DisTop: Discovering a Topological representation to learn diverse and rewarding skills

Anonymous authors
Paper under double-blind review

Abstract

An efficient way for a deep reinforcement learning agent to explore in sparse-rewards settings can be to learn a set of skills that achieves a uniform distribution of terminal states. We introduce DisTop, a new model that simultaneously learns diverse skills and focuses on improving rewarding skills. DisTop progressively builds a discrete topology of the environment using an unsupervised contrastive loss, a growing network and a goal-conditioned policy. Using this topology, a state-independent hierarchical policy can select which skill to execute and learn. In turn, the new set of visited states allows an improved learnt representation. If the agent gets overloaded by the number of skills, the agent can autonomously forget the skills unrelated to its eventual task. Our experiments emphasize that DisTop is agnostic to the ground state representation and that the agent can discover the topology of its environment whether the states are high-dimensional binary data, images, or proprioceptive inputs. We demonstrate that this paradigm is competitive on MuJoCo benchmarks with state-of-the-art algorithms on both single-task dense rewards and diverse skill discovery without rewards. By combining these two aspects, we show that DisTop outperforms a state-of-the-art hierarchical reinforcement learning algorithm when rewards are sparse. We believe DisTop opens new perspectives by showing that bottom-up skill discovery combined with dynamic-aware representation learning can tackle different complex state spaces and reward settings.

1 Introduction

In reinforcement learning (RL), an autonomous agent learns to solve a task by interacting with its environment (Sutton & Barto, 1998) thus receiving a reward that has to be hand-engineered by an expert to efficiently guide the agent. However, when the reward is sparse or not available, standard methods do not well-explore their environment (Aubret et al., 2019). One way to improve exploration is to reward the agent with a prediction-based (Burda et al., 2019) or count-based (Bellemare et al., 2016) flat intrinsic bonuses, making the agent reach unvisited or surprising states. However such methods only learn one policy that sequentially targets new areas; this sequential exploration prevents the simultaneous exploration of several new areas, which often makes the agent forgets how to reach them, thereby hurting exploration (Ecoffet et al., 2021).

This issue partially motivated methods where an agent hierarchically commits to a temporally extended behavior named skills or *options* (Sutton et al., 1999).While it is possible to learn hierarchical skills with an extrinsic (*i.e* task-specific) reward (Bacon et al., 2017), it does not fully address the exploration issue. In contrast, if an agent learns skills in a *bottom-up* way with intrinsic rewards, it makes the skills task-independent (Aubret et al., 2019): they are learnt without access to an extrinsic reward. It follows that, to explore without extrinsic rewards, the intrinsic objective of an agent may be to acquire a large and diverse repertoire of skills. While the methods that follow this paradigm (Pong et al., 2020b; Eysenbach et al., 2019b) manage to learn a large set of different skills that simultaneously explore different areas, they are learnt during an unsupervised pre-training phase, preventing online skill specialization to a task. In contrast, we assume that an agent should both explore by learning diverse skills and specialize its skills to a task when provided, without separating the learning process in different phases, making possible to use the same algorithm on different reward settings (sparse, dense, no-rewards). Similarly, extra assumptions like accessing an expertly built goal/state space (*e.g* (x,y) coordinates (Nachum et al., 2018b)) are not always available, thus, an algorithm should be agnostic to the state space. In other words, **an agent should be able to learn interesting skills whatever are the reward setting and state space**, typical of how humans manage to learn in different settings with different modalities (images, proprioceptive, . . .).

In this paper, we introduce a new way to learn a goal space and select the goals, bridging the gap between acquiring skills that reach a uniform density distribution of terminal embedded states and solving a particular task. We propose a new model that progressively **Dis**covers a **Top**ological representation (**DisTop**) of the environment. DisTop simultaneously optimizes two new components: 1- it learns a continuous representation of its states using a contrastive loss that brings together consecutive states while avoiding the distortion of the state representation, *i.e* the maximal embedded L2 distance covered by one action is similar everywhere; 2- this property allows DisTop to build a meaningful discrete topology of the environment with a new self-organizing growing network named Off-policy Embedded Growing Network (OEGN). Thanks to the clusters, DisTop easily estimates the expected extrinsic rewards and the count-based density of clusters using few parameters. With these values, a state-independent Boltzmann policy autonomously selects which skills to execute and learn. Finally, DisTop trains a goal-conditioned deep RL policy to reach embedded states.

Our experimental contribution shows the genericity and efficiency of selecting skills to improve using a topological representation. This contribution is 3-folds: 1- we show that DisTop competes with a state-of-the-art (SOTA) algorithm specific to single-task dense rewards settings thanks to its ability to select rewarding goals; 2- we demonstrate that, because it selects goals that maximize both the density of visited states and the extrinsic rewards, DisTop competes with/outperforms benchmark-specific SOTA algorithms on diverse skills learning benchmarks and sparse rewards environments; 3- we demonstrate that DisTop outperforms a recent method that shares the properties of DisTop.

2 Background

2.1 Goal-conditioned reinforcement learning for skill discovery

In episodic RL, the problem is modelled with a Markov Decision Process (MDP). An agent interacts with an unknown environment in the following way: the agent gets a state $s_0 \in S$ that follows an initial state distribution $\rho_0(s)$. Its policy π selects actions $a \in A$ to execute depending on its current state $s \in S$, $a \sim \pi(\cdot|s)$. Then, a new state is returned according to the transition dynamics $s' \sim p(\cdot|s, a)$. The agent repeats the procedure until it reaches a particular state or exceeds a fixed number of steps T. This loop is called an episode. An agent learns π to maximize the expected cumulative discounted reward $\mathbb{E}_\pi\left[\sum_{t=0}^{T} \gamma^t R(s_{t-1}, a_{t-1}, s_t)\right]$ where γ is the discount factor. The policy π_θ can be approximated with a neural network parameterized by θ (Haarnoja et al., 2018). In our case, the agent tries to learn *skills*, defined as a policy that tries to target a goal-state (or *goal*). To this end, it computes an intrinsic reward according to the chosen skill (Aubret et al., 2019). Following *Universal Value Function Approximators* (Schaul et al., 2015), the intrinsic reward and the skill's policy become dependent on the chosen goal g_t: $r_t^g = R_\omega(s_{t-1}, a_{t-1}, s_t, g_t)$ and $\pi_\theta^g(\cdot|s_t) = \pi_\theta(\cdot|s_t, g_t)$. With this scheme, we can relabel an interaction with an arbitrary goal and recompute the intrinsic reward (Andrychowicz et al., 2017).

2.2 Skew-Fit

Skew-Fit (Pong et al., 2020a) strives to learn goal-conditioned policies that visit a maximal entropy of states, where all states are considered as possible goals. To generate unvisited goal-states, it learns a generative model that incrementally increases the entropy of generated goals and makes visited states progressively tend towards a uniform density distribution. Assuming it learns a parameterized generative model $q_\psi(s)$, Skew-Fit uses importance sampling to skew the state density distribution closer to the uniform one $q_\psi(s)^{\alpha_{skew}}$ where $\alpha_{skew} < 0$. α_{skew} defines the importance of low-density states; if $\alpha_{skew} = 0$, the density distribution will stay similar, if $\alpha_{skew} = -1$, it strongly over-weights low-density states to approximate $U(S)$. In practice, they show that directly sampling states $s \sim \frac{1}{Z} q_\psi(s)^{\alpha_{skew}}$, with Z being the normalization constant reduces the variance.

2.3 Contrastive learning and InfoNCE

Contrastive learning aims at discovering a similarity metric between data points (Chopra et al., 2005). First, positive pairs that represent similar data and fake negative pairs are built. Second

Algorithm 1: Summary of the learning algorithm.

```
OEGN ← two connected random nodes.
foreach learning iteration do
    Sample $BatchSize * Ratio$ clusters $c_1 \sim \pi^{high}(c)$, equation 5 in section 3.2.
    Sample $BatchSize * (1 - Ratio)$ clusters $c_2 \sim p_{\alpha_{skew}}(c)$, equation 3 in section 3.2.
    Sample buffers $\mathbb{B}_1 \sim Uniform(\{\mathbb{B}^G_{c_1}, \mathbb{B}^S_{c_1}\})$, $\mathbb{B}_2 \sim Uniform(\{\mathbb{B}^G_{c_2}1, \mathbb{B}^S_{c_2}\})$.
    Build minibatch of interactions $concat(i_1 \sim \mathbb{B}_1, i_2 \sim \mathbb{B}_2)$, section 3.2.
    Update the representation function $\phi_\omega$ by maximizing equation 2 in section 3.1.
    Update $\omega'$ with an exponential moving average of $\omega$.
    Compute embeddings of goals $g = \phi_{\omega'}(s_g)$ and observations $e = \phi_{\omega'}(s)$.
    Relabel interactions (change goals) (cf. appendix B.5).
    Update $\pi^g_\theta$ with rewards $r = ||e - g||_2$ and SAC (Haarnoja et al., 2018).
    Update OEGN, section 3.1 and algorithm 2 in appendix B.3 .
```

they are given to an algorithm that computes a data representation able to discriminate the positive from the negative ones. InfoNCE (van den Oord et al., 2018) proposed to build positive pairs from states that belong to the same trajectory, making the process scale to all input representations. The negative pairs are built with states randomly sampled. For a sampled trajectory, it builds a set $\mathbb{B}$ that concatenates $N-1$ negative states and a positive one s_{t+p}. Then, it maximizes $\mathbb{L}^N_{InfoNCE} = \mathbb{E}_{\mathbb{B}}\big[\log \frac{f_p(s_{t+p}, v_t)}{\sum_{s_j \in \mathbb{B}} f_p(s_j, v_t)}\big]$ where v_t represents an aggregation of previous states learnt with an autoregressive model, p indicates the additional number of steps needed to select the positive sample and f_p are positive similarity functions. The numerator brings together states that are part of the same trajectory and the denominator pushes away negative pairs.

3 Method

Problem statement. We are interesting in learning skills that simultaneously maximize the state entropy and extrinsic rewards when provided, thereby supposing the ability to select extrinsically rewarding/unvisited states as goals. Furthermore, we want our approach to be agnostic to the ground state space, meaning that it should work even in very unstructured state spaces.

