N°d'ordre NNT : xxx

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° 512
InfoMaths

Spécialité de doctorat : Informatique

Soutenue publiquement le 30/11/2021, par :
Arthur Aubret

Learning Increasingly Complex Skills through Deep Reinforcement Learning using Intrinsic Motivation

Devant le jury composé de :

Dutech, Alain Rapporteur
Professeur des Universités, Université Nancy 2
Filliat, David Rapporteur
Professeur des Universités, École nationale supérieure de techniques avancé
Aussem, Alexandre Examinateur
Professeur des Universités, Université Claude-Bernard Lyon 1
Oudeyer, Pierre-Yves Examinateur
Directeur de recherche, INRIA Centre Inria Bordeaux Sud-Ouest

Hassas, Salima Directrice de thèse
Professeur des Universités, Université Claude-Bernard Lyon 1
Laëtitia, Matignon Co-directrice de thèse
Maître de Conférences, Université Claude-Bernard Lyon 1

Acknowledgements

I would first like to thank my supervisors, Laëtitia and Salima. Your empathy and kindness made these 3 years enjoyable to me and I appreciated your caring and constructive remarks. But above all, you let me the chance to autonomously experiment my own research while supporting me with strategic advices and critical methodologies. Salima, now that you made the famous home barbecue, I can peacefully end my thesis and I will not be there to harass you anymore !

I thank all the members of the SyCoSMA team for being involved in our regular meetings. Finally, almost everyone participated ! Especially Mathieu, for your dynamic implication and for always being open to scientific discussions. Overall, I hope we will have the opportunity to regularly meet again !

I would like to mention all the persons that made this experience socially pleasant and interesting. This particularly includes Theo, Rémy, Yacine, Mehdi, Simon, Julien, Huan, Antoine, but also Bastien, Mouna, Pierre and other temporary interns. Simon, I'm sure we only got a glimpse of all your games ! Mehdi, I hope you will manage to improve your football skills and one day you may make a true nutmeg.

Finally, I would like to particularly thank my brother Antoine for his wise advices and comforting speeches; I hope we will have the opportunity to make a paper together ! I am grateful to my partner, Clarisse, whose supported me all along the scientific journey despite some hard times. That is only the beginning and like in artificial intelligence, the path will be long and tough; however, nowadays, that is a three-way adventure that begins.

Résumé

En apprentissage par renforcement (RL), un agent apprendre à résoudre une tâche en interagissant avec son environnement. Afin de faire passer à l'échelle ces agents sur des tâches complexes, les méthodes récentes ont proposé avec succès d'intégrer les méthodes d'apprentissage profond au RL, créant le domaine d'apprentissage profond par renforcement (DRL). Cependant, la signification sémantique d'une tâche est toujours pourvoyée par une fonction de récompense experte qui guide l'agent dans son processus d'apprentissage. Ce paradigme contraste avec la manière dont les animaux et humains apprennent: les travaux de psychologie suggèrent que les humains sont intrinsèquement motivés à acquérir de nouvelles connaissances à propos de leur environnement.

Dans cette thèse, notre objectif est d'étudier comment la motivation intrinsèque permet de résoudre les problèmes expérimentés par le DRL.

Tout d'abord, nous mettons en évidence comment les motivations intrinsèques actuelles attaquent certains problèmes du DRL. Nous classifions et formalisons les méthodes et analysons leurs limites. Afin d'exhiber leur importance, nous mettons en avant que ces verrous peuvent empêcher un agent d'apprendre des compétences et représentations de l'environnements de plus en plus complexes. Ce sont des éléments clés pour faire apprendre des agents autonomes comme des humains.

Á partir de cette analyse, nous introduisons deux nouveaux modèles qui peuvent apprendre des compétences diverses et spécifiques à une tâche de bout en bout. Le premier, ELSIM, construit un arbre discret de compétences dans la direction des récompenses de l'environnement. Nos résultats montrent que ca paradigme d'apprentissage améliore l'exploration dans des environnements avec des récompenses éparses et permet d'utiliser des compétences sur différentes tâches corrélées. Nous mettons en avant les inconvénients d'ELSIM et proposons un autre modèle, DisTop, pour les corriger. DisTop construit progressivement une topologie de l'environnement en utilisant une fonction de coût contrastive, un réseau auto-organisé et une politique dépendante d'objectifs. L'agent peut alors intelligemment contrôler quelles compétences apprendre ou oublier. De cette manière, DisTop est compétitif avec des algorithmes de l'état de l'art sur trois types de tâches différentes, incluant une tâche hiérarchique avec des récompenses éparses.

Pour conclure la thèse, nous discutons des perspectives du domaines et des directions futures de notre recherche.

Mots-clés

Apprentissage par renforcement, Motivation intrinsèque, Apprentissage développemental, Apprentissage tout au long d'une vie, Apprentissage de représentations.

Abstract

In reinforcement learning (RL), an agent learns to solve a task by interacting with its environment. In order to scale these RL agents on high-dimensional complex tasks, recent methods successfully proposed to integrate deep learning methods in RL, creating the field of deep RL (DRL). However, the semantic meaning of a task still derives from an expert reward function that guide the agent in its learning process. This paradigm contrasts with how animals and humans learn: psychology suggests that humans are intrinsically motivated to autonomously acquire knowledge about their environment.

In this dissertation, our objective is to investigate how intrinsic motivation can solve the issues experienced by DRL algorithms.

First of all, we stress out how intrinsic motivations currently address some issues of DRL algorithms. We classify them, analyze the current limitations of these methods. To highlight their importance, we emphasize that these deadlocks may prevent an agent to learn both increasing complex skills and environment representations. These are key elements to make autonomous agent learn in a more human-like way.

Following this analysis, we introduce two new models that can learn diverse and task-specific skills in an end-to-end way. The first one, ELSIM, builds a discrete tree of skills in the direction of the feedbacks of the environment. Our results show that this learning paradigm favors exploration in sparse-rewards environments and allows to reuse skills over different correlated tasks. We highlight the drawbacks of ELSIM and propose a novel model, DisTop, that tackle them. It progressively builds a discrete topology of the environment using an unsupervised contrastive loss, a self-organizing network and a goal-conditioned policy. Then, DisTop can smartly control which skill to improve or forget. This way, DisTop competes with state-of-art algorithms on three different benchmarks, including a hierarchical environment with sparse rewards. To conclude the dissertation, we discuss the outlooks of the domain and future directions for research.

Keywords

Reinforcement learning, Intrinsic motivation, Developmental learning, Lifelong learning, Representation learning

Contents

Chapter 1

Introduction

1.1 Context

In this dissertation, we consider the setting where an agent interacts with its environment through reinforcement learning (RL)[Sutton and Barto, 2018]. In RL, an agent learns to solve a task by maximizing the cumulative reward it gathers throught its interactions with its environment. An example of such agent could be a wheeled robot that perceives its environment with a camera (state) and moves (action) to post letters (task). While these works have been limited to very simple tasks for a long time, recent works proposed to merge the generalization power of deep neural networks with standard approaches of RL, creating the field of *deep reinforcement learning* (DRL)[Mnih et al., 2015]. These approaches have demonstrated their ability to learn specific tasks using only ground high-dimensional inputs [Mnih et al., 2015] and ground actions [Haarnoja et al., 2018]. As a typical example, the Deep Q-Network (DQN) [Mnih et al., 2015] outperforms humans performance on several atari games using only ground pixels. More broadly, using neural networks frees an agent from the need of expertly building features representing high-level semantic knowledge of the world.

Despite these recent successes, most of the works in DRL focus on agents that learn tasks specified by a human, *e.g* solving an atari game. This is a consequence of the way the learning process is modelled: for a given task, an expert exports the semantic of the task in a reward function that can be maximized by the agent. This reward function has to be expertly built, it may be a score if an agent tries to solve a game or a distance function when the agent learns to reach a target position. Typically, this reward function has to be dense and well-structured to avoid unexpected learnt behaviors [Ng et al., 1999] and to give meaningful feedbacks to the agent.

On another side, unlike RL, humans from birth onward try to learn, explore even though they do not get external incentives [Ryan and Deci, 2000]. This mechanism inspired the emergence of developmental learning [Cangelosi and Schlesinger, 2018, Piaget and Cook, 1952, Oudeyer and Smith, 2016], which is based on the trend that babies, or more broadly organisms, have to spontaneously explore their environment [Gopnik et al., 1999] and acquire new skills [Barto, 2013]. By skills, we mean a behavior that strives to reach a goal, which can be a set of states in the environment.

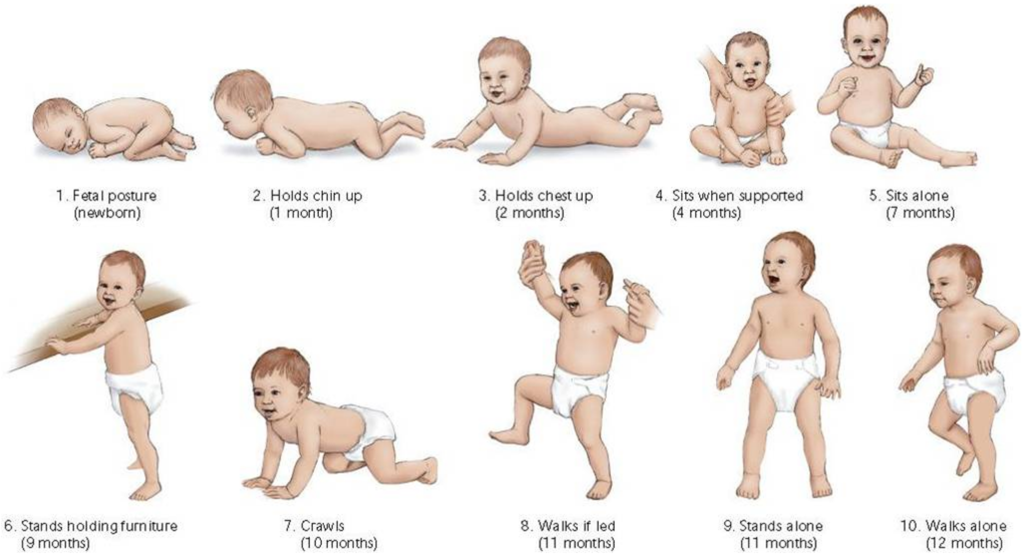

1. Fetal posture (newborn)
2. Holds chin up (1 month)
3. Holds chest up (2 months)
4. Sits when supported (4 months)
5. Sits alone (7 months)
6. Stands holding furniture (9 months)
7. Crawls (10 months)
8. Walks if led (11 months)
9. Stands alone (11 months)
10. Walks alone (12 months)

Figure 1.1: Illustration of the different locomotion learning steps during early infant.

Typically, babies can autonomously select their goals and pursue them by interacting with the environment [Oudeyer et al., 2013, Baldassarre et al., 2014]. By selecting harder and harder goals, a baby incrementally and continually learns new skills and accumulates a plethora of very diverse and increasingly complex skills. Such motivation to learn is called an motivation (IM) [Ryan and Deci, 2000].

Learnt skills can be locomotion skills, grasping skills, climbing or whatever results from playing or interacting with its environment. For example, Figure 1.1[1] displays the locomotion learning steps of a child during its first year: a baby will consecutively start to turn over its stomach, sit down, stand up, crawl, walk with help of objects and then without help; finally it will learn to run faster and faster and will move backward. These skills can then be hierarchically organized to create higher level skills. For example, grasping and walking skills can be organized to create a new skill like *opening a window*. The resulting interactions with the environment may also ground the symbolic meaning of the different objects that compose the world: a child understands the concept of an object according to how it interacts with it [Brooks, 1991, Stoytchev, 2009, Varela et al., 2017]. All this knowledge and these skills are often useless in the short term, but reveals to be precious in the long term for the survival of an organism [Baldassarre and Mirolli, 2013b]. This learning scheme can also be found in most of animals, making it strongly correlated with the appearance of intelligence. It suggests that the combination of open-ended learning and autonomous development through intrinsic motivation may be the hallmark of intelligence [Baldassarre et al., 2014].

[1] Source: `https://edu.glogster.com/glog/infant-and-toddler-development/`

1.2 Problem statement and outline

1.2.1 Problem statement

In this work, we hypothetize that intrinsic motivation can bring key missing elements to DRL methods. In particular, we would like to emancipate from an expert supervision so that agents autonomously and continually learn to interact with their environment. This being done, agents could learn increasingly complex behaviors without supervision, like humans do. The objective of this work is to study **how to take advantage of intrinsic motivations to solve the issues experienced by DRL algorithms**. Throughout our scientific journey, we first give an overview of the basics of DRL and intrinsic motivation, then we detail three contributions that respectively correspond to three chapters and we finally preserve the last chapter for a discussion about the outlooks and limitations of our work. In the following, we sum up the chapters and our contributions.

1.2.2 Background of DRL and intrinsic motivation

The first chapter focuses on the background of DRL and intrinsic motivation. In a first time, we review a key algorithm of RL and its extension to DRL. In a second time, we take a close look on intrinsic motivation, we define intrinsic motivation from a psychological perspective and highlight its properties so that we may fruitfully instantiate it in the RL framework. Finally we exhibit how intrinsic motivation can integrate the RL framework by reformulating it on the basis of hierarchical RL [Sutton et al., 1999], goal-conditioned RL [Schaul et al., 2015] and information theory [Cover and Thomas, 2012]. This gives us the opportunity to investigate the interests of using intrinsic motivation in RL.

1.2.3 Survey of IM in RL:

Analysis of IM in RL: In our first contribution, we study the current role of intrinsic motivation in the context of DRL. We thoroughly analyze and identify how intrinsically motivated DRL agents tackle the different issues specific to DRL approaches. Based on this study, we propose a novel classification of the methods based on their technical and theoretical contribution. It follows that we exhibit the limitations of the different classes of approaches, highlight ongoing research, identify the outlooks of the domain and unify intrinsic motivations under one principle.

More specifically, this work showcases the importance of learning hierarchical skills with intrinsic motivation. We name this a *bottom-up* skill discovery, since skills are learnt independently from any task. Briefly, one of our results is the observation that learning hierarchically organized *bottom-up* skills can solve a large number of issues experienced by DRL algorithms.

Reformulation of our problem statement Based on our analysis, we hypothetize that an agent can overcome shortcomings of DRL if it sequentially learns hierarchically organized skills. This way, we reformulate the objective of our work: **how can an agent learn increasingly complex hi-**

erarchically organized skills with intrinsic motivation ? This is the motivation of our next contributions.

1.2.4 End-to-end learning of reusable skills

Scientific issues. In our analysis, we exhibit that previous approaches mostly learn the skills during an unsupervised pre-training phase (or developmental period [Metzen and Kirchner, 2013]), this prevents end-to-end exploration and the specialization of skills to a given task or goal. If it is not possible to specialize skills according to a goal, it may also prevent an integration in a hierarchically organized structure. Based on this simple observation, we hypothetize that an agent has to be able to simultenaously learn to solve a task or a goal and learn its *bottom-up* skills; it results that its learning process has to be *end-to-end*, *i.e* without pre-training phase. To validate the hypothesis, we temporally puts aside the hierarchical organization of skills and focus on how to discover skills with intrinsic motivation in an end-to-end way.

ELSIM: end-to-end learning of reusable skills through intrinsic motivation. The second contribution is a model, ELSIM, that improves a previous approach in order to learn a discrete set of skills in an end-to-end way. In particular, it progressively builds a tree of skills in the direction of the feedbacks of the environment.

This improvement opens new perspectives: 1- the agent can now directly solves a task or a goal, making it suitable to be incorporated in a hierarchical setting; 2- it can explore its environment even though it does not get extrinsic rewards, thus improving exploration. While being end-to-end, skills can still be used across several tasks. Our proof-of-concept experiments validate our hypothesis.

However, our thorough analysis highlight that: firstly, exploration is sub-optimal; secondly, learning a discrete set of skills makes it difficult to integrate it in a continually growing hierarchical architecture. These issues motivate our third contribution.

1.2.5 Discovering a topological representation while learning diverse skills

The third contribution is a model, DisTop, that learns a topological state-goal representation of its environment while learning the skills to navigate inside it. It overcomes the limitations of ELSIM by learning a continuous goal representation and its discretized version, it selects the skills to improve according to their density (for exploration) or their interest relatively to a task. The agent still learns in an end-to-end way, keeping the best properties of ELSIM.

In our experiments, we show that: 1- while DisTop enjoys the properties of ELSIM, it clearly outperforms it on several benchmarks; 2- by forgetting skills and smartly targetting skills to improve, it gets competitive results with state-of-the-art methods on dense rewards single-task, multi-skill discovery benchmarks without rewards and hierarchical tasks with sparse rewards.

With our model, we expect further works to exhibit the possibility and interest of using DisTop while using a hierarchical policy.

1.2.6 Discussion

In the last chapter, we will summarize our contributions and highlight their perspectives and limitations. In particular we will discuss the questions raised by our works: how to learn a good representation of the environment ? Can we use a unified concept of intrinsic motivation to guide the model proposals ? These questions may be critical for future research in developmental learning, both in the short and long term.

Chapter 2

Background of intrinsic motivation and deep RL

The motivation of an autonomous agent can be two-folds: either it tries to solve a task given by an external entity, in which case it can learn through RL, or it acts according to its own will, in which case it is *intrinsically motivated*. Both methods have different properties and origins. Intrinsic motivation is a concept derived from the psychological literature [Deci and Ryan, 2010] to describe the spontaneous inclination of humans to learn and explore the environment [Ryan and Deci, 2000]. This definition is intuitive but it is unclear how it could be integrated in a computational framework in order for an autonomous agent to enjoy the potential benefits of an intrinsic motivation. In contrast, DRL agents evolve in a well-defined computational framework where an agent perceives the ground state of the environment and executes ground low-level actions to solve one single-task. The main algorithms of RL and DRL are built inside this framework. It results that integrating the intrinsic motivation in the computational framework of RL requires to revisit the standard vision of RL.

In this chapter, we review the basics of RL, DRL, intrinsic motivation and their association. In Section 2.1, we define the framework of RL; this includes the modelisation of the problem as a Markov Decision Process, the core algorithms that drive RL and their recent extensions to DRL algorithms. In particular we highlight a recent state-of-the-art method [Haarnoja et al., 2018] we will use throughout the rest of the dissertation. Then, in Section 2.2, we focus on developmental learning, in particular lifelong learning and intrinsic motivation: we define them and identify the key properties that characterize an intrinsic motivation. With all these elements in hand, in Section 2.3, we describe the computational framework that merges intrinsic motivation and DRL. Importantly, within this model, we give general key elements that guarantee an efficient learning process to an intrinsically motivated agent.

Throughout the dissertion, we will assume the reader has a basic understanding of *deep learning* methods, in particular: Convolutional neural networks (CNN), Multi-layer perceptrons (MLP), stochastic gradient descent (SGD), optimizers . . . . For further informations about these aspects, we refer to Goodfellow et al. [2016].

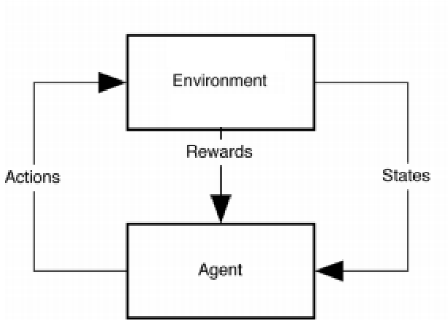


Figure 2.1: Illustration of the interaction loop in RL.

2.1 Reinforcement learning

This section is dedicated to the description of the reinforcement learning framework, including the modeling of the problem and the core algorithms that adress it.

2.1.1 Markov Decision Process

Considering an autonomous agent that interacts in its environment, we can model this as a Markov Decision Process (MDP)[Puterman, 2014]. An MDP is defined by:

- S the set of possible states;
- A the set of possible actions;
- T the transition function $T : S \times A \times S \rightarrow \mathbb{R}$. $T(s', a, s) = p(s'|s, a)$ defines the probability to go in state $s' \in S$ after making an action $a \in A$ in state $s \in S$;
- R the reward function $R : S \times A \times S \rightarrow \mathbb{R}$;
- $\gamma \in [0, 1]$ the discount factor;
- $d_0 : S \rightarrow \mathbb{R}$ the initial probability distribution of states.

In this dissertation, T and d_0 define probabilities (discrete argument) or probability density functions (continuous arguments); these functions are strictly positive and their integral equals one.

The interaction loop of an RL agent is illustrated by Figure 2.1. An agent starts in a state s_0 given by d_0. At each time step t, the agent perceives a state s_t and performs an action a_t. Then, it waits for the feedback from the environment composed of a state s_{t+1} sampled from the transition function $s_{t+1} \sim p(\cdot|s_t, a_t)$, and a reward given by the reward function $r_t = R(s_t, a_t, s_{t+1})$. The agent repeats this interaction loop until the end of an episode. The episode ends when the agent reaches particular states $s_{end} \in S$ or after a fixed number of timesteps $T_{episode}$. At the end of an episode, the agent resets in a starting state $s_0 \sim d_0(\cdot)$ and can repeat the loop. This approach is very generic and allows to model a large number of problems that can be described with an agent interacting with its environment.

It may happen that the agent does not access its full state. For instance, as humans, we do not visually perceive what is behind us and we must infer it from our previous observations. In this case, the MDP can be extended to a Partially Observable Markov Decision Process (POMDP) [Kaelbling et al., 1998]. In comparison with a MDP, it adds a set of possible observations O which defines what the agent can perceive and an observation function $\Omega : S \times O \to \mathbb{R}$ that defines the probability of observing $o \in O$ when the agent is in the state s, *i.e* $\Omega(s, o) = p(o|s)$.

To solve a MDP or POMDP, we need to assess the performance of the algorithm according to a reward-based criteria. For example, a commonly used criteria can be to maximize the discounted cumulated reward gathered throughout an infinite or finite horizon defined by Equation 2.1, where $T_{episode} \in [1; \infty[$. The discount factor $\gamma \in [0; 1]$ defines how much the long-term reward is important in comparison with the short-term reward.

$$J = \left[\sum_{t=0}^{T_{episode}} \gamma^t r_t \right]. \tag{2.1}$$

Similarly, we can also define the finite-horizon undiscounted objective with $T_{episode} \neq \infty$:

$$J = \left[\sum_{t=0}^{T_{episode}} r_t \right]. \tag{2.2}$$

Here, the reward is considered as extrinsic (or as a feedback) because the reward function is provided expertly and specifically for the task. We will now study how standard RL agents manage to solve an MDP.

2.1.2 Reinforcement learning

A reinforcement learning algorithm aims to associate states $s \in S$ to actions $a \in A$ through a stochastic policy $\pi : S \times A \to \mathbb{R}$. In the specific case of $\pi(a|s) = \{0, 1\}$, the policy becomes deterministic. This policy induces a t-steps state distribution that can be recursively defined as:

$$d_t^\pi(s) = \int_S d_{t-1}^\pi(s_{t-1}) \int_A p(s_t|s_{t-1}, a)\pi(a|s_{t-1}) da\, ds_{t-1} \tag{2.3}$$

with $d_0^\pi(s) = d_0$. The goal of the agent is then to find the optimal policy π^* maximizing the cumulative reward defined in Equation 2.1:

$$\pi^* = \arg\max_\pi \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|_{s_0 \sim d_0(\cdot)} \right]. \tag{2.4}$$

The simplest strategy to optimize this objective is to 1- randomly generate policies; 2- execute them a large number of times to estimate the expectation of cumulated reward. This estimation is called a Monte-Carlo approximation; 3- Keep the best policy. But this is strongly inefficient since the number of possible policies scales exponentially with the number of states and actions.

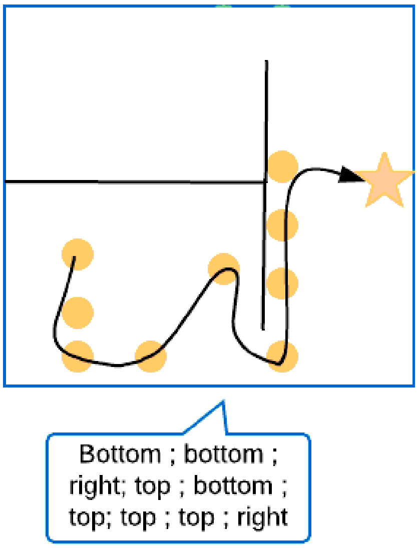


Figure 2.2: Illustration of the difficulty to differentiate optimal actions from sub-optimal ones. The black line represents the trajectory of an orange agent that looks for the star.

To make easier the problem, one can search for the optimal actions in each state of the MDP, dividing the issue into more straightforwards issues. This approach amounts to find the action that maximizes the long-term reward in a state s. Following our criteria, we can maximize the expected discounted gain following a policy π from a state s, noted $V_\pi(s)$ (cf. Equation 2.5), or from a state-action tuple, noted $Q_\pi(s, a)$ (cf. Equation 2.6). It allows to measure the impact of the state-action tuple in obtaining the cumulative reward [Sutton and Barto, 1998].

$$V_\pi(s) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t, a_t)}} \left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|_{s_0=s} \right). \tag{2.5}$$

$$Q_\pi(s, a) = \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t, a_t)}} \left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|_{s_0=s, a_0=a} \right). \tag{2.6}$$

This computation also deals with what is called the *credit assignment problem*, *i.e* the difficulty to identify the value of an action while marginalizing the effect of other actions from the same trajectory. Figure 2.2 illustrates the issue. Before passing the bottleneck, the orange agent made a sub-optimal action; even though it managed to achieve the task, which was reaching the star, it could perform better by fixing this sub-optimal action. Q-learning algorithm tackles this by computing the value of each pair (s, a). When the policy selects the actions, it efficiently identifies the contribution of each action with respect to the final performance of the policy.

There is an optimal Q-function Q^* that has the highest value in all states. This is defined by Equation 2.7.

$$Q^*(s, a) = \max_\pi Q_\pi(s, a) \quad \forall s \in S, \forall a \in A. \tag{2.7}$$

If an agent finds out the optimal Q-function, it will be able to act optimally in the environment according to its given task. More formally, the optimal policy is defined by:

$$\pi^*(a_t|s_t) = \mathbb{1}[a_t = \arg\max_a Q^*(s_t, a)] \tag{2.8}$$

where $\mathbb{1}[\text{cond}]$ is the indicator function which equals 1 if cond is verified, 0 elsewhere. Since the optimal policy derives from the optimal Q-values, the problem amounts to find the optimal Q-function.

One way to approximate the optimal Q-function is to take advantage of the Bellman equation verified by the optimal Q-function [Sutton and Barto, 1998]:

$$Q^*(s_t, a_t) = \mathop{\mathbb{E}}_{s_{t+1} \sim p(\cdot|s_t,a_t)} \big[R(s_t, a_t, s_{t+1}) + \gamma \max_a Q^*(s_{t+1}, a)\big]. \tag{2.9}$$

It tells us that the optimal Q-value of a state-action tuple (s_t, a_t) is the reward that results from the interaction plus the maximum future cumulative reward starting from the next state. One can notice that this approximation of Q is *off-policy*, *i.e* the right-hand part does not depend on the policy [1].

We can now apply the Q-learning algorithm: it uses Equation 2.9 to define a recursive operator that approximates $Q(s_t, a_t)$ using true interactions from the environment. The application of this recursive operator is sometimes called *bootstrapping*. Equation 2.10 describes this recursive approximation where $\alpha \in [0; 1]$ is the learning rate and Q_{prev} denotes the Q-value at the previous learning step [Watkins, 1989].

$$Q(s_t, a_t) = Q_{prev}(s_t, a_t) + \alpha\big[-Q_{prev}(s_t, a_t) + R(s_t, a_t, s_{t+1}) + \gamma \underbrace{\max_a Q_{prev}(s_{t+1}, a)}_{\text{Target}}\big]. \tag{2.10}$$

We recall that an agent does not access the transition function $s_{t+1} \sim p(\cdot|s_t, a_t)$. So it still has to gather true interesting interactions to apply this operator and correct its Q-values. A naive approach would be to apply its optimal policy according to its current estimation of the optimal Q-function, $\pi(a_t|s_t) = \mathbb{1}[a_t = \arg\max_a Q(s_t, a)]$. But under-approximated $Q(s, a)$ would never be corrected, since the agent would never execute sub-optimal actions in s. In contrast, it has to keep exploring using currently sub-optimal actions to fix badly approximated Q-values. Of course, at the opposite, if it only executes random actions, it may never discover interesting parts of its state space. This is called the exploitation-exploration trade-off: how much should the agent follow its actual optimal policy and how much it should explore ? We briefly introduce two standard exploration strategies:

- The simplest one is the method ϵ-greedy. It allows to select with probability ϵ a random action, and with probability $(1 - \epsilon)$ the current optimal action.

- The Boltzmann exploration selects an action a_k according to the probability defined by Equa-

[1] In this dissertation, so as to be concise, we only detail the *off-policy* setting since this is the one we used in the work, but a large different set of algorithms derive from an on-policy setting. Essentially, the difference comes from whether it takes into consideration the current exploration policy of an agent when it approximates its Q-value [Sutton and Barto, 1998].

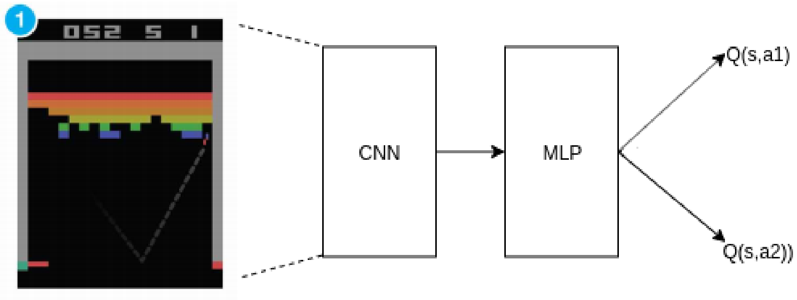


Figure 2.3: Illustration of a Deep Q-network on *Atari Breakout*. The state space is composed of images with dimensions $210 \times 160 \times 3$ and the actions consist in moving the horizontal bar to the left or the right ($|A| = 2$).

tion 2.11 where $\tau \in \mathbb{R}$ is an hyper-parameter that rules how much the agent explores. The greater it is, the more random is the policy. Intuitively, Boltzmann exploration ponderates the probability to select an action according to its current Q-value.

$$\pi(a_k|s) = \frac{e^{\frac{Q(s,a_k)}{\tau}}}{\sum_{a \in A} e^{\frac{Q(s,a)}{\tau}}}. \tag{2.11}$$

Assuming we access a large table that describes the Q-values of all state-action tuples, by iteratively applying the Bellman operator and stochastically acting in the environment, Q-values progressively converge towards the optimal Q-values [Sutton and Barto, 1998].

However, when the states are high-dimensionnal inputs, *e.g* images filled with 84×84 pixels ($84 \times 84 \times 3$ values), the number of states is huge and directly approximating Q-values using a table turns out to be impossible. In this case, a learning agent has to generalize over its state space to efficiently compute Q-values. For similar reasons, it is not clear how one can solve a MDP with a continuous action or state space. The two next sections will be dedicated to summarize the recent advances in the domain based on deep neural networks.

2.1.3 Deep reinforcement learning

To scale on high-dimensional and continuous state and action spaces, recent approaches successfully proposed to approximate Q and π with neural networks, which leaded to the emergence of the field of *deep reinforcement learning*.

Generalizing over the state space

To generalize over the state space, Mnih et al. [2015] proposed to approximate $Q(s, a)$ with a neural network parameterized by θ: $Q_\theta(s, a) \approx Q(s, a)$. This is illustrated in Figure 2.3; the neural network takes in a state (image with $210 \times 160 \times 3$ dimensions) and outputs the Q-values of each of the two possible actions. However it is not possible to directly apply Equation 2.10 due to what has been called the deadly triad [Sutton and Barto, 1998]. When an agent uses complex

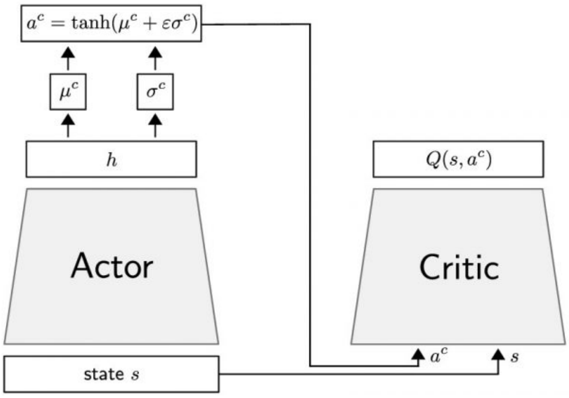


Figure 2.4: Illustration of a typical off-policy variant of an actor-critic architecture (value gradient architecture).

function approximators along with off-policy learning and bootstrapping (like in the Q-learning), the approximation may diverge. Intuitively, this is because modifying a Q-value with Equation 2.9 also modifies its target Q-value (right part of Equation 2.9). It follows that, if some Q-values are more often updated than others, the Q-values can diverge [van Hasselt et al., 2018]. To bypass this issue, the Deep Q-Network (DQN) [Mnih et al., 2015] proposes two mechanisms:

1. To learn on uncorrelated different states, they use a very large buffer $\mathcal{B}$ of interactions and learn from interactions sampled from it.
2. Rather than using the true Q-value as a target, they use a slowly moving approximation $\hat{Q}_{\theta'}$. Parameters θ' are either updated towards θ at a predefined frequency [Mnih et al., 2015] or they slowly move at each learning step using an exponential moving average [Lillicrap et al., 2015].

Equation 2.12 sums up the resulting objective function, which can be maximized using stochastic gradient descent. Using buffers, we have $r_t \sim R(s_t, a_t, s_{t+1})$.

$$\mathbb{L}_{dqn} = \mathop{\mathbb{E}}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{B}} \left[Q_\theta(s_t, a_t) - (r_t + \gamma \max_a \hat{Q}_{\theta'}(s_{t+1}, a)) \right]^2 \tag{2.12}$$

RL with continuous action spaces

To keep generalizing over the state space while generating continuous actions, SVG [Heess et al., 2015] also approximates π with a neural network π_ϕ. Figure 2.4[2] illustrates a typical architecture. An *Actor* neural network takes in a state s and computes the mean $\mu_\phi(s)$ and the co-variance matrix $\sigma_\phi(s)$ of a diagonal Gaussian. Actions are sampled from this parameterized Gaussian distribution. Then, actions are bounded by the action space of the MDP using a $tanh$ function (actions generally range in $[-1; 1]$). After that, a *Critic* neural network computes the Q-value of an incoming (state,action) tuple. The *Critic* objective function is essentially the same as in the previous subsection, but takes in actions deterministically output by the *Actor* to compute the one-dimensional

[2] Image taken from `https://montreal.ubisoft.com`

target objective. Thus, in the following, we focus on how to learn the continuous policy. With this architecture, Equation 2.13 highlights that the cumulative reward, approximated by Q_θ, can be maximized by updating the *Actor* using SGD. $\nabla_{a_t} Q_\theta(s_t, tanh(a_t))$ computes the gradient of action with respect to the Q-value approximation, *i.e* how to change the action to increase the Q-value; $\nabla_\theta \pi_\phi(a_t|s_t)$ translates the required changes of action into an update of the parameters of the policy.

$$\nabla_{\theta^\pi} J \approx \mathbb{E}_{s_t \sim \mathcal{B}} \nabla_{a_t} Q_\theta(s_t, tanh(a_t)) \, \nabla_\theta \pi_\phi(a_t|s_t) \quad \text{with } a_t \sim \mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)) \tag{2.13}$$

To allow back-propagation of the gradient through action sampling $a_t \sim \mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t))$, we can apply the reparameterization trick [Kingma and Welling, 2014]:

1. sample $\epsilon_{noise} \sim \mathcal{N}(0, 1)$;
2. compute $a_t = \mu_\theta(s_t) + \epsilon_{noise} \, \sigma_\theta(s_t)$.

From now, in continuous cases, we will refer to π as being the parameterized distribution induced by ϵ_{noise} with bounded actions. A variant of this architecture, DDPG [Lillicrap et al., 2015], can also learn a deterministic actor. With these methods, the agent typically explores by manually adding either a spherical Gaussian Noise to the actions output by the policy, or a Ornstein-Uhlenbeck process to provide temporally coherent behaviors. In the next subsection, we study a state-of-the-art DRL algorithm based on these improvements.

Soft Actor Critic

Soft Actor-Critic (SAC) [Haarnoja et al., 2018] proposes to maximize the entropy augmented cumulative reward defined by Equation 2.14. By adding an entropy term $H(A) = \sum_{a \in A} p(a) \log p(a)$ weighted by the hyper-parameter α, SAC learns a stochastic actor that directly learns to explore. α directly controls the degree of exploration, the more α is important, the more the agent explores independently from the rewards. They show that it considerably improves and stabilizes the learning process on several environments.

$$\pi^* = \arg\max_\pi \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1})|_{s_0 \sim d_0} + \alpha H(\pi(\cdot|s_t)) \right]. \tag{2.14}$$

It follows two new objectives to learn the Q-values (Equation 2.15) and the policy (Equation 2.16). For the Q-value approximation, the gradient associated to the entropy term is approximated through Monte-Carlo sampling of log-probabilities. Intuitively, the policy outputs a distribution of actions that matches the α-weighted distribution of exponential Q-values over actions. Of course, the repartition of the Q-values over the state space changes considerably in comparison with the usual objective (Equation 2.4), the entropy term strongly favors states where the agent can act stochastically.

$$J_Q(\theta) = \mathop{\mathbb{E}}_{\substack{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{B} \\ a_{t+1}\sim\pi_\phi(\cdot|s_{t+1})}} \left[(Q_\theta(s_t,a_t) - (r_t + \gamma\ (\hat{Q}_{\theta'}(a_{t+1},s_{t+1}) - \alpha\log\pi_\phi(a_{t+1}|s_{t+1})))\right]^2. \tag{2.15}$$

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{B}}\left[\mathbb{E}_{a_t\sim\pi_\phi(\cdot|s_t)}\alpha\log\pi_\phi(a_t|s_t) - Q_\theta(s_t,a_t)\right]. \tag{2.16}$$

Additionally to these objective functions, it is possible to fix the over-estimation bias of Q-learning based models [Fujimoto et al., 2018, Van Hasselt et al., 2016, Hasselt, 2010]. This bias occurs when an agent bootstraps its Q-values using the best possible action in the next state, making Q-values over-estimated in comparison with the true Q-value. Over-estimated Q-values tend to be more often bootstraped making other Q-values also over-estimated. Thus, it can prevent an agent to efficiently converge, being stuck in some over-estimated local optimums. To overcome this issue, TD3 [Fujimoto et al., 2018] proposes to under-estimate Q-values using a double-critic architecture; in fact, an under-estimation is better than an over-approximation since the error may not be recursively propagated. It is possible to incorporate this improvement in the SAC objective; there is now two approximations of the Q-function, $Q^1_{\theta^1}$ and $Q^2_{\theta^2}$ with their own slowly moving approximation $\hat{Q}^1_{\theta'^1}$ and $\hat{Q}^2_{\theta'_2}$. Using $Q^{min}_{\theta'} = \min(\hat{Q}^1_{\theta'^1}(a_{t+1},s_{t+1}), \hat{Q}^2_{\theta'^2}(a_{t+1},s_{t+1}))$, Equation 2.17 sums up the update when the minimum over two Q-function acts as the target Q-value. The same equation applies to update $Q^2_{\theta_2}$.

$$J_{Q^1}(\theta^1) = \mathop{\mathbb{E}}_{\substack{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{B} \\ a_{t+1}\sim\pi_\phi(\cdot|s_{t+1})}} \left[Q_\theta(s_t,a_t) - (r_t + \gamma\ (Q^{min}_{\theta'} - \alpha\log\pi_\phi(a_{t+1}|s_{t+1})))\right]^2. \tag{2.17}$$

Because of its efficiency, most of the methods we will study in the rest of the dissertation are based on SAC. Now that we reviewed some core components of DRL, we will study how we can take advantage of the generalization ability and long-term maximization of DRL agents to build intrinsically motivated agents. We first define two components of develomental learning: lifelong learning and intrinsic motivation, then we study the typical properties of an intrinsic motivation.

