distribution over all states $p(s)$ uniformly. However, training an exploration policy to infer a uniform $p(s)$ is not feasible in PixelEDL since it deals with the high-dimensional pixel space. To overcome this limitation we adopt a non-parametric estimation of $p(s)$ by sampling from a dataset of collected experience. Hence, in PixelEDL the goal of the exploration stage is to collect a dataset of trajectories containing representative states that the learned skills should ultimately cover. PixelEDL adopts a random exploration of the environment through agents that perform random actions within a discrete action space (i.e. move forward, turn left, turn right) and collect the trajectories generated by the environment. For our custom Minecraft map, random policies from agents instantiated in the center of the map are capable of covering a complete set of representative states of the environment given a large number of episodes. In the realistic Minecraft map the random agents do not cover as many representative states as in the custom map but still provide enough coverage of the state space. However, in order to obtain a set

[1] https://imatge-upc.github.io/PixelEDL/

of representative states of the Habitat map we let the agents start in random navigable points of the map at each episode.

2.2. Skill Discovery

The Discovery stage of EDL aims at finding the latent representations z that will ultimately condition the agent policies to learn the skills $\pi(a|s, z)$. Hence, the goal of PixelEDL in this stage is to model $p(z|s)$ as a mapping of the states to their representations and to model $p(z)$ as a categorical distribution of meaningful representations.

Ideally, we aim to obtain representations of the image observations that encode existing similarities and spatial relations within the environment [6]. Furthermore, we aim to find z that are representatives of a meaningful segmentation of the state space. In this work the representations z will be later used to condition a navigation task. Previous works [16] have reported the challenges of unsupervised learning of representations from images that encode valuable features for RL agents in a 3D environment. For modelling $p(z|s)$, we study the performance of two different approaches: (i) a *contrastive* one, that uses a siamese architecture and aims to project positive pairs of input images closer in an embedding space, and (ii) a *reconstruction* one, that use a Variational Autoencoder (VAE) [5] with categorical classes, namely Vector Quantisation VAE (VQ-VAE) [9], to train the model to reconstruct the observations. For the contrastive approach, we use the adaptation to CURL [14] proposed by Stooke et al. [15], namely Augmented Temporal Contrast (ATC). Compared to CURL, in ATC the positive pairs of inputs consist of two image observations belonging to the same exploration trajectory. That is, we train both ATC and VQ-VAE so that a positive pair of inputs consists of two observations of the same trajectory with a delay $d \sim \mathcal{N}(\mu, \sigma^2)$. We experiment with $\mu = 15$ and $\sigma = 5$. Hence, in both ATC and VQ-VAE we perform a data augmentation in the temporal domain. Our experiments indicate that the capabilities of both ATC and VQ-VAE for modelling $p(z|s)$ are promising and we have not yet observed important differences to justify using one over the other.

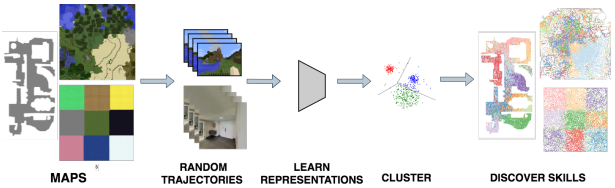


Figure 2. Self-Supervised representation learning and unsupervised skill discovery pipeline.

After the visual representation learning, we model a categorical distribuiton $p(z)$ by clustering the embedding space of the representations. Yarats et. al [17] use a projection of the embeddings onto the prototypes which define a basis of the embedding space to perform the cluster assignments. However, in VQ-VAE, this clustering is implicit in the model since the cluster centroids are actually the representatives of the model's codebook. Also, for ATC we apply a K-means [8] clustering for modelling $p(z)$ with the cluster centroids. After modelling $p(z)$ and $p(z|s)$, we complete the stages of representation learning and skill discovery. Figure 2 summarises the aforementioned pipeline. We provide more details in our project site.

2.3. Learning

Given a model of $p(z)$, we make use of the formulation of Universal Value Function Approximators (UVFA) [12] to train a policy to maximize the MI (1) between the inputs and z. That is, we exploit z as navigation goals or intrinsic objectives to learn the goal-conditioned skills: $\pi(a|s, z)$. Hence, we feed the concatenation of the encoded observation and z to the RL agents. Thus, at each step, the policy predictions depend not only on the current agent state but also on z. EDL [1] maximizes the forward form of the MI (1). That is feasible in EDL because the technique is applied to toy mazes where the states of the MDP are defined by 2D coordinates. In this way, EDL models $p(s|z)$ by variational inference and maximize the MI by deriving a reward that involves computing euclidean distances in the state space of coordinates. However, as in PixelEDL we deal with the pixel space, it is not coherent to match the euclidean distance in the image observation space with the distances in the 3D environment. For this reason we make use of the reverse form of the MI (1) and we model $p(z|s)$ with the encoder that learns latent representations from image observations. Finally, we craft a reward distribution that maximizes the MI (1) between the inputs and the skills by taking into account the distances in the latent space of the representations. Concretely, we assign a positive reward to an action a that positions the agents in a state s only if the encoded image observation is closest to the skill-conditioning z among all $z \sim p(z)$. We use the baseline RL models provided by both Habitat and MineRL for implementing the aforementioned training pipeline. These models are Proximal Policy Optimization (PPO) [13] and Rainbow [4] respectively.

While PixelEDL is capable of learning some of the discovered skills, specially in the custom Minecraft map, it finds more difficulties in the realistic one and in Habitat. We hypothesize that: (i) Rainbow struggles with the latent codes z that encode similar regions of the realistic Minecraft state space; (ii) further tuning of PPO could achieve better results when learning the skills in Habitat, since we already obtain high-quality discovery of these. We provide qualitative results together with a demo video in our project site.

References

[1] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020. 1, 2

[2] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. 1

[3] William H. Guss, Mario Ynocente Castro*, Sam Devlin*, Brandon Houghton*, Noboru Sean Kuno*, Crissman Loomis*, Stephanie Milani*, Sharada Mohanty*, Keisuke Nakata*, Ruslan Salakhutdinov*, John Schulman*, Shinya Shiroshita*, Nicholay Topin*, Avinash Ummadisingu*, and Oriol Vinyals*. Neurips 2020 competition: The MineRL competition on sample efficient reinforcement learning using human priors. *NeurIPS Competition Track*, 2020. 1

[4] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[6] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010. 2

[7] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019. 1

[8] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 2

[9] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017. 2

[10] Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014. 1

[11] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019. 1

[12] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015. 2

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2

[14] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020. 2

[15] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020. 2

[16] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018. 2

[17] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021. 2