Overview. We propose to skew the goal-sampling process in a learnt dynamic-aware representation of states. Algorithm 1 sums up our algorithm. DisTop samples clusters of interactions that are rewarding or rarely visited, the corresponding buffers and finally interactions (s_{prev}, a, s, s_g), where s_g is the original goal-state. Then, it learns to embed the states with a neural network and a contrastive loss such that the embeddings reflect the inherent topology of the environment; this allows to both provide meaningful intrinsic rewards and correctly estimate the dynamic-aware density of states. After that, our growing network dynamically clusters the embedded states. This clustering method allows DisTop to estimate both the mean reward achieved while targeting a state of the cluster and the density of a visited state by counting the number of interactions its cluster contains; it only requires a small number of parameters in contrast with learning a generative model (Lee et al., 2019). Finally, it assigns buffers $\mathbb{B}^G$ and $\mathbb{B}^S$ to clusters and samples goal-states (B^G) or states (B^S) with a skewed version of the visited state density distribution.

At the beginning of each episode, a Boltzmann policy π_{high} selects a cluster c; then DisTop selects a state s that belongs to the buffer B^S_c of the selected cluster and it computes the embedding $g = \phi_{\omega'}(s)$ where weights ω' are an exponential moving average of ω. A goal-conditioned policy π^g_θ, trained to reach g, acts in the environment and discovers new reachable states close to its goal. The interactions made by the policy are stored in a buffer according to their initial objective (B^G) and the embedding of the reached state (B^S) (see section 3.2).

3.1 Learning the topology of the states

In order to learn the discrete topology of the known state-goals, the agent passes through two steps: 1- it learns a continuous representation of states undistorted with respect to the environment dynamics; 2- it clusters this representation and tracks its changes.

Learning an undistorted representation. DisTop strives to learn a *state representation* function ϕ_ω that maps a given state to an embedded space. We call *consecutive states*, pairs of states that are separated by one action, *i.e* pairs for which there exists a transition. To do this, DisTop takes advantage of the InfoNCE loss (cf. section 2.3). Similarly to Li et al. (2021), we do not consider the whole sequence of interactions since this makes it more dependent on the policy. Typically, a constant and deterministic policy would lose the structure of the environment and could emerge once the agent has converged to an optimal deterministic policy. DisTop considers its *local* variant and builds positive pairs with only two consecutive states. This way, it keeps distinct the states that cannot be consecutive and it more accurately matches the topology of the environment. We select our similarity function as $f_\omega(s_t, s_{t+1}) = e^{-k||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2}$ where ϕ_ω is a neural network parameterized by ω and k is a temperature hyper-parameter (Chen et al., 2020). If $\mathbb{B}$ is a batch of N different consecutive pairs, the local InfoNCE objective, LInfoNCE, is described by:

$$\begin{aligned}\mathbb{L}_{LInfoNCE} &= \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[\log \frac{f_\omega(s_t,s_{t+1})}{\sum_{s\in\mathbb{B}} f_\omega(s_t,s)}\big] \\ &\geq \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[-k||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2 - \log(1+\sum_{s\in\mathbb{B}_{s\neq s_{t+1}}} f_\omega(s,s_{t+1}))\big]. \quad (1)\end{aligned}$$

We introduce the lower bound (cf. appendix B.1) since it stabilizes the learning process. Intuitively, equation 1 brings together consecutive states, and pushes away states that are separated by a large number of actions. There are several drawbacks with this objective function. Firstly, the representation may still be strongly **distorted**, *i.e* the agent may cover more distance with one action in some part of the embedded space than in others. In this case, the learnt clusters (see below) may encompass different number of states, preventing their use for density estimation. Secondly, the DRL algorithm requires semantically stable inputs to compute Q-values: if the representation of its goals quickly changes, it cannot take into account the changes and may output bad approximations of Q-values. In contrast with previous approaches (cf. section 5), we tackle these issues by reformulating the learning objective as two-folds constrained maximization (cf. equation 9 in appendix B.1) in order to keep our representation stable and locally undistorted (Indyk, 2001). In practice, in order to perform a stochastic gradient descent, we transform the constraints into penalties and maximize:

$$\begin{aligned}\mathbb{L}_{DisTop} &= \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[-k_c(\max(||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2^2-\delta^2),0) - \log(1+\sum_{s\in\mathbb{B}} f_\omega(s,s_{t+1})) \\ &\quad -\beta||\phi_\omega(s_{t+1})-\phi_{\omega'}(s_{t+1})||_2^2\big]. \quad (2)\end{aligned}$$

Firstly, DisTop forces the distance between consecutive states to be below a threshold δ, avoiding the collapse of parts of the representation; $k_c > 1$ is the dilatation coefficient that brings together consecutive states. Secondly, β encourages our representation to stay consistent over time, weights ω' being a slow exponential moving average of ω. Thus, we avoid using a hand-engineered environment-specific scheduling (Pong et al., 2020a) and update representations in an online way.

By applying this objective function, DisTop learns a consistent, undistorted representation that keeps close consecutive states while avoiding collapse (see Figure 4 for an example). In fact, by modifying k_c and/or k, one can respectively select the level of dilatation and distortion of the representation (cf. appendix A for an analysis). One can notice that the function depends on the density of consecutive states in $\mathbb{B}$; we discuss the density distribution of states that feeds $\mathbb{B}$ in section 3.2.

Clustering the embeddings to obtain a discrete topology Since our representation strives to avoid distortion by keeping close consecutive states, DisTop can match the topology of the embedded environment by clustering the embedded states. Using this topology, the next section details how to bias the goal and state sampling procedure to favor exploration and maximize extrinsic rewards.

In order to adapt the clustering to temporally-related and goal-related interactions, we propose a new variant of a previous method (Marsland et al., 2002), which we call OEGN. OEGN dynamically creates, moves and deletes clusters so that clusters generate a network that covers the whole set of embedded states, independently of their density. Each node/cluster $c \in C$ has a reference vector w_c which represents its position in the input space, an embedding is assigned to its closest cluster, *i.e* $min_c||\phi_{\omega'}(s)-w_c)||_2 \leq \delta_{new}$ where δ_{new} is the radius of the ball and the threshold for creating clusters. δ_{new} is particularly important since it is responsible of the granularity of the clustering: a low value will induce a lot of small clusters, while a high one induce few large clusters that badly approximate the topology (cf. appendix A.1). A learnt topology can be visualized in Figure 2.

Essentially, the *delete* operator removes unused nodes, the *creation* operator creates a node if an embedding falls outside the radius of all clusters and the *moving* operator resembles the k-means clustering. We show section 4 that the *delete* and *creation* operators are critical to adapt the clustering to the dynamically learnt continuous representation (cf. section 4). Edges are added between two clusters when the agent passes from one cluster to the other, and deleted after a cluster-normalized delay of uselessness. Overall, these operators make sure that each embedding belongs to the ball centered on the center of its cluster. Technical details about OEGN can be found in appendix B.3.

3.2 Selecting novel or rewarding skills

While sampling low-density or rewarding states is attractive to solve a task, it is not easy to sample new reachable goals. For instance, using an embedding space $\mathbb{R}^3$, the topology of the environment may only exploit a small part of it, such that randomly sampling goals inside $\mathbb{R}^3$ would mostly result in unreachable goals. Typically, one can see in figure 2 that embeddings of states only exploit a small part of the whole space. To make sure goals are reachable, DisTop generates goals by sampling previously visited states (similarly to Warde-Farley et al. (2019)). To sample the goals-states or states, DisTop approximates the density of states with a density over clusters, samples a cluster, an associated buffer, and a state. This sampling procedure applies for both selecting a goal to execute and sampling a minibatch, but with different desired density distribution of states.

Approximating an arbitrary density distribution of goals and states. We assume that the density of a visited state is reflected by the probability of a state being in its cluster. So we approximate the density of a state with the density parameterized by w, reference vector of $e = \phi_{\omega'}(s)$: $q_w(e) \approx \frac{count(c_e)}{\sum_{c' \in C} count(c')}$ where $count(c_e)$ denotes the number of embedded states that belong to the cluster that contains e. While our approximation $q_w(s)$ decreases the quality of our density approximation, it makes our algorithm efficient: we associate a buffer to each cluster and only compute the density of clusters. In practice, we trade-off the bias of $q_w(s)$ and the instability of OEGN by decreasing δ_{new}: the smaller the clusters are, the smaller the bias is, but the more unstable is OEGN (cf. appendix A).

Using this approximation, we just need to keep states in the buffer of their closest node to sample from an arbitrary density distribution. In practice, we associate to each node a second buffer that takes in interactions according to the proximity of the original goal with respect to the node. This results in two sets of buffers, $\mathbb{B}^G$ and $\mathbb{B}^S$, to respectively sample goal-states and states according to the desired density distribution (*e.g* a uniform one). Formally, in $\mathbb{B}^S$, DisTop selects the cluster $\arg\min_{c \in C} ||\phi(s) - c||_2$; in $\mathbb{B}^G$, DisTop selects the cluster $\arg\min_{c \in C} ||\phi(s_g) - c||_2$.

Sampling goal-states: which density distribution ? In order to sample new exploratory goals for the goal-conditioned policy, DisTop increases the entropy of visited terminal states by skewing the current density distribution of visited states (cf. section 2.1) with:

$$p_{\alpha_{skew}}(c) = \frac{count(c)^{1+\alpha_{skew}}}{\sum_{c' \in C} count(c')^{1+\alpha_{skew}}} = \frac{e^{(1+\alpha_{skew}) \log count(c)}}{\sum_{c' \in C} e^{(1+\alpha_{skew}) \log count(c')}}, \tag{3}$$

where α_{skew} is the skewed parameter (cf. section 2). It is equivalent to sampling with a simple Boltzmann policy π^{high}, using a novelty bonus reward $\log count(c)$ and a fixed temperature parameter $1 + \alpha_{skew}$. We can add a second reward to modify our skewed policy π^{high} so as to take into consideration extrinsic rewards. We associate to a cluster $c \in C$ a value R_c^{ext} that represents the average extrinsic rewards received by the skills associated to its goals :

$$R_c^{ext} = \mathbb{E}_{s \in c,\, g = \phi_\omega(s),\, a_t \sim \pi_\theta^g(\cdot|s_t),\, s_{t+1} \sim p(\cdot|s_t, a_t)} R(s_t, a_t, s_{t+1}). \tag{4}$$

The extrinsic value of a cluster R_c^{ext} is updated with an exponential moving average where α_c is the learning rate. To favour exploration, we can also update the extrinsic value of the neighbors of the final state's cluster with a slower learning rate (cf. appendix B.5). Our sampling rule can then be:

$$\pi^{high}(c) = \text{softmax}_C(t^{ext} \underbrace{R_c^{ext}}_{\text{Sample rewarding goals}} + (1 + \alpha_{skew}) \underbrace{\log count(c)}_{\text{Sample unvisited states}}). \tag{5}$$

Finally, the agent selects a cluster $c_1 \sim \pi^{high}(\cdot)$, its state-buffer $\mathbb{B}^S_{c_1}$ and the agent samples a state with an approximation of the uniform density distribution inside the cluster's c_1 ball of radius δ_{new} (cf. appendix B.4).