2.2 Developmental learning

Developmental learning refers to the design of models that directly draws inspiration from human cognitive development; such model of agent should autonomously build its knowledge by interacting with its world [Guerin, 2011, Nguyen et al., 2021]. In this section, we describe two aspects of developmental learning: a developmental agent (1) continually acts and perceives its environment even though its objective may change and (2) autonomously and actively acquires knowledge through an *intrinsic motivation*.

2.2.1 Definition of lifelong learning

Biological organisms face an uninterrompted stream of data all along their lifetime. During this lifetime, they keep acting according to different objectives that are assigned to them, whether they are self-assigned or attributed by an external entity. Typically, a child may play with its cubes and change its activity to play with, for instance, a puzzle: This ability to switch between activities is called *lifelong learning* or *continual learning* [Thrun and Mitchell, 1995, Parisi et al., 2019, Lesort et al., 2020].

In the machine learning domain, this generally results in an agent that learns to solve several different tasks sequentially. However, such setting introduces several new considerations. Firstly, the agent should be able to transfer and reuse the knowledge acquired while carrying out a task to another task [Parisi et al., 2019]. For instance, if a baby knows how to grasp a ball, it should provide him/her information about how to grasp a cube. This is refered as *transfer learning*. Secondly, the agent must avoid forgetting how to solve previously learnt tasks, which is not trivial when an agent uses neural networks [Parisi et al., 2019, Lesort et al., 2020] since a newly learnt task may overwrite a previous learnt task. For instance, if a child learns to shoot a ball after learning to grasp it, it must not forget how to grasp it. This forgetting issue is refered as *catastrophic forgetting*.

2.2.2 Definition of intrinsic motivation

According to Singh et al. [2010], evolution provides a general intrinsic motivation (IM) function that maximizes a fitness function based on the survival of an individual. Curiosity, for instance, does not immediately produce selective advantages but enables the acquisition of skills providing by themselves some selective advantages. More widely, the use of intrinsic motivation allows to obtain intelligent behaviors which may later serve goals more efficiently than with only a standard reinforcement [Baldassarre and Mirolli, 2013a, Baldassarre, 2011, Lehman and Stanley, 2008]. Typically, a student doing his mathematical homework because he/she thinks it is interesting is intrinsically motivated whereas his/her classmate doing it to get a good grade is extrinsically motivated [Ryan and Deci, 2000]. In this future, the intrinsically motivated student may be more successful in math than the other one. It questions the relevance of using only standard reinforcement methods.

Simply stated, intrinsic motivation is about doing something for its inherent satisfaction rather than to get a positive feedback from the environment [Ryan and Deci, 2000]. Looking at this definition, one can notice that intrinsic motivation is defined by contrast with extrinsic motivation; it highlights the difference between the two paradigms. Intrinsic motivation assumes the agent learns on its own while extrinsic motivation assumes there exits an expert that supervises the learning process.

More rigorously, Oudeyer and Kaplan [2008] explain that an activity *is intrinsically motivating for an autonomous entity if its interest depends primarily on the collation or comparison of information from different stimuli and independently of their semantics*. At the opposite, an extrinsic reward results of an unknown environment static function which does not depend on previous experience of the agent on the considered environment. The main point is that the agent must not have any *a priori* on the semantic of the observations it receives. Here the term *stimuli* does not refer to sensory inputs, but more generally to the output of a system which may be internal or external to the independent entity, thereby including *homeostatic* body variables (temperature, hunger, thirst, attraction to sexual activities . . .) [Baldassarre, 2011, Berlyne, 1965]. Broadly speaking, the motivation of an agent can be internal (*source of motivation*) while still being extrinsic (*why* of the actions). For instance, when an agent is looking for food because of the hunger, hunger is a stimuli coming to the cognitive system of the agent such that it is an internal but extrinsic motivation. As an other example, a child may do his/her homeworks because he/she thinks it will be crucial to latter get a job. While the source of the motivation is internal, the true outcome comes from the environment.

Now that the we clarified the notion of intrinsic motivation, the next section provides a computational perspective on intrinsic motivation and study its properties. An extensive overview of IM can be found in Barto [2013].

2.2.3 Properties of intrinsic motivation

While the previously mentioned definition encompasses the search for desirable stimulus properties such as surprisingness [Berlyne, 1965], an intrinsically motivated entity could also search their exact opposite, such as boredom. It highlights that it is not enough to say that an agent is intrinsically motivated; different kind of intrinsic motivations can co-exist and each one may have different properties. Examples include *"novelty", "surprisingness", "complexity", "ambiguity"* . . . [Oudeyer and Kaplan, 2008, Berlyne, 1965]. However it is possible to exhibit typical properties.

The notion of intrinsic motivation is strongly related to its homeostatic property. The agent, while acting to fulfil its intrinsic motivation is progressively satisfied and the interest of the agent moves towards other things. As an example, this is especially visible when one consider exploratory motivations. Let us consider an agent that moves in a maze. When an agent discovers a new part of its environment, at first it is satisfied and keeps exploring this area; then, one expect it will progressively get bored and may move away, this is in fact the core interest of exploratory behaviors. It highlights that intrinsic motivations do not bring static behaviors but temporary behaviors that rule the search for information.

We can sum up and translate the definition of intrinsic motivation into several key properties that characterize how intrinsic is a motivation for an agent.

- The motivation has to be task-agnostic. By task, we refer to a goal attributed by an external entity (like in the RL framework described in Section 2.3.1).
- The agent must be guided by the information contained in its previous interactions. This directly follows the definition of Oudeyer and Kaplan [2008].
- The motivation has to be dynamic, or in other words, *non-stationary*.
- The agent does not access *a priori* knowledge on its environment.

Looking at these properties, we can see how intrinsic motivation relates to standard learning schemes in *machine learning*. Table 2.1 shows the difference between RL and the use of IM. RL is

Table 2.1: Type of learning. *feedback* here refers to an expert supervision.

	With *feedback*	Without *feedback*
Active	Reinforcement	Intrinsic motivation
Passive	Supervised	Unsupervised

an active process since the agent learns from its interactions with the environment, unlike classification or regression which are supervised methods. Unsupervised learning is a passive learning process, *i.e.* it does not use predefined labels, or in other words, learns without a feedback. Finally, the substitution of the feedback by an intrinsic motivation allows to break free from an expert supervision; however, the difference remains between IM and unsupervised learning in the sense that IM is an active process which implies interactions. In the literature of machine learning, intrinsic motivation-based learning may also be called *self-supervised learning* [Liu et al., 2021, Pathak et al., 2017].

In the next section, we will see how to formally incorporate these properties to make DRL agents intrinsically motivated.

2.3 Intrinsic motivation in DRL

2.3.1 A model of RL with intrinsic rewards

Reinforcement learning is derived from behaviorism [Skinner, 1938] and usually uses extrinsic rewards [Sutton and Barto, 1998]. However Singh et al. [2010] and Barto et al. [2004] reformulated the RL framework to incorporate IM. We can differentiate *rewards*, which are events in the environment, and *reward signals* which are internal stimulis to the agent. Thus, what is named *reward* in the RL community is in fact a *reward signal*. Inside the *reward signal* category, there is a distinction between *primary reward signals* and *secondary reward signals*. The *secondary reward signal* is a local *reward signal* computed through expected future rewards and is related to the value function (cf. Equation 2.6) whereas the *primary reward signal* is the standard *reward signal* received from the MDP.

In addition, rather than considering the MDP environment as the environment in which the agent achieves its task, it suggests that the MDP environment can be formed of two parts: the **external part** which corresponds to the potentiel task and the environment of the agent; the **internal part** which computes the MDP states and the *secondary reward signal* using potentially previous interactions. Consequently, we can consider an intrinsic reward as a *reward signal* received from the MDP environment. The MDP state is no more the external state but an internal state of the agent. However, from now, we will follow the terminology of RL and the term *reward* will refer to the *primary reward signal*.

Figure 2.5 summarizes the new framework: the critic is in the internal part of the agent, it computes the intrinsic reward and deals with the credit assignment. The agent can merge intrinsic rewards and extrinsic rewards in its internal part. The state includes sensations and any form of internal context; in this section we refer to this state as a contextual state. The decision can be a

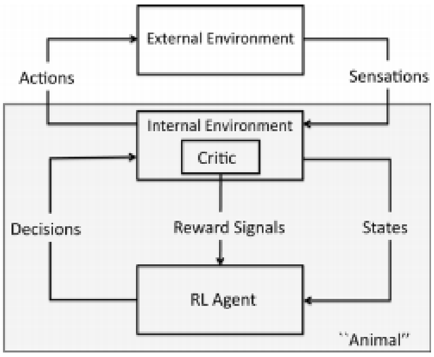


Figure 2.5: Model of RL integrating IM, taken in Singh et al. [2010]. The environment is factored into an internal and external environment, with all reward coming from the former.

high-level decision decomposed by the internal environment into low-level actions.

This conceptual model incorporates intrinsic motivations into the formalism of MDP. Now, we will review how this model is instantiated in practice. Indeed it is possible to extend RL to incorporate the three new components that are intrinsic rewards, high-level decisions and contextual states. We separately study them in the following sections.

2.3.2 Intrinsic rewards and information theory

Throughout our definition of intrinsic motivation, one can notice that the notion of *information* comes up a lot. As we will see in Chapter 3, this is not hazardous and quantifying information proves useful to generate intrinsic rewards. In this section, we provide the basics about information theory and explain how to use it to combine intrinsic and extrinsic rewards. However, we emphasize that intrinsic rewards are not restricted to information measures and their characterization mostly depends one whether the reward function fits the properties of an intrinsic motivation.

Information theory. The Shannon entropy quantifies the mean necessary information to determine the value of a random variable. Let X be a random variable with a law of density $p(X)$ satisfying the normalization and positivity requirements, we define its entropy by:

$$H(X) = -\int_X p(x) \log p(x). \tag{2.18}$$

In other words, it allows to quantify the disorder of a random variable. The entropy is maximal when X follows an uniform distribution, and minimal when $p(X)$ is equal to zero everywhere except in one value, which is a Dirac distribution. From this, we can also define the entropy conditioned on a random variable S. It is similar to the classical entropy and quantifies the mean necessary information to find X knowing the value of an other random variable S:

$$H(X|S) = -\int_S p(s) \int_X p(x|s) \log p(x|s). \tag{2.19}$$

The mutual information allows to quantify the information contained in a random variable X about an other random variable Y. It can also be viewed as the decrease of disorder brought by a random variable Y on a random variable X. The mutual information is defined by:

$$I(X;Y) = H(X) - H(X|Y) \tag{2.20}$$

We can notice that the mutual information between two independent variables is zero (since $H(X|Y) = H(X)$). Similarly to the conditional entropy, the conditional mutual information allows to quantify the information contained in a random variable about an other random variable, knowing the value of a third one. It can be written in various ways:

$$\begin{aligned} I(X;Y|S) &= H(X|S) - H(X|Y,S) && (2.21)\\ &= H(Y|S) - H(Y|X,S) && (2.22)\\ &= H(X|S) + H(Y|S) - H(X,Y|S) \\ &= D_{KL}\Big[p(X,Y|S)||p(X|S)p(Y|S)\Big] && (2.23) \end{aligned}$$

We can see with Equation 2.21 and Equation 2.22 that the mutual information is symmetric and that it characterizes the decrease in entropy on X brought by Y (or inversely). Equation 2.23 defines the conditional mutual information as the Kullback-Leibler divergence [Cover and Thomas, 2012] between distribution $P(Y,X|S)$ and the same distribution if Y and X were independent variables (the case where $H(Y|X,S) = H(Y|S)$).

It is straightforward to generalize mutual information to several random variables, leading to the multi-information [Slonim et al., 2001]:

$$I(X_1,\dots,X_N) = D_{KL}\Big[p(X_1,\dots,X_N)||p(X1)\dots p(X_N)\Big] \tag{2.24}$$

It quantifies the information that variables $X_1,\dots,X_N$ contain about each other. As above, it is described as the discrepancy between the joint distribution $p(X_1,\dots,X_N)$ and the same distribution if variables were independent.

For further information on these notions, the interested reader can refer to Cover and Thomas [2012]. Let us focus again on intrinsic rewards to give an example. By computing the mutual information between actions and the resulting states, an agent can provide intrinsic rewards that measure how much it controls its environment using one action. Other methods compute reward so that the agent learns a policy that maximizes the entropy of visited states. We give more details about this aspect in Chapter 3.

Combining intrinsic rewards and extrinsic reward In practice, there are multiple ways to integrate an intrinsic reward into a RL framework. The main approach is to compute the agent's reward r as a weighted sum of an intrinsic reward r_{int} and an extrinsic reward r_{ext}: $r = \alpha r_{int} + \beta r_{ext}$ [Kakade and Dayan, 2002, Burda et al., 2018, Gregor et al., 2016]. Of course, one of the weighting coefficient α and β can be set to 0. In this version, one can think of the intrinsic reward as an intrinsic bonus. When the extrinsic value function is important to compute the intrinsic reward or when the hyper-parameters α and β may change, the sum can be made at the Q-value level, i.e. $Q(s,a) = \alpha Q_{int}(s,a) + \beta Q_{ext}(s,a)$ [Kim et al., 2019b] where α and β ponderate each Q-value. As noticed in Section 2.2.3, intrinsic rewards generally evolve over time; while it may often decrease to zero [Baldassarre, 2011], this is not a general rule and it may happen that the agent cannot find an optimal stationary policy. Other works propose to learn several Q-values depending of the weighting parameters [Beyer et al., 2019] or learn a set of policies by making neurral networks dependent on the weights [Badia et al., 2019].

2.3.3 Decisions and hierarchical RL

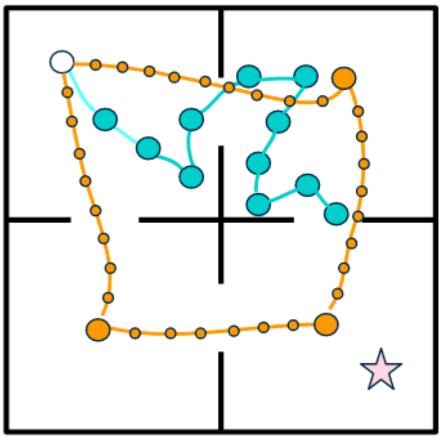


Figure 2.6: Visualization of trajectories of agents that start at the white circle and try to reach the star.The blue agent uses only low-level actions while the orange can set goals.

Hierarchical reinforcement learning (HRL) architectures are adequate candidates to model the decision hierarchy of an agent [Barto and Mahadevan, 2003, Dayan and Hinton, 1993, Sutton et al., 1999]. Dayan and Hinton [1993] introduced the feudal hierarchy, called *Feudal reinforcement learning*. In this framework, a manager selects the goals that workers will try to achieve by selecting low-level actions. Once the worker achieved the goal, the manager can select an other goal, so that the interactions keep going. The manager rewards the RL-based worker to guide its learning process; we formalize this with intrinsic motivation in the next section. Figure 2.6 illustrates the hierarchical trajectories of an orange agent that accesses four high-level goals, each one going into one of the four rooms. At the origin, the hierarchical architectures have been introduced to make easier the long-term credit assignment [Dayan and Hinton, 1993, Sutton et al., 1999]. This problem refers to the fact that rewards can occur with a temporal delay and will only very weakly affect all temporally distant states that have preceded it, although these states may be important to obtain that reward. Indeed, the agent must propagate the reward along the entire sequence of actions (through Equation 2.9) to reinforce the first involved state-action tuple. This process can be very slow when the action sequence is large. This problem also concerns determining which action is decisive for getting the reward, among all actions of the sequence (see also Section 2.1.2. In contrast, if an agent can take advantage of temporally-extended actions, a large sequence of low-level actions become a short sequence of time-extended decisions that make easier the propagation of rewards.

This goal setting mechanism can be extended to create managers of managers so that an agent can recursively define increasingly abstract decisions as the hierarchy of RL algorithms increases. Relatively to Figure 2.5, the internal environment of a RL module becomes the lower level module.

We can model these decisions as *options*. An *option* $op \in \mathcal{O}$ is defined through 3 components:

- A set of starting states $\mathcal{I} \subset S$ from which an *option* can be applied.

- A policy (or worker) that is responsible of achieving the *options* with lower-level actions. This is studied in the next section.
- A completion function $\mathcal{F}$ that specifies the probability of completing the *option* in each state.

Typically, the starting state can derive from d_0 (all *options* start at the beginning of an episode) or the full set of states S (*options can start everywhere*). The completion function can also set a probability 0 everywhere [Eysenbach et al., 2018], in this case, it ends at the same time as an episode. Such specific cases often occur [Eysenbach et al., 2018]. *Options* where originally learnt during a pre-training phase with exclusively extrinsic rewards [Sutton et al., 1999], it was meant to take advantage of expert knowledge on the task. However, in our framework, we are interested in intrinsically motivated agent, so, in the next section, we take a closer look on how to learn the policies that learn to achieve goals using intrinsic motivation. In particular, we will define goals, skills and explain how to build a contextual state.

2.3.4 Goal-parameterized RL

Usually, RL agents solve only one task and are not suited to learn multiple tasks. Thus, an agent is unable to generalize across different variants of a task. For instance, if an agent learns to grasp a circular object, it will not be able to grasp a square object. In the developmental model described in Section 2.3.1, the decisions can be hierarchically organized into several levels where an upper-level takes decision (or sets goals) that a lower-level has to satisfy. This questions: 1- how a DRL algorithm can make its policy dependent on the goal set by its upper-level decision module ? 2- How to compute the intrinsic reward using the goal ? These issues rise up a new formalism based on developmental machine learning [Colas et al., 2020b].

In this formalism, a **goal** is defined by the pair (g, R_G) where $G \subset \mathbb{R}^d$, R_G is a goal-conditioned reward function and $g \in G$ is the d-dimensional goal embedding. This contrasts with the notion of task which is proper to an extrinsic reward function assigned by an expert to the agent.

With such embedding, one can generalize DRL to multi-goal learning, or even to every available goal in the state space, with the Universal Value Function Approximator (UVFA) [Schaul et al., 2015]. UVFA integrates, by concatenating, the state goal embedding g with the state of the agent to create a contextual state $c = (g, s)$. Depending on the semantic meaning of a skill, we can further enhance the contextual states with other actions or states executed after starting executing the skill (cf. Section 3.5).

We can now define the **skill** associated to each goal as the goal-conditioned policy $\pi^g(a|s) = \pi(a|g, s)$; in other words, a skill refers to the sensorimotor mapping that achieve a goal [Thill et al., 2013]. This skill may by learnt or unlearnt according to the expected intrinsic rewards it gathers. It implies that, if the goal space is well-constructed (as often a ground state space for example, $R_G = S$), the agent can generalize its policy across the goal space, *i.e* the corresponding skills of two close goals are similar.

For example, let us consider an agent moving in a closed maze where every position in the maze can be a goal. We can set $G = S$ and set the intrinsic reward function to be the euclidian distance between the goal and the current state of the agent $R_G : S \times G \to \mathbb{R}, (s, g) \to ||s - g||_2$.

This formalism completes the instantiation of the architectures described in Section 2.3.1. Now we will explain how, in practice, one can efficiently learn the goal-conditioned policy.

2.3.5 Efficient learning with goal relabelling

When the goal space is a continuous state space, it is difficult to determine whether a goal is reached or not, since two continuous values are never exactly equal. Hindsight experience replay (HER) [Andrychowicz et al., 2017] tackles this issue by providing a way to learn on multiple objectives with only one interaction. With author's method, the agent can use an interaction done to accomplish one goal to learn on an other goal, by modifying the associated intrinsic reward. This mechanism greatly improves the sample efficiency since it avoids to try all interactions for every goals.

Let us roll out an example. An agent acts in the environment to gather a tuple (s, s', r_g, a, g) where r_g is the reward associated to the goal g. The agent can learn on this interaction, but can also use this interaction to learn other goals; to do so, it can change the goal into a new goal and recompute the reward, resulting in a new interaction $(s, s', r_{g'}, a, g')$. The only constraint for doing this is that the reward function $R(s, a, s', g')$ has to be known, which is the case with an intrinsic reward function. Typically, an agent can have a goal state and a reward function which is 1 if it is into that state and 0 otherwise. At every interaction, it can change its true goal state for its current state and learn with a positive reward.

2.4 Conclusion

We considered the setting where an agent interacts with its environment. We have highlighted several recent advances allowing to scale RL methods to high-dimensional state space and continuous action space (Section 2.1). But one needs other mechanism to make a learning agent truly independent from external interventions. Intrinsic motivation may play this role; while it comes from psychological literature, we emphasized its key properties so that we can use it in computational frameworks in Section 2.2. By theoretically reconsidering the RL framework and taking advantage of how DRL methods learn, we illustrated how intrinsic motivation can be efficiently translated into intrinsic rewards, HRL and goal-conditioned RL (Section 2.3).

However, we notice there are several ways to instantiate and apply an intrinsic motivation; the effects are different for each of them. This is reflected in practice: one may use goal-conditioned RL at a single decision-level or may not use any notions of goals, using only flat intrinsic rewards. Even across similar architectures, intrinsic rewards may be computed using very different heuristics; in addition, it may happen that R_G depends on the structure of G so that computing G may also influence the intrinsic reward. In each method, it may differently overcome a limitation of DRL. This observation motivates our work and we analyze the current literature in the next chapter.

Chapter 3

Survey on intrinsic motivation in DRL

In the previous chapter, we studied the evolution of DRL and gave a general overview on how to formally integrate intrinsic motivations with DRL. In this chapter, we study the role of intrinsic motivation in the framework of DRL.

Indeed, despite the recent improvements of DRL approaches (several are detailed in Section 2.1), they turn out to be most of the time unsuccessful when the rewards are scattered in the environment, as the agent is then unable to learn the desired behavior for the targeted task [Francois-Lavet et al., 2018]. Moreover, the behaviors learned by the agent are hardly reusable, both within the same task and across many different tasks [Francois-Lavet et al., 2018]. It is difficult for an agent to generalize its skills so as to learn to make high-level decisions in the environment. For example, such skill could be *go to the door* using primitive actions consisting in moving in the four cardinal directions; or even to *move forward* controlling different joints of a humanoid robot like in the robotic simulator MuJoCo [Todorov et al., 2012].

For several years now, IM is increasingly used in RL, fostered by important results and the emergence of deep learning. This paradigm offers a greater learning flexibility, through the use of a more general reward function, allowing to tackle the issues raised above when only an extrinsic reward is used. Typically, IM improves the agent ability to explore its environment, to incrementally learn skills independently of its main task, to choose an adequate skill to be improved and even to create a representation of its state with meaningful properties. In addition, as a consequence of its definition, IM does not require additional expert supervision, making it easily generalizable across environments.

A lot of very different works pretend to fit in the framework of IMs by proposing task-agnostic losses [Kulkarni et al., 2016, Zhang et al., 2019, Ostrovski et al., 2017], it follows the introduction of a plethora of erratic objectives. In fact, considering that objective biases (slow features [Wiskott and Sejnowski, 2002], bottleneck research [McGovern and Barto, 2001, Menache et al., 2002], expected cover time [Jinnai et al., 2019] . . .) or assumptions (access to object-oriented representations [Kulkarni et al., 2016], sensory separation [Zhao et al., 2021] . . .) do not question the task-agnosticity of an intrinsic motivation, one can introduce an infinite number of objectives, all with their specific application.

Scope and motivation of our review. In this chapter, we make the choice to study and group together methods that maximize information theoretic objectives. We closely match the definition of IM provided in Section 2.2.2 and we characterize it as the search for correlations among a set of internal variables. This way, **we revisit the notions of surprise, novelty and skill learning and show that they can encompass numerous works.** This allows us to situate a large body of works, to highlight important directions of research and to unify these intrinsic motivations using information theory.

Related works. The overall literature on IM is huge [Barto, 2013] and we only consider its application to DRL. Therefore, our study of IMs is not meant to be exhaustive. Intrinsic motivation currently attracts a lot of attention and several works made a restricted study of the approaches. Colas et al. [2020b] and Amin et al. [2021] respectively focus on the different aspects of skill learning and exploration ; Baldassarre [2019] studies intrinsic motivation through the lens of psychology, biology and robotic ; Pateria et al. [2021] review hierarchical reinforcement learning as a whole, including extrinsic and intrinsic motivations; Linke et al. [2020] experimentally compare different goal selection mechanisms. In contrast with these approaches, we study a large part of objectives all based on intrinsic motivation through the lens of information theory. We assume that our work is in line with the work of Schmidhuber [2008], which postulates that organisms are guided by the desire to compress the information they receive. However, by reviewing the more recent advances in the domain, we formalize the idea of compression with the tools from information theory.

To sum up, in this chapter, we investigate the use of IM in the framework of DRL and consider the following aspects:

- The role of IM in addressing the challenges of DRL.
- Classifying current works through information theoretic principles.
- Important outlooks of IM in RL.
- Proposal of an unified view of IMs.

This chapter is organized as follows. As a first step, we highlight the main current challenges of RL and identify the need for an additional outcome (Section 3.1). Then, we briefly explain our classification (Section 3.2), namely surprise, novelty and skill learning and we detail how current works fit it (respectively Section 3.3, Section 3.4 and Section 3.5). Thereafter, we highlight some important outlooks of the domain (Section 3.6). Finally, in Section 3.7, we unify intrinsic motivations under one information theoretic objective function and discuss its potential role as a universal guiding principle.

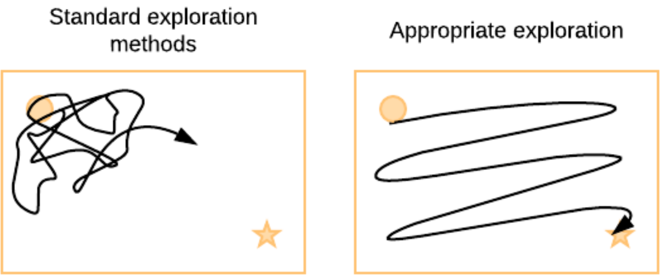


Figure 3.1: Illustration of the sparse reward issue in a very simple setting. The agent, represented by a circle, strives to reach the star. The reward function is one when the agent reaches the star and zero otherwise. On the left side, the agent explores with standard methods such as ϵ-greedy; as a result, it stays in its surrounded area because of the temporal inconsistency of its behaviour. On the right side, we can imagine an ideal exploration strategie where the agent covers the whole state space to discover where rewards are located.

3.1 Challenges

3.1.1 Sparse rewards

Classic RL algorithms operate in environments where the rewards are **dense**, *i.e.* the agent receives a reward after almost every completed action. In this kind of environment, naive exploration policies such as ϵ-greedy [Sutton and Barto, 1998] or the addition of a Gaussian noise on the action [Lillicrap et al., 2015] are effective. More elaborated methods can also be used to promote exploration, such as Boltzmann exploration [Cesa-Bianchi et al., 2017, Mnih et al., 2015] or an exploration in the parameter-space [Plappert et al., 2017, Rückstiess et al., 2010, Fortunato et al., 2017]. In environments with **sparse** rewards, the agent receives a reward signal only after it executed a large sequence of specific actions. The game *Montezuma's revenge* [Bellemare et al., 2015] is a benchmark illustrating a typical sparse reward function. In this game, an agent has to move between different rooms while picking up objects (it can be keys to open doors, torches, ...). The agent receives a reward only when it finds objects or when it reaches the exit of the room. Such environments with sparse rewards are almost impossible to solve with the above mentioned *undirected* exploration policies [Thrun, 1992] since the agent does not have local indications on the way to improve its policy. Thus the agent never finds rewards and cannot learn a good policy with respect to the task [Mnih et al., 2015]. Figure 3.1 illustrates the issue on a simple environment.

This issue stresses out the need for *directed* exploration methods [Thrun, 1992]. While intrinsic motivation can provide such direction, the principle of "optimism in face of uncertainty" [Audibert et al., 2007] can also execute a directed exploration without intrinsic motivation [Thrun, 1992]. Briefly, this principle can incite agents to go in areas with a lot of epistemic uncertainties about its Q-values [Ciosek et al., 2019, Pacchiano et al., 2020]. Yet, it is hard to approximate the epistemic uncertainty and it only slightly improves exploration [Ciosek et al., 2019]. This principle can also relate with some intrinsic motivations when we consider uncertainty about models (see Section 3.3.2).

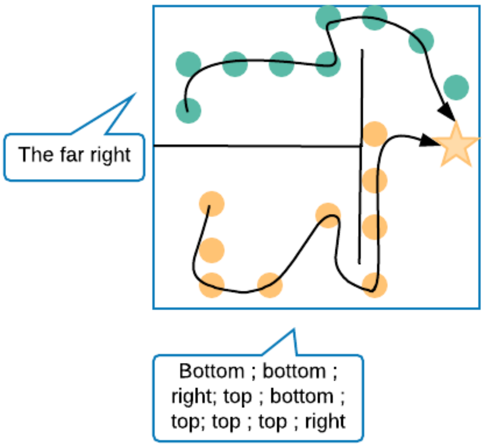

The far right
Bottom ; bottom ; right; top ; bottom ; top; top ; top ; right

Figure 3.2: Illustration of the benefits of using *options*. Agents, represented by circles, have to reach the star. The green agent can use a *skill Go to the far right*; the orange agent can only use primitive actions to reach the star.

Rather than working on an exploration policy, it is common to shape an intermediary dense reward function that adds to the reward associated to the task in order to make the learning process easier for the agent [Su et al., 2015]. However, the building of a reward function often reveals several unexpected errors [Ng et al., 1999, Amodei et al., 2016] and most of the time requires expert knowledge. For example, it may be difficult to shape a local reward for navigation tasks. Indeed, one has to be able to compute the shortest path between the agent and its goal, which is the same as solving the navigation problem. On the other side, the automation of the shaping of the local reward (without calling on an expert) requires too high computational resources [Chiang et al., 2019].

We will see in Section 3.3, Section 3.4 how IM is a valuable method to encourage exploration in a sparse rewards setting.

3.1.2 Temporal abstraction of actions

As argued in Section 2.3.3, skills, through hierarchical RL, are a key element to speed up the learning process since the number of decisions to take is significantly reduced when skills are used. In particular, they make easier the *credit assignment*. Skills can be manually defined, but it requires some extra expert knowledge [Sutton et al., 1999]. To avoid providing hand-made skills, several works proposed to learn them with extrinsic rewards [Bacon et al., 2017, Li et al., 2020b]. However, if an agent rather learns skills in a *bottom-up* way, *i.e* with intrinsic rewards rather than extrinsic rewards, learnt skills become independent from possible tasks. This way, skills can be reused across several tasks to improve transfer learning [Aubret et al., 2020, Heess et al., 2016] and an agent can learn skills even though it does not access rewards, improving exploration when rewards are sparse [Machado et al., 2017]. Let us illustrate both advantages.

Exploration when rewards are sparse. Figure 3.2 illustrates the benefit in terms of exploration when an agent hierarchically uses skills. The green circle can use a skill *Go to the far right*, to reach the rewarding star while the orange agent can only use low-level cardinal movements. The problem of exploration becomes trivial for the agent using skills, since one exploratory action can lead to the reward. In contrast, it requires an entire sequence of specific low-level actions for the other agent to find the reward. This problem arises from the minimal number of specific actions needed to get a reward (see also Section 3.1.1). A thorough analysis of this aspect can be found in [Nachum et al., 2019b].

Reusing skills across several tasks. Skills learnt with intrinsic rewards are not specific to a task. Assuming an agent is required to solve several tasks in a similar environment, *i.e* a single MDP with a changing extrinsic reward function, an agent can execute its discovered skills to solve all tasks. Typically, in Figure 3.2, if both agents learnt to reach the star and we move the star somewhere else in the environment, the green agent would still be able to execute *Go to the far right* and executing this skill may make the agent closer to the new star. In contrast, the orange agent would have to learn a whole new policy.

In Section 3.5, we provide insights on how an agent can discover skills in a *bottom-up* way.

3.2 Classification of methods

In order to tackle the problem of exploration, an agent may want to identify and return in **rarely visited** states or **unexpected** states, which can be quantified with current intrinsic motivations. We will particularly focus on two objectives that address the challenge of exploring with sparse rewards, each with different properties: maximizing novelty and surprise. We formalize novelty and surprise through the lens of information theory (in respectively Section 3.4 and Section 3.3) and the works that instantiate it. Surprise and novelty are specific notions that have often been used in an interchanged way and we are not aware of a currently unanimous definition of novelty [Barto et al., 2013]. The third notion we study, skill learning, focuses on the issue of skill abstraction.

Table 3.1 sums up our taxonomy. We classify intrinsic motivations in three categories of objectives based on information theory that reflects the high-level studied concepts of novelty, surprise and skill learning. In practice, we mostly take advantage of the *mutual information* to provide a quantity for our conceptual objectives. These objectives are compatible with each other and may be used simultaneously, as argued Section 3.6.3. Within each category of objectives, we additionnally highlight several ways to maximize each objective and provide details about the underlying methods of the literature.

Surprise: Following the definition of Itti and Baldi [2009], we reexplore the notion of surprise and quantify it by $I(S';\Phi|h,S,A)$ where h refers to a dataset of interactions and Φ represents a distribution over parameters of a forward/density model. This objective has also been called expected information gain [Sun et al., 2011]. Based on the works we analyze, we study surprise maximization over density models and forward models, which are two ways of measuring the unexpectedness. Surprise can also be maximized through a minimax game on the prediction error using several approximations.

Novelty: Based on the analysis of Barto et al. [2013], we define novelty-seeking behavior as actively maximizing the mutual information between states and a learnt representation of states Z,

<table>
<tr><td colspan="4"><b>Surprise</b>: $I(S';\Phi|h,S,A)$, Section 3.3</td></tr>
<tr><td>Formalism</td><td>Information gain over forward model</td><td>Minimax prediction error</td><td>Information gain over density model</td></tr>
<tr><td>Sections</td><td>Section 3.3.2</td><td>Section 3.3.3</td><td>Section 3.3.4</td></tr>
<tr><td>Rewards</td><td>$D_{KL}(p(\Phi|h,s,a,s')||p(\Phi|h))$</td><td>$||s'-\hat{s}'||_2^2$</td><td>$\frac{1}{\sqrt{\hat{N}(s')}}$</td></tr>
<tr><td colspan="4"><b>Novelty</b>: $I(S;Z)$, Section 3.4</td></tr>
<tr><td>Formalism</td><td>Parametric density</td><td colspan="2">K-nearest neighbors</td></tr>
<tr><td>Sections</td><td>Section 3.4.1</td><td colspan="2">Section 3.4.2</td></tr>
<tr><td>Rewards</td><td>$-\log \rho(s')$</td><td colspan="2">$\log(1+\frac{1}{K}\sum_0^K ||g(s')-nn_k(g(S),g(s'))||_2)$</td></tr>
<tr><td colspan="4"><b>Skill learning</b>: $I(G;f(\mathcal{T}))$, Section 3.5</td></tr>
<tr><td>Formalism</td><td>Fixed goal distribution</td><td>Goal-state achievement</td><td>Proposing diverse goals</td></tr>
<tr><td>Sections</td><td>Section 3.5.1</td><td>Section 3.5.2</td><td>Section 3.5.3</td></tr>
<tr><td>Rewards</td><td>$\log p(g|s')$</td><td>$-||s_g-s'||_2^2$</td><td>$(1+\alpha_{skew})\log p(s_g)$
$\alpha_{skew}<0$
(Goal selection policy)</td></tr>
</table>

Table 3.1: Summary of our taxonomy of intrinsic motivations in DRL. The function f outputs a part of the trajectories $\mathcal{T}$, Z and G are internal random variables respectively denoting state representations and self-assigned goals. Please, refer to the corresponding sections for more details about methods and notations.

$I(S;Z)$. We divide this objective maximization into two kinds of methods: a direct maximization of a parametric entropy of states, and an entropy maximization based on a k-nearest neighbors approximation.

Skill learning: We formalize skill learning as maximizing the mutual information between a goal representation G and a part of a time-extended trajectory $f(\mathcal{T})$, $I(G;f(\mathcal{T}))$ while following G. We will consider two ways to achieve this: 1- fixing the goal distribution; 2- deriving the goal representation from the state space. We will see that the second point also needs to maximize the entropy of goals-states.

We justify our objective within each category and study the different ways to maximize this objective and the advantages/disadvantages. In practice, surprise and novelty are currently maximized as a flat intrinsic motivation, *i.e* without using hierarchical decisions. This mostly helps to improve exploration when rewards are sparse. In contrast, skill learning allows to define time-extended hierarchical skills which enjoy all the benefits argued in Section 3.1.2.

3.3 Surprise

In this section, we study methods that maximize the surprise. Firstly, we formalize the notions of surprise, then we will study three approaches for computing intrinsic rewards based of these notions.

3.3.1 Definition of surprise

In this section, we assume the agent learns either a density model (Section 3.3.4) or a forward model of the environment (Sections 3.3.2 and 3.3.3) parameterized by $\phi \in \Phi$. The density model induces a marginal distribution of state $p(S|\phi)$ and a forward model computes the next-state distribution conditioned on a tuple state-action $p(S'|S,A,\phi)$. Typically, this can be the parameters of a neural network. Trying to approximate the true model, the agent maintains an approximate distribution $p(\Phi|h)$ of models, where $h_t = h$ refers to the ordered history of interactions $((s_0,a_0,s_1),(s_1,a_1,s_2),\dots,(s_{t-1},a_{t-1},s_t))$. In this section, h simulates a dataset of interactions, we use it to clarify the role of the dataset. It is important to notice that the policy feeds this h.

In this case, surprise quantifies the mismatch between an expectation and the true experience of an agent [Barto et al., 2013, Ekman and Davidson, 1994]. In this paper, we refer to the definition of Itti and Baldi [2009], which define it as the discrepancy between a prior distribution of beliefs and the posterior probability distribution following an observation [Itti and Baldi, 2009, Storck et al., 1995]. If an agent maximizes the surprise over a model through interactions with the environment, which is often the case [Barto et al., 2013], it leads to the expected information gain objective [Sun et al., 2011]. Intuitively, the agent returns in states where it experienced an unexpected transition. Using the KL-divergence to assess the discrepancy, surprise can be computed as $D_{KL}(p(\Phi|h_{t+1})||p(\Phi|h_t))$ where $\phi \in \Phi$ are parameters of a model and t denotes the timestep.

In this case, the agent has a prior distribution about model parameters $p(\Phi)$ and this model can be updated using the Bayes rule:

$$p(\phi|h,s,a,s') = \frac{p(\phi|h)\, p(s'|h,s,a,\phi)}{p(s'|h,s,a)}. \tag{3.1}$$

The **expected information gain** [Sun et al., 2011, Little and Sommer, 2013] over a forward or density model parameterized by ϕ can be formulated as:

$$\begin{aligned} IG(h,A,S',S,\Phi) &= \mathop{\mathbb{E}}_{\substack{(s,a)\sim\pi \\ s'\sim p_T(\cdot|s,a,h)}} D_{KL}(p(\Phi|h,s,a,s')||p(\Phi|h)). \\ &\approx I(S';\Phi|h,A,S) \end{aligned} \tag{3.2}$$

where p_T refers, in this section, to the true probability induced by the transition fonction of the environment. Actively maximizing the expected information gain amounts to reduce the uncertainty of the model. We emphasize that $p(\phi|h) = p(\phi|h,a,s)$ since only full transitions provide informations about the true dynamics of the environment. In this case, $p(s'|s,a,h)$ does not refer to the probability induced by the environment, but rather to the probability induced by the current history of transitions. This is stressed out by writing:

$$p(s'|s,a,h) = \sum_{\phi\in\Phi} p(s'|s,a,h,\phi)p(\phi|s,a,h). \tag{3.3}$$

Let us assume ϕ are parameters of a forward model. When the history gets large enough, we hypothetize that the distribution $p(\phi|s,a,h)$ becomes not null only for ϕ being equal to the probability induced by the true environment transition function ϕ_T. In this case, $p(s'|s,a,h)$ indeed refers to the probability of a transition, *i.e* $p(s'|s,a,h) = p(s'|s,a,h,\phi_T)$. Similar reasoning can be obtained if ϕ encodes a density model and if the environment's marginal distribution of states is the uniform one.