Sampling a learning minibatch: which density distribution ? Our goal-conditioned policy π^g_θ needs a uniform density distribution of goals to avoid forgetting previously learnt skills while our representation function ϕ_ω requires a uniform density distribution over visited states to quickly learn the representation of novel areas. Therefore, to construct a minibatch, DisTop samples clusters $\sim p_{\alpha_{skew'}}$ with $\alpha_{skew'} \neq \alpha_{skew}$, randomly takes half of buffers in $\mathbb{B}^G$ and the rest from $\mathbb{B}^S$ and samples interactions. We can also sample a $Ratio$ of clusters with π^{high} if we do not care about forgetting skills (cf. section 4).

4 Experiments

Our experiments aim at studying whether the ability of DisTop to select low-density/rewarding goals in an undistorted dynamic-aware latent space makes the approach generic to different task settings/state spaces. We compare DisTop to SOTA algorithms on three kinds of tasks with different ground state spaces. All curves are a smooth averaged over 5 seeds, except for ablation experiments which are averaged over 3 seeds, completed with a shaded area indicating the mean +/- standard deviation over seeds. We provide used hyper-parameters in appendix C.2 and detail environments and evaluation protocols in appendix C.3. Images of achieved goals are available in appendix C.5.

Is DisTop able to learn diverse skills without rewards ? We assess the diversity of learnt skills on two robotic tasks *Visual Pusher* (a robotic arm moves in 2D and eventually pushes a puck) and *Visual Door* (a robotic arm moves in 3D and can open a door) (Nair et al., 2018). These environments are particularly challenging since the agent has to learn skill from 48x48 images using continuous actions without accessing extrinsic rewards. We compare to the SOTA algorithm Skew-Fit (Pong et al., 2020b), which skews the visited state density distribution in the ground state space with a VAE (Kingma & Welling, 2014) and periodically updates its representation. We use the same evaluation protocol than Skew-Fit: a set of diverse images are sampled, given to the representation function and the goal-conditioned policy executes the skill. In Figure 1(left), we observe that skills of DisTop are learnt quicker on *Visual Pusher* but are slightly worst on *Visual Door*. Since both Skew-Fit and DisTop generates rewards with the L2 distance, we hypothesize that this is due to the structure of the learnt goal space. In practice we observed that DisTop is more stochastic than Skew-Fit, probably because the intrinsic reward function is required to be smooth over consecutive states. It results that a one-step complete change of the door angle creates a small change in the representation, thereby a small negative intrinsic reward. Despite the stochasticity, it is able to reach the goal as evidenced by the minimal distance reached through the episode (Distop(min)). Stochasticity does not bother the evaluation on *Visual Pusher* since the arm moves elsewhere after pushing the puck in the right position. Therefore, **DisTop manages to learn slightly more or less diverse skills than Skew-fit, depending on the environment structure.**

Can DisTop solve non-hierarchical dense rewards tasks? We test DisTop on MuJoCo environments *Halfcheetah-v2* and *Ant-v2* (Todorov et al., 2012), where the agent gets rewards to move forward as fast as possible. We fix the maximal number of timesteps to 300. We compare DisTop to SAC (Haarnoja et al., 2018) and ELSIM (Aubret et al., 2020), a method that follows the same paradigm than DisTop (see section 5). We obtain better results than in the original ELSIM paper using similar hyper-parameters to DisTop. In Figure 1 (Right), we observe that DisTop obtains high extrinsic rewards and clearly outperforms ELSIM. It also outperforms SAC on *Halfcheetah-v2* and is close to SAC on *Ant-v2*. In contrast, we highlight that SAC overfits to dense rewards settings and cannot learn in sparse or no-reward settings (see below). **Despite the genericity of DisTop and the narrowness of SAC, DisTop competes with SAC on two environments.**

Does combining representation learning, entropy of states maximization and task-learning improve exploration on high-dimensional hierarchical tasks ? We evaluate DisTop ability to explore and optimize rewards on image-based versions of *Ant Maze* and *Point Maze* environments (Nachum et al., 2018b). The state is composed of a proprioceptive state of the agent and a top view of the maze ("Image"). Details about state and action spaces are given in appendix C.3.2. In contrast

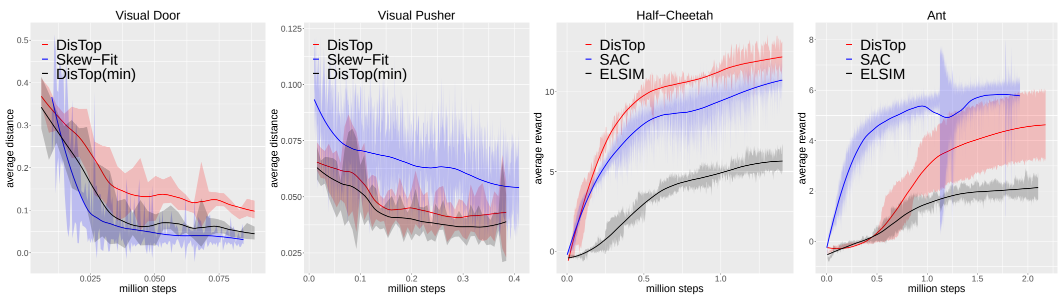


Figure 1: **Left:** Comparison of DisTop and Skew-Fit on their ability to reach diverse states. In the *Visual Pusher* environment, we compare the final distance of the position puck with its desired position; in the door environment, we compare the angle of the door with the desired angle. DisTop(min) is the minimal distance reached through evaluation episode. At each evaluation iteration, the distances are averaged over fifty goals. **Right:** Average rewards gathered throughout episodes of 300 steps while training on *Halfcheetah-v2* and *Ant-v2* environments.

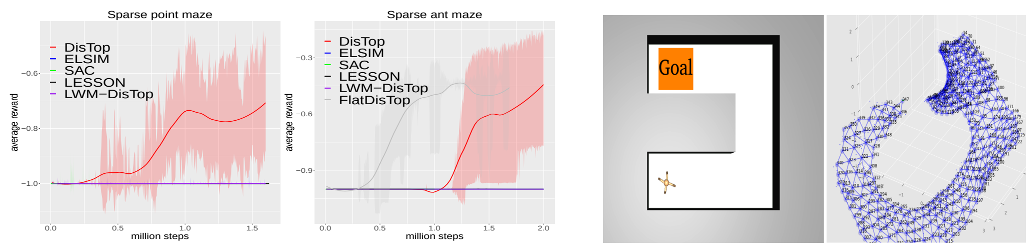


Figure 2: **Left:** Average rewards gathered throughout training episodes. **Right:** Top view of Ant Maze environment with its goal position and an example of OEGN network learnt by DisTop.

with previous methods (Li et al., 2021), we remove the implicit *curriculum* that changes the extrinsic goal across episodes; here we train only on the farthest goal and use a sparse reward function. Thus, the agent gets a non-negative reward only when it gets close to the top-left goal. We compare our method with a SOTA hierarchical method, LESSON (Li et al., 2021) since it performs well on hierarchical environments like mazes, ELSIM, SAC (Haarnoja et al., 2018), a prediction-based flat method mixing LWM (Ermolov & Sebe, 2020) with DisTop (LWM-DisTop), a DisTop's variant that rewards a single policy with a count-based bonus like Strehl & Littman (2008) (FlatDisTop). We only pass the "image" to the representation learning part of the algorithms, assuming that an agent can separate its proprioceptive state from the sensorial state. In figure 2, we can see that DisTop is the only method that manages to regularly reach the goal; in fact, looking at a learnt 3D OEGN network at the right of figure 2, we can see that it successfully represents the U-shape form of the maze. LWM-DisTop too quickly learns to predict making the bonus useless for exploration. FlatDisTop outperforms DisTop, suggesting that DisTop's ability to retain previous skills can slow down its skill discovery. LESSON discovers the goal but does not learn to return to it[1]; we hypothesize that this is because, unlike DisTop, it does maximize the entropy of states, and thus hardly reach the goal. Neither SAC, nor ELSIM find the reward. We suppose that the undirected width expansion of the tree of ELSIM does not maximize the state-entropy, making it spend too much time in useless areas and thus inefficient for exploration. **Therefore, DisTop outperforms two hierarchical methods in two sparse rewards environment by priorizing its goal sampling process on low-entropy areas.**

What are the important components of DisTop? We aim at understanding the importance of each novel component of DisTop. In figure 3a, we compare the performance of DisTop according to different levels of distortion/dilatation. We detail in appendix A the relation between our hyper-parameters and the distortion/dilatation of the representation. We see that fixing a good k_c determines the quality of the density approximation, and thus the novelty of the selected goals; using only the consistency penalty of equation 2 (*No constraints*), the agent learns slower than with constrains, probably because it under-estimates the density of states in some areas. Figure 3b shows the importance of dynamically creating clusters by comparing OEGN with a self-organizing map (SOM) (Kohonen, 1990) that learns with a fix number of nodes. We observed that our OEGN ap-

[1]In *Point Maze*, the best seed of LESSON returns to the goal after 5 millions timesteps

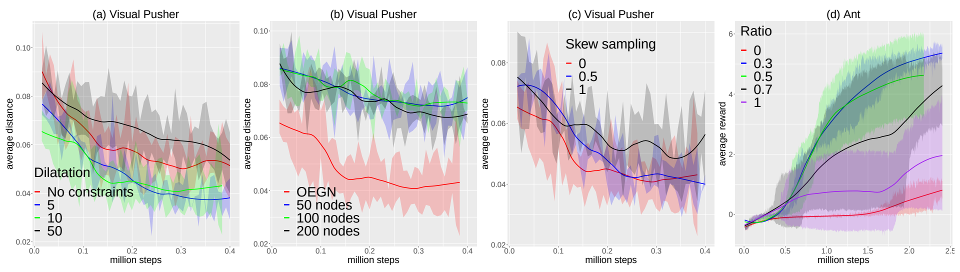


Figure 3: Same environments and evaluation protocols than in figure 1. The agent aims at minimizing the evaluated distance and increasing the average reward. a- comparison of performance in *Visual Pusher* between using no constraints in equation 2 and several values of k_c; b- comparison in *Visual Pusher* of DisTop with OEGN and using a fix number of nodes as in a SOM; c- performance of DisTop in *Visual Pusher* for several skew sampling coefficients $1+\alpha'$; d- evaluation of DisTop in *Ant* for several values of $ratio$.

proximately creates 100 clusters on average on *Visual Pusher*, so we tried 50, 100 and 200 nodes. We clearly see that fix numbers of nodes struggle to generate new skills. We hypothesize that this is due to the resulting bad approximation of the state density: at the beginning, all clusters have a low density while the agent visited few different states. Figure 3c stresses out the importance of skewing the sampling process. The agent focuses its learning process on low-density areas with $1+\alpha_{skew'}$ to speed up representation and policy learning. Overall, DisTop is robust to small changes of $1+\alpha_{skew'}$. Finally, figure 3d shows the importance of sampling a $ratio$ of learning interactions using π^{high} in dense rewards settings (here *Ant*). It results that the agent purposely avoids learning on badly rewarding areas (the policy and representation) so that it does not create clusters in this area and reserves its learning batch for interactions of rewarding skills. At the extremes, trying to learn all skills stuck the agent in local optimas because it tries to retain too much skills ($Ratio = 0$), while learning only in rewarding areas prevents exploration ($Ratio = [1, 0.7]$). To sum up, **our penalties and OEGN are critical to well-approximate a density distribution of visited states and skewing it is crucial to efficiently select the skills to learn on.**

5 Related works

Intrinsic skills in hierarchical RL Some works propose to learn hierarchical skills, but do not introduce a default behavior that maximizes the visited state entropy (Hazan et al., 2019), limiting the ability of an agent to explore. For example, it is possible to learn skills that target a ground state or a change in the ground state space (Nachum et al., 2018b; Levy et al., 2019). These approaches do not generalize well with high-dimensional states. Therefore, one may want to generate rewards with a learnt representation of goals, but current methods still rely on hierarchical random walks (Nachum et al., 2018a; Li et al., 2021) or learn the representation during pre-training (Lu et al., 2019a). We have shown in section 4 that DisTop explores quicker by maximizing the entropy of states in its topological representation. DCO (Jinnai et al., 2019) generates *options* by approximating the second eigenfunction of the combinatorial graph Laplacian of the MDP. It extends previous works (Machado et al., 2017; Bar et al., 2020) to continuous state spaces.

Intrinsic motivation to learn diverse skills. DisTop simultaneously learns skills, their goal representation, and which skill to train on. It contrasts with several methods that focus on selecting *which skill to train on* assuming a good goal representation is available (Florensa et al., 2018; Colas et al., 2018; Zhang et al., 2020). They either select goals according to a curriculum defined with intermediate difficulty, the learning progress (LP) (Oudeyer & Kaplan, 2009) or by imagining new language-based goals (Colas et al., 2020). In addition, DisTop strives to learn either skills that are diverse or extrinsically rewarding. It differs from a set of prior methods that learn only diverse skills during a pre-training phase, preventing exploration for end-to-end learning. Previous methods (Eysenbach et al., 2019b; Florensa et al., 2017) learn a discrete set of skills by maximizing the mutual information (MI) between states and skills, but they hardly generalize over skills. It has been extended to continuous sets of skills, using a generative model (Sharma et al., 2020) or successor features (Hansen et al., 2020; Borsa et al., 2019). In both case, directly maximizing this MI may incite the agent to focus only on simple skills (Campos et al., 2020). DISCERN (Warde-Farley et al., 2019) maximizes the MI between a skill and the last state of an episode using a contrastive loss and the true goal to generate positive pairs. Several methods takes advantages of a VAE to learn skills, but our method better scales to any ground state space (see appendix A.1). Typically, RIG (Nair et al., 2018) uses a VAE to compute a goal representation before training the goal-conditioned policies. Using a VAE, one can define a frontier with a reachability network, from which the agent should start stochastic exploration (Bharadhwaj et al., 2020); but the gradual extension of the frontier is not automatically discovered, unlike approaches that maximize the entropy of states (including DisTop). Skew-Fit (Pong et al., 2020a) learns more diverse skills by making the VAE over-weight low-density states. It is unclear how Skew-Fit could target another density distribution over visited states than a uniform one. Approaches based on LP have already been built over VAEs (Kovač et al., 2020; Laversanne-Finot et al., 2018); we believe that DisTop could make use of LP to improve skill selection.

Skill discovery for end-to-end exploration. Like DisTop, ELSIM (Aubret et al., 2020) discovers diverse and rewarding skills in an end-to-end way. It builds a tree of skills and selects the branch to improve with extrinsic rewards. DisTop outperforms ELSIM for both dense and sparse-rewards settings (cf. section 4). This end-to-end setting has also been experimented through multi-goal distribution matching (Pitis et al., 2020; Lee et al., 2019) where the agent tries to reduce the difference between the density of visited states and a given density distribution (with high-density in rewarding areas). Yet, either they approximate a state density distribution over the ground state space (Lee et al., 2019) or assume a well-structured state representation (Pitis et al., 2020). Similar well-structured goal space is assumed when an agent maximizes the reward-weighted entropy of goals (Zhao et al., 2019).

Dynamic-aware representations. A set of RL methods try to learn a topological map without addressing the problem of discovering new and rewarding skills. Some methods (Savinov et al., 2018; 2019; Eysenbach et al., 2019a) consider a topological map over direct observations, but to give flat intrinsic rewards or make planning possible. Other approaches learn landmarks for planning, but need pre-training with uniform initial state and goal density distributions (Zhang et al., 2021; Huang et al., 2019) or need a uniform initial state density (Laskin et al., 2020), thereby avoiding the exploration challenge. We emphasize that SFA-GWR-HRL (Zhou et al., 2019) hierarchically takes advantage of a topological map built with two GWQ placed over two Slow Feature Analysis algorithms (Wiskott & Sejnowski, 2002); it is unclear whether it can be applied to other environments than their robotic setting. Functional dynamic-aware representations can be discovered by making the distance between two states match the expected difference of trajectories to go to the two states (Ghosh et al., 2019); interestingly, they exhibit the interest of topological representations for HRL and propose to use a fix number of clusters to create goals. Previous work also showed that an active dynamic-aware search of independent factors can disentangle the controllable aspects of an environment (Bengio et al., 2017). Other methods take advantage of temporal contrastive losses for other functional uses; therefore, unlike DisTop, they do not try to learn a topology of the environment by preventing the distortion of the representation. For example, successor representations (Jinnai et al., 2019; Wu et al., 2018) orthogonalize the features of the representation and some methods use a temporally-contrastive representation for imitation (Sermanet et al., 2018), end-to-end task solving (Anand et al., 2019; Stooke et al., 2021) or flat exploration (Yarats et al., 2021; Guo et al., 2021). Zhang et al. (2021) also cluster a dynamic-aware embedding, but they learn a fixed number of size-changing clusters, making them inappropriate for estimating the inherent density distribution of states.

6 Conclusion

We introduced a new model, DisTop, that simultaneously learns a discrete topology of its environment and the skills that navigate into it. This topology allows to efficiently select the skills to improve regardless of the ground state space or reward setting, *i.e* extrinsically rewarding ones or exploratory ones. In contrast with previous approaches, there is no pre-training (Pong et al., 2020b), no random walks (Lu et al., 2019b). It does not need a well-structured goal space like (Pitis et al., 2020) or dense rewards like in (Li et al., 2021). Yet, there are exciting perspectives. HRL and planning based approaches (Nasiriany et al., 2019) could both take advantage of the topology and make easier states discovery. More importantly, disentangling the topology (Bengio et al., 2013) could improve the scalability of the approach since the number of cluster exponentially grows with respect to the number of independent factors of variation.

REPRODUCIBILITY STATEMENT

To improve the reproducibility of the paper, we provide details about our comparison methods, some technical details of DisTop and used hyper-parameters in appendix C. In addition, we provide the code in the supplementary materials and commit to make the github repository public after acceptance.

REFERENCES

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 8769–8782, 2019.

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Annual Conference on Neural Information Processing Systems*, pp. 5048–5058, 2017.

Arthur Aubret, Laetitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

Arthur Aubret, Laëtitia Matignon, and Salima Hassas. ELSIM: end-to-end learning of reusable skills through intrinsic motivation. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera (eds.), *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Proceedings, Part II*, volume 12458 of *Lecture Notes in Computer Science*, pp. 541–556. Springer, 2020.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Amitay Bar, Ronen Talmon, and Ron Meir. Option discovery in the absence of rewards with manifold analysis. In *International Conference on Machine Learning*, pp. 664–674. PMLR, 2020.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1471–1479, 2016.

Emmanuel Bengio, Valentin Thomas, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable features. *CoRR*, abs/1703.07718, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Homanga Bharadhwaj, Animesh Garg, and Florian Shkurti. Leaf: Latent exploration along the frontier. *arXiv e-prints*, pp. arXiv–2005, 2020.

Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Rémi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

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*, pp. 1317–1327. PMLR, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Maxime Chevalier-Boisvert and Lucas Willems. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 539–546. IEEE, 2005.

Cédric Colas, Pierre Fournier, Olivier Sigaud, and Pierre-Yves Oudeyer. CURIOUS: intrinsically motivated multi-task, multi-goal reinforcement learning. *CoRR*, abs/1810.06284, 2018.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter F. Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Aleksandr Ermolov and Nicu Sebe. Latent world models for intrinsically motivated exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ben Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15220–15231, 2019a.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1514–1523. PMLR, 2018.

Bernd Fritzke et al. A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7:625–632, 1995.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal conditioned policies. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=Hye9lnCct7`.

Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Alaa Saade, Shantanu Thakoor, Bilal Piot, Bernardo Avila Pires, Michal Valko, Thomas Mesnard, Tor Lattimore, and Rémi Munos. Geometric entropic exploration. *arXiv preprint arXiv:2101.02055*, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Steven Hansen, Will Dabney, André Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. *Advances in Neural Information Processing Systems*, 32:1942–1952, 2019.

Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pp. 10–33. IEEE, 2001.

Yuu Jinnai, Jee Won Park, Marlos C Machado, and George Konidaris. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2019.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

Grgur Kovač, Adrien Laversanne-Finot, and Pierre-Yves Oudeyer. Grimgep: learning progress for robust goal sampling in visual deep reinforcement learning. *arXiv preprint arXiv:2008.04388*, 2020.