In the following, we will study three objectives: the expected information gain over forward models, an approximation of surprise over forward models leading to a minimax game and the expected information gain over density models.

3.3.2 Information gain over forward model

We first study the works that maximize the expected information gain over forward models. Here, ϕ are parameters of a forward model. Using Equation 3.2, we can extract an intrinsic reward:

$$R(s,a,s') = D_{KL}(p(\Phi|h,s,a,s')||p(\Phi|h)). \tag{3.4}$$

This way, an agent executes actions that provide information about the dynamics of the environment. This allows, on one side, to push the agent towards areas it does not know, and on the other side to prevent attraction towards stochastic areas. Indeed, if the area is deterministic, environment transitions are predictable and the uncertainty about its dynamics can decrease. At the opposite, if transitions are stochastic, the agent turns out to be unable to predict transitions and does not reduce uncertainty. The exploration strategy **VIME** [Houthooft et al., 2016] computes this intrinsic reward by modelling $p(\phi|h)$ with Bayesian neural networks [Graves, 2011]. The interest of Bayesian approaches is to be able to measure the uncertainty of the learned model [Blundell et al., 2015]. This way, assuming a fully factorized Gaussian distribution over model parameters, the KL-divergence has a simple analytic form [Houthooft et al., 2016, Linke et al., 2020], making it easy to compute. However, the interest of the proposed algorithm is shown only on simple environments and the reward can be computationally expensive to compute. Achiam and Sastry [2017] propose a similar method (**AKL**), with comparable results, using deterministic neural networks, which are simpler and quicker to apply. The weak performance of both models is probably due to the difficulty to retrieve the uncertainty reduction by rigorously following the mathematical formalism of information gain.

In **JDRX** [Shyam et al., 2019], authors show that one can maximize the information gain by computing the Jensen-Shannon or Jensen-Rényi divergence between distributions of states induced by several forward models. The more the models are trained on a state-action tuple, the more they will converge to the expectated distribution of next states. Intuitively, the reward represents how much the different transition models disagree on the next-state distribution. Other works also maximize a similar form of disagreement [Pathak et al., 2019, Yao et al., 2021, Sekar et al., 2020] by looking at the variance of predictions among several learnt transition models. These models can also predict in latent spaces [Sekar et al., 2020]. It appears that such methods are competitive with state of the art approaches [Burda et al., 2019]. However the main intrinsic issue is computational since it requires multiple forward models to train.

To conclude, despite the theorical power of the information gain for improving exploration, it remains hard to efficiently estimate it and use it in difficult tasks. In the next paragraph, we review how we can simplify the expected information gain.

3.3.3 Prediction error of forward model

Using a deterministic forward model ϕ, one can also maximize the surprise using several approximations.

$$\begin{aligned} I(S';\Phi|h,A,S) &\approx H_T(S'|h,A,S) - H_T(S'|A,\Phi,S,h) \\ &= - \mathop{\mathbb{E}}_{(s',s,a)\sim\pi} \log p(s'|h,s,a) + \mathop{\mathbb{E}}_{\substack{\phi\sim p(\cdot|h,s,a,s') \\ (s',s,a)\sim\pi}} \log p(s'|s,a,\phi,h) \end{aligned} \tag{3.5}$$

where H_T refers to the entropy with true transitions in its expected part. Equation 3.5 means that, in order to maximize the expected information gain, the agent should go in areas where it reduces the entropy of its approximation; indeed, the right-hand term predicts with one more transition than the left-hand term. Now, we will make the strong assumption that the second term of Equation 3.5 can be optimally maximized through the loss of the predictive model if $H_T(S'|h,A,S)$ is maximized using actions. A fully deterministic environment is a particular case of such setting since the second term would converge to 0 for all transitions. In this case, the agent just has to go in stochastic areas. One can lower-bound the first term using the stochastic forward model:

$$-\underset{(s',s,a)\sim\pi}{\mathbb{E}} \log p(s'|h,a,s) = -\underset{(s',s,a)\sim\pi}{\mathbb{E}} \log \underset{\phi\sim p(\cdot|h)}{\mathbb{E}} p(s'|h,a,s,\phi) \tag{3.6}$$

$$\geq -\underset{\substack{\phi\sim p(\cdot|h)\\(s',s,a)\sim\pi}}{\mathbb{E}} \log p(s'|\phi,h,a,s) \tag{3.7}$$

where we applied the Jensen inequality. The two final terms of Equation 3.5 are essentially similar while being optimized by different parts of the model (actions and forward model), it results a minimax objective. The predictive model strives to reduce the mismatch between true next-states and predictions while the agent acts to maximize it.

One can model $p(s'|h,s,a,\phi)$ with a unit-variance Gaussian distribution in order to obtain a tractable loss. This way, we have:

$$\underset{\substack{\phi\sim p(\cdot|h)\\(s',s,a)\sim\pi}}{\mathbb{E}} -\log p(s'|\phi,h,a,s) \approx \underset{\substack{\phi\sim p(\cdot|h)\\(s',s,a)\sim\pi\\ \hat{s}'\sim p(\cdot|s,a,\phi)}}{\mathbb{E}} -\log \frac{1}{(2\pi)^{d/2}} e^{-0.5(s'-\hat{s}')^T(s'-\hat{s}')} \tag{3.8}$$

$$\propto \underset{\substack{\phi\sim p(\cdot|h)\\(s',s,a)\sim\pi\\ \hat{s}'\sim p(\cdot|s,a,\phi)}}{\mathbb{E}} ||s'-\hat{s}'||_2^2 + Const \tag{3.9}$$

As explained in Section 3.3.1, we assume $p(s'|h,a,s)$ represents the the probability induced by the transition function of the environment and $\hat{s}'$ represents the mean prediction. It follows a line of works that generates intrinsic rewards equal to $||s'-\hat{s}'||_2^2$ where

$$\hat{s}' = \arg\max_{s''\in S} p(s''|h,a,s,\phi) \tag{3.10}$$

assuming ϕ paramaterizes a deterministic forward model. Therefore, the agent should go towards areas where the prediction of the state following a state-action tuple is difficult. Following the objective, we can compute an intrinsic reward as:

$$R(s,a,s') = ||g(s') - g(\hat{s}')||_2^2 \tag{3.11}$$

where g is a generic function (e.g. identity or a learnt one) encoding the state space into a feature space. Equation 3.11 amounts to reward the predictor error of ϕ in the representation g. In the following, we will see that learning a relevant function g is the main challenge.

The first natural idea to test is whether a function g is required. Burda et al. [2019] learn the forward model from the **ground state space** and observe it is inefficient when the state space is large. In fact, the L2[1] distance is meaningless in such high-dimensional state space. In contrast, they raise up that **random features** extracted from a random neural network can be very competitive with other state-of-art methods. However they poorly generalize to environment changes. An

[1] Euclidian distance.

other model, **Dynamic Auto-Encoder (Dynamic-AE)** [Stadie et al., 2015], computes the distance between the predicted and the real state in a state space compressed with an auto-encoder [Hinton and Salakhutdinov, 2006]. g is then the encoding part of the auto-encoder. However this approach only slightly improves the results over Boltzmann exploration on some standard Atari games. Other works also consider a dynamic-aware representation [Ermolov and Sebe, 2020].

These methods are unable to handle the local stochasticity of the environment [Burda et al., 2019]. For example, it turns out that adding random noise in a 3D environment attracts the agent; it passively watches the noise since it is unable to predict the next observation. This problem is also called *the white-noise* problem [Pathak et al., 2017, Schmidhuber, 2010]. This problem emerges by considering only the maximization of Equation 3.7, making us assume environments are deterministic. Therefore, exploration with prediction error breaks down when this assumption is no longer true.

To tackle exploration with local stochasticity, the **intrinsic curiosity module (ICM)** [Pathak et al., 2017] learns a state representation function g end-to-end with an *inverse model* (i.e. a model which predicts the action done between two states). Thus, the function g is constrained to represent things that can be controlled by the agent during next transitions. Secondly, the forward model used in ICM predicts, in the feature space computed by g, the next state given the action and the current state. The prediction error does not incorporate the white-noise that does not depend on actions, so it will not be represented in the feature state space. ICM notably allows the agent to explore its environment in the games *VizDoom* et *Super Mario Bros*. Building a similar action space, **Exploration with Mutual Information (EMI)** [Kim et al., 2019a] significantly outperforms previous works on Atari but at the cost of several complex layers. EMI transfers the complexity of learning a forward model into the learning of a space and action representation through the maximization of $I([s,a];s')$ and $I([s,s'];a)$. Then, the forward model ϕ is constrained to be a simple linear model in the representation space. Furthermore, EMI introduces a *model error* which offloads the linear model when a transition remains strongly non-linear (such as a screen change). However one major drawback of ICM and EMI is the incapacity of their agent to keep in their representation what depends on their long-term control. For instance, in a POMDP, an agent may perceive the consequences of its actions several steps later.

An other way to tackle local stochasticity can be to maximize the improvement of prediction error, or learning progress, of a transition model [Schmidhuber, 1991, Azar et al., 2019, Lopes et al., 2012, Oudeyer et al., 2007, Kim et al., 2020]. However, it turns out to be hard to estimate since the reward depends on the efficiency of the gradient update of the forward model. In its stochastic variant, rewarding the learning progress correlates with the reduction of entropy, thereby merging the information gain formalism of Section 3.3.2.

Conclusion. While these methods perform well in deterministic environments, they struggle to offset the determinism assumption that underpines the focus on Equation 3.7; it results that standard methods focus on the more stochastic areas. Methods that tackle stochasticity may not predict important long-term information about the environment or they need to compute a learning progress measure, which is non-trivial. In the next paragraph, we explore the information gain related to an other kind of models: a density model.

3.3.4 Information gain over density model

Surprisal can also arise by quantifying *the discrepancy between its probability of occurring and the fact that it actually occurred* [Barto et al., 2013]. To quantify this probability of occuring, in this paragraph, we assume the agent tries to learn a density model $\phi \in \Phi$ that approximates the true current marginal density distribution of states $p(s')$. In this setting, we can define the expected information gain over a density model ϕ [Bellemare et al., 2016]:

$$IG(h, S, A, S') \approx \mathop{\mathbb{E}}_{s' \sim \pi} D_{KL}(p(\phi|h, s')||p(\phi|h)). \tag{3.12}$$

We hypothetize that the adversarial training that results from the objective (active maximization of the KL-divergence and density fitting) results in an approximately uniform distribution of states (and auniform density estimation). This may be due to the convexity of the KL-divergence in $p(\phi|h, a, s, s')$ and $p(\phi|h)$ but we leave the proof to future work. To our knowledge, no works directly optimize this objective, but it has been shown that this objective lower-bounds the squared inverse pseudo-count objective [Bellemare et al., 2016], which derives from count-based objectives; in the following, we will review *count* and *pseudo-count* objectives.

To efficiently explore its environment, an agent can count the number of times it visits a state and returns in rarely visited states. Such methods are said to be *count-based* [Strehl and Littman, 2008]. As the agent visits a state, the intrinsic reward associated with this state decreases. It can be formalized with:

$$R(s, a, s') = \frac{1}{\sqrt{N(s')}} \tag{3.13}$$

where $N(s)$ is the number of times that the state s has been visited.

Although this method is efficient and tractable in a tabular environment (with a discrete state space), it hardly scales when states are numerous or continuous since an agent never really returns in the same state. A first solution proposed by Tang et al. [2017], called **TRPO-AE-hash**, is to hash the state space when it is too large. However, these results are only slightly better than those obtained with a classic exploration policy. An other line of works propose to adapt counting to high-dimensional state spaces via *pseudo-counts* [Bellemare et al., 2016]. Essentially, *pseudo-counts* allow the generalization of the count from a state towards neighbourhood states using a learnt density model ϕ. This is defined as:

$$\hat{N}(s') = \frac{p(s'|\phi)(1 - p(s'|\phi')}{p(s'|\phi') - p(s'|\phi)} \tag{3.14}$$

where $\phi'(s)$ computes the density of s after having learnt on s. In fact, Bellemare et al. [2016] show that, under some assumptions, *pseudo-counts* increase linearly with the true counts. In this category, **DDQN-PC** [Bellemare et al., 2016] and **DQN-PixelCNN** [Ostrovski et al., 2017] compute ϕ using respectively a Context-Tree Switching model (CTS) [Bellemare et al., 2014] and a Pixel-CNN density model [Van den Oord et al., 2016]. Although the algorithms based on density models work on environments with sparse rewards, they add an important complexity layer [Ostrovski et al., 2017]. One can preserve the quality of observed exploration while decreasing the computational complexity of the pseudo-count by computing it in a learnt latent space [Vezzani et al., 2019, Martin et al., 2017].

There exists several other well-performing tractable exploration methods like **RND** [Burda et al., 2018], **DQN+SR** [Machado et al., 2018], **RIDE** [Raileanu and Rocktaschel, 2020] or **BeBold** [Zhang et al., 2020b]. These papers argue the reward they propose more or less relate to a count estimation.

Conclusion. Maximizing the information gain over a density model may maximize the pseudo-count, which relates to count-based objectives. They provide interesting feedbacks for exploration, but in practice, pseudo-counts are hard to approximate since they rely on a powerfull density model, a strict online estimation of density and they assume $p(s|\phi)$ strictly increases $\forall s \in S$ [Ostrovski et al., 2017]. In addition, they also struggle with the problem of randomness. For instance, let us assume that one (state, action) tuple can lead to two very different states with 50% chance each. The algorithm will manage to count for both states the number of visits, although it would take twice as long to avoid to be too much attracted. However, these methods do not address the white-noise problem since next states may be randomly generated at every steps. In this case, it is unclear how these methods could resist the temptation of going into this area since the counting associated to this state will never increase.

3.4 Novelty maximization

Novelty quantifies how much a stimuli contrasts with a previous set of experiences [Barto et al., 2013, Berlyne, 1966]. More formally, Barto et al. [2013] defend that *an observation is novel when a representation of it is not found in memory, or, more realistically, when it is not "close enough" to any representation found in memory*. Previous experiences may be collected in a bounded memory or distilled in a learnt representation.

Several works propose to formalize novelty seeking as looking for low-density states [Becker-Ehmck et al., 2021], or similarly (cf. Section 3.4.2), states that are different from others [Lehman and Stanley, 2011, Conti et al., 2018]. In our case, this would result in maximizing the entropy of a state distribution. This distribution can be the t-steps state distribution (cf. Equation 2.3) $H(d_t^\pi(S))$ or the entropy of the stationary state-visitation distribution over a finite horizon T:

$$H(d^\pi(S)) = H(\frac{1}{T}\sum_{t=1}^{T} d_t^\pi(S)). \tag{3.15}$$

This formalization is not perfect and does not fit several intuitions about novelty [Barto et al., 2013]. Barto et al. [2013] criticize such definition by stressing out that very distinct and memorable events may have low probabilites of occuring while not being novel (*e.g* a wedding). They suggest that novelty may rather relates to the acquisition of a representation of the incoming sensory data. Following this definition, we propose to formalize novelty seeking behaviors as those that *actively* maximize the mutual information between states and their representation $I(S;Z) = H(S) - H(S|Z)$ where Z is a low-dimensional space ($|Z| \leq |S|$). This objective is commonly known as the *infomax* principle. [Linsker, 1988, Almeida, 2003, Bell and Sejnowski, 1995, Hjelm et al., 2019]; in our case, it amounts to **actively** learning a representation of the environment. Most of works focus on actively maximizing the entropy of state distribution while a representation learning function minimizes $H(S|Z)$.

There are several ways to maximize the state-entropy, we separate them based on how they maximize the entropy. We found two kind of methods: low-density search and k-nearest neighbors methods.

3.4.1 Direct entropy maximization

The most evident way to maximize the entropy of states consists in maximizing $H(\rho(s))$ where $\rho(s)$ approximates a density model $p(s)$. If we access this density model, it becomes straightforward to discover a policy that maximizes the entropy of a stationary state distribution [Hazan et al., 2019]. But computing $\rho(s)$ is challenging in high-dimensional state spaces. Several methods propose to estimate $\rho(s)$ using variational inference [Zhang et al., 2021, Islam et al., 2019, Lee et al., 2019, Pong et al., 2020] based on autoencoder architectures. In this setting, we can use either Equation 3.16 [Lee et al., 2019] or Equation 3.17 [Pong et al., 2020], assuming z is a compressed latent variable, $p(z)$ a prior distribution [Kingma and Welling, 2014] and $q_{decoder}$ a neural network that ends with a diagonal Gaussian.

$$\begin{aligned} \rho(s) &\approx q_{decoder}(s|z)q_{encoder}(z|s) && (3.16) \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(z)}{q_{encoder}(z|s)}q_{decoder}(s|z) && (3.17) \end{aligned}$$

Equation 3.17 is unbiased but more expensive to compute than Equation 3.16 since it requires decoding several samples. Basically, this estimation allows to reward an agent [Berseth et al., 2020, Lee et al., 2019, Zhang et al., 2021] according to:

$$R(s, a, s') = -\log \rho(s').$$

Within this setting, Pong et al. [2020] and Lee et al. [2019] learn new skills that target these novel states (see also Section 3.5). **MaxRenyi** [Zhang et al., 2021] uses the Rény entropy, a more general version of the Shannon entropy, to give more importance to very low-density states. Islam et al. [2019] propose to condition the state density estimation with policy parameters in order to directly back-propagate the gradient of state-entropy into policy parameters. Although **MaxRenyi** achieves good scores on *Montezuma's revenge* with pure exploration, maximizing the ground state entropy may not be adequate since two closed ground states are not necessarily neighbors in the true environment [Aubret et al., 2021]. Following this observation, **GEM** [Guo et al., 2021] rather maximizes the entropy of the estimated density of states considering the dynamic-aware proximity of states, $H(R)$. However they do not actively consider $H(R|S)$.

Conclusion. Generally speaking, these methods need an accurate density model to provide

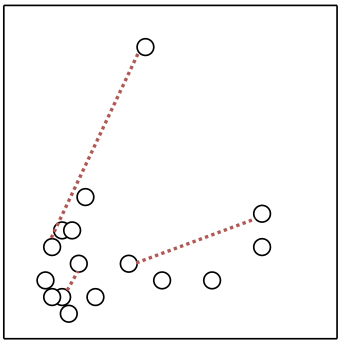


Figure 3.3: Illustration of the correlation between density and the fourth-nearest neighbor distance. Circles represent states and red dotted lines show the distance between a state and its fourth nearest neighbor.

rewards. In the next paragraph, we study methods that avoid learning a density model.

3.4.2 K-nearest neighbors approximation of entropy

Several works propose to approximate the entropy of a distribution using samples and their k-nearest neighbors [Singh et al., 2003, Kraskov et al., 2004]. In fact such objective has already been refered to as novelty [Conti et al., 2018]. Assuming $nn_k(S, s_i)$ is a function that outputs the k-th closest state to s_i in S, this approximation can be written as:

$$H(S) \propto \frac{1}{|S|} \sum_{s_i \in S} \log ||s_i - nn_k(S, s_i)||_2 + \chi(|S|) + Const \tag{3.18}$$

where $\chi(s)$ is the digamma function. This approximation assumes the uniformity of states in the ball centered on a sampled state with radius $||s_i - nn_k(S, s_i)||_2$ [Lombardi and Pant, 2016] but its full form is unbiased with a large number of samples [Singh et al., 2003]. Intuitively, it means that the entropy is proportional to the average distance between states and their neighbors. Figure 3.3 shows how density estimation relates to k-nearest neighbors distance. We clearly see that low-density states tend to be more distant from their nearest neighbors.

Few methods [Mutti et al., 2020] provably relates to such estimations, but several approaches take advantage of the distance between state and neighbors to generate intrinsic rewards, making them related to such entropy maximization. For instance, **APT** [Liu and Abbeel, 2021] proposes new intrinsic rewards based on the k-nearest neighbors estimation of entropy:

$$R(s, a_t, s') = \log(1 + \frac{1}{K} \sum_{0}^{K} ||g(s') - nn_k(g(S), g(s'))||_2) \tag{3.19}$$

where g is a representation function learnt with a contrastive loss based on data augmentation [Srinivas et al., 2020] and K denotes the number of k-nn estimations. By looking for distant state embeddings during an unsupervised pre-training phase, they manage to considerably speed up task-learning in the DeepMind Control Suite. The representation g can also derive from a random encoder [Liu and Abbeel, 2021] or a constrastive loss that ensures the euclidian proximity between consecutive states [Tao et al., 2020, Yarats et al., 2021].

Identifying different states. Instead of relying on euclidian distance, one can try to learn a similarity function. **EX**^{2} [Fu et al., 2017] learns a discriminator to differentiate states from each other: when the discriminator does not manage to differentiate the current state from those in the buffer, it means that the agent has not visited this state enough and it will be rewarded. States are sampled from a buffer, implying the necessity to have a large buffer. To avoid this, some methods distill recent states in a prior distribution of latent variables [Kim et al., 2019b, Klissarov et al., 2019]. The intrinsic reward for a state is then the KL-divergence between a fixed diagonal Gaussian prior and the posterior of the distribution of latent variables. In this case, common latent states fit the prior while novel latents diverge from the prior.

Intra-episode novelty. K-nearest neighbors intrinsic rewards have also been employed to improve intra-episode novelty [Stanton and Clune, 2018]. It contrasts with standard exploration since the agent looks for novel states in the current episode: typically it can try to reach all states after every resets. This setting is possible when the policy depends on all its previous interactions, which is often the case when an agent evolves in a POMDP, since the agent has to be able to predict its value function even though varies widely during episodes. This way, ECO [Savinov et al., 2018b] and Never give up [Badia et al., 2019] uses an episodic memory and learn to reach states that have not been visited during the current episode.

Conclusion K-nn methods turn out to be simple to experiment, but they strongly rely on learnt dynamic-aware representations, their theoretical connection to the rigorous approximation of entropy remains most of the time unclear and the approach badly scales with an increase of the memory size. We note that simple methods can tackle the issue of finding the neighbors by partionning together close states [Yarats et al., 2021]. We observe efficient exploration and the methods easily translate to intra-episode exploration.

3.4.3 Conclusion

In this section, we reviewed works that maximize novelty to improve exploration with flat policies. We formalized novelty as actively discovering a representation according to the infomax principle despite that most of works maximize the entropy of states. But works manages to learn a representation that match the inherent structure of the environment [Tao et al., 2020]. It suggests that it is most of the time enough to learn a good representation. For instance, Guo et al. [2021] and Tao et al. [2020] compute a reward based on a learnt representation, but perhaps a bad representation tends to be located in low-density areas. It would result that active representation entropy maximization correlates with state-conditional entropy minimization.

We are not aware of a lot of methods that actively learn a representation maximizing $I(R;S)$. Yet, we stress out two methods that strive to actively learn a representation of states. In CRL [Du et al., 2021] and NOR [Nachum et al., 2019a], the agent plays a minimax game. A module learns a representation function with a constrastive loss and the agent actively challenges the representation by looking for states with a large loss.

3.5 Skill learning

In our everyday life, nobody has to think about having to move his arms' muscles to grasp an object. A command to take the object is just issued. This can be done because an acquired skill can be effortlessly reused.

Skill abstraction denotes the ability of an agent to learn a representation of diverse skills. We formalize skill abstraction as maximizing the mutual information between the goal $g \in G$ and some of the rest of the contextual states $f(\tau) \in f(\mathcal{T})$, denoted as $I(G; f(\mathcal{T}))$ where $\tau \in \mathcal{T}$ is a trajectory and f a function that extracts a subpart of the trajectory (last state for example). The definition of f depends on the wanted semantic meaning of a skill. Let s_0 refers to the state at which the skill started and s a random state from the trajectory, we highlight two settings based on the literature:

- $f(\mathcal{T}) = S$, the agent learns skills that target a particular state of the environment [Eysenbach et al., 2018].

- $f(\mathcal{T}) = \mathcal{T}$, the agent learns skills that follow a particular trajectory. This way, two different skills can end in the same state if they cross different areas [Co-Reyes et al., 2018].

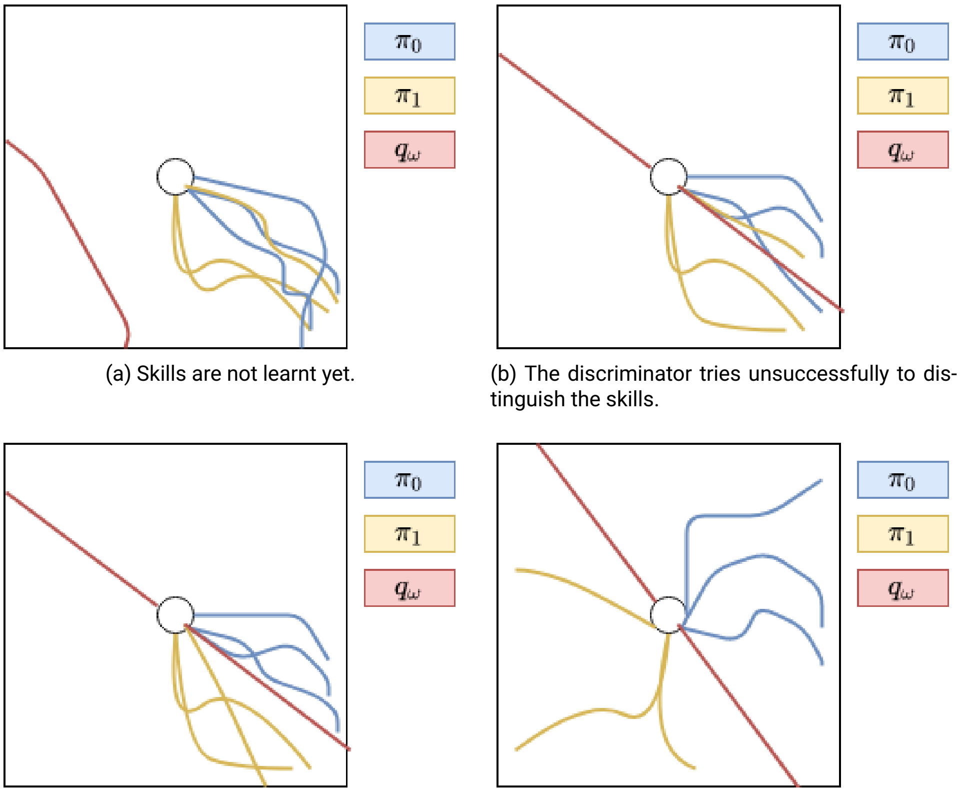
Most of works maximize $I(G; S)$ so that, unless stated otherwise, we refer to this objective. In the following, we will study the different ways to maximize $I(G; S)$ which can be written under its reversed form $I(S; G) = H(G) - H(G|S)$ or forward form $I(G; S) = H(S) - H(S|G)$ [Campos et al., 2020]. In particular, we emphasize that:

$$-H(G|S) = \sum_{g \in G, s \in S} p(g, s) \log p(g|s) \tag{3.20}$$

$$= \mathop{\mathbb{E}}_{\substack{g \sim p(g) \\ s \sim \pi^g}} \log p(g|s) \tag{3.21}$$

where, to simplify, $p(g)$ is the current distribution of goals (approximated with a buffer) and $s \sim \pi^g$ denotes the distribution of states that results from the policy that achieves g. Note that $p(g, s) = p(s|g)p(g)$.

In this section, we first focus on methods that assume they can learn all skills induced by a given goal space/goal distribution and they assign parts of trajectories to every goal. The second set of methods directly derives the goal space from visited states, so that there are two different challenges that we treat separately: the agent has to learn to reach a selected goal and it must maximize the diversity of goals it learns to reach. We make this choice of decomposition because some contributions focus on only one part of the objective function.



(a) Skills are not learnt yet.

(b) The discriminator tries unsuccessfully to distinguish the skills.

(c) Each skill learns to go in the area assigned to it by the discriminator.

(d) Skills locally spread out by maximizing action entropy, as standardly done in DRL [Haarnoja et al., 2018].

Figure 3.4: The agent (circle) starts an episode in the center of the environment, colors denote the trajectories of their corresponding skills.

3.5.1 Fixing the goal distribution

The first approach assumes the goal space is arbitrarily provided except for the semantic meaning of a goal. In this setting, the agent samples goals uniformly from G, ensuring that $H(G)$ is maximal, and it progressively assigns all possible goals to a part of the state space. To do this assignment, the agent maximizes the reward provided by Equation 3.21:

$$R(g, s, a, s') = -\log q_\omega(g|s') \tag{3.22}$$

where $q_\omega(g|s')$ represents a learnt discriminator (often a neural network) that approximates $p(g|s')$.

At first, we focus on discrete number of skills, where $p(g)$ represents a uniform categorical distribution. Figure 3.4 sums up the learning process with two discrete skills: 1- skills and discriminator q are randomly initialized; 2- the discriminator tries to differentiate the skills with states s from its trajectories, in order to approximate $p(g|s)$; 3- skills are rewarded with Equation 3.22 in order to make them go in the area assigned to it by the discriminator; 4- finally, skills are clearly distinguishable and target different parts of the state space. **SNN4HRL** [Florensa et al., 2017] and **DIAYN** [Eysenbach et al., 2018] implement this procedure by approximating g with, respectively, a partition-based normalized count and a neural network. **VALOR** [Achiam et al., 2018] also uses a neural network, but discriminate discrete trajectories. In this setting, the agent executes one skill per episode.

Maximizing $I(G; S|S_0)$ like **VIC** [Gregor et al., 2016] or $I(G; S_0|S)$ with R-VIC [Baumli et al., 2021] make it hard to use a uniform (for instance) $H(G|S_0)$, because every skill may not be executable everywhere in the state space. Therefore, they also maximize the entropy term with another reward bonus similar to $\log p(g|s_0)$. They learn discriminable skills, but still struggle to combine them on complex benchmarks [Baumli et al., 2021]. Keeping $p(g)$ uniform, **DADS** [Sharma et al., 2020] maximizes the forward form of mutual information $I(S; G|S_0) = H(S|S_0) - H(S|G, S_0)$ by approximating $p(s|s_0)$ and $p(s|s_0, g)$. This method makes possible to plan over skills and can combine several locomotion skills. However this requires several conditional probability density estimation on the ground state space, which may badly scale on higher-dimensional environments.

These methods tend to stay close from their starting point [Campos et al., 2020] and do not learn skills that cover the whole state space. In fact, it is easier for the discriminator to overfit over a small area than to make a policy go in a novel area, this results with a lot of policies that target a restricted part of the state space [Choi et al., 2021]. Accessing the whole set of true possible states and deriving the set of goals by encoding states can considerably improve the coverage of skills [Campos et al., 2020].

Approaches for a better coverage of states. Hetereogenous methods address the problem of overfitting of the discriminator. The naive way can be to regularize the learning process of the discriminator. **ELSIM** (Chapter 4) takes advantages of L2 regularization and progressively expand the goal space G to cover larger areas of the state space and Choi et al. [2021] propose to use spectral normalization [Miyato et al., 2018]. More consistent dynamic-aware methods may further improve regularization; however it remains hard to scale the methods to a large number of skills which are necessary to scale to a large environment. In above-mentioned methods, the number of skills greatly increases [Achiam et al., 2018, Aubret et al., 2020] and the discrete skill embedding does not provide information about proximity of skills. Therefore learning a continuous embedding may be more efficient.

Continuous embedding. The prior uniform distribution $p(g)$ is far more difficult to set in a continuous embedding. One can introduce the **continuous DIAYN** with a prior $p(G) = \mathcal{N}(0^d, I)$ where d is the number of dimensions, or the **continuous DADS** with a uniform distribution over $[-1; 1]$ [Sharma et al., 2020], yet it remains unclear how the skills could adapt to complex environments, where the prior does not globally fit the inherent structure of the environment. **VISR** [Hansen et al., 2020] seems to, at least partially, overcome this issue with a long unsupervised training phase and successor features. They uniformly sample goals on the unit-sphere and computes the reward as a dot product between unit-normed goal vectors and successor features $\log q_\omega(g|s) = \phi_{successor}(s)^T g$.

Conclusion. This set of methods manages to learn discrete skills that can be combined, yet, despite regularization, discrete skills struggle to cover a very large state space [Aubret et al., 2020] (cf. Chapter 4). Successfull adaptations to scale it up to large states spaces currently rely on the relevance of successor features. In the next two sections, we study how to maximize the mutual information by assuming the goal space derives from the state space.

3.5.2 Achieving a state-goal

In this section, we review how current methods maximize the goal achievement part of the objective of the agent, $-H(S_g|S)$ where S_g refers to the goal-relative embedding of states. We temporally set aside $H(S_g)$ and we will come back to this in the next subsection, Section 3.5.3, mainly because the two issues are tackled separately in the literature.

Obviously, maximizing $-H(S_g|S)$ can be written:

$$-H(S_g|S) = \sum_{S_g} p(s_g, s) \log p(s_g|s) \tag{3.23}$$

$$= \mathop{\mathbb{E}}_{\substack{s_g \sim p(s) \\ s \sim \pi^g}} \log p(s_g|s) \tag{3.24}$$

where, to simplify, $p(s)$ is the current distribution of states (approximated with a buffer) and $s \sim \pi^g$ denotes the distribution of states that results from the policy that achieves g. If $\log p(s_g|s')$ is modelled as an unparameterized Gaussian with a unit-diagonal co-variance matrix, we have $\log p(s_g|s') \propto -||s_g - s'||_2^2 + Const$ so that we can reward an agent according to:

$$R(s_g, s, a, s') = -||s_g - s'||_2^2. \tag{3.25}$$

Trivially, it means that if the goal is a state, the agent must minimize the distance between its state and the goal state. To achieve this, it can take advantage of a goal-conditioned policy $\pi^{s_g}(s)$.

Ground state space. This way, **Hierarchical Actor-Critic (HAC)** [Levy et al., 2019] directly uses the state space as a goal space to learn three levels of option (the options from the second level are selected to fulfill the chosen option from the third level). A reward is given when the distance between states and goals (the same distance as in Equation 3.25) is below a threshold and they take advantage of HER to avoid to directly use the threshold. Similar reward functions can be found in Pitis et al. [2020] and Zhao et al. [2019]. Related to these works, **HIRO** [Nachum et al., 2018] uses as a goal the difference between the initial state and the state at the end of the option $f(\mathcal{T}) = S_f - S_0$.

This approach is relatively simple and does not require extra neural networks. However, there are two problems in using the state space in the reward function. Firstly, a distance (like L2) makes little sense in a very large space like images composed of pixels. Secondly, it is difficult to make a manager policy learn on a too large action space. Typically, an algorithm having as goals images can imply an action space of $84 \times 84 \times 3$ dimensions for a goal-selection policy (in the case of an image with standard shape). Such a wide space is currently intractable, so these algorithms can only work on low-dimensional state spaces.

Learning a representation of goals. To tackle this issue, an agent can learn low-dimensional embedding of space ϕ_e and maximize the reward of Equation 3.26 using a goal-conditioned policy $\pi^{\phi_e(s_g)}(s)$:

$$R(s_g, s, a, s') = -||\phi_e(s_g) - \phi_e(s')||_2^2. \tag{3.26}$$

Similarly to Equation 3.25, this amounts to maximize $-H(S_g|S)$. **RIG** [Nair et al., 2018] proposes to build the feature space independently with a variational auto-encoder (VAE); but this approach can be very sensitive to distractors (i.e. useless features for the task or goal, inside states) and does not allow to correctly weight features. Similar approaches also encode part of trajectories [Kim et al., 2021, Co-Reyes et al., 2018] for similar mutual information objectives. **SFA-GWR-HRL** [Zhou et al., 2019] uses unsupervised methods like the algorithms of *slow features analysis* [Wiskott and Sejnowski, 2002] and *growing when required* [Marsland et al., 2002b] to build a topological map. A hierarchical agent then uses nodes of the map, representing positions in the world, as a goal space. However the authors do not compare their contribution to previous approaches.

Other approaches learn a state embedding that captures the proximity of states with contrastives losses. For instance, **DISCERN** learns the representation function by maximizing the mutual information between the last state representation and the state-goal representation. Similarly to works in Section 3.5.1, the fluctuations around the objective allow to bring states around s_g closer to it in the representation. More explicitly, the representation of NOR [Nachum et al., 2019a] maximizes $I(\phi_e(S_{t+k}); \phi_e(S_t), A_{t:t+k})$ and the one of LESSON [Li et al., 2021b] and DisTop (cf. Chapter 5) approximately maximizes $I(\phi_e(S_{t+1}); \phi_e(S_t))$; LESSON and NOR target a change in the representation and manage to navigate in a high-dimensional maze while learning the intrinsic euclidian structure of the mazes. Their skills can be reused on several environments. However, experiments are made in 2-dimensional embedding spaces and it remains unclear how relevant may be goals as state changes in an embedding space with higher dimensions. The more the number of dimensions increase, the more difficult it will be to distinguish possible skills from impossible skills in a state. DisTop targets goal-state, it has more difficulties to navigate in large environments, but also work in non-maze environments. We discuss again this issue in the next section.

Conclusion. To sum up, representation learning methods allows to learn state-based skills over complex state spaces. Learning this representation function combined with the use of the euclidian distance as reward function amounts to learn a particular form of reward function in addition for providing pre-computed features to the goal-conditioned policy. In the next paragraph, we study how to maximize $H(S)$ so that to make sure learnt skills are diverse.

3.5.3 Proposing diverse state-goals

To make sure the agent maximizes the mutual information between its goals and all visited states, it must sample a diverse set of goal-states. In other words, it has to maximize $H(S_g)$ but through goal selection rather than with an intrinsic bonus as in Section 3.4. Similarly to works on novelty (cf. Section 3.4), such entropy maximization along with skill acquisition (cf. Section 3.5.2) tackles the exploration challenge, but without facing catastrophic forgetting (cf. Section 3.6.1) since the agent does not forget its skills.

A naive approach would be to generate random values in the goal space, but this faces a considerable problem: the set of achievable goals is often a very small subset of the entire goal space. To tackle this, a first approach can be to explicitly learn to differentiate these two sets of goals [Florensa et al., 2018, Racaniere et al., 2019], using for example a Generative Adversarial Networks (GAN) [Florensa et al., 2018, Goodfellow et al., 2014], but it is ineffective in complex environments

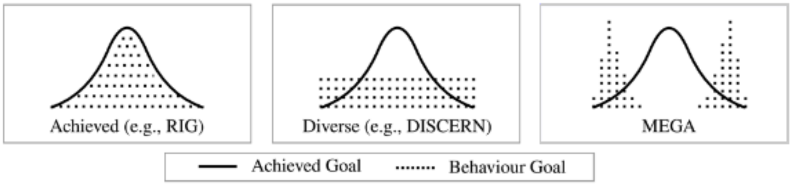


Figure 3.5: Examples of state-goals selection strategies, extracted and adapted from Pitis et al. [2020]. In the RIG strategy, the agent samples goals according to its current distribution of states; the DISCERN strategy tries to samples uniformly and the MEGA strategy prioritizes very low density states.

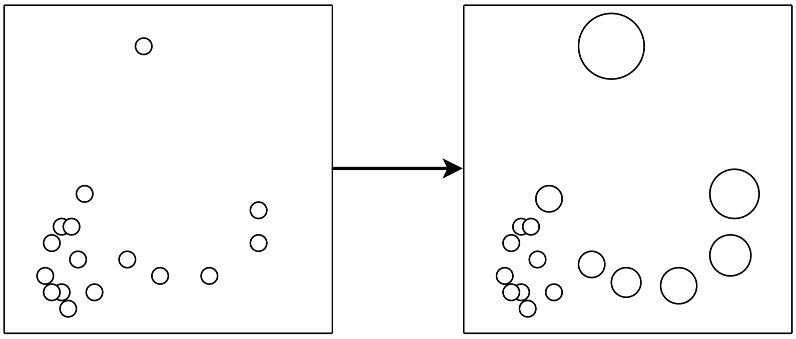


Figure 3.6: Illustration of the reweighting process. **Left**: probability of visited states to be selected as goals before density-reweighting. **Right**: probability of visited states to be selected as goals after density-reweighting. This figure simplifies the figure of Pong et al. [2020].