Michael Laskin, Scott Emmons, Ajay Jain, Thanard Kurutach, Pieter Abbeel, and Deepak Pathak. Sparse graphical memory for robust planning. 2020.

Adrien Laversanne-Finot, Alexandre Pere, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. In *Conference on Robot Learning*, pp. 487–504. PMLR, 2018.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *CoRR*, abs/1906.05274, 2019.

Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019.

Siyuan Li, Lulu Zheng, Jianhao Wang, and Chongjie Zhang. Learning subgoal representations with slow dynamics. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=wxRwhSdORKG`.

Xingyu Lu, Stas Tiomkin, and Pieter Abbeel. Predictive coding for boosting deep reinforcement learning with sparse rewards. *CoRR*, abs/1912.13414, 2019a.

Xingyu Lu, Stas Tiomkin, and Pieter Abbeel. Predictive coding for boosting deep reinforcement learning with sparse rewards. *CoRR*, abs/1912.13414, 2019b.

Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. A laplacian framework for option discovery in reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2295–2304. PMLR, 2017.

Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. A self-organising network that grows when required. *Neural Networks*, 15(8-9):1041–1058, 2002.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3303–3313. 2018b.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9209–9220, 2018.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14814–14825, 2019.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7750–7761. PMLR, 2020.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020a.

Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020b.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=SygwwGbRW`.

Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphaël Marinier, Marc Pollefeys, Timothy P. Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

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

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pp. 1–18, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ IROS*, pp. 5026–5033. IEEE, 2012.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

David Warde-Farley, Tom Van de Wiele, Tejas D. Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. In *International Conference on Learning Representations*, 2018.

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

Lunjun Zhang, Ge Yang, and Bradly C Stadie. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning*, pp. 12611–12620. PMLR, 2021.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33, 2020.

Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7553–7562. PMLR, 2019.

Xiaomao Zhou, Tao Bai, Yanbin Gao, and Yuntao Han. Vision-based robot navigation through combining unsupervised learning and hierarchical reinforcement learning. *Sensors*, 19(7):1576, 2019.

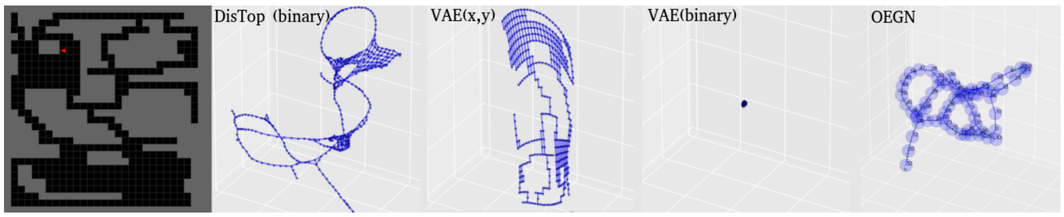


Figure 4: Visualization of the representations learnt by a VAE and DisTop on the environment displayed at the far left. From left to right, we respectively see a- the rendering of the maze; b- the continuous representation learnt by DisTop with 900-dimensional binary inputs; c- a VAE representation with true (x,y) coordinates; d- a VAE representation with 900-dimensional binary inputs; e- OEGN network learnt from binary inputs.

A Ablation study

In this section, we study the impact of the different key hyper-parameters of DisTop. Except for our main results presented in section 4, we average the results over 3 random seeds. For visualization of topologies, we select the most viewable one among 3 learnt topologies; while we did not observe a lot of variance through the seeds, the 3D angle of rotation can bother our perception of the topology.

A.1 Understanding the building of topologies

In this subsection, we aim at qualitatively studying the properties of our topology and their building process. To do this, we make sure that the learnt representation is visualizable. Therefore we adapted the *gym-minigrid* environment (Chevalier-Boisvert & Willems, 2018) (cf. Figure 4) where a randomly initialized agent moves for fifty timesteps in the four cardinal directions. Each observation is either a 900-dimensional one-hot vector with 1 at the position of the agent (*binary*) or the (x,y) coordinates of the agent. Interactions are given to the representation learning algorithm. Unless otherwise stated, we display as a node the learnt representation of each possible state and connect states that are reachable in one step.

Does DisTop discover the environment topology even though the ground state space is unstructured ? In this experiment, we analyze the representation learnt by a standard Variational Auto Encoder (VAE) and DisTop. In Figure 4, we clearly see that DisTop is able to discover the topology of its environment since each connected points are distinct but close with each other. In contrast, the VAE representation collapses since it cannot take advantage of the proximity of states in the ground state space; it only learns a correct representation when it gets (x,y) positions, which are well-structured. This toy visualization highlights that, unlike VAE-based models, **DisTop does not depend on well-structured ground state spaces to learn a suitable environment representation for learning skills.**

Can we control the dilatation and distortion of our representations ? The analysis of our objective function $\mathbb{L}_{DisTop}$ shares some similarities with standard works on contrastive learning (Chopra et al., 2005). However, we review it to clarify the role of the representation with respect to the interactions, the reward function and OEGN. In particular we emphasize two desideratas for the continuous representation: 1- two maximally distant consecutive states need to have a similar distance in the embedding space, in order for our density approximation to be consistent with the dynamics; 2- we have to avoid adding local optimas between two states with respect to the L2 distance, in comparison with the ground true topology. In addition, the discrete representation must accurately match the continuous one. We base our analysis on figure 5, please notice that the last line of figure 5 output an OEGN network, not the enviroment continuous representation.

First, we can study the influence of the distortion parameter k_c that brings closer consecutive states in $\mathbb{L}_{DisTop}$ (cf. equation 2). We can see in the second line of figure 5 that the distortion parameter k_c rules the global dilatation of the learnt representation. A low temperature k (fourth line of figure 5), in equation 2, prevents non-consecutive states to be close with each other, and thus, flattens the representation and allows to capture the *global* structure of the environment; a high temperature

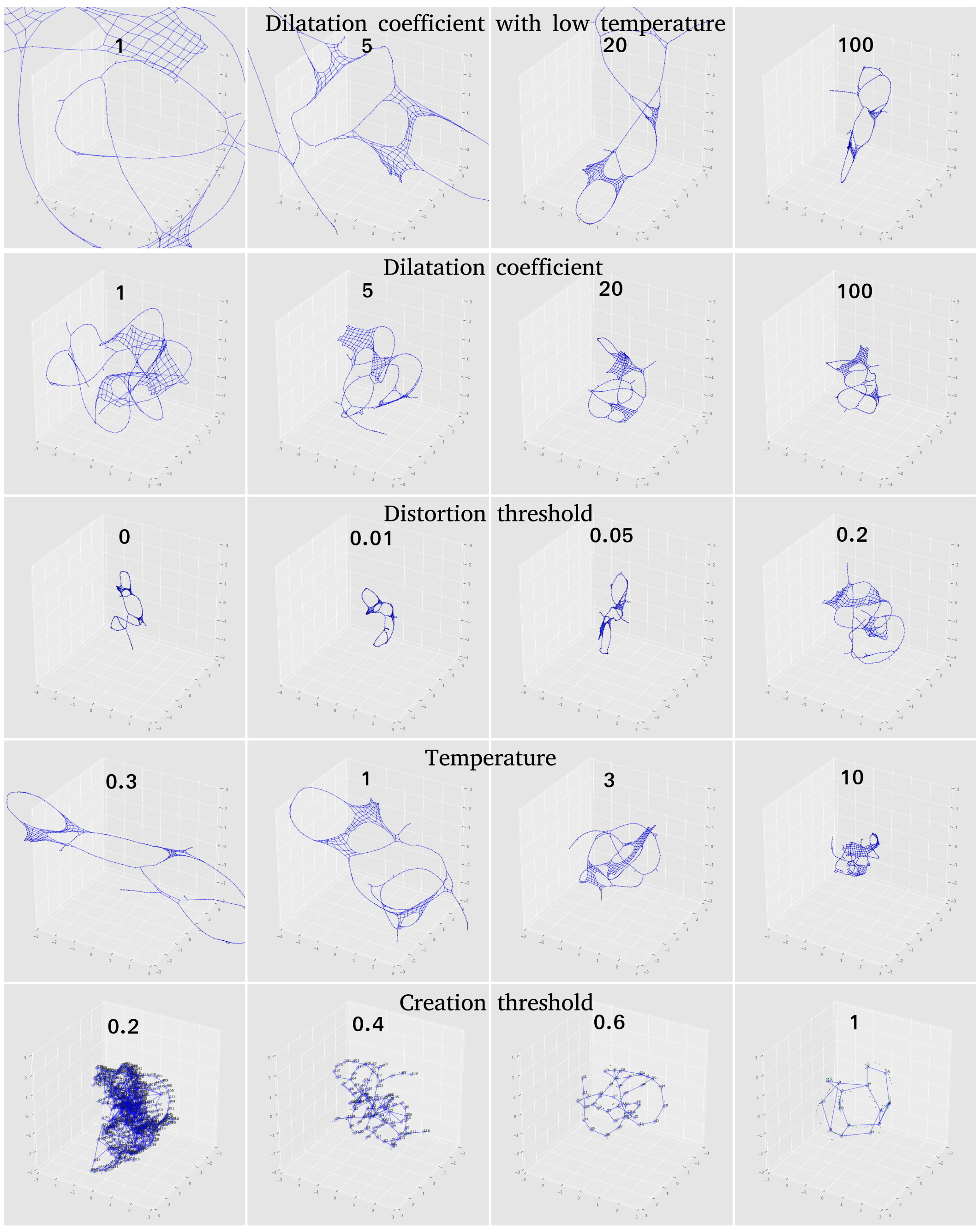


Figure 5: Different Topologies learnt on the gridworld displayed in figure 4. Each line corresponds to a hyper-parameter and we display the corresponding value on each square. All hyper-parameters are equals to $k = 3$, $k_c = 10$, $\delta = 0.1$, $delta_{new} = 0.6$ except for the changing one in each line, the second line for which we change the temperature $k = 0.5$ and the third line for which we change the dilatation coefficient $k_c = 100$.