[Pong et al., 2020]. Other works obtain good results on imagining new goals, but using either a particularly structured goal space [Colas et al., 2020b] or dataset [Khazatsky et al., 2021]. In contrast, an agent can simply set a previously met state as a goal, this way, it ensures that goals are reachable, since they have already been achieved. In the rest of this section, we focus on this set of methods.

Figure 3.5 illustrates different strategies for sampling goals using previously met states. In **RIG** [Nair et al., 2018], the agent randomly samples states as goals from its buffer, but it does not increase the diversity of states, and thus, the diversity of learnt skills. Pong et al. [2020] showed theoretically and empirically that, by sampling goals following a α-more uniform distribution over the support of visited states than the "achieved" distribution, the distribution of states of the agent can converge to the uniform distribution. Intuitively, the agent just samples more often low-density goals: this setting typically applies in the two right-most distributions of Figure 3.5 and we illustrate it in Figure 3.6. There are several ways to increase the importance of low-density goal-states that we introduce in the following.

Density estimation in the ground state space. DISCERN [Warde-Farley et al., 2019] proposes to sample uniformly over the support of visited stated with a simple procedure. Every time the agent wants to add an observation to its buffer, it randomly samples an other observation from its buffer and only keeps the one that is the farthest to all other states of the buffer. This way, it progressively builds an uniform distribution of states inside its buffer. However, it uses the euclidian distance to compare images, which may not be relevant. Other approaches select the state that has the lower density (**OMEGA**) [Pitis et al., 2020] according to a kernel density estimation or use the rank of state-densities [Zhao and Tresp, 2019] estimated with a Variational Gaussian Mixture Model [Blei and Jordan, 2006]. In contrast with them, **Skew-fit** [Pong et al., 2020] provides more flexibility on how uniform one want its distribution of states. **Skew-fit** extends RIG and learns a parameterized generative model $q_\phi(S) \approx p(S)$ and skews the generative model (VAE) with the ratio:

$$q_\phi(s)^{\alpha_{skew}}. \tag{3.27}$$

where $\alpha_{skew} < 0$ determines the speed of uniformisation. This way it gives more importance to low-density states. Then it weights all visited states according to the density approximated by the generative model at the beginning of each epoch, which is made of a predefined number of timesteps. Skew-fit manages to explore image-based environments very efficiently. As shown in [Aubret et al., 2021] (cf. Chapter 5), this ratio applied on a discrete number of skills, amount to rewards a Boltzmann goal-selection policy with:

$$R(s_g) = (1 + \alpha_{skew}) \log p(s_g). \tag{3.28}$$

Density reweighting by partitioning the embedding space. With a different objective, **GRIM-GREP** [Kovač et al., 2020] partitions the VAE embedding of Skew-fit with a Gaussian Mixture Model [Rasmussen et al., 1999] to estimate the learning progress of each partition and avoid distractors. The density weighting can also operate in a learnt embedding. **HESS** [Li et al., 2021a] partitions the embedding space of LESSON and rewards with a variant of a count-based bonus (see Section 3.3). It improves exploration in a two-dimensional latent embedding but the size of partitions may not scale well if the agent considers more latent dimensions. In contrast, **DisTop** [Aubret et al., 2021] (cf. Chapter 5) dynamically clusters a dynamic-aware embedding space using a variant of a Growing When Required [Marsland et al., 2002b]; they estimate the density of state according to how much its partition contains states and skew the distribution of sampled similarly to Skew-fit. HESS and DisTop demonstrate their ability to explore and navigate with an ant inside complex mazes without extrinsic rewards.

Conclusion. Entropy maximization methods improves over standard skill learning methods by learning to reach as many states as possible. We expect further works to show the ability to scale to even more complex environments, with higher-dimensional latent structure [Li et al., 2021a].

3.6 Outlooks of the domain

In this section, we take a step back and thoroughly analyze the results of our overall review. We first study the exploration process of flat intrinsic motivation in comparison with hierarchical intrinsic motivations in Section 3.6.1; then, this will motivate our focus on the challenges induced by learning a deep hierarchy of skills in Section 3.6.2. Finally, in Section 3.6.3, we discuss how flat and hierarchical intrinsic motivation can and should cohabit in such hierarchy.

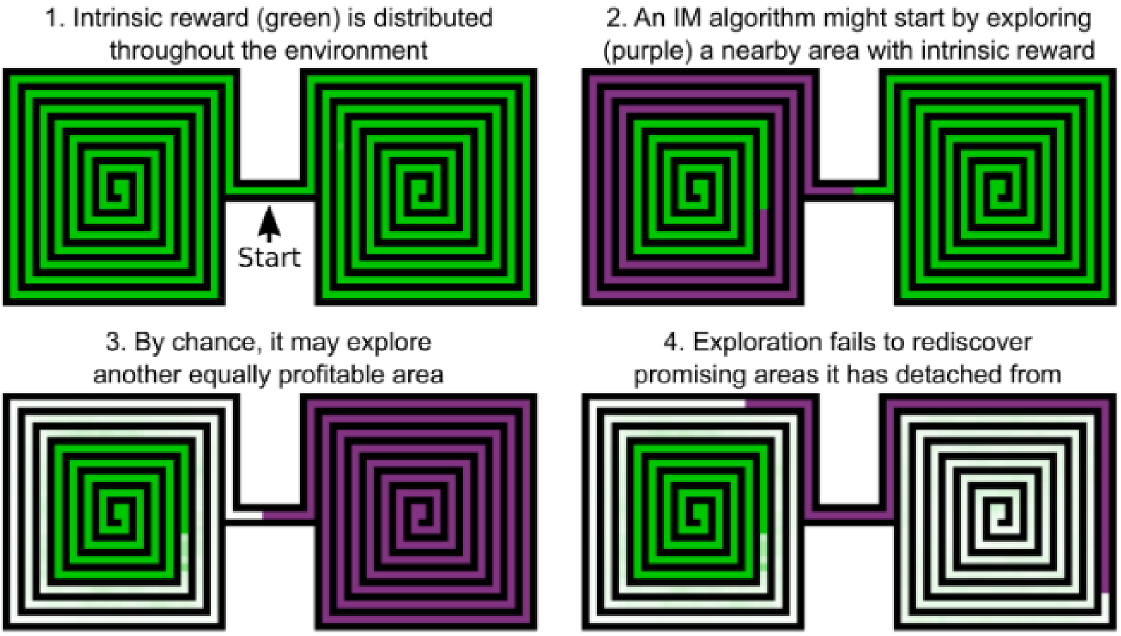

1. Intrinsic reward (green) is distributed throughout the environment
Start
2. An IM algorithm might start by exploring (purple) a nearby area with intrinsic reward
3. By chance, it may explore another equally profitable area
4. Exploration fails to rediscover promising areas it has detached from

Figure 3.7: Illustration of the *detachment* issue. Image extracted from Ecoffet et al. [2019]. Green color represents intrinsically rewarding areas, white color represents no-rewards areas and purples areas are currently being explored. (1) The agent starts to learn and has not explored the environment yet. (2) It discovers the rewarding area at the left of its starting position and explores it. (3) It consumed close intrinsic rewards on the left part, thus it prefers gathering the right-part intrinsic rewards. (4) Due to catastrophic forgetting, it forgot how to reach the intrinsically rewarding area on the left.

3.6.1 Long-term exploration, detachment and derailment

The most challenging used benchmarks in flat intrinsic motivations (surprise and novelty) are *DMLab* and *Montezuma's revenge*, yet very sparse reward games such as *Pitfall!* are not currently addressed and should be investigated. In *Pitfall!*, the first reward is reached only after multiple rooms where it requires specific action sequences to go through each room. State of the art on IM methods [Ostrovski et al., 2017] achieve 0 mean reward in this game. At the opposite, imitation RL methods [Aytar et al., 2018, Hester et al., 2018] are insensitive to such a specific reward, and thus, exceed IM methods with a mean reward of 37232 on *Montezuma's revenge* and 54912 on *Pitfall!*. Even though these methods use expert knowledge, this performance gap exhibits their resilience to long-term rewards. Compared with flat intrinsic reward methods, which do not exceed a 10000 score on *Montezuma's revenge* [Burda et al., 2018] and hardly achieve a score on *Pitfall!* [Ostrovski et al., 2017], it shows that flat IMs is still far from solving the overall problem of exploration.

Furthermore, we want to emphasize that the challenge is harder when the intrinsic reward itself is sparse [Burda et al., 2018]. In *Montezuma's revenge*, it is about avoiding to use a key too quickly in order to be able to use it later. In every day life, it can be about avoiding to spend money too quickly. In fact, it looks like there is an exploration issue in the intrinsic reward function. Intrinsic reward can guide the exploration at the condition that the agent finds this intrinsic reward. There may be two reasons causing the intrinsic reward to be sparse:

1. The first comes from partial observability, with which most models are incompatible. Typically, if an agent has to push a button and can only see the effect of this pushing after a long sequence of actions, density models and predictive models may not provide meaningfull intrinsic rewards. There would be a too large distance between the event "push a button" and the intrinsic reward.
2. Figure 3.7 illustrates the second issue, called *detachment* [Ecoffet et al., 2019, 2021]. It results from a distant intrinsic reward coupled with catastrophic forgetting. Simply stated, the RL agent can forget the presence of an intrinsic reward in a distant area: this is hard to maintain the correct Q-value that derives from a distant currently unvisited rewarding area. This is emphasized in on-policy settings.

Pursuing such distant intrinsic reward may be even harder due to the possible *derailment* issue [Ecoffet et al., 2019, 2021]. Essentially, an agent may struggle to execute a long sequence of specific actions needed to reach a distant rewarding area because the local stochasticity incites local dithering all along the sequence. Detachment motivates the need for a hierarchical exploration [Ecoffet et al., 2021] and derailment motivates frontier-based exploration [Bharadhwaj et al., 2020], which consists in deterministically reaching the area to explore before starting exploration.

3.6.2 Deeper hierarchy of skills

According to Brooks [1991], *everything is grounded in primitive sensor motor patterns of activation*. This *everything* refers to the structure of the world and agent affordances. Capturing this knowledge amounts to form concept representations and reusable skills [Weng et al., 2001], use it

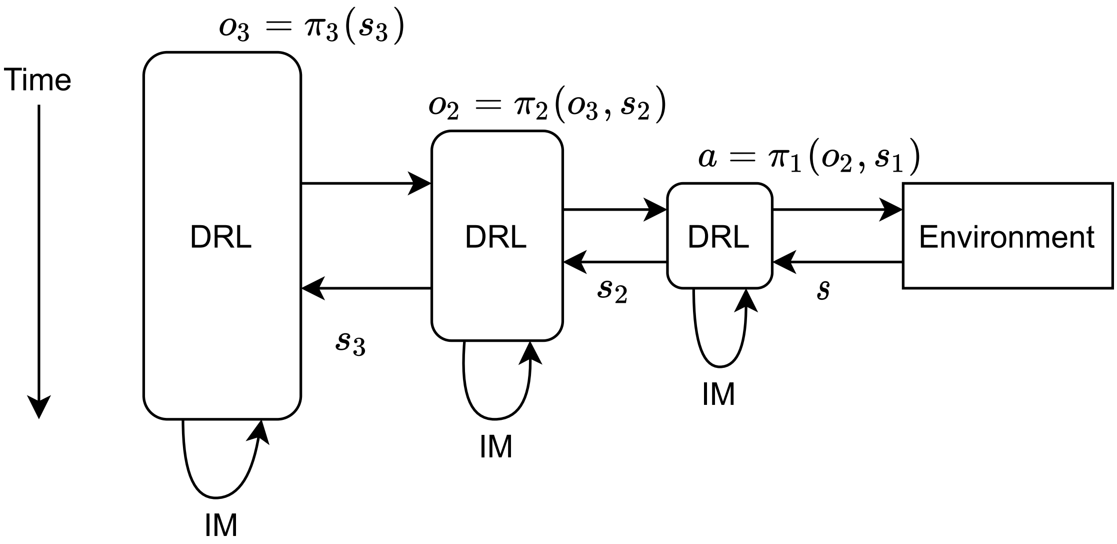


Figure 3.8: Illustration of a three-levels HRL architecture where each DRL algorithm learns with intrinsic motivation. Each incoming state is built based on the original state and the dynamics of the lower-level policies.

as a basis for new skills [Prince et al., 2005], explore the environment to find new interesting skills, autonomously self-generate goals in accordance with the level and morphology of the agent.

Most works presented in Section 3.5 abstract actions on a restricted number of hierarchies (generally one hierarchy). This is necessary to well-understand the mechanism of abstraction, but we want to argue that imposing deeper hierarchies could considerably enhance the semantic comprehension of the environment of an agent. Organisms are often assumed to deal with composition of behaviors, which in turn serve as building block for more complex behaviors [Flash and Hochner, 2005]. Using a limited vocabulary of skills makes easier avoiding the curse of dimensionality associated to the redundancy of a whole set of ground behaviors.

Figure 3.8 displays a simplified example of such architecture. We saw in Section 3.5 that obtaining good skills essentially relies on the quality of the learnt representation which can depend on the potentially temporally-extended dynamics [Nachum et al., 2019a, Aubret et al., 2021, Li et al., 2021b]. The representation can be learnt using the slowness principle [Wiskott and Sejnowski, 2002] which assumes temporally close states are similar. By configuring the time-extension of the representation, one may focus on different semantic parts of the state space.

This can be illustrated in Aubret et al. [2021] (cf. Chapter 5): 1- the agent can learn a very low level representation that provides skills that can manipulate torques of a creature; 2- skills can also orientate an agent in a maze assuming it accesses the maze representation. While they do not try to combine and learn several representations at the same time, further works could consider separate different parts of states (*e.g.* agent positions and object positions [Zhao et al., 2021]) or learning these representations at different time scales.

Skill focus. In a developmental process, multi-level hierarchical RL questions the ability of the agent to learn all policies of the hierarchy simultenaously. This obviously relates to the ability of organisms to continually learn throughout their lifetime; but in more practical way, it may allow to focus the learning process of skills that are interesting for higher-level skills. This focus avoids learning everything in the environment [Aubret et al., 2021], which is hard and obviously not done by biological organisms.

Critical periods and lifelong learning. Considering a goal representation that changes over time introduces new issues for the agent. In this case, the goal-conditioned policy may be perturbed by the changes of inputs and may no longer be able to reach the goal [Li et al., 2021a]. Current methods consider 1- developmental periods (unsupervised pre-training [Metzen and Kirchner, 2013]); 2- to modify the representation every k-steps epochs [Pong et al., 2020]; 3- to impose slowly changes of the representation [Aubret et al., 2021, Li et al., 2021a]. Further works may thoroughly investigate the relation and transitions between these methods since they can relate to the concept of critical periods [Hensch, 2004, Konczak, 2004]. Critical periods assume that the brain is more plastic at some periods of development in order to acquire specific knowledge. Despite this mechanism, the brain slowly keeps learning throughout the lifetime. In the hierarchy of skills, the introduction of a new level may first result in a quick/plastic learning process, followed by slower changes.

3.6.3 The role of flat intrinsic motivations

In Section 3.6.1, we essentially criticized the limited role that flat intrinsic motivation like surprise or novelty can play in favor of exploration and we hypothetized in Section 3.6.2 that deeper hierarchies could make emerge an understanding of more complex affordances. Then, what could be the roles of surprise and novelty ?

Novelty. We saw in Section 3.4 that novelty seeking behaviors allow to learn a correct representation of the whole environment; this can be a basis for learning diverse skills. While some methods consider a goal as a state and manage to avoid using novelty bonuses [Pong et al., 2020], this is harder to do when skills have a different semantic (like a change in the state space). Nachum et al. [2019a] provide a meaningful example of this: the agent acts to simultaneously discover a representation of the environment and achieve upper-level goals.

Surprise. We leave aside the interest of surprise for learning a forward model that could be used for planning [Hafner et al., 2019] and rather focus on the learning process. Surprise amounts to look for the learning progress of forward models so that, in a hierarchy of skills, it quantifies whether skills can currently be better learnt or not. This links surprise to curriculum learning [Bengio et al., 2009], *i.e* can we find a natural order to efficiently learn skills ? For example, assuming an agent want to learn to reach state-goal in a maze, it would be smarter to learn to start learning skills that target goals close to its starting position and to progressively extend its goal selection while learning other skills. Several strategies have been proposed to smartly hierarchically select goals [Colas et al., 2019, Linke et al., 2020], yet it often does not consider intrinsic skills [Colas et al., 2019].

To sum up, we propose that the role of surprise and novelty may rather be to support the learning of skills. Novelty seeking helps to learn the representation required by the skill learning module and surprise speeds up the maximization of the skill learning objective. Considering this, it would result several surprises and novelties: an agent can experiment a novel or surprise interaction for a level of decision (injure the toy while walking), yet it does not mean other levels would be surprised (it is still on the same road). This emphasizes the multi-dimensionality and relativity of the notion of surprisingness ou novelty [Berlyne, 1960], only a part of the incoming stimuli may arouse the agent.

3.7 Unifying intrinsic motivations

In this section, we more thoroughly investigate the links between information theory and intrinsic motivation. We saw in Section 2.2.2 that an intrinsic motivation is defined by how it is computed, *i.e* the comparison on previous data. While this definition encompasses the search for desirable stimulus properties such as surprisingness, novelty or a large number of distinctly described motivations [Berlyne, 1960], an intrinsically motivated entity could also search their exact opposite, such as boredom. Therefore, the comparison of data appears more as a *property* of an IM rather than as its definition. It follows that IM appears to the DRL community like a fuzzy concept that admits a lot of different approaches. Despite this, intrinsic motivation could be a core component in the development of organisms [Guerin, 2011]. As such, a precise formalism may converge to a computational principle guiding the development of agents.

In order to quantify the information processed by an agent, previous works propose to model the sensori-motor loop of an agent interacting with its environment using a Bayesian network [Pearl, 2014]; This modeling allows to quantitatively analyze the perception-action loop [Touchette and Lloyd, 2004, Klyubin et al., 2004a]. This framework opens the path for a quantitatization of intrinsic motivations.

We noticed in Section 3.6 that maximizing the correlations of some variables can/could lead to complex behaviors that reflect the emergence of *increasingly more complex cognitive structures as a result of interactions with the environment* [Guerin, 2011, Zlatev and Balkenius, 2001]. Having argues in Section 3.6.3 for the compatibility of intrinsic motivations, we now propose to go considerably further and to join current forms of intrinsic motivations with the information-theoretic framework of the perception-action loop in order to exhibit the information-theoretic general principle that underpins current IMs. Precisely, we formally show that maximizing the multi-information of a simple hierarchical cognitive model amounts, to some extent, to simultaneously maximizing novelty, surprise and skill learning objectives

Our methodology is two-folds:

- Proposing a plausible cognitive architecture, modelled as a Bayesian Network. It essentially requires variables with causal dependencies.
- Deriving local maximization terms from the multi-information of the model.

Proposing a complete cognitive architecture explaining the full complexity of human behaviors is out of the scope of our study. This rather aims to demonstrate the validity of our objective recipe.

In the following, we first introduce Bayesian networks in Section 3.7.1, then we explictly give the assumptions (Section 3.7.2) that we use in Section 3.7.3 to derive the objective that unifies skill learning, novelty and surprise. We end the section with a discussion about this objective.

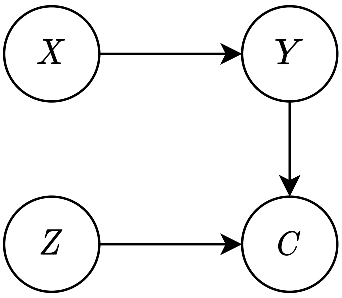


Figure 3.9: Simple example of Bayesian network.

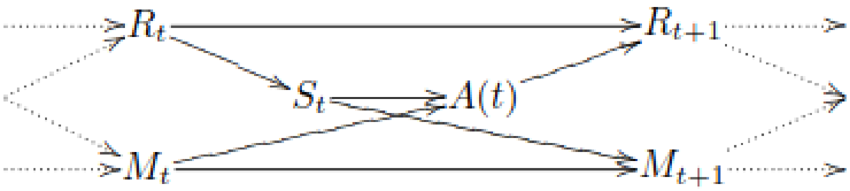


Figure 3.10: Bayesian network which sums up the perception-action loop, extracted from Klyubin et al. [2004a]. S are sensory inputs, R the true state of the environment, A ground actions on the agent, M the memory of the agent. We use a different notation in comparison with the rest of the dissertation to remain faithful to the original figure.

3.7.1 Bayesian networks

Bayesian networks (or graphical models) are directed acyclic graphs for which 1-vertices correspond to random variables which represent part of the state of the system; 2-edges reflect statistical dependencies between these variables, *i.e* a causal relationship. Figure 3.9 illustrates a simple example of graphical model. Given the dependencies, a joint probability $p(X, Y, C, Z)$ is conform to the graphical model if $p(X, Y, C, Z) = p(X)p(Y|X)p(Z)p(C|Z, Y)$. More generally, if $(V_0, \ldots, V_N)$ are the $N+1$ vertices of a graph, and $Pa(V)$ sums up the parents of V with respect to the edges, we can write [Pearl, 2014]:

$$P(V_0, \ldots, V_N) = \prod_{i=0}^{N} P(V_i|Pa(V_i)). \tag{3.29}$$

These models are convenient to model action-perception loops [Touchette and Lloyd, 2004, Klyubin et al., 2004a, Levine, 2018] and allow to compute information theoretic measures. In this setting, parameters, actions, states, decisions etc...are all random variables. Figure 3.10 shows the graphical model induced by a typical perception-action loop which is unrolled through time. This kind of unrollment is typical of Dynamical Bayesian Network [Dagum et al., 1992]. While we could also use a Structured Graphical Model [Pearl, 2010], we assume that the standard graphical model is simpler and makes our results more understandable.

When variables can be described with a graphical model, the multi-information of the whole set of variables can be rewritten as the sum of mutual information between a node and its parents [Friedman et al., 2001, Slonim et al., 2001]:

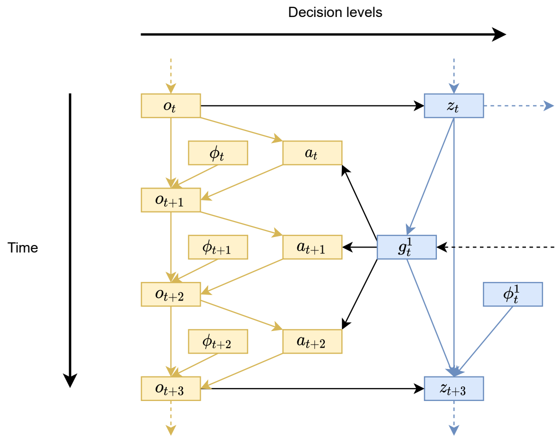


Figure 3.11: Bayesian network which sums up a simplified HRL-based cognitive model of an agent. o are observations, ϕ a distribution of parameters encoding a forward model, a ground actions of the agent, z representations of the state of the agent, g^k the k-level decisions and ϕ^k the k-level forward model. Black arrows represent inter-hierarchies interactions, yellow and blue components represent respectively the first decision level and the second decision level. We assume the model is consistent through time. In practice, all variables are dependent on the content of the memory and the decisions/actions depend on DRL parameters but we omit these here for clarity since it does not bring out more informations for our analysis.

$$MI(V_0, \dots, V_N) = \sum_{i=0}^{N} I(V_i; Pa(V_i)). \tag{3.30}$$

3.7.2 Assumptions about agent's cognitive model

A plausible candidate for a simplified cognitive architecture can be inspired from hierarchical reinforcement learning for two reasons. First there are bio-inspired correlations between hierarchical behaviors in humans [Botvinick, 2008, Mussa-Ivaldi and Bizzi, 2000] and the *option framework* [Botvinick et al., 2009]. Indeed, human behaviors are building blocks that can be hierarchically or serially combined to perform more complex building blocks, which can be combined again [Mussa-Ivaldi and Solla, 2004, Mussa-Ivaldi and Bizzi, 2000]. Secondly it is widely used in computational models since it can bring out complex behaviors, as argued in Section 3.6.2. We also introduce the concept of representations that have been discovered in the brain [Quiroga et al., 2005], *e.g* some neurons respond to particular objects.

We model the cognitive model of an agent with a Bayesian network [Pearl, 2014], as in Friston et al. [2017]. However, in contrast with them, we do not assume an agent is a generative model and only follows the internal causality of its processing. Thereby we do not explicitly model the hidden variables of the environment and rather consider an implicit and abstract understanding of the dynamics at several levels of representations. This directly follows the Bayesian brain hypothesis [Knill and Pouget, 2004] which assumes that the brain models uncertainty and performs Bayesian inference over variables. However, Figure 3.11 shows the Bayesian Network B of a decision-making step at the second level of a HRL framework, assuming the high-level decisions last for three timesteps. Of course, the framework can be made generic over a decision's level-dependent duration l^k. In our case, the probability distributions are defined over a memory of the agent, which is filled with new interactions. We made some common-sense assumptions: first, we assume the agent learns a forward model ϕ that integrates the perception-action loop. It does not access the true state and the agent is agnostic to it but rather learns a representation of the environment that allows to produce high-level decisions; the information contained in the representation grows the higher up the hierarchy it is. The decisions impact the time-extended representation, as well as the ground actions of the agent.

One can notice that we can easily add more hierarchies to the model; we can recursively reproduce the interactions between low-level and high-level decisions so that high-level modules become low-level modules for the next level of the hierarchy.

In practice, the true causal model that usually results from the interactions of the agent may be essentially similar, except for the links between z_t and z_{t+3} and the absence of the hidden causal variables of the environment.

Maximizing the multi-information of such architecture is particularly hard since each combination of time-observation is a random variable. For instance, the probability $p(O_0 = o)$ may be different from $p(O_{25} = o)$. In addition, in a lifelong learning scenario without episodes, the timesteps t may infinitely increase, leaving one value for each variable at each timestep. To overcome this issue, we take inspiration from off-policy RL [Mnih et al., 2015] and the brain [Wilson and McNaughton, 1994] and assume the agent considers a restricted set of time-independent variables while replaying its interactions. This facilitates the memorization of interactions so that the agent only needs to keep local ordering (a variable and its direct parents) rather than the whole time-index. Thus, we write $p(V_t) = p(V|t) = P(V)$ and $p(V_t|Pa(V_t)) = p(V|Pa(V))$ where V can represent any random variables and Pa represent the parents. In other words, we assume that the joint distribution of tuples of locally dependent variables is time-independent. In this case we have, with $t = 0, \dots, N$:

$$
\begin{aligned}
\sum_t I(X_t; X_{t-1}, Y_t) &= \sum_t I(X'; X, Y|t) \\
&= \sum_t H(X'|t) - H(X'|X, Y, t) \\
&= \sum_t - \sum_{X'} p(x', t) \log p(x'|t) + \sum_{X',X,Y} p(x', x, y, t) \log p(x'|x, y, t) \\
&= -\sum_{X'} \sum_t p(x'|t)p(t) \log p(x'|t) + \sum_{X'} \sum_{X,Y} \sum_t p(x', x, y|t)p(t) \log p(x'|x, y, t) \\
&\overset{(1)}{=} -\sum_{X'} \sum_t p(x')p(t) \log p(x') + \sum_{X'} \sum_{X,Y} \sum_t p(x', x, y)p(t) \log p(x'|x, y) \\
&\overset{(2)}{=} -\sum_{X'} \sum_t p(x') \frac{1}{N} \log p(x') + \sum_{X'} \sum_{X,Y} \sum_t p(x', x, y) \frac{1}{N} \log p(x'|x, y) \\
&= \sum_t \frac{1}{N} \Big[- \sum_{X'} p(x') \log p(x') + \sum_{X'} \sum_{X,Y} p(x', x, y) \log p(x'|x, y) \Big] \\
&= I(X'; X, Y) \qquad (3.31)
\end{aligned}
$$

where we applied $p(V|t) = p(V)$ in (1) and noticed that $p(t) = \frac{1}{N}$ in (2). This essentially means that, by storing interactions in memory, the agent considerably simplifies the objective to time-independent variables. In the next section, we use this results to derive the multi-information of the model.

3.7.3 Derivation of core objectives

Now we will use Equation 3.31 to derive the locally structured multi-information of the model B:

$$
\begin{align}
MI^B &\overset{(1)}{=} \sum_t I(O_{t+1}; O_t, A_t, \phi) + I(A_t; G^1_t, O_t) + I(Z_t; \phi^1, G^1_{t-3}, Z_{t-3}, O_t) + I(G^1_t; G^2_t, Z_t) \notag \\
&\overset{(2)}{=} I(O'; O, A, \phi) + I(A; G^1, O) + I(Z'; \phi^1, G^1, Z, O') + I(G^1; G^2, Z) \tag{3.32} \\
&\overset{(3)}{=} \underbrace{I(O'; \phi|A, O)}_{\text{Low-level surprise}} + \underbrace{I(O'; A, O)}_{\text{Controllability}} + \underbrace{I(A; G^1, O)}_{\text{Low-level policy}} + \underbrace{I(Z'; Z, O')}_{\text{Novelty}} \notag \\
&+ \underbrace{I(Z'; \phi^1|G^1, Z, O')}_{\text{High-level surprise}} + \underbrace{I(Z'; G^1|Z, O')}_{\text{Skill learning}} + \underbrace{I(G^1; G^2, Z)}_{\text{High-level policy}} \tag{3.33}
\end{align}
$$

Since our model is a directed acyclic graph, in (1) we can rewrite the multi-information as the sum of mutual informations between the nodes and their parents using Equation 3.30. In (2), we simply apply Equation 3.31 to each term. Finally, in (3), we unroll the first and third terms with the chain rule of mutual information. Let us now analyze each term of Equation 3.33:

Surprise: two terms relate to surprise, one per level of the hierarchy. It directly refers to our formalism of surprise in Section 3.3 where the agent acts to reduce its uncertainty over its forward or density model. This allows to explore the environment through different levels of decisions.

Controllability: controllability has been discussed in Section 3.3 and relates to the ability of the agent to control through actions the observations it gathers [Touchette and Lloyd, 2004]. This way, the agent can act to avoid stochastic areas. This may also be maximized through evolution of sensors and actuators [Klyubin et al., 2005]. In upper-levels of the hierarchy, this term can be retrieve through the skill learning term, however, in this setting, the policy underpinning a skill defines the semantic meaning of a skill, so that no skills lead to intrinsically stochastic areas [Eysenbach et al., 2018].

Policies: these terms essentially means the agent must learn policies dependent on the current representations and high-level decisions while each action should have low marginal probabilities. This is most of the time respected since current exploration policies often progressively converge to an almost deterministic policy when they become able to solve a task. This is the case, for instance of decaying ϵ-greedy, where ϵ progressively converges to almost 0, or Boltzmann exploration. Along with exploration through surprise and novelty, it allows for directed exploration.

Novelty: we discussed about novelty seeking behaviors in Section 3.4, it essentially allows to learn a representation of the environment. This time, we propose to maximize $I(Z';Z,O')$ rather than $I(Z';O')$. We argued in Section 3.4 that novelty aims at dynamically discovering a representation of the environment, but what kind of representations ? Equation 3.33 suggests that this representation should capture both low-level features and temporally-extended features. This is in line with our analysis in Section 3.6.2 which emphasizes the role of the slow moving principle for learning representations. One can see this as transforming the temporal information into abstract representations, without the need for explicit memory-based networks [Greff et al., 2016]. Conceptually, this shares ideas with clockwork recurrent neural networks [Koutnik et al., 2014] which has been applied on classification of sequences of data. Since Z' must keep informations about all O' and add information relatively to previous Z, the representation size should grow along with the hierarchy in order to capture more information about observations. If the agent transmits ground representations towards high levels of the hierarchy (maximizing the objective), this can be deducted with $Z' = (Z'', O')$ and:

$$\begin{aligned} I(Z';Z,O') &= I(Z'',O';Z,O') \\ &= I(Z'';Z,O'|O') + I(O';Z,O') \\ &= I(Z'';Z|O') + H(O'). \end{aligned} \tag{3.34}$$

Considering a large hierarchy of decisions and representations, we generalize this to an arbitrary level:

$$I(Z'^k;Z^k,Z'^{k-1}) = I(Z''^k;Z^k|Z'^{k-1}) + H(Z'^{k-1}). \tag{3.35}$$

In practice, we just need to make possible $H(O'^{k-1}) < H(Z'^k)$ which may be a consequence of $|Z'^k| > |O'^{k-1}|$. Now let us further investigate why different temporal resolution should encode different representations. We can write:

$$I(Z'^k;Z^k,O'^{k-1}) = H(Z^k) + H(O'^{k-1}) - I(O'^{k-1};Z^k) - H(Z^k,O'^{k-1}|Z'^k). \tag{3.36}$$

Equation 3.36 tells us that, under size constraints, the agent has to minimize $I(O'^{k-1};Z^k)$, thereby encoding different informations about the observations with Z^k and O'^{k-1}. The causal model directly implies that O'^{k-1} should focus over short-term information and Z^k over longer-term information since Z^{k-1} does not access local information.

Skill learning: Finally, we observe the last component we are looking for when maximized through the low-level policy: a skill learning objective that we described in Section 3.5. This suggests an agent should not look for ground states or trajectories [Eysenbach et al., 2018], but rather a change in the input space [Gregor et al., 2016]. In our case, this appears that the skill learning should focus on changes in the temporally-extended part of the representation.

As a result, the multi-information of a hierarchical model can indeed explain surprise, novelty and skill learning. As emphasized by Equation 3.33, the roles of $I(Z';\phi^1,G^1,Z,O')$ and $I(O';O,A,\phi)$ are essential for such derivation; the two others only incite the policies to act according to their goal and actual observations/representations.

3.7.4 Discussion

We hope future works will investigate whether multi-information would provide explanations for other phenomenas, like visual tracking [Eckmann et al., 2020], vergence control [Zhang et al., 2014], disentanglement [Bengio et al., 2009], language [Vygotsky, 1980] .... Multi-information principle arises in a bottom-up way by studying works on intrinsic motivation but the building of a Bayesian cognitive model should be intertwinned with biological evidences about the structure of the cortex, in particular the interactions between cortical areas [Mumford, 1992, Felleman and Van Essen, 1991]. In fact, we expect that an investigation about correlated neurobiology aspects to provide more evidences of the principle.

Widening the explanation strength of multi-information could make it a plausible candidate to be a guiding principle for action and perception, thereby competing with the free-energy principle [Friston, 2010]. Yet, we highlight that the multi-information shares some properties with the free-energy principle which itself has been justified with neurobiological studies [Friston, 2005]. This includes minimizing the conditional entropy of representations, or errors of prediction, and the building of a causal model. The most proeminent difference lies in the fact that multi-information maximizes the total amount of information, or marginal entropy of states, representations and decisions. This is a critical add-on since it allows to tackle the dark room issue of the free-energy principle: in their case, an agent staying motionless in the darkness minimizes its prediction error and thus, acts optimally. In the case of multi-information, the dark-room scenario does not maximize the marginal entropy of observations/representations, and thus, does not maximize the objective.

However, we think that the most important advantage of multi-information in comparison with the free-energy comes from our methodology: by extrapolating multi-information principle from machine learning objectives, it deeply connects with *the truth on the ground*, making it useful for deriving new objectives. In contrast, to the best of our knowledge, the free-energy principle composes with only few applications in complex environments [Berseth et al., 2020].

3.8 Conclusion

In this survey, we have presented the current challenges faced by DRL: namely 1- learning with *sparse rewards* through exploration; 2- *building a hierarchy of skills* in order to make easier credit assignment, exploration with *sparse rewards* and *transfer learning*.

We identified several types of IM to tackle these issues, that we classified into three categories based on the maximized information theoretic objective, which are *surprise*, *novelty* and *skill learning*. Surprise and novelty based intrinsic motivations implicitly improve exploration while skill learning allows to create a hierarchy of skills.

Looking for surprise maximizes the mutual information between a model parameters and the next state, knowing the previous state, the action and the history of interactions. We have shown that it can be maximized through three set of works: information gain over predictive models, density models or prediction errors. Novelty seeking can be assimilated to learning a representation of the environment, through the maximization of mutual information between states and their representation. The most important term is the state-entropy maximization. Finally, using skill learning objective that amount to maximize the mutual information between a goal and a part of trajectories of the corresponding skill, an agent can learn hierarchies of temporally-extended skills.

The three objectives are compatible and we have discussed how they could interact to provide a more robust exploration process along with reusable and hierarchical skills, a quick and focused skill acquisition and multi-semantic representations. In addition, we showed that these objectives can by summarized by considering the multi-information of a simple hierarchical cognitive model. We expect that further works will show the value of multi-information as a way to induce a large set of complex behaviors and cognitive abilities.

The core part of the developmental architecture presented in Section 3.6.2 is mostly based on the integration of bottom-up hierarchical skill discovery in HRL. We recall that we identified three advantages of discovering skills in a bottom-up way:

1. The time extended commitment resulting from a hierarchical random walk avoids the usual wanderlust due to the sparsity of rewards.
2. Objective functions for skill abstraction encourages skill diversity, which favors exploration of the state space.
3. Skills can be hierarchically executed and transfered across several different tasks.

In particular, such exploration pathways may be robust against *detachment*, which is a hard issue for flat exploration methods. All together, these elements highlight the potential role of bottom-up skill discovery to tackle RL deadlocks. In the rest of the dissertation, we propose to investigate how bottom-up hierarchical skill discovery could concretely be at the origin of this developmental architecture. Thus we reformulate and refine our problem statement: **Can an agent continually learn increasingly complex hierarchical skills using DRL and intrinsic motivation ?**

As a first step, we put aside the hierarchical combination of skills and focus on their discovery. In fact, we hypothetize that, even though an agent does not take advantage of a straightforward credit assignment or hierarchical random walk, exploration and skills transfer can still benefit from bottom-up skill discovery. In the next chapter, we introduce a novel model that validate this hypothesis.

Chapter 4

End-to-end learning of reusable skills through intrinsic motivation

4.1 Introduction

In the previous chapter, we thoroughly studied what is missing to DRL agents to tackle some of their more important issues (exploration, transfer learning and credit assignment problem). We proposed a developmental architecture and noticed that it could be based on a bottom-up skill discovery. In this chapter, we thoroughly investigate how to discover such skills, we identify the shortcomings of the related approaches and propose a novel model that tackle them.

Several works [Eysenbach et al., 2018, Achiam et al., 2018, Pong et al., 2020] recently proposed to intrinsically learn skills (see definition in Section 2.3.4) by maximizing the mutual information between states $s \in S$ and goals $g \in G$, $I(G, S)$, such that different states are covered by the learned skills. While they discover a great diversity of skills, these works neither learn skills sequentially, nor execute skills sequentially [Eysenbach et al., 2018], but rather always sample goals uniformly.

Learning skills uniformly brings up several issues:

1. Most skills target uninteresting parts of the environment relatively to some tasks; thereby it requires prior knowledge about which features to diversify [Eysenbach et al., 2018, Sharma et al., 2020]. This is illustrated in the first step of Figure 4.1, the agent learns a skill that goes to the top part of the environment, even though there are no extrinsic rewards.

2. Time-extended skills used in a hierarchical setting are often sub-optimal for a task. With diversity heuristic, skills are indeed not expressive enough to efficiently target a goal [Eysenbach et al., 2018, Achiam et al., 2018]. This is showed in the second step of Figure 4.1, even though the agent knows which of its skill performs the best, it may often reach sub-optimal areas.

3. The agent suffers from catastrophic forgetting when it tries to learn a task while learning skills [Eysenbach et al., 2018]. This issue appears in steps two and three of Figure 4.1, while

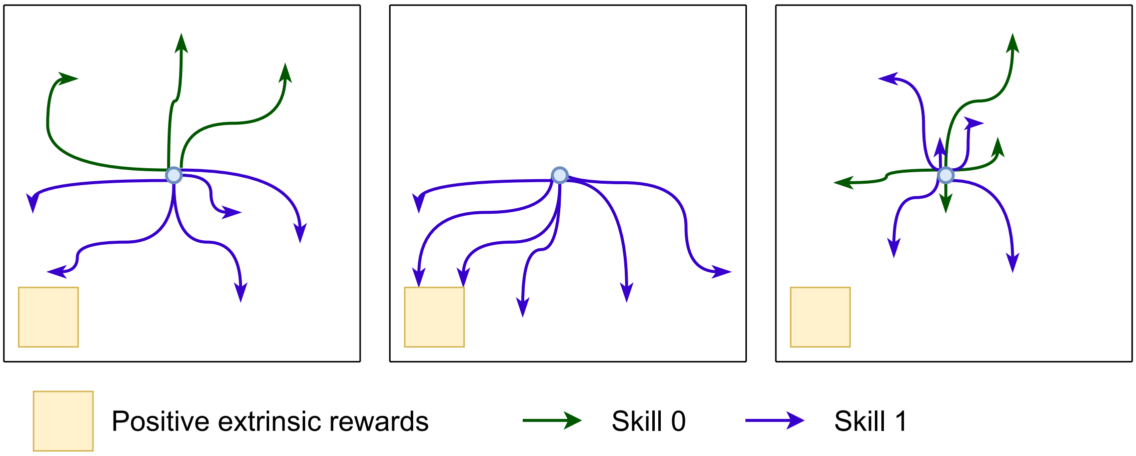


Figure 4.1: Illustration of the problem inherent to previous skill learning methods. The blue circle represents the starting position of the agent and it executes one skill par episode. 1) the agent learns diverse skills, skill 0 goes up and skill 1 goes down; 2) it executes its rewards skill, *i.e* skill 1; 3) It forgets the difference between skills 1 and 0.

executing only skill 1, the agent progressively forgets what differentiates skills 1 and 0, making skills collapse.

Let us suppose there is a hierarchical module that provides goals to the skill learning algorithm. Again, the algorithm will not focus its learning process on upper-levels assignments and will discover skills in uninterested areas of the state space. Thus, the issue is two-folds: 1- it prevents an agent to learn a task; 2- it prevents skill discovery methods to be efficiently integrated in a hierarchical way. In fact, uniform skill discovery occurs during a *developmental period* [Metzen and Kirchner, 2013] which is just an unsupervised pretraining. However, a truly open-ended learning agent does not consider pre-training phases and always expands its repertoire of skill, whether high-level goals or tasks are provided to it. To implement such open-ended learning agent, hierarchical skills should be discovered in a *continual learning* framework (cf. Section 2.2.1).

In the rest of the chapter, we propose to **improve the approaches for continually learning increasingly difficult skills with diversity heuristics**. We introduce ELSIM (End-to-ended Learning of reusable Skills through Intrinsic Motivation), a method for learning representations of skills in a bottom-up way. The agent autonomously builds a tree of abstract skills where each skill is a refinement of its parent. First of all, skills are learned independently from the tasks but along with tasks; it guarantees they can be easily transferred to other tasks and may help the agent to explore its environment. Secondly, the agent selects a skill to refine with extrinsic or intrinsic rewards, and learns new sub-skills; it ensures that the agent learns specific skills useful for tasks through an intelligent *curriculum*, among millions of possible skills.

We believe our paradigm, by removing the requirement of an unsupervised pretraining, makes compatible skill discovery methods with an open-ended architecture. Therefore, we emphasize three properties of our ELSIM method :

Learning is bottom-up : the agent does not require an expert supervision to expand the set of skills. It can use its skills to solve different sequentially presented tasks or to explore its environment.

Learning is end-to-end : the agent never stops training and keeps expanding its tree of skills. It gradually self-improves and avoids catastrophic forgetting.

Learning is focused : the agent only learns skills useful for its high-level extrinsic/intrinsic objectives when provided.

Our contributions are the following: we introduce a new curriculum algorithm based on an adaptation of diversity-based skill learning methods. Our objective is not to be competitive when the agent learns one specific goal, but **to learn useful and reusable skills along with sequentially presented goals in an end-to-end fashion**. We show experimentally that ELSIM achieves **good asymptotic performance** on several single-task benchmarks, **improves exploration** over standard DRL algorithms and manages to easily **reuse its skills**. Thus, this is a step towards *lifelong learning* agents.

This chapter is organized as follows. First, we provide more details on diversity-based intrinsic motivation (Section 4.2.1) and exhibit how ELSIM relates to existing works that learn skills in a *lifelong* learning scenario. In Section 4.3, the core of our method is presented. Then, we explain and visualize how ELSIM works on simple gridworlds and compare its performances with state-of-the-art DRL algorithms on single and sequentially presented tasks learning (Section 4.4). We sum up the method in Section 4.5 just before we take a step back, will evaluate how ELSIM could be integrated in a hierarchical setting and discuss the limitations of ELSIM in Section 4.6. Based on this discussion, we investigate a potential solution in Section 4.7.

4.2 Background

4.2.1 Obtaining diverse skills through mutual information objective

One way to learn, without extrinsic rewards, a set of different skills is to use the objective discussed in Section 3.5.1. We already discussed the intuition of the method in Section 3.5.1, here we provide more technical details.

In DIAYN [Eysenbach et al., 2018], learned skills should be as **diverse** as possible (different skills should visit different states) and **distinguishable** (it should be possible to infer the goal from the states visited by the skill). It follows that the learning process of DIAYN is 4-step with two learning parts: 1- the agent samples one skill from an uniform distribution; 2- the agent executes the skill (randomly initialized); 3- a discriminator q_ω learns to categorize the resulting states to the assigned skill; 4- at the same time, these approximations reward (cf. Equation 4.4).

The global objective can be formalized as maximizing the MI between the set of skills G and states S' visited by skills, defined by [Gregor et al., 2016]:

$$I(G;S') = \mathbb{H}(G) - \mathbb{H}(G|S') \tag{4.1}$$

$$= \mathbb{E}_{\substack{g \sim p(g) \\ s' \sim p(s'|\pi_\theta^g, s)}} [\log p(g|s') - \log p(g)] \tag{4.2}$$

where π_θ^g is the skill associated to the goal $g \in G$ and is parameterized by θ; $p(g)$ is the uniform distribution of skills the agent samples on; and $p(g|s')$ is the probability to infer g knowing the next state s' and skills. This MI quantifies the reduction in the uncertainty of G due to the knowledge of S'. By maximizing it, states visited by a skill π^g have to be informative about the given goal g.

A bound on the MI can be used as an approximation to avoid the difficulty to compute $p(g|s')$ [Barber and Agakov, 2003, Gregor et al., 2016] :

$$I(G, S') \geq \mathbb{E}_{\substack{g\sim p(g)\\ s'\sim p(s'|\pi_\theta^g, s)}} [\log q_\omega(g|s') - \log p(g)] \tag{4.3}$$

where $q_\omega(g|s')$ is the **discriminator** approximating $p(g|s')$. In our case, the discriminator is a neural network parameterized by ω. q_ω minimizes the standard cross-entropy $-\mathbb{E}_{g\sim p(g|s')} \log q_\omega(g|s')$ where $s' \sim \pi_\theta^g$.

To discover skills, it is more efficient to set $p(g)$ to be uniform as it maximizes the entropy of G [Eysenbach et al., 2018]. Using the uniform distribution, $\log p(g)$ is constant and can be removed from Equation 4.3. It follows that one can maximize Equation 4.3 using an intrinsic reward to learn the skill $g \in G$ [Eysenbach et al., 2018]:

$$r^g(s') = \log q_\omega(g|s'). \tag{4.4}$$

Similarly to [Eysenbach et al., 2018], we use an additional entropy term to encourage the diversity of covered states. In practice, this bonus is maximized through the use of DRL algorithms: Soft Actor Critic (SAC) [Haarnoja et al., 2018] for continuous action space and Deep Q network (DQN) with Boltzmann exploration [Mnih et al., 2015] for discrete one.

4.2.2 Related works

Since we already studied works related to skill learning in Section 3.5, we focus on sequential learning methods and other *lifelong* learning aspects tackled in the literature. We also compare our technical contribution to previous approaches.

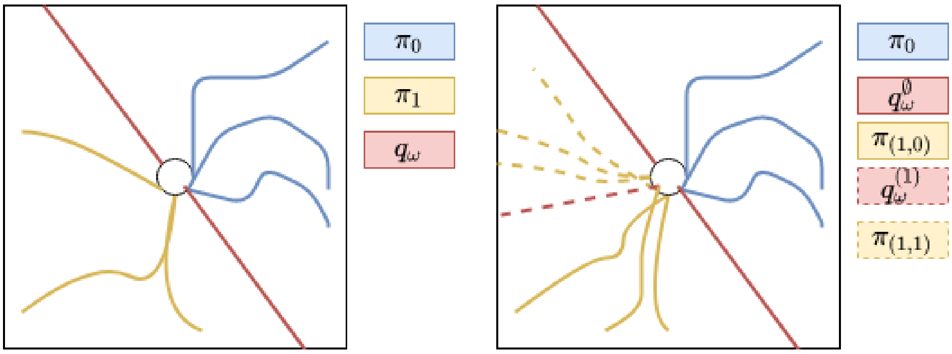
Sequential learning. Contrary to methods that learn skills uniformly (Section 4.2.1), some methods manage to execute skills sequentially, considering skills as *options* and hierarchically running several of them inside one episode. However, either they do not explore efficiently [Nachum et al., 2019a, Levy et al., 2019], or explore with a hierarchical random walk [Nachum et al., 2018, Li et al., 2021b]. While this is more efficient than a low-level random walk [Li et al., 2021b], it is not as efficient as discovering novel states through novel skill discovery. Deep Covering Options (DCO) explores efficiently [Jinnai et al., 2019] but it is unclear whether it can learn a large set of options. Approaches also learn skills directly with the tasks [Bacon et al., 2017, Li et al., 2020a], but skills are biased towards a task and this prevents exploration when rewards are sparse.

Continual learning. Other works proposed a *lifelong learning* architecture. Some assume that skills are already learned and learn to reuse them; for example, Hierarchical Deep Reinforcement Learning Network (H-DRLN) [Tessler et al., 2017] uses a hierarchical policy to choose between ground actions and skills. They also propose to distill previously learned skills into a larger architecture, making their approach scalable. In contrast, we tackle the problem of learning skills in an end-to-end fashion, thereby our approach may be compatible. Similarly to us, Continual Curiosity driven Skill Acquisition (CCSA) [Kompella et al., 2017] addresses the catastrophic forgetting problem by freezing the learning of some experts. They mix two unsupervised learning methods to find and represent goal states, and then learn to reach them. However, their unsupervised algorithm only extracts linear features and they manually define a first set of skills. One particular aspect of continual learning is Meta-RL: how can an agent learn how to learn ? Traditional methods assume there exists a task distributions and try to generalize over it [Finn et al., 2017, Duan et al., 2016]; this task distribution serves as prior knowledge. In [Gupta et al., 2018], the authors address this issue and apply Model-Agnostic Meta-Learning (MAML) [Finn et al., 2017] on an uniform distribution of tasks learned by DIAYN [Eysenbach et al., 2018]. However, learning is neither focused, nor end-to-end. In the continuity of this work, Curricula for Unsupervised Meta-Reinforcement Learning (CARML) [Jabri et al., 2019] mixes the objective of DADS [Sharma et al., 2020] and Meta-RL; it alternates between generating trajectories of the distribution of tasks and fitting the task distribution to new trajectories. While CARML discovers diverse behaviors with pixel-level state space, it cannot learn a global objective end-to-end like ELSIM.

State abstraction. Our method can be viewed as a way to perform state abstraction [Li et al., 2006]. Rather than using this abstraction as inputs to make learning easier, we use it to target specific states. The application of our refinement method bounds the suboptimality of the representation, while the task-independent clustering ensures that skills are transferable. In contrast to our objective, existing methods usually tackle suboptimality for a task without addressing transfer learning or exploration [Akrour et al., 2018, Abel et al., 2016]. The *k*-d tree algorithm [Friedman et al., 1977] has been used to perform state abstraction over a continuous state space [Uther and Veloso, 1998], but as above, the splitting process takes advantage of extrinsic reward and previously defined partitions are not adapted throughout the learning process. In the domain of developmental robotics, Robust Intelligent Adaptive Curiosity (RIAC) and SAGG-RIAC [Baranes and Oudeyer, 2009, 2010] already implement a splitting algorithm building a tree of subregions in order to efficiently explore the environment and learn a forward model. More precisely, they split the state space to maximize either the sum of variance of interactions already collected or the difference of learning progress between subregions. However, these heuristics do not scale to larger continuous environments. In contrast, we assign states to subregions according to the proximity of states and use these subregions as reusable skills to solve several tasks. Adaptive skills adaptive partitions (ASAP) [Mankowitz et al., 2016] partitions the goal space, but does not use intrinsic motivation and the partitions are limited to hyper-plans.

4.3 Method

In this section, we first give an overview of our method and then detail the building of the tree of skills, the learning of the skill policy, the selection of the skill to refine and how ELSIM integrates this in an end-to-end framework.



(a) Two skills are learnt.

(b) The agent separates skill 1 into two new sub-skills (1,0) and (1,1)

Figure 4.2: The agent (circle) starts an episode in the center of the environment, colors denote the trajectories of their corresponding skills.

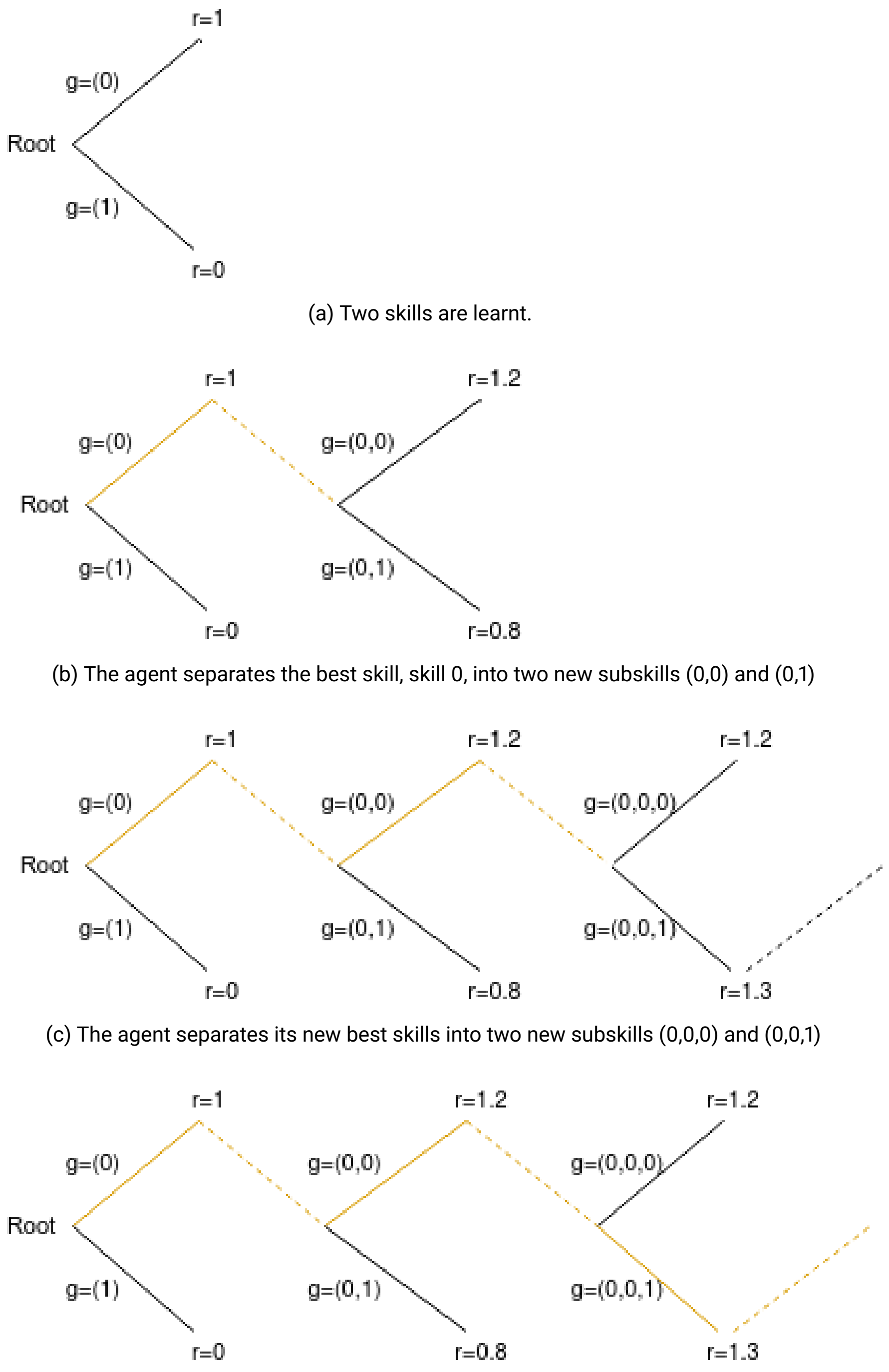
4.3.1 Overview: building a tree of skills

To get both bottom-up skills and interesting skills relatively to some tasks, our agent has to choose the skills to improve thanks to the extrinsic rewards, but we want that our agent improves its skills without extrinsic rewards. The agent starts by learning a discrete set of diverse and distinguishable skills using the method presented in Section 4.2.1. Once the agent clearly distinguishes these skills using the covered skill-conditioned states with its discriminator, it splits them into new sub-skills. For instance, for a creature provided with proprioceptive data, a *moving forward* skill could be separated into *running* and *walking*. The agent only trains on sub-skills for which the parent skill is useful for the global task. Thus it incrementally refines the skills it needs to accomplish its current task. If the agent strives to sprint, it will select the skill that provides the greater speed. The agent repeats the splitting procedure until its skill either reach the maximum number of splits or become too deterministic to be refined.

The first splitting is illustrated in Figure 4.2a and Figure 4.2b, where the agent is a circle that moves in the squared environment. The **hierarchy of skills** is maintained using a tree where each node refers to an abstract skill that has been split and each leaf is a skill being learned.

We formalize the hierarchy using sequence of letters where a letter's value is assigned to each node:

- The set of skills G is the set of leaf nodes. A skill $g \in G$ is represented by a sequence of $k+1$ letters : $g = (l^0, l^1, ..., l^k)$. When g is split, a letter is added to the sequence of its new sub-skills. For instance, the skill $g = (l^0 = 0, l^1 = 1)$ can be split into two sub-skills $(l^0 = 0, l^1 = 1, l^2 = 0)$ and $(l^0 = 0, l^1 = 1, l^2 = 1)$.
- The vocabulary V refers to the values which can be assigned to a letter. For example, to refine a skill into 4 sub-skills, we should define $V = \{0, 1, 2, 3\}$.
- The length $L(g)$ of a skill is the number of letters it contains. Note that the length of a skill is always larger than its parent's.
- $l^{:k}$ is the sequence of letters preceding l^k (excluded).



(a) Two skills are learnt.

(b) The agent separates the best skill, skill 0, into two new subskills (0,0) and (0,1)

(c) The agent separates its new best skills into two new subskills (0,0,0) and (0,0,1)

(d) The agent keeps expanding its tree of skill in the direction of the feedback from the environment.

Figure 4.3: The yellow path represents the goal selected to be executed and splitted and r denotes the average extrinsic rewards gathered by a skill and . Here, $|V| = 2$.

Figure 4.3 illustrate how ELSIM builds a tree. The agent first learns to distinguish two skills, then it recursively split it best extrinsically rewarding skill into new subskills.

We use two kind of policies: the first are the goal-conditioned policies defining a skill. The learning of these skills is described in Section 4.3.2. The second type of policy is task-dependent and responsible to choose which skill to execute; we call it the **tree-policy** (see Section 4.3.3).

4.3.2 Learning skills

In this section, we detail how skills are learned. We adapt the method presented in Section 4.2.1 to our hierarchical skills context. Two processes are simultaneously trained to obtain diverse skills: the skills learn to maximize the intrinsic reward (cf. Equation 4.4), which requires to learn a discriminator $q_\omega(g|s')$. This is difficult because it requires to provides a lot of trajectories from each π^g.

Given our hierarchic skills, we can formulate the probability inferred by the discriminator as a product of the probabilities of achieving each letter of g knowing the sequence of preceding letters, by applying the chain rule:

$$\begin{aligned} r^g(s') &= \log q_\omega(g|s') = \log q_\omega(l^0, l^1, \dots, l^k|s') \\ &= \log \prod_{i=0}^{k} q_\omega(l^i|s', l^{:i}) = \sum_{i=0}^{k} \log q_\omega(l^i|s', l^{:i}) \\ &= \sum_{i=0}^{k-1} \log q_\omega(l^i|s', l^{:i}) + \log q_\omega(l^k|s', l^{:k}). \end{aligned} \tag{4.5}$$

Gathering this value is difficult and requires an efficient discriminator q_ω. As it will be explained in Section 4.3.4, in practice, we use **one different discriminator for each node** of our tree: $\forall i,\ q_\omega(l^i|s', l^{:i}) \equiv q_\omega^{:i}(l^i|s')$ where i indicates the level in the tree. For instance, if $|V| = 2$, one discriminator $q_\omega^{\emptyset}$ will be used to discriminate $(l^0 = 0)$ and $(l^0 = 1)$ but an other one, $q_\omega^{l^0=0}$ will discriminate $(l^0 = 0, l^1 = 0)$ and $(l^0 = 0, l^1 = 1)$.

Training several discriminators at the same time induce several issues, 1- we can not simulteneaously learn all of them; 2- we must avoid to learn correlated discriminators that similarly partition the state space.

Primary learning for the leaves discriminators. It would be difficult for the discriminators to learn over all letters at once; the agent would gather states for several inter-level discriminators at the same time and a discriminator would not know which part of the gathered states it should focus on. This is due to the fact that discriminators and skills simultaneously train. Furthermore, there are millions of possible combinations of letters when the maximum size of sequence is large. We do not want to learn them all.

To address these issues, we introduce **a new curriculum learning algorithm that refines a skill**

only when it is distinguishable. When discriminators successfully learn, they progressively extends the sequence of letters; in fact, we split a skill (add a letter) only when its discriminator has managed to discriminate the values of its letter. Let's define the following probability:

$$p^{:k}_{finish}(l^k) = \mathbb{E}_{s_{final} \sim \pi^{:k+1}} \left[q^{:k}_{\omega}(l^k | s_{final}) \right]. \tag{4.6}$$

where s_{final} is the state reached by the skill at the last timestep. We assume the discriminator $q^{:k}_{\omega}$ has finished its primary learning phase when: $\forall v \in V,\ p^{:k}_{finish}(l^k = v) \geq \delta$ where $\delta \in [0,1]$ is an hyperparameter. Choosing a δ close to 1 ensures that the skill is learned, but a skill always explores, thereby it may never reach an average probability of exactly 1; we found empirically that 0.9 works well.

To approximate Equation 4.6 for each letters' value v, we use an exponential moving average $p^{:k}_{finish}(l^k = v) = (1-\beta) p^{:k}_{finish}(l^k = v) + \beta q^{:k}_{\omega}(l^k = v | s_{final})$ where $s_{final} \sim \pi^{:k+1}$ and $\beta \in [0;1]$. Since we use buffers of interactions (see Section 4.3.4), we entirely refill the buffer before the split.

Learning uncorrelated discriminated areas. Let us reconsider Equation 4.5. The left-hand part represents the reward assigned by the previously learned discriminators. It forces the skill to stay close to the states of its parent skills since this part of the reward is common to all the rewards of its parent skills. In contrast, the right-hand part represents the reward assigned by the discriminator that actively learns a new discrimination of the state space. Since the agent is constrained to stay inside the area of previous discriminators, the new discrimination is **uncorrelated** from previous parent discriminations. In practice, we increase the importance of previous discriminations with a hyper-parameter $\alpha \in \mathbb{R}$:

$$r^g(s') = \log q_{\omega}(l^k | s', l^{:k}) + \alpha \sum_{i=0}^{k-1} \log q_{\omega}(l^i | s', l^{:i}). \tag{4.7}$$

This hyper-parameter is important to prevent the agent to deviate from previously discriminated areas to learn more easily the new discrimination.

4.3.3 Learning which skill to execute and train

For each task, a stochastic policy, called *tree-policy* and noted π_T (with T the tree of skills), is responsible to choose the skill to train by navigating inside the tree at the beginning of a task-episode. This choice is critical in our setting: while expanding its tree of skills, the agent cannot learn to discriminate every leaf skill at the same time since discriminators need states resulting from the skills. We propose to **choose the skill to refine according to its benefit in getting an other reward** (extrinsic or intrinsic), thereby ELSIM executes and learns only interesting skills (relatively to an additional reward).

To learn the *tree-policy*, we propose to model the tree of skills as an MDP solved with a Q-learning and Boltzmann exploration. The action space is the vocabulary V; the state space is the set of nodes, which include abstract and actual skills; the deterministic transition function

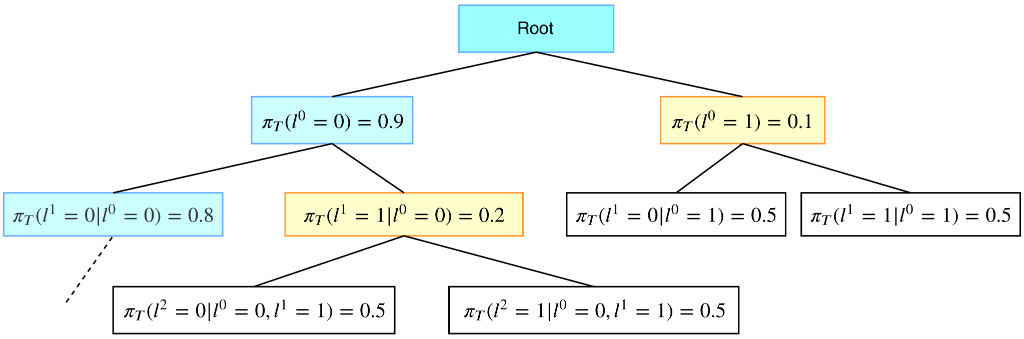

Root
$\pi_T(l^0 = 0) = 0.9$
$\pi_T(l^0 = 1) = 0.1$
$\pi_T(l^1 = 0|l^0 = 0) = 0.8$
$\pi_T(l^1 = 1|l^0 = 0) = 0.2$
$\pi_T(l^1 = 0|l^0 = 1) = 0.5$
$\pi_T(l^1 = 1|l^0 = 1) = 0.5$
$\pi_T(l^2 = 0|l^0 = 0, l^1 = 1) = 0.5$
$\pi_T(l^2 = 1|l^0 = 0, l^1 = 1) = 0.5$

Figure 4.4: Representation of a part of the tree of skills with $|V| = 2$ and the value of *tree-policy* in each node. White nodes are actual leaves of the tree (there are no discriminators since there are no children to discriminate). Yellow nodes represent nodes for which the discriminator can not differentiate its sub-skills; the *tree-policy* samples uniformly. Nodes are blue when the discriminator can distinguish its sub-skills; the *tree-policy* samples using Q-values.

is the next node selection; if the node is not a leaf, the reward function R_T is 0, else this is the discounted reward of the skill executed in the environment divided by the maximal episode length. Each episode starts with the initial state as the root of the tree, the *tree-policy* selects the next nodes using Q-values. Each episode ends when a leaf node has been chosen, i.e. a skill for which all its letters has been selected; the last node is always chosen uniformly (see Section 4.3.4).

Let us roll out an example using the *tree-policy* displayed in Figure 4.4. The episode starts at the root of the tree; the *tree-policy* samples the first letter, for example it selects $l^0 = 0$. Until it reaches a leaf-node, it samples new letters, e.g. $l^1 = 1$ and $l^2 = 0$. The *tree-policy* has reached a leaf, thereby it will execute and learn the skill $(0, 1, 0)$. Then, the state-action tuple $((0, 1), (0))$ is rewarded with the scaled discounted reward of the task. This reward is propagated via the Q-learning update to previous state-action tuples $((\emptyset), (0))$ and $((0), (1))$ to orientate the *tree-policy* to $(0, 1, 0)$.

The MDP evolves during the learning process since new letters are progressively added. The Q-values of new skills are initialized with their parent Q-values. However, Equation 4.5 ensures that adding letters at the leaf of the tree monotonically increases Q-values of their parent nodes. The intuition is that, when splitting a skill, at least one of the child is equal or better than the skill of its parent relatively to the task. We experimentally show this in Section 4.4.2. The resulting curriculum can be summarized as follows: the tree will be small at the beginning, and will grow larger in the direction of feedbacks of the environment.

We now sum up the process of the *tree-policy*: 1-an agent runs an episode inside the MDP of skills; the sequence of actions represents a skill; 2- the agent executes the skill; 3- the *tree-policy* is rewarded according to how well the skill fits the task and the Q-learning applies. When the tree policy gathers a constant (possibly null) reward, it becomes uniform. The full algorithm of the *tree-policy* is given in Appendix A.1.

4.3.4 Simultaneous training of the *tree-policy* and skills

The MI objective requires the skill distribution to remain uniform (cf. Equation 4.4), however that is not our case: the agent strives to avoid some useless skills while focusing on others. In our preliminary experiments, ignoring this leads us to catastrophic forgetting of the learned skills since discriminators forget how to categorize states of the skills they never learn on. To bypass this issue and sample uniformly, we assign to each node i of our tree a replay buffer containing interactions of the skill with the environment, a RL algorithm and a discriminator ($q_{\omega}^{:i}$). At each split, skills and buffers of a node are copied to its children; for the first node, its skills are randomly initialized and its buffer is empty.

This way, the entire training is off-policy: the skill fills the replay buffer while the discriminator and skills learn from the interactions that are uniformly extracted from their buffers. We divide the lifetime of a node into two phases, before and after the split: 1-the **learning phase** during which next letter's values are sampled uniformly; the *tree-policy* is uniform at this node; 2-the **exploitation phase** during which the *tree-policy* chooses letters with its Boltzmann policy (Section 4.3.3).

Then, at each step, the agent runs the *tree-policy* to select the discriminator in the learning phase that will learn. The discriminator samples a mini-batch of data from its children's (all leaves) buffers and learns on it. Then, all children skills learn from the intrinsic feedback of the same interactions, output by the selected discriminator and all its parents according to Equation 4.5.

Once a node enters the exploitation phase, an hyper-parameter η regulates the probability that each parent's discriminator learn on its children data. Their learning interactions are recursively sampled uniformly on their children. This post-exploration learning allows a node to expand its high-reward area. Without this mechanism, different uncovered states of the desired behaviour may be definitively attributed to different fuzzy skills, as shown in Section 4.4.1. The full learning algorithm is given in Appendix A.2.

Figure 4.4 gives an example of a potential tree and how different phases coexist; the skills starting by $(0,1)$ seem to be the most interesting for the task since each letter sampling probability is high. Skills $(0,1,0)$, $(0,1,1)$, $(1,0)$ and $(1,1)$ are being learned, therefore the sampling probability of their last values is uniform.

4.4 Experiments

The first objective of this section is to study the behavior of our ELSIM algorithm on basic gridworlds to make the visualization easier. The second purpose is to show that ELSIM can scale with high-dimensional environments. We also compare its performance with a non-hierarchical algorithm SAC [Haarnoja et al., 2018] in a single task setting. Finally we show the potential of ELSIM for transfer learning.

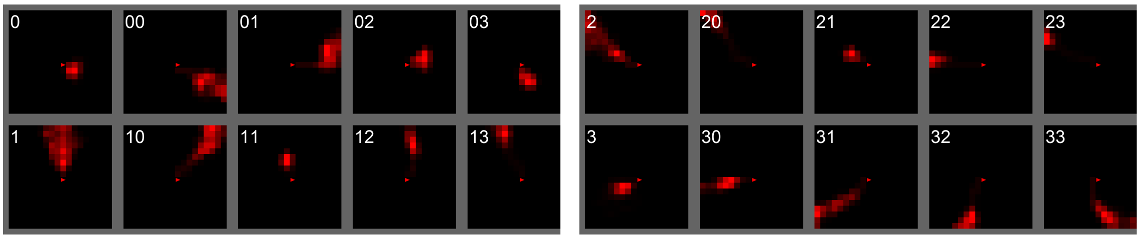


Figure 4.5: Different states covered by the agent while doing a skill. The first and sixth columns display the skills learned with a message length equal to 1; once the learning has been completed, the agent refines each skill into four new sub-skills, displayed on each row.

4.4.1 Study of ELSIM in gridworlds

In this section, we analyze how skills are refined on simple gridworlds adapted from gym-minigrid [Chevalier-Boisvert and Willems, 2018]. Unless otherwise stated, **there is no particular task (or extrinsic reward)**, thereby the *tree-policy* is uniform.

Experimental setup. The observations of the agent are its coordinates; its actions are the movements into the four cardinal directions. Our hyperparameters can be found in Appendix A.3 To maximize the entropy of the skill with a discrete action space, we use the DQN algorithm [Mnih et al., 2015]. The agent starts an episode in the middle of the grid (see figures) and an episode resets every 100 steps, thus the skill lasts 100 steps. At the end of the training phase, the skills of all the nodes are evaluated through an evaluation phase lasting 500 steps for each skill. In all figures, each tile corresponds to a skill that is displayed at the top-left of the tile. We set $|V| = 4$, so that there are four skills at the beginning of the tree and each skill splits into four new subskills. Figure 4.6, Figure 4.5 and 4.8 display the density of the states visited by skills during the evaluation phase: the more red the state, the more the agent goes over it.

Can ELSIM refine a high-stochastic skills into low-stochastic skills ? The first and sixth column of Figure 4.5 shows, for each possible skill g when $L(g) = 1$, the states covered by the agent during the evaluation phase of the skills. We can see that the agent clearly separates its skills since the states covered by one skill are distinct from the states of the other skill. In our example, the first skill (0) makes the agent go at the right of the grid, the second one (1) at the top, the third one (2) at the left and the fourth one at the bottom. In contrast, columns two to five and seven to eleven of Figure 4.5 show the skills learned with $L(g) = 2$. As evidenced by goals' numbers, the four rightmost skills are the refinement of the leftmost fuzzy policy on the same row. We see that the refinement allows to get lower-stochastic policies. For example, skill (1) is very fuzzy while its children target very specific areas of the world. This emphasizes the benefits of using more latent variables to control the environment.

Do the split of skills improve the exploration of an agent ? Figure 4.6 shows some skills learned in an environment of 4 rooms separated by a bottleneck. The full set of skills is displayed in Appendix A.5. We first notice that the agent clearly separates its first skills $(0), (1), (2), (3)$ since the states covered by one skill are distinct from the states of the other skills. However it does not escape from its starting room when it learns these first skills. When it develops the skills close to bottlenecks, it learns to go beyond and invests new rooms. It is clear for skills (1) and (2) which, with one refinement, respectively explore the top-right (skill $(1, 0)$) and bottom-left (skills

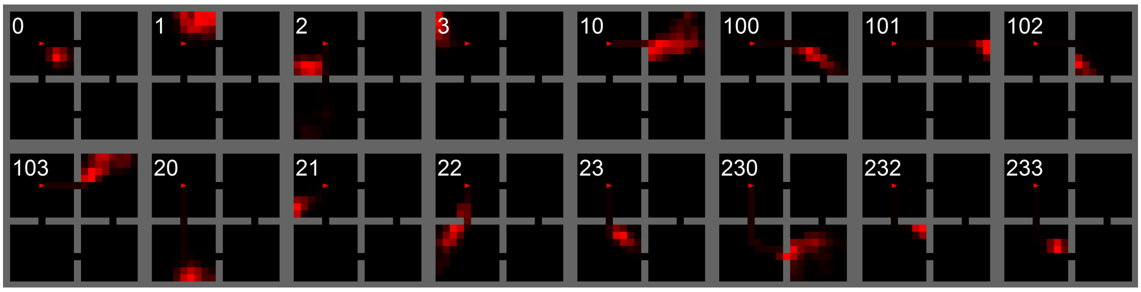


Figure 4.6: Some skills learned by the agent in an environment composed of four rooms.

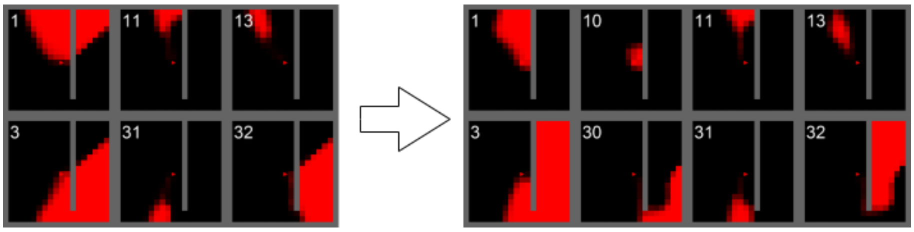


Figure 4.7: Discriminator's probability of achieving skills (1), (3) and their sub-skills in every state (*i.e.* $q(g|s)$). The more red the state, the more rewarding it is for the skill. The left side corresponds to the preliminary stage of the learning process (timestep 128.10^4); the right side corresponds to the end of the learning process (timestep 640.10^4).

$(2,0),(2,2),(2,3)$) rooms. With a second refinement, $(2,3,0)$ even manages to reach the farthest room (bottom-right). This stresses out that the refinement of a skill also allows to expand the states covered by the skill, and thus can improve the exploration of an agent when the rewards are sparse in an environment.

Does the split of skills correct a wrong over-generalization of a parent skill ? Figure 4.7 shows the evolution of the intrinsic reward function for some skills (see Appendix A.6 for the full set of skills). The environment contains a vertical wall and settings are the same as before, except the Boltzmann parameter set to 0.5. At the beginning, skill (1) is rewarding identically left and right sides of the wall. This is due to the generalization over coordinates and to the fact that the agent has not yet visited the right side of the wall. However it is a wrong generalization because left and right sides are not close to each other (considering actions). After training, when the agent begins to reach the right side through the skill $(3,2)$, it corrects this wrong generalization. The reward functions better capture the distance (in actions) between two states: states on the right side of the wall are attributed to skill (3) rather than (1). We can note that other parts of the reward function remain identical.

Can the agent choose which skill to develop as a priority ? In this part, we use the same environment as previously, but states on the right side of the wall give an extrinsic reward of 1. Thus the agent follows the *tree-policy* to maximize its rewards, using Boltzmann exploration, and focus its refinement on rewarding skills. Figure 4.8 shows all the parent skills of the most refined skill which reaches $L(g)=6$. The agent learns more specialized skills in the rewarding area than when no reward is provided (cf. Appendix A.7 for the full set of skills learned).

Summary. We illustrated the following properties of ELSIM: 1- it expands a previously learned rewarding area when it discovers new states; 2- we show in Section 4.4.2 that it improves exploration when the rewards are sparse; 3- adding letters corrects over-generalization of their parent

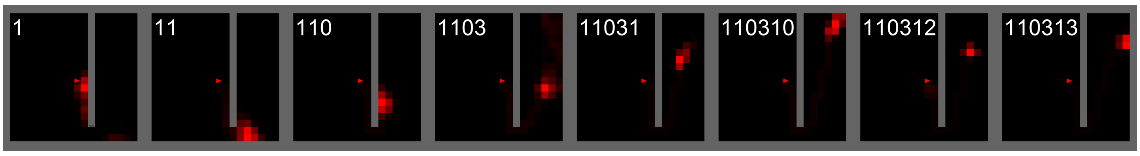


Figure 4.8: One path in the tree of skills learned by the agent with an extrinsic reward of 1 on the upper right side of the wall.