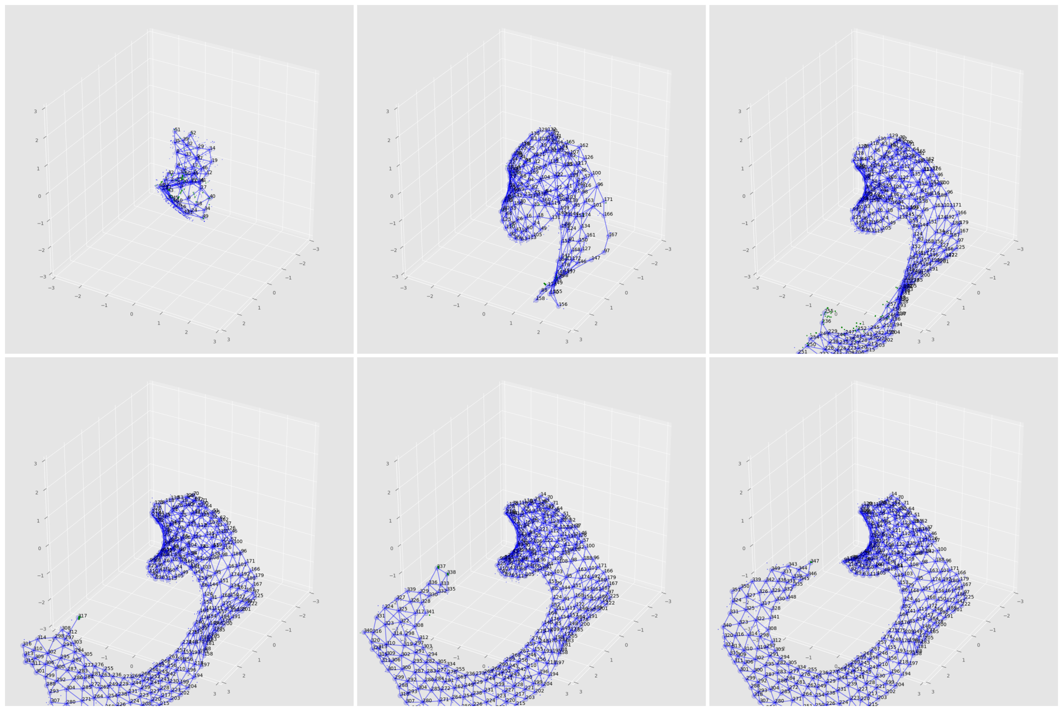


Figure 6: Progressively learnt topology throughout training. We selected six intermediary OEGN network with an equal intermediate number of timesteps. The topology is extracted from one of our experiments in section 4

curves the topology of the environment and the L2 rewards may admit several local optimas, which is not desirable. However one can see that a low temperature can also distort the representation, states in rooms (group of inter-connected nodes) tend to be closer with each other than bordering states, such that an action covers a larger distance in the border than in a room. In fact, in the borders, the agent often hurts the wall and stays in the same position, so, in comparison with large rooms, the *bring together* is less important than the *move away* part. We can prevent this by increasing the dilatation coefficient (first line of figure 5) such that all consecutive states get limited by the distortion threshold δ, which prevents consecutive embeddings to be equal in equation 2. Overall, relying on δ also becomes interesting when the number of states increases due to the exploration of the agent: the agent progressively compresses its representation since interesting negative samples are less frequent. In the third line of figure 5, we see that we can control the the minimal allowed distance by playing with δ.

The size of the clusters of OEGN needs to be hyper-parameterized with respect to the dilatation of the representation. The bottom line of Figure 5 emphasizes the importance of the creation threshold parameter δ_{new} that rules the radius of cluster in section 3.1. With a low $\delta_{new} = 0.2$, clusters do not move and a lot of clusters are created. OEGN waits a long time to delete them. With a high $\delta_{new} = 1$, the approximation of the topology becomes very rough, and states that belong to very different parts of the topology are classified in the same cluster; this hurts our density approximation. In addition, it may become hard to discover new clusters with random walks when clusters' radius is too large.

To sum up, there are 4 important messages: 1- we need a low temperature k to easily differentiate negative samples from the anchor; 2- to prevent a possible distortion, we increase k_c; 3- to prevent a collapse of the representation, we bound the minimal distance with δ; 4- δ_{new} must be set according to the dilatation of the representation, fixed with k_c, to correctly approximate a density distribution over states.

Does the discrete network manages to self-organize to track the continuous representation ? In contrast with the two previous paragraphs, we consider an agent that starts its episodes at the same

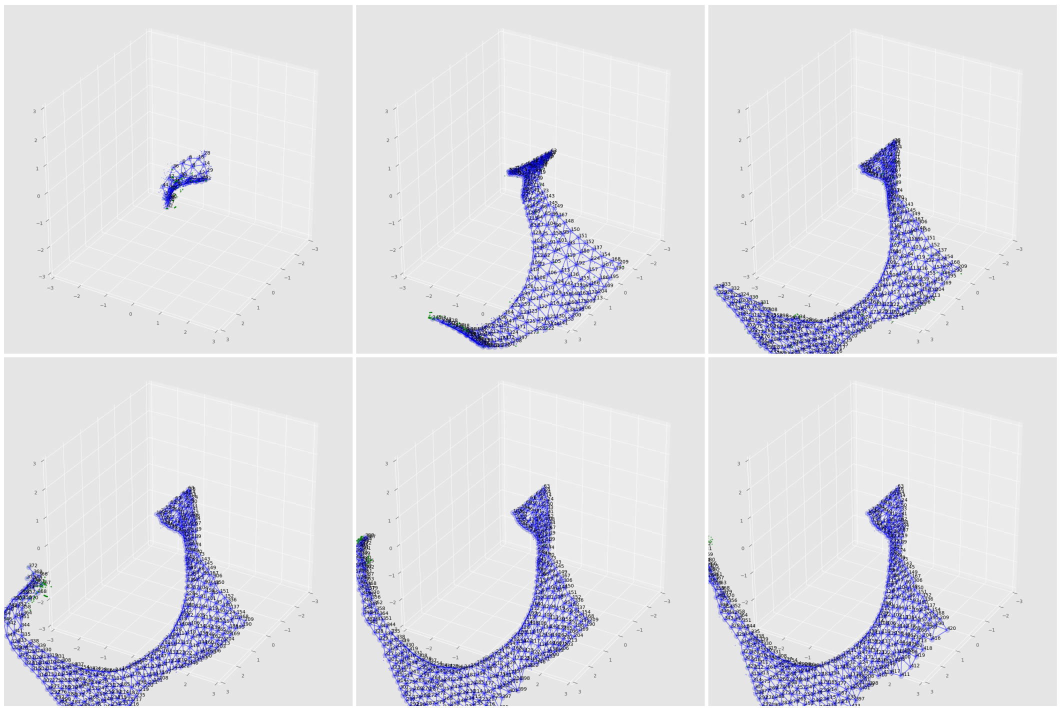


Figure 7: Progressively learnt topology throughout training. We selected six intermediary OEGN network with an equal intermediate number of timesteps. In comparison with figure 6, the temperature k, the dilatation coefficient k_c and the creation threshold δ_new are respectively set to 1, 50 and 0.3.

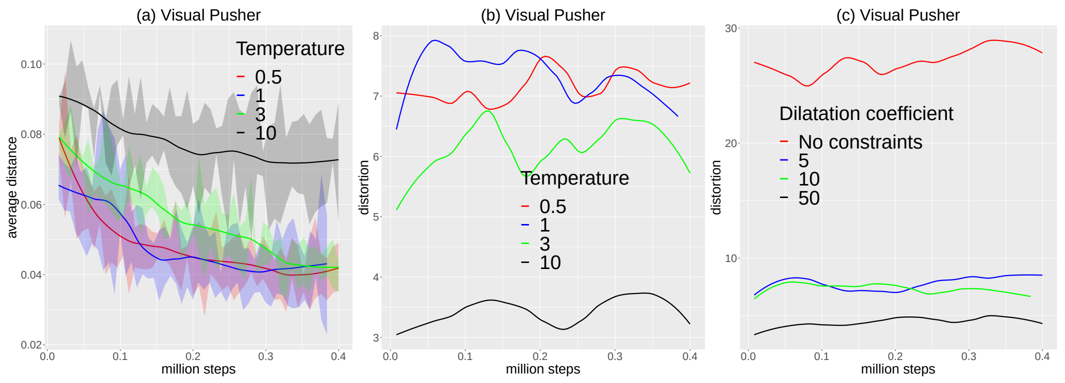


Figure 8: Distance between the final puck position and the wanted puck position in *Visual Pusher*. a- comparison for different values of temperature k; b- comparison of distortion for different values of temperature k; c- comparison of distortion for different values of k_c.

place and acts in the sparse ant maze experiment described in section 4. Figure 6 and figure 7 show how DisTop progressively builds its discrete topology (OEGN) of the environment. We only show topologies of the maze since its ground structure is appropriate for 3D visualization. We see that the network manages to progressively expand as the agent discovers the maze and learns its continuous representation. **At the end of the learning process, the topologies resemble the original U-shape maze.**

A.2 Quantitative analysis

In this section, we complement our results of the ablation conducted in section 4.

Is the temperature hyper-parameters important ? In figure 8a, we see the impact of the temperature hyper-parameter. Overall, a low temperature makes easier the learning process as hypothesized in section A.1; probably because it reduces the number of local optimas for the reward function.

Does the temperature and dilatation coefficient impact the distortion in complex environments ? To gain insights about the distortion of the representation, we propose to compute the ratio between the maximal distance between two consecutive embedded states and the mean one. Intuitively, if the distortion is 2, it means that there exists an area where the agent executes twice larger steps than in the rest of the environment. Therefore, when computing the state density with its clusters, the agent would use twice more clusters and thus, under-estimate the density of the area. Figure 8c show the distortion for several values of temperature; essentially, we see that the distortion increases when the temperature decreases, as visually experimented in appendix A.1. However, the distortion looks to be upper-bounded, probably because the squared dilatation in equation 2 does not allow this. Figure 8d stresses out the importance of our penalties: the agent learns a very distorted representation without them. We also see that the dilatation coefficient can reduce the distortion, as emphasized in appendix A.1.

B Complement informations on DisTop

In this section, we provide additional details about DisTop.

B.1 Derivation of the contrastive loss

$$\begin{aligned}
\mathbb{L}_{InfoNCE} &= \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[\log \frac{f_\omega(s_t,s_{t+1})}{\sum_{s\in\mathbb{B}} f_\omega(s,s_{t+1})}\big] \\
&= \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[-k||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2 - \log(\sum_{s\in\mathbb{B}} f_\omega(s,s_{t+1}))\big] && (6)\\
&= \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[-k||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2 - \log(f_\omega(s_t,s_{t+1}) + \sum_{s\in\mathbb{B}_{s\neq s_t}} f(s,s_{t+1}))\big] && (7)\\
&\geq \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}\big[-k||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2 - \log(1 + \sum_{s\in\mathbb{B}_{s\neq s_t}} f(s,s_{t+1}))\big] && (8)
\end{aligned}$$

In the last line of equation 8, we upper-bound $f_\omega(s_t, s_{t+1})$ with 1 since $e^{-v} < 1$ when v is positive. The logarithmic function is monotonic, so the negative logarithm inverses the bound.