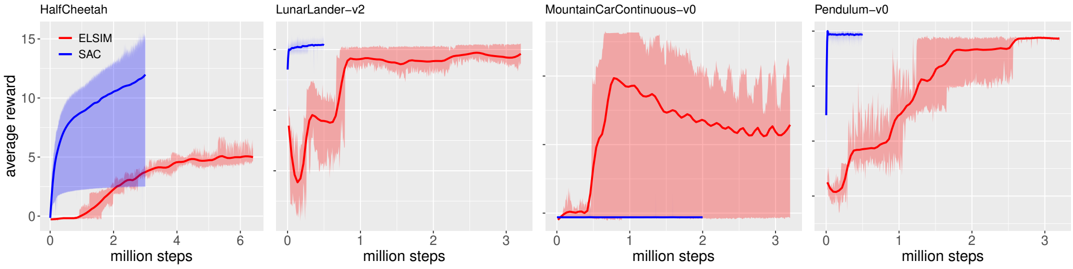


Figure 4.9: Average reward per episode in classical environments (*HalfCheetah-v2* [Todorov et al., 2012], *LunarLanderContinuous-v0* [Shariff and Dick, 2013], *MountainCarContinuous-v0* [Moore, 1990] and *Pendulum-v0*) for SAC and ELSIM (averaged over 4 seeds). We use our own implementation of SAC except for *HalfCheetah* for which the blue curve is the average reward of SAC on 5 seeds, taken from [Haarnoja et al., 2018]. We stopped the simulation after convergence of SAC.

discriminator; 4- it can focus the skill expansion towards task-interesting areas.

4.4.2 Performance on a single task

In this part, we study the ability of ELSIM to be competitive on continuous benchmarks **without any prior knowledge**. Action and state spaces of used environments are summarized in Table 4.1. Unless stated otherwise, used hyper-parameters can be found in Appendix A.4. For ELSIM, we set the maximum skill length to 10, which is reached in *HalfCheetah*.

Figure 4.9 respectively shows the average reward per episode for different environments. Shaded areas color are upper-bounded (resp. lower-bounded) by the maximal (resp. minimal) average reward.

First, the *MountainCarContinuous* environment represents a challenge for the exploration as it is a sparse reward environment: the agent receives the reward only when it reaches the goal. In this environment, ELSIM outperforms SAC by getting a higher average reward. It confirms our results (cf. Section 4.4.1) on **the positive impact of ELSIM on the exploration**. There is a slight decrease after reaching an optima, in fact, ELSIM keeps discovering skills after finding its optimal skill. On

Environment	State space	Action space
HalfCheetah	17	6
MountainCarContinous	2	1
Pendulum	3	1
LunarLander	8	2

Table 4.1: Summary of environments and the size of action and state spaces.

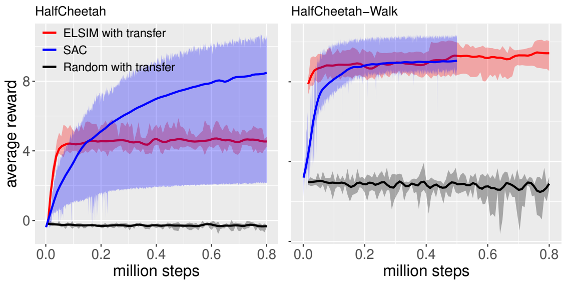


Figure 4.10: Average reward per episode in *HalfCheetah* and *HalfCheetah-Walk*. We use our own implementation of SAC for *HalfCheetah-Walk*. The black curve is the average reward of a random *tree-policy* that uses the transfered skills.

Pendulum and *LunarLander*, ELSIM achieves the same asymptotic average reward than SAC, even though ELSIM may require more timesteps. On *HalfCheetah*, SAC is on average better than ELSIM. However we emphasize that **ELSIM also learns other skills**. For example in *HalfCheetah*, ELSIM learns to walk and flip while SAC, which is a non-hierarchical algorithm, only learns to sprint.

4.4.3 Transfer learning

In this section, we evaluate the interest of ELSIM for transfer learning. We take skills learned by intra-policies in Section 4.4.2, reset the *tree-policy* and restart the learning process on *HalfCheetah* and *HalfCheetah-Walk*. *HalfCheetah-Walk* is a slight modification of *HalfCheetah* which makes the agent target a speed of 2 (cf. Appendix A.8 for more details on the new reward function). skills learning was stopped in *HalfCheetah*.

In contrast with our previous experiment, the performance exclusively depends on the exploration of the tree. Therefore we use MBIE-EB [Strehl and Littman, 2008] to more efficiently explore the tree. This method adds a count-based bonus to the Q-value:

$$\tilde{Q}(s,a) = Q(s,a) + \frac{\beta}{\sqrt{n(s,a)}} \tag{4.8}$$

where $n(s,a)$ is the number of time we chose a in s and β is an hyper-parameter. We set it to 10 for classic HalfCheetah and 2 for walking HalfCheetah. The agent selects the action that has the larger $\tilde{Q}$.

Figure 4.10 shows that the *tree-policy* learns to reuse its previously learned skills on *HalfCheetah* since it almost achieves the same average reward as in Figure 4.9. On *HalfCheetah-Walk*, we clearly see that the agent has already learned skills to walk and that it easily retrieves them. In both environments, ELSIM learns faster than SAC, which learns from scratch. It demonstrates that skills learned by ELSIM can be used for **other tasks** than the one it has originally been trained on.

4.5 Conclusion

We proposed ELSIM, a novel algorithm that continually refines discrete skills using a recently defined diversity heuristic [Eysenbach et al., 2018]. To do so, the agent progressively builds a tree of different skills in the direction of a high-level objective. As shown in Section 4.4, **ELSIM expands the area associated to a skill** thanks to its exploratory behavior which comes from adding latent variables to the overall policy. **ELSIM also focuses its training on interesting skills relatively to some tasks**. Even though the agent is often learning a task, the skills can be defined independently from a specific task and we showed that ELSIM possibly makes them **transferable across different tasks** of a similar environment. Since the agent does not need extrinsic reward to learn, we show that it can **improve exploration** on sparse rewards environments. We trained ELSIM on relatively simple benchmarks, however, it is a strong proof-of-concept that emphasizes the potential ability of bottom-up skill discovery to tackle exploration and transfer learning.

Let us recall that our final objective is to integrate this paradigm of skill discovery in a hierarchical scenario where several upper-level modules set different higher-level goals inside one episode. Therefore, in the next section, we assess the ability of ELSIM to integrate a hierarchical setting. We also discuss its exploration and transfer performance.

4.6 Limitations of ELSIM

ELSIM learns a discrete set of skills and, when it does not access rewards, it explores uniformly over its set of skills while building the tree of skills. In this section, we will explain why these properties hurt the integration of ELSIM in a hierarchical framework and why it sub-optimally bounds both its exploration process and its ability to reuse skills. We will sketch some outlooks, however these observations will incite us to tackle the problem of continual bottom-up skill discovery in a different way in the next chapter.

4.6.1 Sub-optimal exploration

ELSIM is able to keep improving its skills in an end-to-end way, even though it does not access rewards. However it uniformly samples skills to execute, and thus uniformly tries to improve them. Such uniform discovery is, in fact, strongly sub-optimal. To illustrate this, let us reconsider our previous examples. In Figure 4.5, the agent starts in the middle of an empty environment, it can explore all around its starting position and each skill can be improved; in this situation, the exploration process is going well: its uniform sampling matches an uniform distribution of the state space.

It contrasts with our experiment in the four rooms environment (Figure 4.6) from which we display the third skill in Figure 4.11. By uniformly sampling skills at each level of its tree, it more thoroughly focuses on the top-left room. Indeed, 2 of its first-level skills (0 and 3) can not leave the room. Uniform sampling over each level of the tree hierarchy results in at

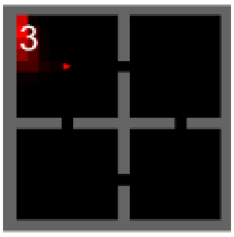


Figure 4.11: Skill 3 of Figure 4.6.

least 50% of skills sampled in the top-left room, despite they can not be improved. There is no mechanism for inciting **the agent to focus on skills that can be improved or novel areas**. This issue is highlighted on much more complicated benchmarks in the next chapter.

This issue could be addressed by taking advantage of learning progress measures [Oudeyer and Kaplan, 2009] or count-based entropy maximization [Watters et al., 2019, Pong et al., 2020]. In the next section, we will focus on the limitations proper to the discretization of the goal space.

4.6.2 Discrete set of skills

Currently, our method allows to avoid the problem of catastrophic forgetting, but the counterpart is an increase of the memory footprint, which is a recurrent issue in methods based on trees. ELSIM creates novel neural networks at each level of the hierarchy, it prevents the transfer of knowledge between the different neural networks. But even though one neural network may approximate several same-level policies, the input goal space is strongly unstructured (one hot encoding vector) so that the goal-conditioned policy of an agent can not generalize on it: the distance between two goal representations does not give information about the proximity of the resulting skills.

This generalization issue also occurs on the top-down vertical representation of the tree. Knowing that a skill performs badly does not give sufficient information about the performance of skills lower in the tree, making samples unefficient. This is stressed out by our transfer learning expriment in Figure 4.10: while ELSIM indeed finds the correct already learnt skill, it took almost 100 000 steps to find out which one is the best, even though it takes advantage of the tree. This problem is even more prominent if one wants to integrate ELSIM in a multi-level hierarchical architecture. Assuming the goal space is the action space of an upper-level agent, its upper-level action space would be a very large discrete action space, which is intractable for a DRL agent: typical approaches that scale to large action spaces take advantage of a continuous embedding of actions [Dulac-Arnold et al., 2015].

To sum up, the discrete set of goals is not adequate, neither as an input for a low-level goal-conditioned policies, nor as an output for a high-level policy. These observations suggest that a continuous embedding would more easily integrate a hierarchical organization and make easier transfer learning. There are two alternatives to build a continuous embedding:

1. We can map the discrete tree representation to a continuous space.
2. The agent should directly learn a continuous embedding.

In the last section of the chapter, we briefly study the relevance of the first option: can we map a discrete tree to a continuous embedding ? By highlighting our preliminary investigation, we mean to provide explanation and a basic overview of this research outlook. We more thoroughly study the second option in the Chapter 5.

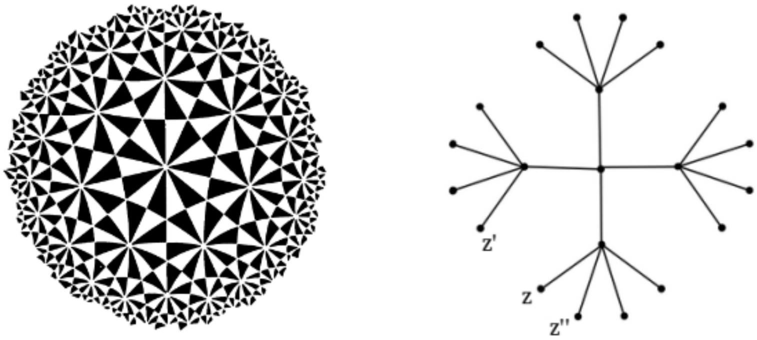


Figure 4.12: **Left:** Illustration extracted from. "Circle Limit I" by M.C. Escher? It illustrates how the Poincare disc models the hyperbolic space. The area of each tile is **constant** in the hyperbolic space while the area gradually decreases as tiles get closer to the boundary in an euclidian space. **Right:** A tree output in an euclidian space modified. Source: `https://bjlkeng.github.io/posts/`

4.7 Learning a continuous hyperbolic embedding of skills

This section provides basic materials about how to learn a continuous embedding of trees so that we can discuss if ELSIM could integrate a hierarchical setting. It ends up with a discussion on the perspectives of this approach.

4.7.1 A short introduction to hyperbolic spaces

Creating a continuous embedding of the tree of skills is not trivial; this embedding should be induced with a metric that captures how much skills are similar. Similarity between skills here refers to the length of the path in the tree that connects skills, this comes from how we build our tree of skills (Section 4.3). For instance, in the right part of Figure 4.12, z is far from z' since three nodes separate them, but close to z'' since one node separate them. While the induced metric usually follows an euclidian distance, the required dimensionality grows too fast with the number of levels when the metric is intrinsically hierarchical like in a tree [Linial et al., 1995, Sala et al., 2018], the required dimension of the embeddings may be unbounded. This is especially due to the fact that the number of nodes of a tree grows exponentially with the depth of a tree. More specifically, assuming each node of a tree has b children, with maximum depth l, we can compute the number of nodes as $n = 1 + b^{l-1}$. The right part of Figure 4.12 illustrates this issue: the euclidian space quickly saturates and, although nodes z and z' should be very different, their euclidian distance is low.

Recent studies rather suggest that hyperbolic embeddings may more appropriately fit the hierarchical structure of a tree than embeddings provided with an euclidian metric [Nickel and Kiela, 2017, Sonthalia and Gilbert, 2020, Law et al., 2019, Chamberlain et al., 2017]. Essentially, hyperbolic geometry is a non-Euclidian geometry that studies spaces provided with a constant negative sectional curvature. This curvature modifies the notion of distance between points so that the area of a disc grows exponentially with its radius (instead of quadratically for an uncurved euclidian geometry). It follows that hyperbolic geometry can be thought as a continuous version of a tree [Krioukov et al., 2010], making intuitive its relevance for embedding a tree.

Several models exist to study an hyperbolic geometry, but we focus on the Poincare Ball, since this is largely used in machine learning [Nickel and Kiela, 2017, Khrulkov et al., 2020], even though other models may be used [Nickel and Kiela, 2018]. The Poincare Ball model corresponds to the Riemannian manifold with $(\mathcal{B}_c^d, g_p^c)$ where $c < 0$ is an arbitrary negative curvature coefficient, $\mathcal{B}_c^d$ is the d-dimensional open-ball of radius $\frac{1}{\sqrt{|c|}}$ and $g_p^c = \frac{2}{1+c||z||_2^2} g_e$ is its metric tensor. g_e is the euclidian metric tensor, *i.e* it defines a dot-product. Using this model, all points belong to $\mathcal{B}_c^d$ and we can analytically compute the distance between two points using Equation 4.9.

$$d_p^c(z, z') = \frac{1}{\sqrt{|c|}} arcosh\big(1 + 2c\frac{||z - z'||_2^2}{(1 + c||z||_2^2)(1 + c||z'||_2^2}\big). \tag{4.9}$$

The Poincare disc model $\mathcal{B}_1^2$ illustrated in the left part of Figure 4.12 shows how distances are altered when points get closer to the boundary of the ball: they exponentially increase with small variations. This is reflected in the distance estimation of Equation 4.9, the denominator exhibits that, when vectors are closed to the boundary, distances tend to be larger than those of vectors close to the origin of the ball. Consequently, the center of the ball is close to all points relatively to the boundary points; this gives an intuitive understanding of the induced hierarchy: the root of the tree should be close to the origin and leaves should be close to the boundary of the ball.

4.7.2 Hyperbolic skill embedding

In practice, there are two ways to compute an hyperbolic embedding in a Poincare Ball. We can use optimization methods based on SGD, contrastive learning and hyperbolic distance [Nickel and Kiela, 2017]. Equation 4.10 provides an example of loss function which simultaneously finds out embeddings of all nodes, using a dataset $\mathcal{D}$ of connected nodes. The numerator brings together connected nodes while the denominator moves away unconnected nodes.

$$\mathcal{L}_Z = \sum_{(z,z') \in \mathcal{D}} \log \frac{e^{-d_p^c(z,z')}}{\sum_{z_n | (z,z_n) \notin \mathcal{D}} e^{-d_p^c(z,z_n)} + 1} \tag{4.10}$$

Alternatively, one can analytically embeds a provided tree [Sala et al., 2018, Sarkar, 2011] using geometric operators [Brannan et al., 1999]. This method, called *Sarkar's construction*, progressively embeds nodes by starting from the root and adding nodes with a depth-first strategy. In both cases, our preliminary experiments highlight several shortcomings. We briefly detail them below.

Scaling factor. An example of already learnt skill embedding resulting from Sarkar's construction applied on skills previously stored is displayed in Figure 4.13. We see that skill $(0, 0, 0)$ is close to skill $(2, 2)$, even though we consider an hyperbolic distance. In fact Sarkar's construction trades off how quickly close-to-the-root nodes gets closer to the boundary with a scale factor τ. A high scale factor will set a lot of nodes very close to the boundary. This keeps the distances of the original tree but makes embeddings sensitive to the precision of float numbers [Sala et al., 2018]: for an embedding e, $\frac{1}{\sqrt{(c)}} - e$ should be different from 0, but as it gets closer to 0 (*e.g* 10^{-15}), it may be confounded with 0. A low scale factor reverses the trade-off. This issue is shared by the

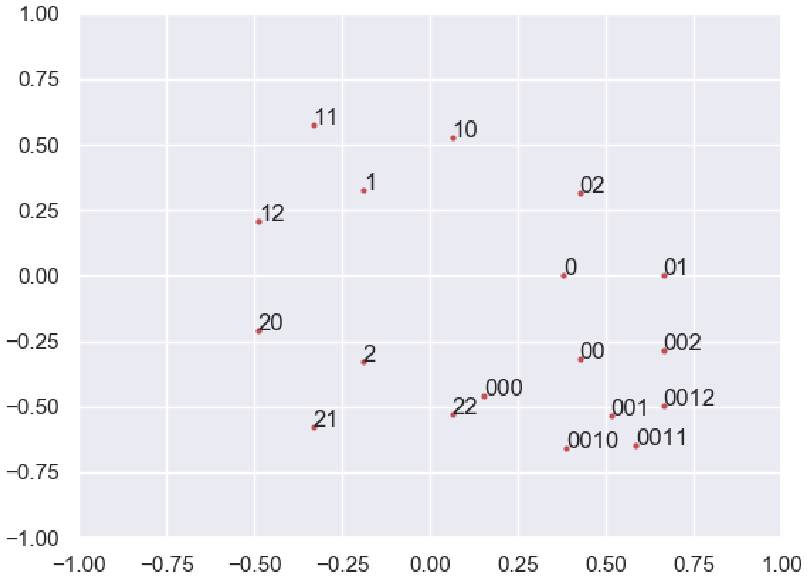


Figure 4.13: Embedding of a set of skills in the Poincare disc.

optimization-based method, for which the scaling factor can be simulated through a regularization that incites embeddings to not deviate too much from the origin. Ongoing research suggests that hyperbolic models different from the Poincare Ball could tackle this issue [Nickel and Kiela, 2018].

Hyperbolic DRL. Let us recall that our objective is to consider the skills as *options* selected by an upper-level DRL policy. However, despite recent advances in hyperbolic deep learning and hyperbolic Normal distributions [Ganea et al., 2018, Mathieu et al., 2019, Khrulkov et al., 2020, Pennec, 2006, Nagano et al., 2019], it remains unclear how a high-level DRL algorithm could output hyperbolic goals. It is made worse by the fact that it should output *possible* goals, *i.e* goals that correspond to existing goals inside the whole Poincare Ball.

Graph versus tree. Figure 4.5 highlighted that ELSIM can refine a high-stochastic skill into a low-stochastic one. This is interesting because it highlights that the tree of skills captures a hierarchical structure of the state space built by the discriminators; close-to-the-origin skills target larger part of the state space whereas leaf skills target smaller parts. However, in practice, the tree and skills most of time do not follow this semantic. This is emphasized by Figure 4.6 and Figure 4.14. A new skill does not only target a more specific area, it targets a totally *different area* than the area of its parent. At the end, looking at the state space and learnt skills, the goal representations of the top-right and the bottom-right skills are very different although they access close areas. This error may happen every time the agent finds a novel path between otherwise different goals, as a consequence of exploration. It may complicate generalization over the goal space in novel areas.

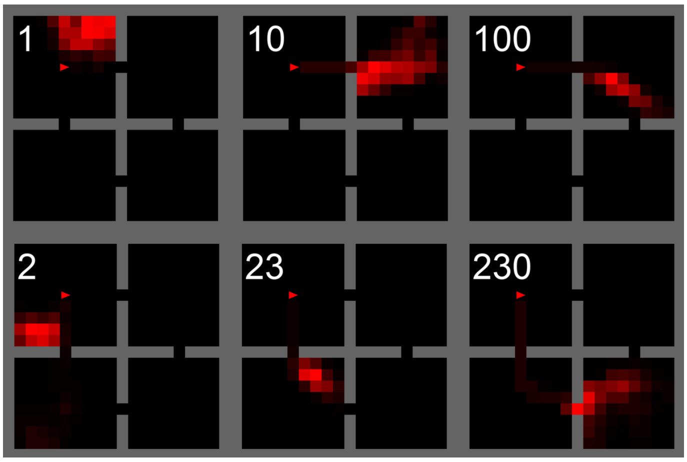


Figure 4.14: Skills (1,0,0) and (2,3,0) and their parents. Extracted from Figure 4.6.

4.7.3 Conclusion

These observations suggest that our hierarchical representation is not adequate to explore and match the structure of the environment, a graph may be more appropriate by being able to handle such setting. However, further works on hyperbolic embeddings may open new research perspectives by considering a continuous hierarchical abstraction of the state space. We expect further improvements in hyperbolic DRL algorithms to open this path. In the next chapter, we thoroughly investigate our second option which was about directly learning a continuous representation of the state space. In particular, we will introduce a new model that aims to tackle the shortcomings of ELSIM.

Chapter 5

Discovering a topological representation to learn diverse and rewarding skills

In the last chapter, we stressed out several mandatory properties to learn skills that can be simultaneously learnt and organized in a hierarchical way. We sum up these *desideratas*:

1. An agent should stop trying to improve a skill if it can not extend it.
2. Learning a continuous skill representation may make easier both a hierarchical integration of skills and the generalization of a goal-conditioned policy.
3. A graph-based representation of the environment may better fit the structure of the environment.

In particular, an efficient way to explore an environment can be to learn diverse skills that maximize the state entropy [Hazan et al., 2019, Pong et al., 2020]. To follow such principle, we propose that the agent should not focus on *what* to learn, as commonly done in RL, but rather on ***where* to learn in its environment.** Thus the agent should learn a representation of states that keeps the structure of the environment to be able to select where it is worth learning, instead of learning a representation that fits a wanted behavior which is most of the time unknown. Indeed, a standard DRL agent tackles the issue of learning a representation of its states from scratch using only extrinsic rewards. In a sparse-reward environment, the agent turns out to be unable to locate the reward and learn a representation of its surrounding.

In this chapter, we introduce a new way to learn a continuous goal space and select the goals, keeping the learning process end-to-end. We propose a new model that progressively **Dis**covers a **Top**ological representation (**DisTop**) of the environment. DisTop bridges the gap between acquiring skills that reach a uniform distribution of terminal embedded states and solving a task. It makes diversity-based skill learning suitable for end-to-end exploration in a single-task setting irrespective of the ground state space. DisTop simultaneously optimizes three components:

1. it learns a continuous representation of its states using a contrastive loss that brings together consecutive states;

2. this representation allows the building of a discrete topology of the environment with a new variation of a *Growing When Required* network [Marsland et al., 2002a]. Using the clusters of the topology, DisTop can sample from an almost-arbitrary distribution of visited embedded states using a very small set of parameters;

3. it trains a goal-conditioned deep RL policy to reach embedded states.

Upon these 3 components, a hierarchical state-independent Boltzmann policy selects the cluster of skills to improve according to an extrinsic reward and a diversity-based intrinsic reward.

We show, through DisTop, that the paradigm of choosing *where* to learn and *what* to forget using a learnt discrete topology is more generic than previous approaches. Our contribution is 4-folds: 1- we visualize the representation learnt by DisTop and exhibit that, unlike previous approaches [Pong et al., 2020], DisTop is agnostic to the shape of the ground state space and works with states being images, high-dimensional proprioceptive data or high-dimensional binary data; 2- we demonstrate that DisTop clearly outperforms our previous method ELSIM; 3- we show that DisTop achieves similar performance with state-of-the-art (SOTA) algorithms on both single-task dense rewards settings [Haarnoja et al., 2018] and multi-skills learning benchmarks [Pong et al., 2020]; 4- we show that it improves exploration over a state-of-the-art hierarchical method on hard hierarchical benchmarks.

In Section 5.1, we first describe the basic elements required to understand DisTop . Then, in Section Section 4.3 we describe the motivation of DisTop and its core components. After that, we describe the experiments that support the effectiveness of DisTop (Section 5.3). Finally, we explain how our work relates with previous approaches (Section 5.5) and we discuss the perspectives and limitations of DisTop (Section 4.5).

5.1 Background

In this section, we introduce the key tools required to understand how DisTop works. This includes the framework of entropy maximization of Skew-fit, basic elements on contrastive learning and a well-known topology-mapping growing network.

5.1.1 Goal-conditioned RL for skill discovery

Skew-Fit [Pong et al., 2020] strives to learn goal-conditioned policies that visit a maximal entropy of states. We already discussed Skew-fit in Section 3.5.3 but we deepen some aspects in this section.

To generate goal-states with high entropy, they learn a generative model, *i.e* a model that generates states, that incrementally increases the entropy of generated state-goals and makes visited states progressively tend towards a uniform distribution. Assuming a parameterized generative model of states $q_\psi(s)$ is available, an agent would like to learn ψ to maximize the log-likelihood of

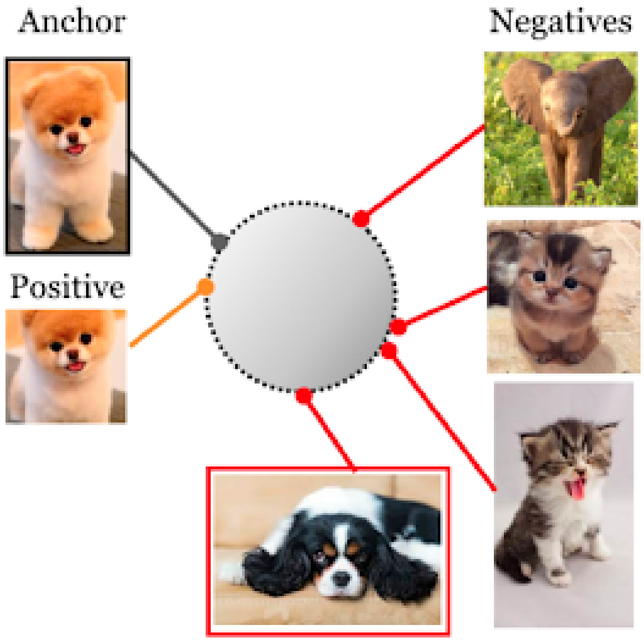


Figure 5.1: Illustration of an embedding of a unsupervised contrastive loss. Extracted from https://ai.googleblog.com/2021/06/extending-contrastive-learning-to.html.

the state distribution according to a uniform distribution: $\mathbb{E}_{s\sim U(S)} \log q_\psi(s)$. But sampling from the uniform distribution $U(S)$ is hard since the set of reachable states is unknown.

Skew-Fit uses importance sampling and approximates the true ratio with a skewed distribution of the generative model $q_\psi(s)^{\alpha_{skew}}$ where $\alpha_{skew} < 0$. This way, it maximizes $\mathbb{E}_{s\sim\mathbb{B}}\, q_\psi(s)^{\alpha_{skew}} \log q_\psi(s)$ where s are uniformly sampled from the buffer $\mathbb{B}$ of the agent. α_{skew} defines the importance given to low-density states; if $\alpha_{skew} = 0$, the distribution will stay similar, if $\alpha_{skew} = -1$, it strongly overweights 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.

5.1.2 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 they are given to an algorithm that computes a data representation able to discriminate the positive from the negative ones. Figure 5.1 illustrates the learning process when no labels are provided. The agent samples an anchor and embeds in a low-dimensional state represented by a circle. Then a process transforms the image in positive examples and the agent other data points denoted labeled as negatives, here images of elephants and cats. The agent brings together the anchor and its transformation in the embedding space and separates the anchor from negative samples. In unsupervised learning, the positive sample building process is a critical step since the whole embedding builds over this. Several methods [Laskin et al., 2020, Schwarzer et al., 2020] propose to build them using data augmentation methods; this includes random shifts, crops or color jitter.

InfoNCE [van den Oord et al., 2018] proposed to build positive pairs from states that belong

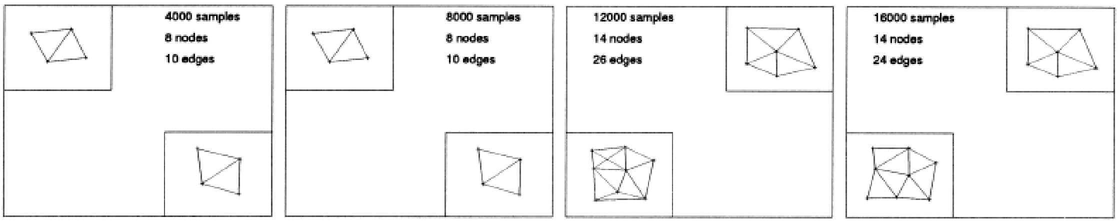


Figure 5.2: Illustration of the networks learnt while varying the input distribution. Extracted from Marsland et al. [2002a].

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, they build a set $\mathcal{B}$ that concatenates $N-1$ negative states and a positive one s_{t+p}. Then, they maximize $\mathbb{L}^{N}_{InfoNCE} = -\mathbb{E}_{\mathcal{B}}\left[\log \frac{f_p(s_{t+p},v_t)}{\sum_{s_j \in \mathcal{B}} f_p(s_j,v_t)}\right]$ 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. InfoNCE maximizes a lower bound of the mutual information $I(v_t, s_{t+p})$.

One can also see such optimization function as metric learning [Tschannen et al., 2020]. The objective of metric learning is to learn a function $g : \mathbb{R}^d \rightarrow \mathbb{R}^p$ that maps semantically similar inputs $x \in \mathbb{R}^d$ onto metrically close points in $\mathbb{R}^p$. In the case of InfoNCE, we can set $f_p(x, y) = sim(g(x), g(y))$ where sim is a similarity function that defines the wanted metric (euclidian distance for example).

5.1.3 Growing when required

Several methods introduced unsupervised neural networks to learn a discrete representation of a fixed input distribution [Kohonen, 1990, Fritzke et al., 1995, Marsland et al., 2002a]. 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., 2002a] since it updates the structure of the network (creation and deletion of edges and nodes) and does not impose a frequency on node addition. Figure 5.2 shows the network learnt while facing points sampled inside the two squares. The network globally matches the input distribution and quickly changes its topology along with the input distribution (between the second and third squares).

In the following, we will summarize the algorithm.It starts with 2 nodes and an edge that connects them. A d dimensional weight vector w_i is associated to each node i, denoting its position in the input space. Input points x_t are sequentially given to the network and each one go through several steps:

1. get the closest node c_t^1 to x_t (firing one) and its second closest node c_t^2;
2. add (or update) a winning edge with $age = 0$ between the two closest nodes;
3. if the distance between c_t^1 and x_t exceeds a threshold and the c_t^1 firing count is greater than

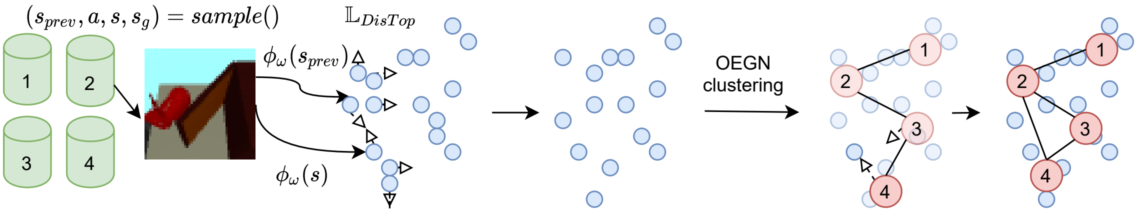


Figure 5.3: Illustration of a learning step of our growing network and contrastive loss (cf. text). Cylinders are buffers associated to each cluster, blue circles are states embedded with ϕ and pink circles represent clusters. The image is an example of state in the *Visual Door* [Nair et al., 2018] environment.

an other threshold, create a new node c' halfway and connect it to c_t^1 and c_t^2;

4. if no node is added, move c_t^1 and its neighbors towards x_t according to the respective learning rates α and $\alpha_{neighbors}$;

5. increment the age of the edges of c_t^1 and the firing count of c_t^1;

6. the age of an edge is deleted when it exceeds a threshold and a node is deleted when it does not have neighbors. The procedure repeats until a stopping criterion is met.

We provide additional details about the algorithm in Section 5.2.2.

5.2 Method

In this section, we introduce our model. Firstly, we give an overview of the method, then we will successively present the core components of the model, namely a new state-representation learning method (Section 5.2.2), a way to discretize the state-representation in order to bias state sampling (Section 5.2.2), how we mix the previous sampling with extrinsic rewards (Section 5.2.3) and technical details about the reward computation and our goal-relabeling strategies (Section 5.2.4).

5.2.1 Overview

As in the previous chapter, we define the interest of an agent so that its default behavior is to learn skills that achieve a uniform distribution of terminal states while focusing its skill discovery in extrinsically rewarding areas when a task is provided. If the agent understands the true intrinsic environment structure and can execute the skills that navigate in the environment, a hierarchical policy would just have to select the state to target and to call for the corresponding skill. However, this is difficult to learn both skills and the structure of the environment. First the ground representation of states may give no clue of this structure and may be high-dimensional. Second, the agent has to autonomously discover new reachable states. Third, the agent must be able to easily sample goals where it wants to learn.

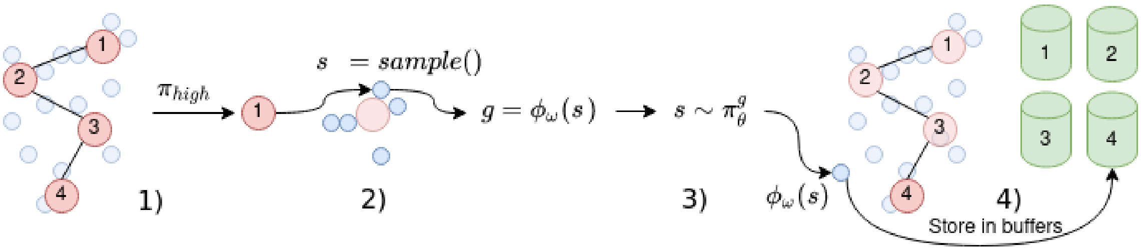


Figure 5.4: Illustration of the selection of a skill and an interaction of the skill. See text for explanations.

DisTop tackles theses challenges by executing skills that target low-density or previously rewarding visited states in a learnt topological representation. We refer to a state that acts like a goal as a **goal-state**. Figure 5.3 illustrates how DisTop learns a topological representation:

1. It samples an interaction (s_{prev}, a, s, s_g) from different buffers (see below), where s_g is the original goal-state and embeds the associated states with a neural network ϕ_ω (cf. Section 5.2.3). Here, it selects the second buffer.
2. Then, ϕ_ω is trained with a new contrastive loss that brings together consecutive states in the embedded space (cf. Section 5.2.2). This makes the embeddings of states reflect the topology of the environment, which is important to make the next step meaningful.
3. After that, our growing network dynamically clusters the embedded states in order to uniformly cover the embedded state distribution. In Figure 5.3, the two nearests clusters, three and four gets closer to the input embedding. DisTop approximates the probability density of visiting a state with the probability of visiting its cluster (cf. Section 5.2.3).

Finally, it assigns buffers to clusters, and can sample goal-states or states with almost-arbitrary density over the support of visited states, *e.g* the uniform or reward-weighted ones (cf. Section 5.2.3).

Figure 5.4 describes how DisTop gathers states:

1. At the beginning of each episode, a state-independent Boltzmann policy π_{high} selects a cluster.
2. Then a goal-state s is selected that belongs to the buffer of the selected cluster and the agent finally computes its representation $g = \phi_{\omega'}(s)$ where weights ω' are a slow exponential moving average of ω.
3. A goal-conditioned policy π^g_θ, trained to reach g, acts in the environment and discovers new reachable states close to its goal.
4. The interactions made by the policy are stored in two buffers: according to their initial objective and according to the embedding of the reached state.

We will now detail each step of the DisTop process.

5.2.2 Learning the topology of the states

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

Learning an undistorted continuous representation of states. DisTop strives to learn a **state representation** function ϕ_ω that maps a given state to an embedded space. To learn this function, we propose to maximize the constrained mutual information between the consecutive states resulting from the interactions of an agent. We call **consecutive states**, pair of states that are separated by one action, *i.e* pairs for which there exists a transition. We define the binary operator $\mathbb{1}(s, s')$ to be equal to 1 if s and s' follow each other, 0 otherwise.

To learn ϕ_ω, DisTop takes advantage of the InfoNCE loss (cf. Section 5.1.2). In contrast with previous approaches [Lu et al., 2019, van den Oord et al., 2018], we want to keep our representation locally undistorted [Lesort et al., 2018, Indyk, 2001], so that it reflects the true topology of the environment and we can apply a L2-based clustering algorithm on this representation. We call **undistorted** a representation that respects:

$$||\phi_\omega(s) - \phi_\omega(s')||_2 \leq k_c, \quad \text{if } \mathbb{1}(s, s'). \tag{5.1}$$

$$||\phi_\omega(s) - \phi_\omega(s')||_2 > k_c, \quad \text{otherwise.} \tag{5.2}$$

To compute such representation, 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 consecutive states. This way, it keeps distinct the states that cannot be consecutive. We propose to select our unique 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, $||_2$ is the L2 norm and k is a temperature hyper-parameter [Chen et al., 2020]. If $\mathcal{B}$ is a batch of $N-1$ negative states and 1 consecutive state, the local InfoNCE objective, LInfoNCE, is described by Equation 5.5.

$$\begin{aligned}
\mathbb{L}_{InfoNCE} &= \mathop{\mathbb{E}}_{(s_t,s_{t+1}) \in \mathcal{B}} \big[\log \frac{f_\omega(s_t, s_{t+1})}{\sum_{s \in \mathcal{B}} f_\omega(s, s_{t+1})}\big] \\
&= \mathop{\mathbb{E}}_{(s_t,s_{t+1}) \in \mathcal{B}} \big[-k||\phi_\omega(s_t) - \phi_\omega(s_{t+1})||_2 - \log(\sum_{s \in \mathcal{B}} f_\omega(s, s_{t+1}))\big] && (5.3) \\
&= \mathop{\mathbb{E}}_{(s_t,s_{t+1}) \in \mathcal{B}} \big[-k||\phi_\omega(s_t) - \phi_\omega(s_{t+1})||_2 - \log(f_\omega(s_t, s_{t+1}) + \sum_{s \in \mathcal{B}_{s \neq s_t}} f(s, s_{t+1}))\big] && (5.4) \\
&\geq \mathop{\mathbb{E}}_{(s_t,s_{t+1}) \in \mathcal{B}} \big[-k||\phi_\omega(s_t) - \phi_\omega(s_{t+1})||_2 - \log(1 + \sum_{s \in \mathcal{B}_{s \neq s_t}} f(s, s_{t+1}))\big] && (5.5)
\end{aligned}$$

In the last line of Equation 5.5, 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. We introduce this bound since we found it to stabilize the learning process.

Intuitively, Equation 5.5 brings together consecutive states, and pushes away states that are separated by a large number of actions. This is illustrated in the second step of Figure 5.3.