Then, equation 9 formulates our objective as a constrained maximization problem, where ϵ is a small positive constant.

$$\begin{aligned}
\max_\omega \quad \mathbb{E}_{(s_{t+1})\in\mathbb{B}} - \log(1+\sum_{s\in\mathbb{B}} f_\omega(s,s_{t+1})) \quad \text{s.t.} \quad & \mathbb{E}_{(s_t,s_{t+1})\in\mathbb{B}}||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2^2 \leq \delta^2 \\
& \mathbb{E}_{s_{t+1}\in\mathbb{B}}||\phi_\omega(s_{t+1})-\phi_{\omega'}(s_{t+1})||_2^2 < \epsilon
\end{aligned} \quad (9)$$

Firstly, DisTop enforces the distance between consecutive states to be below a threshold δ (first constraint of equation 9). Secondly, it forces our representation to stay consistent over time (second constraint of equation 9).

B.2 Algorithm overview

Figure 9 illustrates algorithm 1, which sums up the learning loop of DisTop. See main text for explanations (section 3).

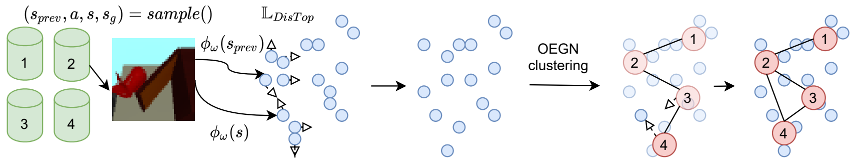


Figure 9: Illustration of a learning step of our growing network and contrastive loss (cf. text). Cylinders are buffers associated to each cluster, blue circles are states embedded with ϕ and pink circles represent clusters. The image is an example of state in the *Visual Door* (Nair et al., 2018) environment.

Algorithm 2: Algorithm of OEGN (red) and GWQ (blue)

```
Initialize network with two random nodes, set theirs attributes to 0 and connect them. foreach
learning iteration do
  Sample a tuple $s_i, c_i, s_i^{prev} \leftarrow sample(\mathbb{B})$.
  Embed states: $e_i \leftarrow \phi_{\omega'}(s_i)$; $e_i^{prev} \leftarrow \phi_{\omega'}(s_i^{prev})$
  Sample an input $e_i \leftarrow sample(\mathbb{B})$.
  $closest \leftarrow min_{nodes}(||nodes - e_i||_2)$.
  Increase error count of $c_i$ by 1.
  Reset error count of $closest$ to 0
  Apply $DeleteOperator()$.
  If a node is deleted, stop the learning iteration
  Apply $CreationOperator()$.
  Apply $MovingOperator()$.
  Apply $EdgeOperator()$.
```

B.3 OEGN AND GWQ

Several methods introduced unsupervised neural networks to learn a discrete representation of a fixed input density distribution (Kohonen, 1990; Fritzke et al., 1995; Marsland et al., 2002). The objective is to have clusters (or nodes) that cover the space defined by the input points. Each node $c \in C$ has a reference vector w_c which represents its position in the input space. We are interested in the *Growing when required* algorithm (GWQ) (Marsland et al., 2002) since it updates the structure of the network (creation and deletion of edges and nodes) and does not impose a frequency of node addition. OEGN is an adaptation of GWQ that intuitively works as follows: assuming a new low-density embedded state is discovered, there are two possibilities: 1- the balls currently overlap: OEGN progressively moves the clusters closer to the new state and reduces overlapping; 2- the clusters almost do not overlap and OEGN creates a new cluster next to the embedded state. Algorithm 2 describes the major steps of OEGN and GWQ. Operators and relative changes are described below. Specific operations of OEGN are colored in red, the ones of GWQ in blue and the common parts are black. We define $e = \phi_{\omega'}(s)$ and $closest(e) = min_{c \in C}(||c - e||_2)$ for all $s \in \mathbb{B}$.

Our modifications are made to 1- make the network aware of the dynamics and goals; 2- adapt the network to the dynamically changing true density distribution.

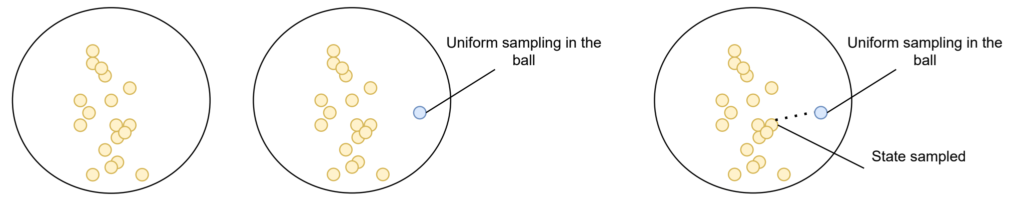
Delete operator: Delete c_i if $c_i.error$ is above a threshold δ_{error} to verify that the node is still active; delete the less filled neighbors of two neighbors if their distance is below δ_{prox} to avoid too much overlapping; check it has been selected n_{del} times before deleting it. Delete if a node does not have edges anymore.

Both creation and moving operators: Check that the distance between the original goal g_i and the resulting embedding $||g_i - e_i||_2$ is below a threshold $\delta_{success}$.

Creation operator: Check if $||closest - e_i||_2 > \delta_{new}$. Verify $closest$ has already been selected δ_{count} times by the goal-selection policy. If the conditions are filled, a new node is created at the position of the incoming input e_i and it is connected to $closest$. Update a firing count of the winning

Parameters	Value
Age deletion threshold δ_{age}	600
Error deletion count δ_{error}	600
Proximity deletion threshold δ_{prox}	$0.4 \times \delta_{new}$
Count creation threshold δ_{count}	3
Minimal number of selections n_{del}	10
Learning rate α	0.001
Neighbors learning rate $\alpha_{neighbors}$	0.000001
Number of updates per batch	~ 32

Table 1: Fixed hyper-parameters used in OEGN. They have not been tuned.



(a) States that belong to the buffer of a cluster represented by the circle.

(b) The agent randomly samples a vector in the ball associated to the cluster.

(c) The agent selects the closest embedded state to the vector.

Figure 10: Illustration of how an agent samples interactions from a buffer.

node and its neighbors and check it is below a threshold. A node is created halfway between the winning node and the second closest node. The two winning nodes are connected to the new node.

Moving operator: If no node is created or deleted, which happens most of the time, we apply the moving operator described by eq. 10 assuming $closest = \arg\min_c (\phi_{\omega'}(s) - w_c)^\top (\phi_{\omega'}(s) - w_c)$. In our case, we use a very low $\alpha_{neighbors}$ to avoid the nodes to concentrate close to high-density areas.

$$w_j = \begin{cases} w_j + \alpha(\phi_{\omega'}(s) - w_j), & \text{if } j = closest \\ w_j + \alpha_{neighbors}(\phi_{\omega'}(s) - w_j), & \text{if } j \in neighbors(closest) \\ w_j, & \text{otherwise.} \end{cases} \tag{10}$$

Edge operator: edges are added or updated with attribute $age = 0$ between the winning node of e_i and the one of e_i^{prev}. Edges with $age = 0$ are added or updated between the two closest nodes of e_i. When an edge is added or updated, increment the age of the other neighbors of $closest$ and delete edges that are above a threshold δ_{age}.

Except for δ_{new} and $\delta_{success}$, we emphasize that thresholds parameters have not been fine-tuned and are common to all experiments; we give them in Table 1.

B.4 STATE SAMPLING

Once the cluster is selected (cf. section 3.2), we keep exploring independently of the presence of extrinsic rewards. Assuming we have a ball of embedded states (figure 10a), to approximate a uniform density distribution over visited states that belongs to the cluster, DisTop samples a vector in the ball of radius δ_{new} that surrounds the center of its cluster (figure 10b). It rejects the vector and samples another one if it does not belong to the cluster. Finally, it selects the closest embedded state to the random vector and extracts the corresponding interaction (figure 10c). For example, one can imagine a ball with 100 states close to the center of the ball and two states on its surface; with a random sampling system, the agent would give priority to states close to the center. In contrast, by computing the state that is the closest to a uniformly sampled point in the ball, our preliminary experiments suggested it increases the uniformity of selection inside the ball and favors exploration.

B.5 Technical details

Number of negative samples. In practice, to learn ϕ_ω we do not consider the whole batch of states as negative samples. For each positive pair, we randomly sample only 10 states within the batch of states. This number has never been tuned.

Relabeling strategies. We propose three relabeling strategies;

1. π^{high} relabelling: we take samples from $\mathbb{B}^S$ and relabel the goal using clusters sampled with π^{high} and randomly sampled states. This is interesting when the agent focuses on a task; this gives more importance to rewarding clusters and allows to forget uninteresting skills. We use it in *Ant* and *Half-Cheetah* environments.
2. Uniform relabelling: we take samples from $\mathbb{B}^S$ and relabel the goal using states sampled from from $\mathbb{B}^S$. When $\alpha_{skew} \approx 0$, this is equivalent to relabeling uniformly over the embedded state space. This is used for maze and robotic experiments.

Neighbors update. In order to speed up the propagation of extrinsic rewards through the nodes of the network, we propose to relabel the true high level action (the goal) with the achieved action (achieved goal). Furthermore, it forces the agent to periodically explore again goals with potential. The achieved goal can be computed by embedding the final state with ϕ. Then, we update the neighbors with an exponential moving average of the achieved average extrinsic reward.

C Details about our experiments

C.1 Comparison methods

LESSON: we used the code provided by the authors and reproduced some of their results in dense rewards settings. Since the environments are similar, we used the same hyper-parameter as in the original paper (Li et al., 2021).

LESSON requires a task to learn skills since it explores with hierarchical random walks, making a fair comparison unrelevant in no/dense-rewards task.

Skew-Fit: since we use the same evaluation protocol, we directly used the results of the paper. In order to fairly compare DisTop to Skew-Fit, the state given to the DRL policy of DisTop is also the embedding of the true image. We do not do this in other environments. We also use the exact same convolutional neural network (CNN) architecture for weights ω as in the original paper of Skew-Fit. It results that our CNN is composed of three convolutional layers with kernel sizes: 5x5, 3x3, and 3x3; number of channels: 16, 32, and 64; strides: 3, 2 and 2. Finally, there is a last linear layer with neurons that corresponds to the topology dimensions d. This latent dimension is different from the ones of Skew-Fit, but this is not tuned and set to 10. T

That is unclear to us how to fairly adapt Skew-fit to solve a particular task, thus, we do not evaluate Skew-fit in experiments with tasks.