There are several drawbacks with this objective function. Firstly, the representation may still be strongly distorted, making a clustering algorithm inefficient since the pushing away term (right-hand term) can override the other one. Secondly, the DRL algorithm requires semantically stable inputs to compute Q-values: if the representation of its goals quickly changes, it can not take into account the changes and may output bad approximations of Q-values. To tackle this, Equation 5.6 reformulates the objective as a constrained maximization. Firstly, DisTop forces the distance between consecutive states to be below a threshold δ (first constraint of Equation 5.6). Secondly, it enforces our representation to stay consistent over time, *i.e* lower than a close to 0 constant ϵ (second constraint of Equation 5.6). In consequence, we avoid using a hand-engineered environment-specific scheduling [Pong et al., 2020] and update representations in an online fashion.

$$\begin{aligned}\max_{\omega} \underset{(s_{t+1})\in\mathcal{B}}{\mathbb{E}} -\log(1+\sum_{s\in\mathcal{B}} f_\omega(s,s_{t+1})) \quad \text{s.t.} \quad & \underset{(s_t,s_{t+1})\in\mathcal{B}}{\mathbb{E}} ||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2 \leq \delta \\ & \underset{s_{t+1}\in\mathcal{B}}{\mathbb{E}} ||\phi_\omega(s_{t+1})-\phi_{\omega'}(s_{t+1})||_2^2 \leq \epsilon\end{aligned} \tag{5.6}$$

Transforming the constraints into penalties, the agent maximizes Equation 5.7. k_c is the temperature hyper-parameter that brings closer consecutive states, β is the coefficient that slows down the speed of change of the representation. In practice we set $k_c > 1$ to avoid distortions (cf. Section 5.4).

$$\begin{aligned}\mathbb{L}_{DisTop} = & \underset{(s_t,s_{t+1})\in\mathcal{B}}{\mathbb{E}} \big[-k_c(ReLU(||\phi_\omega(s_t)-\phi_\omega(s_{t+1})||_2-\delta)) - \log(1+\sum_{s\in\mathcal{B}} f_\omega(s,s_{t+1})) \\ & -\beta||\phi_\omega(s_{t+1})-\phi_{\omega'}(s_{t+1})||_2^2\big]\end{aligned} \tag{5.7}$$

By applying this objective function, DisTop learns a consistent, not distorted representation that keeps close consecutive states while avoiding collapse (see Figure 5.7 for an example). In fact, by increasing k_c and/or k, one can select the level of distortion of the representation (cf. Section 5.4 for an analysis). One can notice that the function depends on the distribution of consecutive states in $\mathcal{B}$; we experimentally found that using tuples (s_t, s_{t+1}) from sufficiently stochastic policy is enough to keep the representation stable. We discuss the distribution of states that feed $\mathcal{B}$ in Section 5.2.4. 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. In the following, we will study how DisTop takes advantage of this specific representation to sample diverse or rewarding state-goals.

Mapping the continuous representation to 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 will detail how to bias the goal and state sampling procedure to favor exploration and the maximization of an extrinsic reward.

In order to adapt the building process to temporally-related and goal-related interactions, we adapt the GWQ and propose the Off-policy Embedded Growing Network (OEGN). OEGN dynamically creates, moves and deletes clusters so that clusters generates a network that uniformly cover the whole set of embedded states, independently of their density; this is illustrated in the two last steps of Figure 5.3. Each node (or cluster) $c \in C$ has a reference vector w_c which represents its position in the input space. The update operators make sure that all embedded states $\phi_\omega(s)$ are within a ball centered on the center of its cluster, *i.e* $min_c\,(\phi_\omega(\mathrm{s})\text{-}\mathrm{w_c})^\top(\phi_\omega(\mathrm{s})\text{-}\mathrm{w_c}) \leq \delta_{new}$ 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: if it is low, we will have a lot of small clusters, else we will obtain few large clusters that badly approximate the topology. The algorithm works as follows: assuming a new low-density embedded state is discovered (rare state located at the border of a ball), 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. In the case of a high-density embedded state, it does not much modify the spread of balls. A learnt discrete topology can be visualized at the far right of Figure 5.7.

Algorithm 1 Algorithm of OEGN (red) and GWQ (blue)

```
Initialize network with two random nodes, set theirs attributes to 0 and connect them.
for each learning iteration do
    Sample a tuple $s_i, c_i, s_i^{prev} \leftarrow sample(\mathcal{B})$.
    Embed states: $e_i \leftarrow \phi_{\omega'}(s_i)$; $e_i^{prev} \leftarrow \phi_{\omega'}(s_i^{prev})$
    Sample an input $e_i \leftarrow sample(\mathcal{B})$.
    $closest \leftarrow min_{c \in C}(||c - e_i||_2)$.
    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 $EdgeOperator()$.
end for
```

Algorithm 1 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. Following Algorithm 1, we define $e = \phi_{\omega'}(s)$ and $closest(e) = min_{c \in C}(||c - e||_2)$ for all $s \in \mathcal{B}$.

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 is connected to $closest$. Update a firing count of the winning 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 Equation 5.8. In our case, we use a very low $\alpha_{neighbors}$ to avoid the nodes to concentrate close to high-density areas.

$$w_j = \begin{cases} w_j + \alpha(\phi_{\omega'}(s) - w_j), & \text{if } j = closest \\ w_j + \alpha_{neighbors}(\phi_{\omega'}(s) - w_j), & \text{if } j \in neighbors(closest) \\ w_j, & \text{otherwise.} \end{cases} \tag{5.8}$$

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}.

In the next section, we detail how we take advantage of the topology to bias the state and goal sampling process.

5.2.3 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 $\mathbb{R}^{10}$, the topology of the environment may only exploit a small part of it, making most of the goals pointless. Similarly to previous works [Warde-Farley et al., 2019], DisTop generates goals by sampling previously visited states. To sample the states, DisTop first samples a cluster, and then samples a state that belongs to the cluster.

Building a skewed distribution To sample more often low-density embedded states, we assume that the density of a visited state is reflected by the marginal probability of its cluster. So we approximate the density of a state with the density parameterized by w, reference vector of $e = \phi_{\omega'}(s)$:

$$q_w(\phi_{\omega'}(e)) \approx \frac{count(c_s)}{\sum_{c' \in C} count(c')} \tag{5.9}$$

where $count(c_s)$ denotes the number of interactions that belong to the cluster that contains e. Using this approximation, we skew the distribution very efficiently by first sampling a cluster with the probability given by

$$p_{\alpha_{skew}}(c) = \frac{count(c)^{1+\alpha_{skew}}}{\sum_{c' \in C} count(c')^{1+\alpha_{skew}}} \tag{5.10}$$

where α_{skew} is the skewed parameter (cf. Section 5.1.1).

While our approximation $q_w(s)$ decreases the quality of our skewed objective, it makes our algorithm very efficient: we associate a buffer to each cluster and only weight the distribution of clusters; unlike Pong et al. [2020], DisTop does not weight each visited state , but only a limited set of clusters. In practice we can trade-off the bias and the instability of OEGN by decreasing the radius of the balls δ_{new}: the smaller the clusters are, the smaller the bias is, but the more unstable are OEGN and the sampling distribution (see Section 5.4). So, in contrast with Skew-Fit, we do not need to compute the approximated density of each state, we just need to keep states in the buffer of their closest node. 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: $\mathcal{B}^G$ and $\mathcal{B}^S$ to respectively sample goal-states and states with the skewed distribution. Formally, given an interactions (s_g, s, s', a) :

- in $\mathcal{B}^S$, DisTop selects the cluster $\underset{c \in C}{\arg\min}\ ||\phi(s) - c||_2$;
- in $\mathcal{B}^G$, DisTop selects the cluster $\underset{c \in C}{\arg\min}\ ||\phi(s_g) - c||_2$.

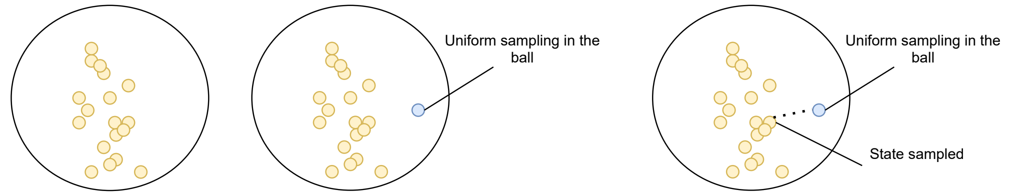
In the next sections, we will detail how we use this skewed distribution over $\mathcal{B}^G$ and B^S for sampling low-density goal-states (Section 5.2.3) and sampling learning interactions (Section 5.2.4).

Sampling from the skewed distribution It is not easy to sample new reachable goals. For instance, using an embedding space $\mathbb{R}^{10}$, the topology of the environment may only exploit a small part of it, making most of the goals pointless. Similarly to previous works [Warde-Farley et al., 2019], DisTop generates goals by sampling previously visited states. To sample the states, DisTop first samples a cluster, and then samples a state that belongs to the cluster.

Sampling a cluster: To increase the entropy of states, DisTop samples goal-states with the skewed distribution defined in Section 5.2.2, which can be reformulated as:

$$p_{\alpha_{skew}}(c) = \frac{e^{(1+\alpha_{skew})\log count(c)}}{\sum_{c' \in C} e^{(1+\alpha_{skew})\log count(c')}}. \tag{5.11}$$

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 $1 + \alpha_{skew}$. In practice, we can use a different α'_{skew} than in Section 5.2.2 to trade-off the importance of the novelty reward in comparison with an extrinsic reward (see below) or decrease the speed at which we gather low-density states [Pong et al., 2020].



(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 5.5: Illustration of how an agent samples interactions from a buffer.

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 that represents the mean average extrinsic rewards received by the skills associated to its goals :

$$r_c = \mathop{\mathbb{E}}_{s \in c, g=\phi_\omega(s)} \frac{1}{T} \sum_{t=0}^{T} \mathop{\mathbb{E}}_{a_t \sim \pi_\theta^g(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t,a_t)} R(s_t, a_t, s_{t+1}). \tag{5.12}$$

The extrinsic value of a cluster R_c^{ext} is updated with an exponential moving average $R_c^{ext} = (1 - \alpha_c) * R_c^{ext} + \alpha_c * r_c$ where α_c is the learning rate. To favor exploration, we can also update the extrinsic value of the cluster's neighbors with a slower learning rate. Our sampling rule can then be :

$$\pi^{high}(c) = \text{softmax}_C(t^{ext} R_c^{ext} + (1 + \alpha'_{skew}) \log count(c)) \tag{5.13}$$

Sampling a state: Once the cluster is selected, we keep exploring independently of the presence of extrinsic rewards. To approximate a uniform 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 5.5b). 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 5.5c).

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, we want to simulate uniform sampling in the ball that corresponds to the selected cluster. 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.

5.2.4 Learning goal-conditioned policies

Training We now briefly introduce the few mechanisms used to efficiently learn the skills π_θ^g that achieve sampled goals. Our implementation of the goal-conditioned policy is trained with Soft

Actor-Critic (SAC) [Haarnoja et al., 2018] and the reward is the L2 distance between the state and the goal in the learnt representation: $r_t^g(s_{t-1}, a_{t-1}, s_t, s_t^g) = ||\phi_{\omega'}(s_t) - \phi_{\omega'}(s_t^g)||_2$ like Pong et al. [2020].

Our goal-conditioned policy π_θ^g needs a uniform distribution of goals to avoid forgetting previously learnt skills. Our representation function ϕ_ω requires a uniform distribution over visited states to quickly learn the representation of novel areas. In consequence, DisTop samples a cluster $c \sim p_{\alpha_{skew}}$ and randomly takes half of the interactions from $\mathcal{B}^G$ and the rest from $\mathcal{B}^S$. We can also sample a ratio of clusters with π^{high} if we do not care about forgetting skills (cf. Section 5.4). θ, ω and w are learnt over the same mini-batch and we relabel the goals extracted from $\mathcal{B}^S$ as detailed below.

Relabeling strategies. To increase the learning efficiency, we proposed three relabeling strategies:

1. π^{high} relabelling: we take samples from $\mathcal{B}^S$ 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. Indeed, since it does not learn of these skills, it forgets how to execute them and the corresponding nodes of the graph gets deleted.

2. Uniform relabelling: we take samples from $\mathcal{B}^S$ and relabel the goal using states sampled from from $\mathcal{B}^S$. When $\alpha_{skew} \approx 0$, this is equivalent to relabeling uniformly over the embedded state space. It is close to the stategy used by Pong et al. [2020].

3. Topological relabelling: we take samples from both $\mathcal{B}^S$ and $\mathcal{B}^G$ and relabel each goal with a state that belongs to a neighboring cluster. This is interesting when the topology is very large, making an uniform relabelling inefficient.

5.3 Experiments

Our experiments aim at studying whether the ability of DisTopto select skills to learn and forget makes the approach generic to different task settings. We compare DisTop to three SOTA algorithms on three kinds of tasks with very different ground state spaces, thereby we compare DisTop to: the SOTA algorithm SAC [Haarnoja et al., 2018] on dense rewards task; the SOTA algorithm Skew-Fit [Pong et al., 2020] on no-rewards task; SAC and the SOTA algorithm LESSON [Li et al., 2021b] on sparse rewards task. We also compare DisTop to ELSIM, which follows the same paradigm than ours, on the ability to explore the environment and solve a dense-reward task. That is unclear to us how to fairly adapt Skew-fit to solve a particular task and we can not assess the skills diversity of SAC since it does not learn skills. Similarly, LESSON requires a task to learn skills and it explores with hierarchical random walk, making a fair comparison unrelevant in no/dense-rewards task. We also compare the learnt representation with the representation learnt by a VAE to exhibit its properties.

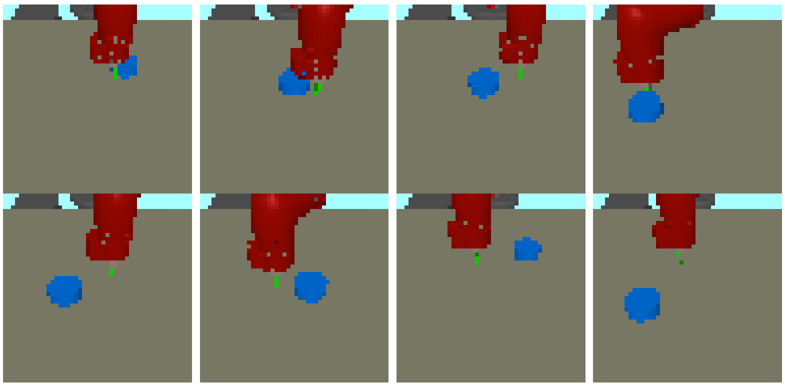


Figure 5.6: Examples of 8 terminal states after the execution of 8 different skills learnt in *Visual Pusher*.

All curves are a smooth averaged over 5 seeds, completed with a shaded area that represent the mean +/- the standard deviation over seeds. Used hyper-parameters are described in Appendix B.3, environments and evaluation protocols details are given in Appendix B.4. Figure 5.6 displays some skills learnt by our algorithm, but further videos and images are available in Appendix B.6.

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., 2020], which skews the goal distribution in ground the state space with a VAE [Kingma and Welling, 2014] and periodically updates its representation. We use the same evaluation protocol than Skew-Fit: a set of images are sampled, given to the representation function and the goal-conditioned policy executes the skill. In Figure 5.8, 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 by *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 diverse skills, but may more or less fluctuate according to the overall proximity of states.**

Does DisTop discover the environment topology even though the ground state space is unstructured ? In this experiment, we analyze the representation learnt by a VAE [Kingma and Welling, 2014] and DisTop. To make sure that the best representation is visualizable, we adapted the *gym-minigrid* environment [Chevalier-Boisvert and Willems, 2018] (cf. Figure 5.7) where a randomly initialized agent moves for fifty timesteps in the four cardinal directions. We use the ground state space, either an unstructured one with 900-dimensional one-hot vector with 1 at the position of

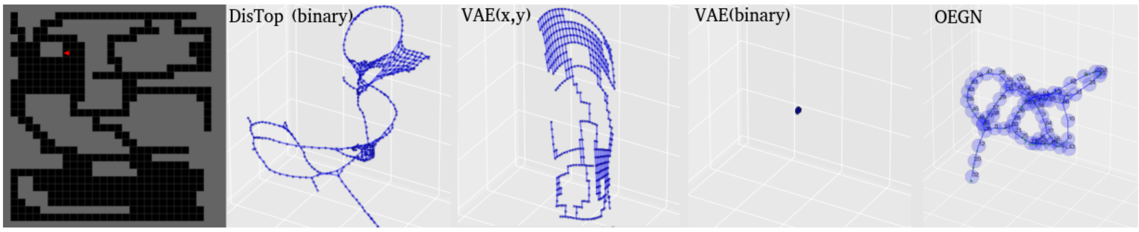


Figure 5.7: Visualization of the representations learnt by a VAE and DisTopon the gridworld displayed at the far left. From left to right, we respectively see a- the rendering of the maze; b- the continuous representation learnt by DisTop with 900-dimensional binary inputs; c- a VAE representation with true (x,y) coordinates; d- a VAE representation with 900-dimensional binary inputs; e- OEGN network learnt from binary inputs.

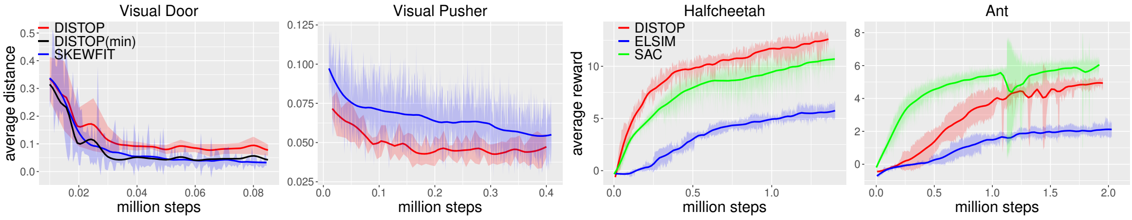


Figure 5.8: **Left:** Comparison of DisTop and Skew-Fit on their ability to reach diverse states. In the *Visual Pusher* environment, we compare the final distance of the position puck with its desired position; in the door environment, we compare the angle of the door with the desired angle. DisTop(min) 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.

the agent (*binary*) or a structured one composed of (x,y) coordinates of the agent. Interactions are given to the representation learning algorithm. Except for OEGN, we display as a node the learnt representation of each possible state and connect states that are reachable in one step. For OEGN, we simply displays the learnt network. In Figure 5.7, we clearly see that DisTop discovers the topology of its environment since each connected points are distinct but close with each other. Similary, OEGN network closely resembles the environment topology. 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 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 our implementation of SAC [Haarnoja et al., 2018], a SOTA algorithm, and to ELSIM, the method introduced in the previous chapter which follows the same paradigm than DisTop. We obtain better results than our previous experiments with ELSIM using similar hyper-parameters to DisTop. In Figure 5.8 (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 of SAC's favourite environments.**

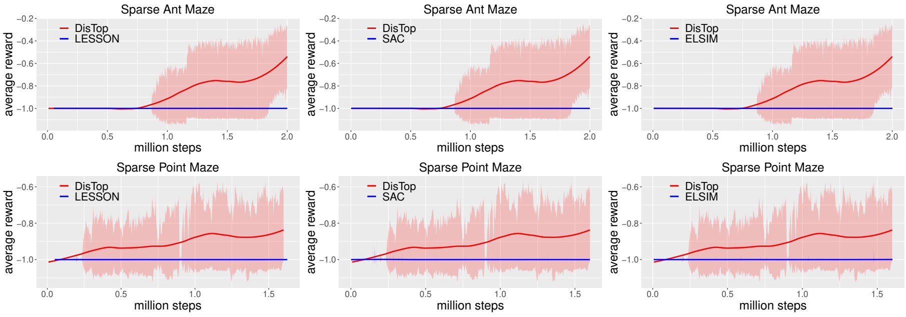


Figure 5.9: Average rewards throughout training episodes. We use a plot for each comparison to avoid curve overlaps.

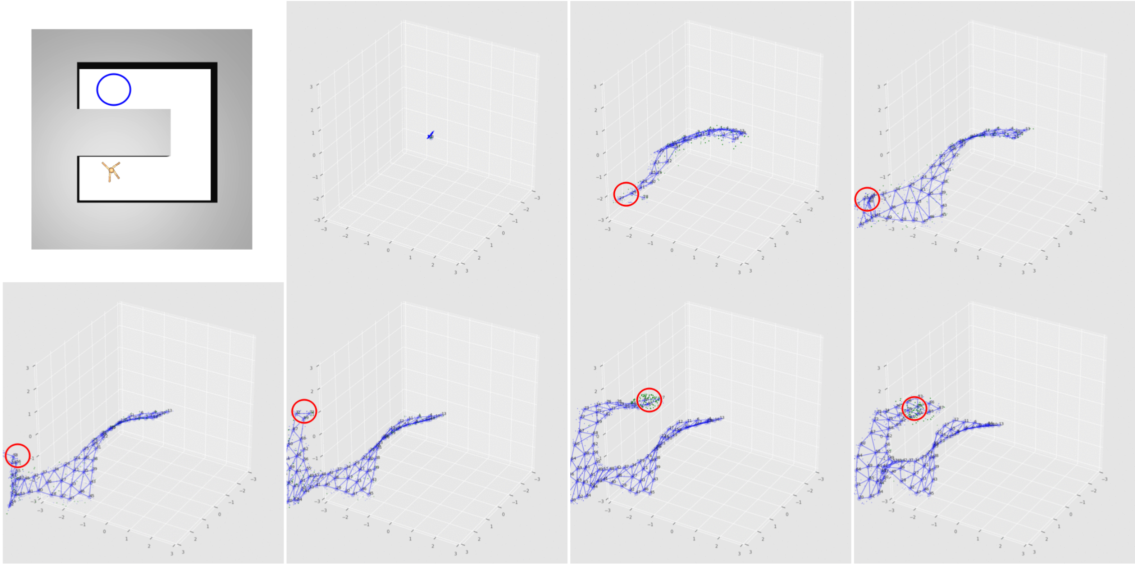


Figure 5.10: Top-down view of Ant Maze environment and visualization of the topology that is progressively learnt by DisTop. The blue circle represents the extrinsically rewarding area. Small green points represent selected goal-states and the red circle highlights the area under exploitation.

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., 2018]. 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 B.4.2. In contrast with previous methods [Nachum et al., 2018, Li et al., 2021b], 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-right goal. We compare our method with a SOTA hierarchical method, LESSON [Li et al., 2021b] since it performs well on hierarchical environments with extrinsic rewards like mazes, ELSIM and our implementation of SAC [Haarnoja et al., 2018]. For ELSIM, LESSON and DisTop, 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 5.9, 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 in Figure 5.10, we can see that it successfully represents the U-shape form of the maze and sets goals close to the extrinsic goal. LESSON discovers the goal but does not learn

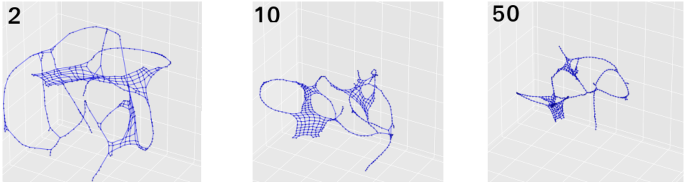


Figure 5.11: Different Topologies learnt on the gridworld displayed in Figure 5.7. From left to right, we show the learnt topology with $k_c = 2$, $k_c = 10$, $k_c = 50$.

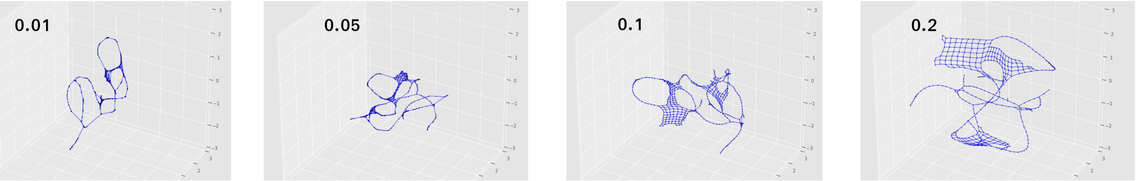


Figure 5.12: Different Topologies learnt on the gridworld displayed in Figure 5.7. From left to right, we show the learnt topology with $\delta = 0.01$, $\delta = 0.05$, $\delta = 0.1$, $\delta = 0.2$.

to return to it[1]; we hypothesize that this is because, unline 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.**

5.4 Ablation study

In this section, we study the impact of the different key hyper-parameters of DisTop. Except for previous results (entitled "paper"), we average the results over 3 random seeds. For visualizations based on the topology, the protocol is the same as the one described in Section 5.3. In addition, we select the most viewable one among 3 learnt topologies; while we did not observe variance in our analysis through the seeds, the 3D angle of rotation can bother our perception of the topology.

Controlling the distortion of the representation. 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 Figure 5.11, we study the influence of the distortion parameter k_c that brings closer consecutive states in $\mathbb{L}_{DisTop}$ (cf. Equation 5.7). We can see that the distortion parameter k_c rules the global dilatation of the learnt representation. A low k_c also increases the distortion of the representation, which may hurt the quality of the clustering algorithm. k_c competes with k, the temperature hyper-parameter of the similarity function in Equation 5.7. As we can see in Figure 5.13, k rules the minimal allowed distance between very different states. So, there is a trade-off between a low

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

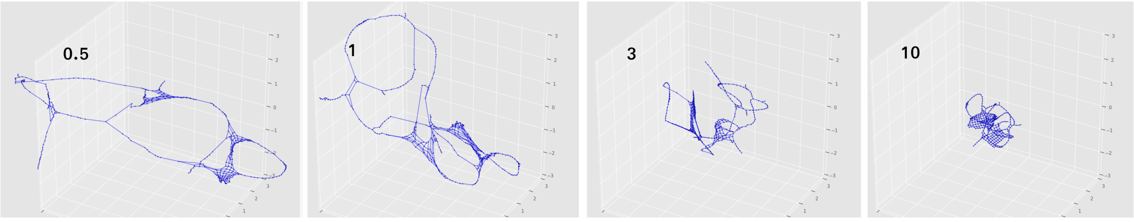


Figure 5.13: Different Topologies learnt on the gridworld displayed in Figure 5.7. From left to right, we show the learnt topology with $k = 0.5$, $k = 1$, $k = 3$, $k = 10$.

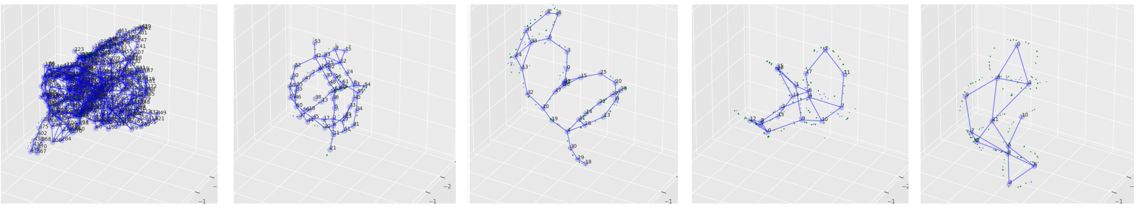


Figure 5.14: Different OEGN networks learnt according to δ_{new}. From left to right, we show the OEGN network with $\delta_{new} = 0.2$,$\delta_{new} = 0.4$,$\delta_{new} = 0.6$,$\delta_{new} = 0.8$,$\delta_{new} = 1$

k that distorts the representation, and a high k that allows different states to be close with each other. With a high k, the L2 rewards may admit several local optimas.

In Figure 5.12, we see that the distortion threshold δ, which prevents consecutive embeddings to be equal in Equation 5.7, also impacts the distortion of the representation, however it mostly limits the compression of the representation in the areas the agent often interacts in. 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. δ limits such asymmetries and keeps large rooms dilated. Overall, this asymmetry also occurs 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.

The size of the clusters of OEGN has to match the distortion of the representation. Figure 5.14 emphasizes the importance of the creation threshold parameter δ_{new} that rules the radius of cluster in Section 5.2.2. 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.

Selection of interactions: Figure 5.15 shows the importance of the different hyper-parameters that rule the sampling of goals and states in Equation 5.13 and Equation 5.10. In Ant environment, we see that the agent has to sample a small ratio of learning interactions from π^{high} rather than $p_{\alpha_{skew}}$; it speeds up its learning process by making it focus on important interactions relatively to extrinsic rewards. Otherwise, it remains hard to learn all skills at the same time. However, the learning process becomes unstable if it deterministically samples from a very small set of clusters (ratios 0.9 and 0.7).

Then we evaluate the importance of $1 + \alpha'_{skew}$ on *Visual Pusher*. We see that the agent learns quicker with a low $1 + \alpha'_{skew}$. It hardly learns when the high-level policy almost does not over-

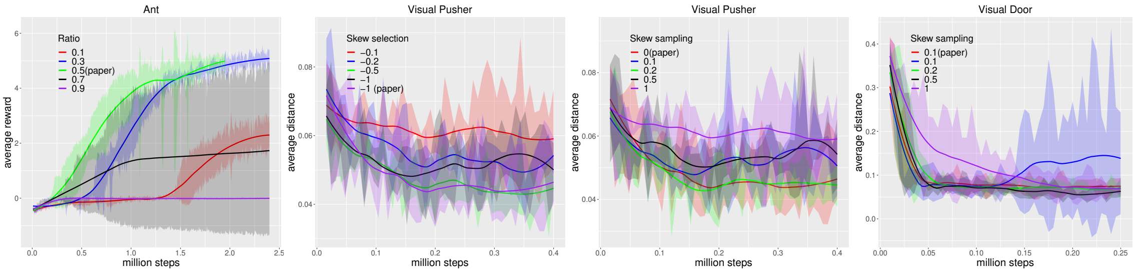


Figure 5.15: Different learning curves showing the impact of the choice of interactions. 1- We study the impact choosing learning interactions with π^{high} rather than $p_{\alpha_{skew}}$. 2- we study the importance of $1+\alpha'_{skew}$ in *Visual Pusher*. 3 and 4- we assess the importance of $1+\alpha_{skew}$ in *Visual Pusher* and *Visual Door*.

sample low-density clusters (-0.1). This makes our results consistent with the analysis provided in the paper of Skew-Fit [Pong et al., 2020].

In the last two graphics of Figure 5.15, we show the impact of $1+\alpha_{skew}$; we observe that the agent learns quicker when $1+\alpha_{skew}$ is close to 0. It highlights that the agent quicker learns both a good representation and novel skills by sampling uniformly over the clusters.

Overall, we also observe that some seeds become unstable since they coincide with large deletions of clusters. We expect that an hyper-parameter search on the delete operators of OEGN may solve this issue in these specific cases.

5.5 Related works

In this section, we explain how our work relates to previous approaches.

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., 2018, Levy et al., 2019]. These approaches do not generalize well with high-dimensional states. To address this, one may want to generate rewards with a learnt representation of goals. NOR [Nachum et al., 2019a] bounds the sub-optimality of such representation to solve a task and LESSON [Li et al., 2021b] uses a slow dynamic heuristic to learn the representation. In fact, it uses an InfoNCE-like objective function; this is similar to [Lu et al., 2019] which learns the representation during pre-training with random walks. Above-mentioned methods use a hierarchical random walk to explore the environment, we have shown in Section 5.3 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.

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. Some of them maximize the mutual information (MI) between a set of states and skills. Typically, DIAYN [Eysenbach et al., 2018], VALOR [Achiam et al., 2018] and SNN [Florensa et al., 2017] learn a discrete set of skills, but hardly generalize over skills. It has been further extended to continuous set 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. Unlike us, they use the true goal to generate positive pairs and a L2 distance over pixels to define a strategy that improves the diversity of skills. In addition, unlike VAE-based models, our method better scales to any ground state space (see Section 5.3). Typically, RIG [Nair et al., 2018] uses a VAE [Kingma and Welling, 2014] to compute a goal representation before training the goal-conditioned policies. Using a VAE, it is possible to 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., 2020] further extended RIG to improve the diversity of learnt skills by making the VAE over-weight low-density states. Unlike DisTop, it is unclear how Skew-Fit could target another distribution over states than a uniform one. Approaches based on learning progress (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 avoid distractors or further improve skill selection.

Skill discovery for end-to-end exploration. Like DisTop, ELSIM (cf. Chapter 4) 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 5.3). 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 distribution (with high-density in rewarding areas). Yet, either they approximate a 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., 2018b,a, Eysenbach et al., 2019] consider a topological map over direct observations, but to give flat intrinsic rewards or make planning possible. 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 and 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 standard temporal contrastive losses use the 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].

5.6 Conclusion

We introduced a new model, DisTop, that simultaneously learns a discrete topology of its environment and the skills that navigate into it. In contrast with previous approaches [Pitis et al., 2020, Pong et al., 2020], there is no pre-training, particular scheduling [Pong et al., 2020] or random walks [Lu et al., 2019]. It manages to select whether it wants to forget a skill, does not need a well-structured goal space like Pitis et al. [2020] or dense rewards as required by Li et al. [2021b]. Our main take-away message is as follow: **computing a discrete topology of the environment allows to control which skills to forget, improve or explore. With this control capacity, DisTop is generic enough to compete with SOTA algorithms on three very different reward settings and state spaces.** Yet, there are limitations and exciting perspectives: HRL and planning based approaches [Nasiriany et al., 2019] could both take advantage of the topology and make easier states discovery; Frontier-based exploration [Bharadhwaj et al., 2020] could also be explored to reduce skill stochasticity. Disentangling the topology [Bengio et al., 2013] could improve the scalability of the approach: currently, the number of created cluster may exponentially grow with respect to the number of independent factors.

Chapter 6

Conclusion

The concept of intrinsic motivation has been introduced in the psychological literature to describe the tendency of organisms to actively understand the structure of the world. Intrinsic motivation is able to describe parts of complex organism behaviors, highlighting its potential interest to develop agents that learn and grow like humans. On an other side, the computational framework of DRL has recently shown substantial improvements in being able to provide learning agents that can achieve complex tasks in difficult environments. Yet there are critical limitations in standard DRL: an agent requires handmade feedback to guide its learning process. This dissertation aimed to study how intrinsic motivations can complement DRL.

6.1 Synthesis of contributions

The first part of our work (Chapter 2) consisted in describing the framework of DRL and defining intrinsic motivation from a computational perspective. This way, we then managed to highlight that DRL and intrinsic motivation can be theoretically used together through goal-conditioned policy, hierarchical RL and information theory. This provided the basis of a computational framework for developmental DRL. Having these key elements in hand, Chapter 3 studied on-going research in this domain under the perspective of information theory. We proposed/reconsidered mathematical information theoretic definitions of the notions of surprise seeking, novelty seeking, and skill learning. This way, this can serve as a categorization to encompass a large body of works that tackle DRL issues like exploration in sparse rewards or skill abstraction.

Our analysis stressed out the interest of learning abstract skills: it allows surprise and novelty seeking behaviors to avoid catastrophic forgetting, and thus, avoid the collapse of the exploration process. Learning more hierarchies of abstract skills could also be crucial to imitate the hierarchy of skills learnt by biological organisms. However, we noticed that, to learn deep hierarchies of skill, an agent must be able to focus its skill discovery on what interests its upper-level skills. The analogy with humans clearly testifies this: there is no need for all humans to learn all gymnastic skills, most humans only learn the main locomotion skills like walking, turning or running. To be able to select what to improve, the agent must simultaneously select the skills it wants to improve and to learn them.

This motivated the rest of the dissertation. In Chapter 4, we introduced ELSIM, which is a method that learns skills in a bottom-up way while being able to select which skill to improve according to extrinsic rewards. In particular, ELSIM creates a tree of discrete skills that either expands widthwise when the agent wants to explore or expands depthwise if the agent knows which skills it wants to improve according to extrinsic feedbacks. We showed that ELSIM improves transfer learning and exploration when rewards are sparse in comparison with a SOTA DRL algorithm. Despite the properties of ELSIM, we also evidended that ELSIM is unable to efficiently explore and can not easily integrate a deep hierarchy of DRL agents. In addition, the tree of skills can not capture the graph-based structure of an environment.

To tackle the challenges of ELSIM, we proposed another model, DisTop (Chapter 5). DisTop learns a continuous representation of goal-states that matches the topological structure of the environment. This representation is then discretized into a graph. Thanks to this structured representation, DisTop selects which node is worth to be improved according to both exploration and extrinsic rewards. If it needs to focus on particular skills, it also forgets uninteresting skills. Therefore, DisTop enjoys the same properties than ELSIM, but using a continuous representation of goals-states. This makes DisTop eligible for an integration in a deep hierarchy of DRL agents.

In the following section, we try to take a step back to project ourselves on what remains to be done in the domain of intrinsic motivation.

6.2 Outlooks of research

Our works focused on analyzing the potential of intrinsic motivation to build open-ended learning agents. We proposed two models that can learn to achieve intrinsically generated goals while learning to achieve a task in an end-to-end way. In this section, we discuss the lessons from our works on two time-scales: 1- the short-term perspective and 2- the long-term perspective.

6.2.1 On learning a representation of the environment

Here, we focus on the short-term perspectives based on lessons we learnt about our survey analysis and our models DisTop and ELSIM.

ELSIM and DisTop only consider one hierarchy of skills. Typically, the agent selects a goal (in the graph or tree) and it executes the corresponding skill in the environment. We argued in Section 3.6.1 for the need of deep composite hierarchies of skills and we hope that further works will quantitatively exhibit the interest of using several hierarchies of skills that correspond to different temporal scales. But, as highlighted in Section 3.5, the nature of skills essentially depends on the learnt representation of states that underpins the reward function. Currently, there are clearly two distinct approaches, one based on VAE [Pong et al., 2020] and the other based on temporal (and marginally data-augmentation based for images) contrastive losses (cf. Chapter 5). In the first set of methods, the disentangled representation opens a large set of possibilities for affordance learning [Khazatsky et al., 2021], while contrastive losses capture the temporal proximity whatever the structure of the ground state space is. We perceive two challenges for active contrastive representation learning:

Clarifying actual losses. Contrastive losses are the subject of intensive research [Chen et al., 2020, Srinivas et al., 2020, Schwarzer et al., 2020]. In particular, several works (including DisTop) introduce temporally contrastive losses with different structures [Li et al., 2021b, Oord et al., 2018, Warde-Farley et al., 2019, Nachum et al., 2019a], [Shu et al., 2020, Ermolov and Sebe, 2020, Tao et al., 2020, Yarats et al., 2021] and non-contrastive losses also show similar properties [Choi et al., 2021, Zhang et al., 2020a]. Their specific properties are currently unclear since each representation is sometimes learnt in a different context for a different purpose. The same mutual information objective can lead to a spherically distributed representation [Ermolov and Sebe, 2020], a distorted one Li et al. [2021b], an undistorted one (like DisTop, cf. Chapter 5), action-dependent ones [Nachum et al., 2019a] and our preliminary analysis highlights that some losses do not dynamically adapt [Choi et al., 2021]. Studying the properties of these representations under one common framework could enlighten the path of representation learning.

Disentangling the representation. The biggest challenge for contrastive losses may be to be able to disentangle this representation to better capture the causal model of the world. This is consistent with our analysis of the difficulty of DisTop to tackle environments with several factors of variation. In the case of DisTop, such disentangled continuous representation could open the path for a disentangled discrete representation, *i.e* a disentangled topology of the environment. In this case, contrastive loss may go beyond the VAE approach.

6.2.2 Multi-information as a universal guiding princple

We discussed in Section 3.7 how we can unify intrinsic motivations with multi-information over a hierarchical agent's model. Furthermore, we shortly argued that this single objective could be a plausible candidate for being a universal guiding principle that rules action selection and perception in organisms. In addition, the multi-information may clarify the role of ontogenetic learning and phylogenetic learning. Ontogenetic learning may refer to the multi-information maximization while phylogenetic learning could refer to learning the overall structure of the brain, considering the trade-off between processing information and minimizing the required energy, *i.e* following the parsimony principle [Polani, 2009, Polani et al., 2007].

That is a very strong hypothesis that deserves to be verified since it could open the path to the derivation of precise objective functions. In fact, this principle transforms the problem of proposing a learning objective into the problem of proposing a cognitive architecture composed of elementary components and inductive biases. Fortunately, such proposals can be enlightened with the help of neurobiology, in contrast to the problem of proposing objectives, which hardly takes inspiration from neuroscience. Briefly, this may strengthen the relation between the domain of neurobiology and machine learning. In results, our recipe may directly tackle the actual problem of erratically proposing a plethora of objectives for an agent.

To illustrate this aspect, let us naturally ask the following question: what does multi-information teach us about the objective function of DisTop ? Let us discuss the three objectives derived from multi-information:

Novelty could improve the exploration process by inciting to go into badly represented areas of the state space.

Surprise may allow DisTop to target improvable skills, this includes skills to improve in sparse reward areas, but most importantly, it would make DisTop avoid purely stochastic areas where the agent could not learn skills. We currently avoid such setting in our experiments.