ELSIM: we use the code provided by the authors. We noticed they used very small batch sizes and few updates, so we changed the hyper-parameters and get better results than in the paper on *Half-Cheetah*. We set the batch size to 256 and use neural networks with 2×256 hidden layers. The weight decay of the discriminator is set to $1 \cdot 10^{-4}$ in the maze environment and $1 \cdot 10^{-3}$ in *Ant* and *Half-Cheetah*. In *Ant* and *Half-Cheetah*, the learning process was too slow since the agent sequentially runs up to 15 neural networks to compute the intrinsic reward; so we divided the number of updates by two. In our results, it did not bring significant changes.

Parameters	Values $RP/RD/MA/MC/SAM/SPM$	Comments
DRL algorithm SAC		
Entropy scale	$0.1/0.05/0.2/0.2/0.1/0.2$	
Hidden layers	$3/3/3/3/4/4$	
Number of neurons	512	Smaller may work
Learning rate	$5 \cdot 10^{-4}$	Not tuned
Batch size	$256/256/512/512/512/512$	Works with both
Smooth update	0.001	Not tuned.
Discount factor γ	$0.99/0.99/0.99/0.99/0.996/0.996$	Tuned for mazes
Representation ϕ_ω		
Ratio	$0/0/0.5/0.5/0/0$	see appendix A
Learning rate	$1 \cdot 10^{-4}$	Not tuned
Number of neurons	256 except robotic images (CNN)	Not tuned
Hidden layers	2 except robotic images (CNN)	Not tuned
Distortion threshold δ	SPM: 0.01 else 0.1	Tuned on SPM
Dilatation coefficient k_c	SPM: 50 else 20	See appendix A
Consistency coefficient β	RD: 1	Not tuned
Smooth update α_{slow}	0.001	Not tuned
Temperature k	$1/1/3/3/3/1$	See appendix A
Topology dimensions d	$10/10/3/3/3/3$	Not tuned
OEGN and sampling		
Creation threshold δ_{new}	$0.8/0.6/0.4/0.6/0.4/0.6$	See appendix A
Success threshold $\delta_{success}$	$\infty/\infty/0.2/0.2/\infty/\infty$	
Buffers size	$[8/8/5/5/5/5] \cdot 10^3$	
Skew sampling $1 + \alpha_{skew}$	0	See appendix A
Updates per steps	$2/2/0.5/0.5/0.25/0.25$	
High-level policy π^{high}		
Learning rate α_c	0.05	Tuned
Neighbors learning rate	0.05	Not fine-tuned
Skew selection $1 + \alpha'_{skew}$	$-1/-1/0/0/-50/-50/$	See appendix A
Reward temperature t^{ext}	$0/0/50/10/100/100$	

Table 2: Hyper-parameters used in experiments. RP, RD, MA, MC, SAM, SPM respectively stands for Robotic Visual Pusher, Robotic Visual Door, MuJoCo Ant, MuJoCo Half-Cheetah, Sparse Ant Maze, Sparse Point Maze.

SAC: we made our own implementation of SAC. We made a hyper-parameter search on entropy scale, batch size and neural networks structure. Our results are consistent with the results from the original paper (Haarnoja et al., 2018).

LWM-DisTop: For fairness in the sampling process, we adapt LWM (Ermolov & Sebe, 2020) to the topology of DisTop. It essentially rewards an interaction with the prediction error in the embedded space.

FlatDisTop: One can think of FlatDisTop as a variant of TRPO-AE-SimHash (Tang et al., 2017), but replacing the autoencoder and the hashing procedure by our contrastive-based representation and OEGN. The agent rewards an interaction (s_{prev}, a, s, s_g) with $\frac{1}{\sqrt{count(c_s)}}$ where c_s is the cluster of $\phi_{\omega'}(s)$.

C.2 Hyper-parameters

Table 2 shows the hyper-parameters used in our main experiments. We emphasize that tasks are very heterogeneous and we did not try to homogenize hyper-parameters across environments.

C.3 Environment details

C.3.1 Robotic environments

Robotic Environments and evaluation protocols are as described in (Pong et al., 2020a). For convenience, we sum up again some details here.

Visual Door: a MuJoCo environment where a robotic arm must open a door placed on a table to a target angle. The state space is composed of 48x48 images and the action space is a move of the end effector (at the end of the arm) into (x,y,z) directions. Each direction ranges in the interval [-1,1]. The agent only resets during evaluation in a random state. During evaluation, goal-states are sampled from a set of images and given to the goal-conditioned policy. At the end of the 100-steps episode, we measure the distance between the final angle of the door and the angle of the door in the goal image.

Visual Pusher: a MuJoCo environment where a robotic arm has to push a puck on a table. The state space is composed of 48x48 images and the action space is a move of the end effector (at the end of the arm) in (x,y) direction. Each direction ranges in the interval [-1,1]. The agent resets in a fixed state every 50 steps. During evaluation, goal-states are sampled randomly in the set of possible goals. At the end of the episode, we measure the distance between the final puck position and the puck position in the goal image.

C.3.2 Maze environments

Maze environments are described in (Nachum et al., 2018a) and we used the code modified by (Li et al., 2021). For convenience, we provide again some details and explain our sparse version. The environment is composed of 8x8x8 fixed blocks that confine the agent in a U-shaped corridor displayed in Figure 2.

Similarly to (Li et al., 2021), we zero-out the (x,y) coordinates and append a low-resolution top view of the maze to the proprioceptive state. This ”image” is a 75-dimensional vector. In our sparse version, the agent gets 0 reward when its ground distance to the target position is below 1.5 and gets -1 reward otherwise. The fixed goal is set at the top-left part of the maze.

Sparse Point Maze: the proprioceptive state is composed of 4 dimensions and its 2-dimensional action space ranges in the intervals [-1,1] for forward/backward movements and [-0.25,0.25] for rotation movements.

Sparse Ant Maze: the proprioceptive state is composed of 27 dimensions and its 8-dimension action space ranges in the intervals [-16,16].

C.4 Computational resources

Each simulation runs on one GPU during 20 to 40 hours according to the environment. Here are the settings we used:

- Nvidia Tesla K80, 4 CPU cores from of a Xeon E5-2640v3, 32G of RAM.
- Nvidia Tesla V100, 4 CPU cores from a Xeon Silver 4114, 32G of RAM.
- Nvidia Tesla V100 SXM2, 4 CPU cores from a Intel Cascade Lake 6226 processors, 48G of RAM. (Least used).

C.5 Example of skills

Figures 11, 12, 13 show examples skills learnt in respectively *Visual Door*, *Visual Pusher* and *Ant Maze*. Additional videos of skills are available in supplementary materials.

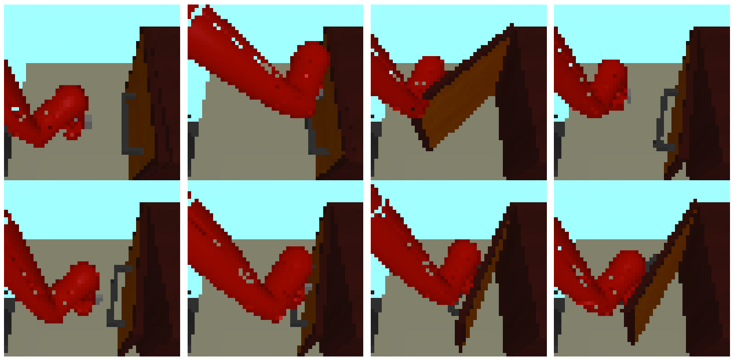


Figure 11: Examples of 8 skills learnt in *Visual Door*.

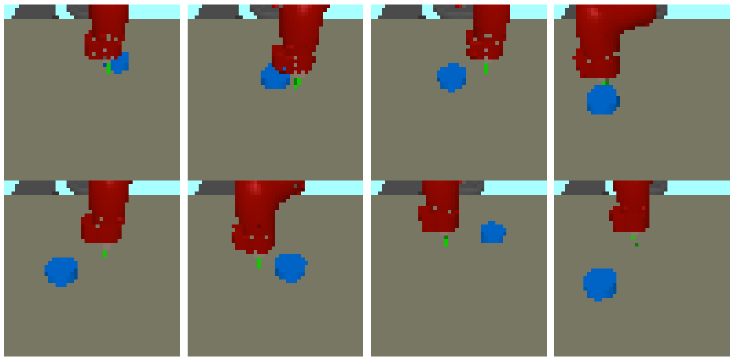


Figure 12: Examples of 8 skills learnt in *Visual Pusher*.

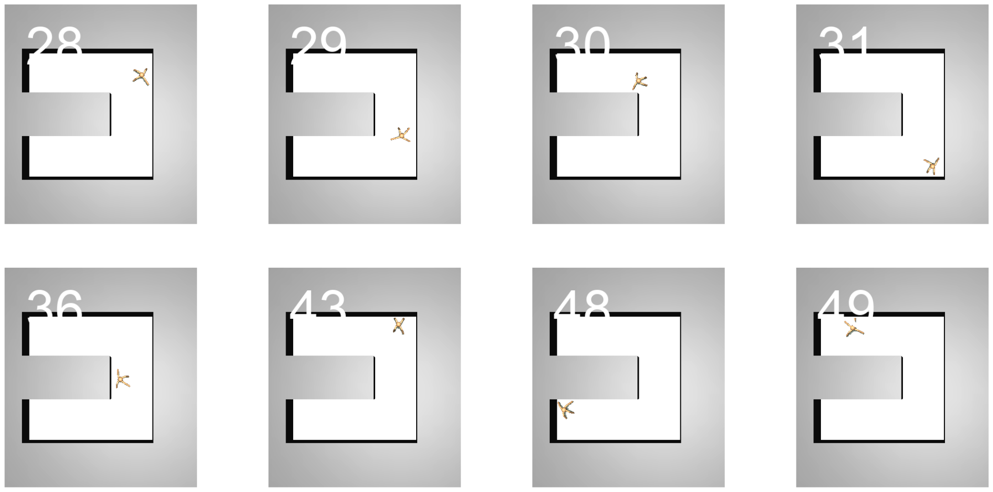


Figure 13: Examples of 8 skills learnt in *Ant Maze*.