Skill learning: the skill learning term of the multi-information suggests to maximize $I(R'; G^1|R, O')$ rather than $I(R'; G^1)$ for the goal-conditioned policy. Thus, DisTop agent should learn skills that impose a *change* in the representation rather than skills that target a particular area of the environment. Along with surprise, the agent may also avoid making skills that can not be achieved in particular states.

Overall, even though the discrete topological representation has several advantages presented in Chapter 5, this suggests that DisTop could also benefit from using objective functions as the one of HESS [Li et al., 2021a] and NOR [Nachum et al., 2019a], which integrate skills as changes in the representation space, incoroporate novelty-seeking behaviors and autonomously compute temporally coherent features. This example ilustrates how instrinsic motivation as a guiding principle could help us to define future objectives.

6.3 Conclusion

Developmental approaches attract a lot of attention, because of the ability of the theory to explain the development process of humans [Nguyen et al., 2021]. Our pathway is composed of two parts: firstly, we argued that information-theoretic intrinsic motivations in DRL and lifelong learning are plausible candidates for a computational theory of developmental learning, despite the fact that they were currently mostly addressed separately. Secondly, we introduced two models that allow to integrate the lifelong learning paradigm in works on intrinsic motivation, it follows that choosing which skills to keep and deepen makes the learning models generic to several types of tasks. This may be a key aspect to integrate skill discovery in a multi-level hierarchy of skills. To go further in this direction, we argued for the need of further evidences showing that our unification of intrinsic motivations based on the multi-information can be a universal guiding principle of autonomous development. This may provide a methological way to keep exploring the path towards human-like learning agents.

Bibliography

David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *ICML*, pages 2915–2923, 2016.

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Riad Akrour, Filipe Veiga, Jan Peters, and Gerhard Neumann. Regularizing reinforcement learning with state abstraction. In *IROS*, pages 534–539. IEEE, 2018.

Luís B Almeida. Misep–linear and nonlinear ica based on mutual information. *Journal of Machine Learning Research*, 4(Dec):1297–1318, 2003.

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

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*, pages 8769–8782, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183, 1954.

Arthur Aubret, Laëtitia Matignon, and Salima Hassas. Elsim: End-to-end learning of reusable skills through intrinsic motivation. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2020.

Arthur Aubret, Salima Hassas, et al. Distop: Discovering a topological representation to learn diverse and rewarding skills. *arXiv preprint arXiv:2106.03853*, 2021.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Tuning bandit algorithms in stochastic environments. In *International conference on algorithmic learning theory*, pages 150–165. Springer, 2007.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Nihat Ay and Daniel Polani. Information flows in causal networks. *Advances in complex systems*, 11(01):17–41, 2008.

Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo Avila Pires, Jean-Bastien Grill, Florent Altché, and Rémi Munos. World discovery models. *arXiv preprint arXiv:1902.07685*, 2019.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, et al. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2019.

Gianluca Baldassarre. What are intrinsic motivations? a biological perspective. In *2011 IEEE international conference on development and learning (ICDL)*, volume 2, pages 1–8. IEEE, 2011.

Gianluca Baldassarre. Intrinsic motivations and open-ended learning. *arXiv preprint arXiv:1912.13263*, 2019.

Gianluca Baldassarre and Marco Mirolli. Intrinsically motivated learning systems: an overview. In *Intrinsically motivated learning in natural and artificial systems*, pages 1–14. Springer, 2013a.

Gianluca Baldassarre and Marco Mirolli. Intrinsically motivated learning systems: An overview. In Gianluca Baldassarre and Marco Mirolli, editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 1–14. Springer, 2013b.

Gianluca Baldassarre, Tom Stafford, Marco Mirolli, Peter Redgrave, Richard M Ryan, and Andrew Barto. Intrinsic motivations and open-ended development in animals, humans, and robots: an overview. *Frontiers in psychology*, 5:985, 2014.

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

Adrien Baranes and Pierre-Yves Oudeyer. R-iac: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169, 2009.

Adrien Baranes and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration for active motor learning in robots: A case study. In *IROS*, pages 1766–1773, 2010.

David Barber and Felix V Agakov. The im algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, pages 201–208, 2003.

Andrew Barto, Marco Mirolli, and Gianluca Baldassarre. Novelty or surprise? *Frontiers in psychology*, 4:907, 2013.

Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 112–19, 2004.

Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. Relative variational intrinsic control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6732–6740, 2021.

Philip Becker-Ehmck, Maximilian Karl, Jan Peters, and Patrick van der Smagt. Exploration via empowerment gain: Combining novelty, surprise and learning progress. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

Marc Bellemare, Joel Veness, and Erik Talvitie. Skip context tree switching. In *International Conference on Machine Learning*, pages 1458–1466, 2014.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents (extended abstract). In *IJCAI*, pages 4148–4152. AAAI Press, 2015.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

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.

Daniel E Berlyne. Conflict, arousal, and curiosity. 1960.

Daniel E Berlyne. Structure and direction in thinking. 1965.

Daniel E Berlyne. Curiosity and exploration. *Science*, 153(3731):25–33, 1966.

Glen Berseth, Daniel Geng, Coline Devin, Nicholas Rhinehart, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. Smirl: Surprise minimizing rl in dynamic environments. 2020.

Lucas Beyer, Damien Vincent, Olivier Teboul, Sylvain Gelly, Matthieu Geist, and Olivier Pietquin. Mulex: Disentangling exploitation from exploration in deep rl. *arXiv preprint arXiv:1907.00868*, 2019.

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

David M Blei and Michael I Jordan. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

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.

Matthew M Botvinick. Hierarchical models of behavior and prefrontal function. *Trends in cognitive sciences*, 12(5):201–208, 2008.

Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

Diane Bouchacourt and Marco Baroni. Miss tools and mr fruit: Emergent communication in agents learning about object affordances. *arXiv preprint arXiv:1905.11871*, 2019.

David A. Brannan, Matthew F. Esplen, and Jeremy J. Gray. *Geometry*. Cambridge University Press, 1999. doi: 10.1017/CBO9780511807503.

Rodney A Brooks. Elephants don't play chess. *Robotics and autonomous systems*, 6(1-2):3–15, 1990.

Rodney A Brooks. Intelligence without reason. 1991.

Neil Bruce and John Tsotsos. Saliency based on information maximization. In *Advances in neural information processing systems*, pages 155–162, 2005.

Neil DB Bruce and John K Tsotsos. Saliency, attention, and visual search: An information theoretic approach. *Journal of vision*, 9(3):5–5, 2009.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

Martin V Butz. Toward a unified sub-symbolic computational theory of cognition. *Frontiers in psychology*, 7:925, 2016.

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.

Angelo Cangelosi and Matthew Schlesinger. From babies to robots: the contribution of developmental robotics to developmental psychology. *Child Development Perspectives*, 12(3):183–188, 2018.

Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. Emergent communication through negotiation. In *International Conference on Learning Representations*, 2018.

Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*, pages 6284–6293, 2017.

Benjamin Paul Chamberlain, James R. Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *CoRR*, abs/1705.10359, 2017.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in vaes. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2615–2625, 2018.

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

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

Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, 2019.

Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. Variational empowerment as representation learning for goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pages 1953–1963. PMLR, 2021.

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, pages 539–546. IEEE, 2005.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32:1787–1798, 2019.

Paul Cisek. Cortical mechanisms of action selection: the affordance competition hypothesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1485):1585–1599, 2007.

Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

John D. Co-Reyes, Yuxuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In Jennifer G. Dy and Andreas Krause, editors, *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*, pages 1008–1017. PMLR, 2018.

Cédric Colas, Pierre-Yves Oudeyer, Olivier Sigaud, Pierre Fournier, and Mohamed Chetouani. Curious: Intrinsically motivated modular multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 1331–1340, 2019.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems*, 33, 2020a.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey. *arXiv preprint arXiv:2012.09830*, 2020b.

Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5032–5043, 2018.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Paul Dagum, Adam Galper, and Eric Horvitz. Dynamic network models for forecasting. In *Uncertainty in artificial intelligence*, pages 41–48. Elsevier, 1992.

P Dayan and GE Hinton. Feudal reinforcement learning. nips'93 (pp. 271–278), 1993.

Peter Dayan and Geoffrey E Hinton. Varieties of helmholtz machine. *Neural Networks*, 9(8):1385–1403, 1996.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Edward L Deci and Richard M Ryan. Intrinsic motivation. *The corsini encyclopedia of psychology*, pages 1–2, 2010.

Dobromir G Dotov, Lin Nie, and Matthieu M De Wit. Understanding affordances: History and contemporary development of gibson's central concept. *Avant: the journal of the philosophical-interdisciplinary vanguard*, 2012.

Yilun Du, Chuang Gan, and Phillip Isola. Curious representation learning for embodied intelligence. *arXiv preprint arXiv:2105.01060*, 2021.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Samuel Eckmann, Lukas Klimmasch, Bertram E Shi, and Jochen Triesch. Active efficient coding explains the development of binocular vision and its failure in amblyopia. *Proceedings of the National Academy of Sciences*, 117(11):6156–6162, 2020.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *CoRR*, abs/1901.10995, 2019. URL http://arxiv.org/abs/1901.10995.

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

Jeffrey A Edlund, Nicolas Chaumont, Arend Hintze, Christof Koch, Giulio Tononi, and Christoph Adami. Integrated information increases with fitness in the evolution of animats. *PLoS Comput Biol*, 7(10):e1002236, 2011.

Paul Ed Ekman and Richard J Davidson. *The nature of emotion: Fundamental questions.* Oxford University Press, 1994.

Edward J Engelken and Kennith W Stevens. Saccadic eye movements in response to visual, auditory, and bisensory stimuli. *Aviation, space, and environmental medicine*, 1989.

Aleksandr Ermolov and Nicu Sebe. Latent world models for intrinsically motivated exploration. *Advances in Neural Information Processing Systems*, 33, 2020.

Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, Narayanaswamy Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2525–2534. PMLR, 2019.

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, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15220–15231, 2019.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *CoRR*, abs/1802.06070, 2018.

Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47, 1991.

Maria Laura Filippetti, Mark H Johnson, Sarah Lloyd-Fox, Danica Dragovic, and Teresa Farroni. Body perception in newborns. *Current Biology*, 23(23):2413–2416, 2013.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

Tamar Flash and Binyamin Hochner. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology*, 15(6):660–666, 2005.

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*, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1514–1523, 2018.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11 (3-4):219–354, 2018.

Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM (TOMS)*, 3(3):209–226, 1977.

Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. Multivariate information bottleneck. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 152–161, 2001. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=95&proceeding_id=17.

Karl Friston. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456):815–836, 2005.

Karl Friston. The free-energy principle: a rough guide to the brain? *Trends in cognitive sciences*, 13 (7):293–301, 2009.

Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2): 127, 2010.

Karl Friston. Life as we know it. *Journal of the Royal Society Interface*, 10(86):20130475, 2013.

Karl Friston, James Kilner, and Lee Harrison. A free energy principle for the brain. *Journal of physiology-Paris*, 100(1-3):70–87, 2006.

Karl Friston, Rick Adams, Laurent Perrinet, and Michael Breakspear. Perceptions as hypotheses: saccades as experiments. *Frontiers in psychology*, 3:151, 2012a.

Karl Friston, Christopher Thornton, and Andy Clark. Free-energy minimization and the dark-room problem. *Frontiers in psychology*, 3:130, 2012b.

Karl Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas Fitzgerald, and Giovanni Pezzulo. Active inference and epistemic value. *Cognitive neuroscience*, 6(4):187–214, 2015.

Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, Giovanni Pezzulo, et al. Active inference and learning. *Neuroscience & Biobehavioral Reviews*, 68:862–879, 2016.

Karl J Friston, Jean Daunizeau, and Stefan J Kiebel. Reinforcement learning or active inference? *PloS one*, 4(7):e6421, 2009.

Karl J Friston, Thomas Parr, and Bert de Vries. The graphical brain: belief propagation and active inference. *Network Neuroscience*, 1(4):381–414, 2017.

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

Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2577–2587, 2017.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5350–5360, 2018.

Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. Auto-encoding total correlation explanation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1157–1166. PMLR, 2019.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal conditioned policies. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hye9lnCct7.

James J Gibson. The theory of affordances. *Hilldale, USA*, 1(2):67–82, 1977.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Alison Gopnik, Andrew N Meltzoff, and Patricia K Kuhl. *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co, 1999.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016.

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

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

Frank Guerin. Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review*, 26(2):209–236, 2011.

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.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *CoRR*, abs/1806.04640, 2018.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

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.

Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

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

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. *arXiv preprint arXiv:1510.09142*, 2015.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Takao K Hensch. Critical period regulation. *Annu. Rev. Neurosci.*, 27:549–579, 2004.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR, 2017.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=Bklr3j0cKX.

Joachim Hoffmann, Michael Berner, Martin V Butz, Oliver Herbort, Andrea Kiesel, Wilfried Kunde, and Alexandra Lenhard. Explorations of anticipatory behavioral control (abc): A report from the cognitive psychology unit of the university of würzburg. *Cognitive Processing*, 8(2):133–142, 2007.

Bernhard Hommel. The theory of event coding (tec) as embodied-cognition framework. *Frontiers in Psychology*, 6:1318, 2015.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

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

Riashat Islam, Raihan Seraj, Pierre-Luc Bacon, and Doina Precup. Entropy regularization with discounted future state distribution in policy gradient methods. *arXiv preprint arXiv:1912.05104*, 2019.

Laurent Itti and Pierre Baldi. Bayesian surprise attracts human attention. *Vision research*, 49(10): 1295–1306, 2009.

Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 10519–10530, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049, 2019.

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.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Sham Kakade and Peter Dayan. Dopamine: generalization and bonuses. *Neural Networks*, 15(4-6): 549–559, 2002.

Alexander Khazatsky, Ashvin Nair, Daniel Jing, and Sergey Levine. What can i do here? learning new skills by imagining visual affordances. *arXiv preprint arXiv:2106.00671*, 2021.

Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020.

Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. EMI: Exploration with mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3360–3369, Long Beach, California, USA, 09–15 Jun 2019a. PMLR. URL http://proceedings.mlr.press/v97/kim19a.html.

Jaekyeom Kim, Seohong Park, and Gunhee Kim. Unsupervised skill discovery with bottleneck option learning. In *International Conference on Machine Learning*, pages 5572–5582. PMLR, 2021.

Kuno Kim, Megumi Sano, Julian De Freitas, Nick Haber, and Daniel Yamins. Active world model learning with progress curiosity. In *International conference on machine learning*, pages 5306–5315. PMLR, 2020.

Youngjin Kim, Wontae Nam, Hyunwoo Kim, Ji-Hoon Kim, and Gunhee Kim. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International Conference on Machine Learning*, pages 3379–3388, 2019b.

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

Martin Klissarov, Riashat Islam, Khimya Khetarpal, and Doina Precup. Variational state encoding as intrinsic motivation in reinforcement learning. 2019.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Organization of the information flow in the perception-action loop of evolved agents. In *Proceedings. 2004 NASA/DoD Conference on Evolvable Hardware, 2004.*, pages 177–180. IEEE, 2004a.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Tracking information flow through the environment: Simple cases of stigmerg. In *In: Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, edited by Pollack, J.* MIT Press, 2004b.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 128–135. IEEE, 2005.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Representations of space and time in the maximization of information flow in the perception-action loop. *Neural computation*, 19 (9):2387–2432, 2007.

David C Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719, 2004.

Christof Koch and Gilles Laurent. Complexity and the nervous system. *Science*, 284(5411):96–98, 1999.

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

Varun Raj Kompella, Marijn Stollenga, Matthew Luciw, and Juergen Schmidhuber. Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247:313–335, 2017.

Jürgen Konczak. Neural development and sensorimotor control. *Available at SSRN 3075656*, 2004.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International Conference on Machine Learning*, pages 1863–1871. PMLR, 2014.

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.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arXiv preprint arXiv:1807.01521*, 2018.

Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pages 3672–3681. PMLR, 2019.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

Joel Lehman and Kenneth O Stanley. Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*, pages 37–56. Springer, 2011.

Alexander Lelais, Jonas Mahn, Vikram Narayan, Chong Zhang, Bertram E Shi, and Jochen Triesch. Autonomous development of active binocular and motion vision through active efficient coding. *Frontiers in neurorobotics*, 13:49, 2019.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 2018.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

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

Alexander C. Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a.

Alexander C. Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

Siyuan Li, Jin Zhang, Jianhao Wang, and Chongjie Zhang. Efficient hierarchical exploration with stable subgoal representation learning. *arXiv preprint arXiv:2105.14750*, 2021a.

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

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Jessica Lindblom and Tom Ziemke. Social situatedness: Vygotsky and beyond. 2002.

Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

Cam Linke, Nadia M Ady, Martha White, Thomas Degris, and Adam White. Adapting behavior via intrinsic reward: A survey and empirical study. *Journal of Artificial Intelligence Research*, 69: 1287–1332, 2020.

Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Ralph Linsker. Perceptual neural organization: Some approaches based on network models and information theory. *Annual review of Neuroscience*, 13(1):257–281, 1990.

Daniel Ying-Jeh Little and Friedrich Tobias Sommer. Learning and exploration in action-perception loops. *Frontiers in neural circuits*, 7:37, 2013.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Damiano Lombardi and Sanjay Pant. Nonparametric k-nearest-neighbor entropy estimator. *Physical Review E*, 93(1):013310, 2016.

Luca Lonini, Sébastien Forestier, Céline Teulière, Yu Zhao, Bertram E Shi, and Jochen Triesch. Robust active binocular vision through intrinsically motivated learning. *Frontiers in neurorobotics*, 7:20, 2013.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pages 206–214, 2012.

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

Max Lungarella and Olaf Sporns. Information self-structuring: Key principle for learning and development. In *Proceedings. The 4th International Conference on Development and Learning, 2005*, pages 25–30. IEEE, 2005.

Max Lungarella and Olaf Sporns. Mapping information flow in sensorimotor networks. *PLoS computational biology*, 2(10), 2006.

Max Lungarella, Teresa Pegors, Daniel Bulwinkle, and Olaf Sporns. Methods for quantifying the informational structure of sensory and motor data. *Neuroinformatics*, 3(3):243–262, 2005.

Marios C Machado, Marc G Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2295–2304. JMLR. org, 2017.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. *arXiv preprint arXiv:1807.11622*, 2018.

Donna L Maier, Shyamala Mani, Stacy L Donovan, Dan Soppet, Lino Tessarollo, James S McCasland, and Karina F Meiri. Disrupted cortical map and absence of cortical barrels in growth-associated protein (gap)-43 knockout mice. *Proceedings of the National Academy of Sciences*, 96(16):9397–9402, 1999.

Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. Adaptive skills adaptive partitions (asap). In *Advances in Neural Information Processing Systems*, pages 1588–1596, 2016.

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

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

Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2471–2478, 2017. doi: 10.24963/ijcai.2017/344.

Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincar\'e variational auto-encoders. *arXiv preprint arXiv:1901.06033*, 2019.

Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. 2001.

Alicia P Melis and Dirk Semmann. How is human cooperation different? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1553):2663–2674, 2010.

Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*, pages 295–306. Springer, 2002.

Jan Hendrik Metzen and Frank Kirchner. Incremental learning of skill collections based on intrinsic motivation. *Frontiers in neurorobotics*, 7:11, 2013.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Andrew William Moore. Efficient memory-based learning for robot control. 1990.

David Mumford. On the computational architecture of the neocortex. *Biological cybernetics*, 66 (3):241–251, 1992.

Ferdinando A Mussa-Ivaldi and Emilio Bizzi. Motor learning through the combination of primitives. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 355 (1404):1755–1769, 2000.

Ferdinando A Mussa-Ivaldi and Sara A Solla. Neural primitives for motion control. *IEEE Journal of Oceanic Engineering*, 29(3):640–650, 2004.

Mirco Mutti, Lorenzo Pratissoli, and Marcello Restelli. A policy gradient method for task-agnostic exploration. 2020.

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, editors, *Advances in Neural Information Processing Systems 31*, pages 3303–3313. 2018.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019a.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019b.

Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *International Conference on Machine Learning*, pages 4693–4702. PMLR, 2019.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 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, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14814–14825, 2019.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

Phuong DH Nguyen, Yasmin Kim Georgie, Ezgi Kayhan, Manfred Eppe, Verena Vanessa Hafner, and Stefan Wermter. Sensorimotor representation learning for an “active self” in robots: a model survey. *KI-Künstliche Intelligenz*, 35(1):9–35, 2021.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6338–6347, 2017.

Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

Hans-Christoph Nothdurft. Salience and target selection in visual search. *Visual Cognition*, 14(4-8): 514–542, 2006.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2721–2730, 2017. URL `http://proceedings.mlr.press/v70/ostrovski17a.html`.

Pierre-Yves Oudeyer and Frederic Kaplan. How can we define intrinsic motivation? In *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund: LUCS, Brighton*. Lund University Cognitive Studies, Lund: LUCS, Brighton, 2008.

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

Pierre-Yves Oudeyer and Linda B Smith. How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, 8(2):492–502, 2016.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

Pierre-Yves Oudeyer, Adrien Baranes, and Frédéric Kaplan. Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints. In Gianluca Baldassarre and Marco Mirolli, editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 303–365. Springer, 2013.

Aldo Pacchiano, Philip Ball, Jack Parker-Holder, Krzysztof Choromanski, and Stephen Roberts. On optimism in model-based reinforcement learning. *arXiv preprint arXiv:2006.11911*, 2020.

Daniel P Palomar and Sergio Verdú. Lautum information. *IEEE transactions on information theory*, 54(3):964–975, 2008.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pages 5062–5071, 2019.

Judea Pearl. Causal inference. *Causality: Objectives and Assessment*, pages 39–58, 2010.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006.

Jean Piaget and Margaret Cook. *The origins of intelligence in children*, volume 8. International Universities Press New York, 1952.

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*, pages 7750–7761. PMLR, 2020.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Daniel Polani. Information: currency of life? *HFSP journal*, 3(5):307–316, 2009.

Daniel Polani, Olaf Sporns, and Max Lungarella. How information and embodiment shape intelligent information processing. In *50 years of artificial intelligence*, pages 99–111. Springer, 2007.

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*, pages 7783–7792. PMLR, 2020.

Christopher Prince, Nathan Helder, and George Hollich. Ongoing emergence: A core concept in epigenetic robotics. 2005.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Tom Quick, Kerstin Dautenhahn, Chrystopher L Nehaniv, and Graham Roberts. On bots and bacteria: Ontology independent embodiment. In *European Conference on Artificial Life*, pages 339–343. Springer, 1999.

R Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

Sebastien Racaniere, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*, 2019.

Roberta Raileanu and Tim Rocktaschel. RIDE: rewarding impact-driven exploration for procedurally-generated environments. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Carl Edward Rasmussen et al. The infinite gaussian mixture model. In *NIPS*, volume 12, pages 554–560, 1999.

Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.

Philippe Rochat. Self-perception and action in infancy. *Experimental brain research*, 123(1-2):102–109, 1998.

Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 1, pages 35–40. IEEE, 1999.

Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1 (1):14–24, 2010.

Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Changing the environment based on empowerment as intrinsic motivation. *Entropy*, 16(5):2789–2819, 2014a.

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

Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011.

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, 2018a. URL https://openreview.net/forum?id=SygwwGbRW.

Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018b.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

Jürgen Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991.

Jürgen Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In *Workshop on Anticipatory Behavior in Adaptive Learning Systems*, pages 48–76. Springer, 2008.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

Philipp Schwartenbeck, Johannes Passecker, Tobias U Hauser, Thomas HB FitzGerald, Martin Kronbichler, and Karl J Friston. Computational mechanisms of curiosity and goal-directed exploration. *eLife*, 8:e41703, 2019.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

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)*, pages 1134–1141. IEEE, 2018.

Anil K Seth and Gerald M Edelman. Environment and behavior influence the complexity of evolved neural networks. *Adaptive Behavior*, 12(1):5–20, 2004.

Roshan Shariff and Travis Dick. Lunar lander: A continous-action case study for policy-gradient actor-critic algorithms, 2013.

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*, 2020.

Rui Shu, Tung Nguyen, Yinlam Chow, Tuan Pham, Khoat Than, Mohammad Ghavamzadeh, Stefano Ermon, and Hung Bui. Predictive coding for locally-linear control. In *International Conference on Machine Learning*, pages 8862–8871. PMLR, 2020.

Pranav Shyam, Wojciech Jaskowski, and Faustino Gomez. Model-based active exploration. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5779–5788, 2019. URL http://proceedings.mlr.press/v97/shyam19a.html.

Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23(3-4):301–321, 2003.

Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

B. F. Skinner. The behavior of organisms. In *New York: Appleton*, 1938.

Noam Slonim, Nir Friedman, and Naftali Tishby. Agglomerative multivariate information bottleneck. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 929–936, 2001.

Rishi Sonthalia and Anna C. Gilbert. Tree! I am no tree! I am a low dimensional hyperbolic embedding. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Olaf Sporns and Max Lungarella. Evolving coordinated behavior by maximizing information structure. In *Artificial life X: proceedings of the tenth international conference on the simulation and synthesis of living systems*, volume 10, page 323. Citeseer, 2006.

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

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Christopher Stanton and Jeff Clune. Deep curiosity search: Intra-life exploration can improve performance on challenging deep reinforcement learning problems. 2018.

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

Jan Storck, Sepp Hochreiter, Jürgen Schmidhuber, et al. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the international conference on artificial neural networks, Paris*, volume 2, pages 159–164. Citeseer, 1995.

Alexander Stoytchev. Some basic principles of developmental robotics. *IEEE Transactions on Autonomous Mental Development*, 1(2):122–130, 2009.

Hans Strasburger, Ingo Rentschler, and Martin Jüttner. Peripheral vision and pattern recognition: A review. *Journal of vision*, 11(5):13–13, 2011.

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.

Pei-Hao Su, David Vandyke, Milica Gasic, Nikola Mrksic, Tsung-Hsien Wen, and Steve Young. Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems. *arXiv preprint arXiv:1508.03391*, 2015.

Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011.

Raphael Suter, Djordje Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, pages 6056–6065. PMLR, 2019.

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

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

Ruo Yu Tao, Vincent François-Lavet, and Joelle Pineau. Novelty search in representational space for sample efficient exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Samuel F Taylor, Naftali Tishby, and William Bialek. Information and fitness. *arXiv preprint arXiv:0712.4382*, 2007.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, 2017.

Céline Teulière, Sébastien Forestier, Luca Lonini, Chong Zhang, Yu Zhao, Bertram Shi, and Jochen Triesch. Self-calibrating smooth pursuit through active efficient coding. *Robotics and Autonomous Systems*, 71:3–12, 2015.

Guy Theraulaz and Eric Bonabeau. A brief history of stigmergy. *Artificial life*, 5(2):97–116, 1999.

Serge Thill, Daniele Caligiore, Anna M Borghi, Tom Ziemke, and Gianluca Baldassarre. Theories and computational models of affordance and mirror systems: an integrative review. *Neuroscience & Biobehavioral Reviews*, 37(3):491–521, 2013.

Brittany L Thomas, Jenni M Karl, and Ian Q Whishaw. Independent development of the reach and the grasp in spontaneous self-touching by human infants in the first 6 months. *Frontiers in psychology*, 5:1526, 2015.

Valentin Thomas, Emmanuel Bengio, William Fedus, Jules Pondard, Philippe Beaudoin, Hugo Larochelle, Joelle Pineau, Doina Precup, and Yoshua Bengio. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1802.09484*, 2018.

Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46, 1995.

Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Giulio Tononi and Olaf Sporns. Measuring information integration. *BMC neuroscience*, 4(1):31, 2003.

Giulio Tononi, Gerald M Edelman, and Olaf Sporns. Complexity and coherency: integrating information in the brain. *Trends in cognitive sciences*, 2(12):474–484, 1998.

Hugo Touchette and Seth Lloyd. Information-theoretic approach to the study of control systems. *Physica A: Statistical Mechanics and its Applications*, 331(1-2):140–172, 2004.

Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

William TB Uther and Manuela M Veloso. Tree based discretization for continuous state space reinforcement learning. In *Aaai/iaai*, pages 769–774, 1998.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

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

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018.

Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? *Advances in Neural Information Processing Systems*, 32:14245–14258, 2019.

Francisco J Varela, Evan Thompson, and Eleanor Rosch. *The Embodied Mind, revised edition: Cognitive Science and Human Experience*. MIT press, 2017.

Stephan A Verschoor and Bernhard Hommel. Self-by-doing: The role of action for self-acquisition. *Social Cognition*, 35(2):127–145, 2017.

Giulia Vezzani, Abhishek Gupta, Lorenzo Natale, and Pieter Abbeel. Learning latent state representation for speeding up exploration. *arXiv preprint arXiv:1905.12621*, 2019.

TN Vikram, Céline Teulière, Chong Zhang, Bertram E Shi, and Jochen Triesch. Autonomous learning of smooth pursuit and vergence through active efficient coding. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 448–453. IEEE, 2014.

Claes Von Hofsten and Kerstin Rosander. Development of smooth pursuit tracking in young infants. *Vision research*, 37(13):1799–1810, 1997.

Lev Semenovich Vygotsky. *Mind in society: The development of higher psychological processes*. Harvard university press, 1980.

Jonathan Waddington and Christopher M Harris. Human optokinetic nystagmus: A stochastic analysis. *Journal of Vision*, 12(12):5–5, 2012.

Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. In *International Conference on Learning Representations*, 2019.

Martin Ward. A definition of abstraction. *Journal of Software Maintenance: Research and Practice*, 7(6):443–450, 1995.

David Warde-Farley, Tom Van de Wiele, Tejas 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*, 2019. URL https://openreview.net/forum?id=r1eVMnA9K7.

Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1):66–82, 1960.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P. Burgess, and Alexander Lerchner. COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration. *arXiv e-prints*, art. arXiv:1905.09275, May 2019.

Juyang Weng, James McClelland, Alex Pentland, Olaf Sporns, Ida Stockman, Mriganka Sur, and Esther Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504): 599–600, 2001.

Robert W White. Motivation reconsidered: The concept of competence. *Psychological review*, 66 (5):297, 1959.

Charles Wilmot and Jochen Triesch. Learning abstract representations through lossy compression of multi-modal signals. *arXiv preprint arXiv:2101.11376*, 2021.

Matthew A Wilson and Bruce L McNaughton. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, 1994.

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

Jeremy M Wolfe. Guided search 2.0 a revised model of visual search. *Psychonomic bulletin & review*, 1(2):202–238, 1994.

Jeremy M Wolfe. Visual search. 2015.

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

Yoshiyuki Yamamoto, David W Stock, and William R Jeffery. Hedgehog signalling controls eye degeneration in blind cavefish. *Nature*, 431(7010):844–847, 2004.

Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9593–9602, 2021.

Yao Yao, Li Xiao, Zhicheng An, Wanpeng Zhang, and Dijun Luo. Sample efficient reinforcement learning via model-ensemble exploration and exploitation. *arXiv preprint arXiv:2107.01825*, 2021.

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

Alfred L Yarbus. Eye movements during perception of complex objects. In *Eye movements and vision*, pages 171–211. Springer, 1967.

Jeffrey M Zacks, Nicole K Speer, Khena M Swallow, Todd S Braver, and Jeremy R Reynolds. Event perception: a mind-brain perspective. *Psychological bulletin*, 133(2):273, 2007.

Chong Zhang, Yu Zhao, Jochen Triesch, and Bertram E Shi. Intrinsically motivated learning of visual motion perception and smooth pursuit. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1902–1908. IEEE, 2014.

Chuheng Zhang, Yuanying Cai, Longbo Huang, and Jian Li. Exploration by maximizing renyi entropy for reward-free RL framework. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 10859–10867, 2021.

Jingwei Zhang, Niklas Wetzel, Nicolai Dorka, Joschka Boedecker, and Wolfram Burgard. Scheduled intrinsic drive: A hierarchical take on intrinsically motivated exploration. *arXiv preprint arXiv:1903.07400*, 2019.

Lunjun Zhang, Ge Yang, and Bradly C Stadie. World model as a graph: Learning latent landmarks for planning. *arXiv preprint arXiv:2011.12491*, 2020a.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*, 2020b.

Rui Zhao and Volker Tresp. Curiosity-driven experience prioritization via density estimation. *arXiv preprint arXiv:1902.08039*, 2019.

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

Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual information state intrinsic control. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

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.

Qingpeng Zhu, Jochen Triesch, and Bertram E Shi. Joint learning of binocularly driven saccades and vergence by active efficient coding. *Frontiers in neurorobotics*, 11:58, 2017.

Tom Ziemke. What’s that thing called embodiment? In *Proceedings of the annual meeting of the cognitive science society*, volume 25, 2003.

Jordan Zlatev and Christian Balkenius. Introduction: Why òepigenetic roboticsó? 2001.

List of abbreviations

Abbreviation	Label
DRL	Deep reinforcement learning
RL	Reinforcement learning
IM	Intrinsic motivation
CNN	Convolutional Neural Network
MLP	Multi-layer Perceptron
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
SOTA	State-of-the-art

List of Tables

List of Figures

Appendix A

Complements on ELSIM experiments

A.1 *Tree-policy* algorithm

Algorithm 2 shows how ELSIM runs an episode in the environment without the learning part of the intra-skill policy and the discriminator. There are 3 steps: 1-an agent runs an episode inside the MDP of skills; the sequence of actions represents a skill; 2- the agent executes the intra-skill policy of the skill; 3- the *tree-policy* is rewarded according to how well the intra-skill policy fits the task and the Q-learning applies.

Algorithm 2 *Tree-policy* of ELSIM

```
  Environment $env$, episode length $ep_{len}$, learning rate $\phi$.
  Tree $T$, Tree-policy $\pi_T$.
  Leaves' policies and buffers: $\forall g_{leaves} \in leaves(T),\ \pi_\theta^{g_{leaves}},\ \mathbb{B}^{g_{leaves}}$.
                                                      ▷ It selects a skill to execute
  $node \leftarrow root(T)$
  while $not(leaf(node))$ do
      $node \leftarrow Boltzmann(\pi_T(node))$
  end while
  $r_{tree} \leftarrow 0$
                                       ▷ It runs the intra-skill policy in the environment
  $obs \leftarrow env.reset()$
  $done \leftarrow false$
  while $not(done)$ do
      $action \leftarrow \pi^{node}(obs)$
      $next\_obs,\ reward,\ done \leftarrow env.step(action)$
      Fill $\mathbb{B}^{node}$ with $obs, done, next\_obs, action$
      $obs \leftarrow next\_obs$
      $r_{tree} \leftarrow r_{tree} + \gamma_T \frac{reward}{ep_{len}}$
  end while
                                          ▷ It learns Q-values of the tree-policy
  $parent \leftarrow parent(node)$
  $Q(parent, node) = (1 - lr_T) \times Q(parent, node) + lr_T \times r_{tree}$
  while $not(root(node))$ do
      $parent \leftarrow parent(node)$
      $Q(parent, node) = \max_{n \in children(node)}(Q(node, n))$
  end while
```

A.2 Learning algorithm

Algorithm 3 shows one learning step in ELSIM for discriminators and intra-skill policies. The agent executes its *tree-policy* on its tree; the discriminators of the intermediary encountered nodes learn with probability η on a batch of interactions of their recursively and uniformly chosen children leaves. The last discriminator learns with probability 1 over a batch of its leaves' interactions. This batch is labeled with the intrinsic reward computed through Equation 4.5 and leaves' intra-skill policies learn with this batch.

Algorithm 3 Learning step of intra-skill policies and discriminators

```
$batch\_size$, Tree $T$, Tree-policy $\pi_T$.
Discriminators: $\forall g \in nodes(T),\ q_\omega^g$.
Leaves' policies and buffers: $\forall g_{leaves} \in leaves(T),\ \pi_\theta^{g_{leaves}},\ \mathbb{B}^{g_{leaves}}$.
                                                        ▷ It selects a node to learn on
$node \leftarrow root(T)$
while $not(leaves(children(node)))$ do
    if $random() < \eta$ then
        $i \leftarrow 0$                    ▷ Discriminators in exploitation phase learn with probability $\eta$
        $batch \leftarrow \emptyset$
        while $i < batch\_size$ do
            $node2 \leftarrow node$
            while $not(leaf(node2))$ do
                $node2 \leftarrow Uniform(children(node2))$
            end while
            $ADD(batch, sample(\mathbb{B}^{node2}))$
        end while
        $Cross\_entropy(batch, q_\omega^{node})$
    end if
    $node \leftarrow Boltzmann(\pi_T(node))$
end while
                               ▷ Learn the discriminator and intra-skill policies of the last node
$i \leftarrow 0$
$batch \leftarrow \emptyset$
while $i < batch\_size$ do
    $leaf \leftarrow Uniform(children(node))$
    $ADD(batch, sample(\mathbb{B}^{leaf}))$
end while
$Cross\_entropy(batch, q_\omega^{node})$
for each $leaf \in children(node)$ do
    Learn $\pi_\theta^{leaf}$ with Equation 4.5 and SAC
end for
```

A.3 Hyper-parameters on gridworlds

Table A.1 shows hyperparameters used in gridworlds experiments. Hyper-parameters of the discriminator and DRL networks are identical. *Tree-policy* parameters are used only when there is a high-level goal.

Parameters	Symbol	Values
DRL		
Boltzmann coefficient	α_{DQN}	1
Buffer size	$\mathbb{B}_{size}$	10k
Hidden layers	hl_{SAC}	2x64
Learning rate	lr_{DQN}	0.001
Gamma	γ	0.98
Episode duration	D	100
Batch size	$batch\_size$	64
Parallel environments	n	16
target smoothing coefficient	τ	0.005
ELSIM		
Discriminators hidden layers	lr_D	2x64
Split threshold	δ	0.9
Sampling probability	η	0.5
Average coefficient	β	0.02
Vocabulary size	$\|V\|$	4
Old discriminations scale	α	1
Tree-policy		
Gamma	γ_T	1
Boltzmann coefficient	α_T	20
Learning rate	lr_T	0.05

Table A.1: Hyper-parameters used for gridworld experiments.

A.4 Hyper-parameters on continuous environments.

Since the goal of ELSIM is to learn in a continual learning setting, we record the performances of ELSIM in training mode, *i.e.* with its stochastic policies. Table A.2 shows hyper-parameters used in experiments on continuous environments. Learning rate of the discriminator and DRL networks are identical. In addition to these hyper-parameters, we set the weight decay of discriminators to 0.01 in the exploitation phase. We also manually scale the intra-skill rewards and entropy coefficient α_{SAC} to get lower-magnitude value function, thus, we divide rewards and entropy coefficient by 5 and clamp the minimal reward by (-2). SAC algorithms used by ELSIM do not use a second critic (but our implementation of SAC does).

Classic environments . On MountainCar, Pendulum and LunarLander , we changed α_{sac} to 0.1.

Our implementation of SAC . Our implementation of SAC use the same hyper-parameters as in Haarnoja et al. [2018], but with a buffer size of 100000 and a learning rate of 0.001.

Parameters	Symbol	Values
DRL		
Entropy coefficient	α_{SAC}	0.25
Buffer size	$\mathbb{B}_{size}$	20k
SAC hidden layers	hl_{SAC}	2x128
Learning rate	lr_{SAC}	0.001
Discount	γ	0.98
Episode duration	D	500
Batch size	$batch\_size$	128
Parallel environments	n	16
target update parameter	τ	0.005
ELSIM		
Discriminators hidden layers	lr_D	2x64
Split threshold	δ	0.9
Sampling probability	η	0.5
Average coefficient	β	0.02
Vocabulary size	$\|V\|$	4
Old discriminations scale	α	2
Tree-policy		
Gamma	γ_T	1
Boltzmann coefficient	α_T	5
Learning rate	lr_T	0.05

Table A.2: Hyper-parameters used for experiments on continuous environments.

Transfer learning . In experiments on transfer learning, we fix the number of parallel environments to 1 and reduce the length of an episode to 200. For the first 20 updates of a node, the *tree-policy* remains uniform.

A.5 Skills learned in four rooms environment

Figure A.1 shows the complete set of learned skills in four rooms environment. It completes skills displayed by Figure 4.6.

A.6 Skills learned with a vertical wall

Figure A.2 shows the evolution of the reward function for the complete set of learned skills. It completes skills displayed by Figure 4.7.

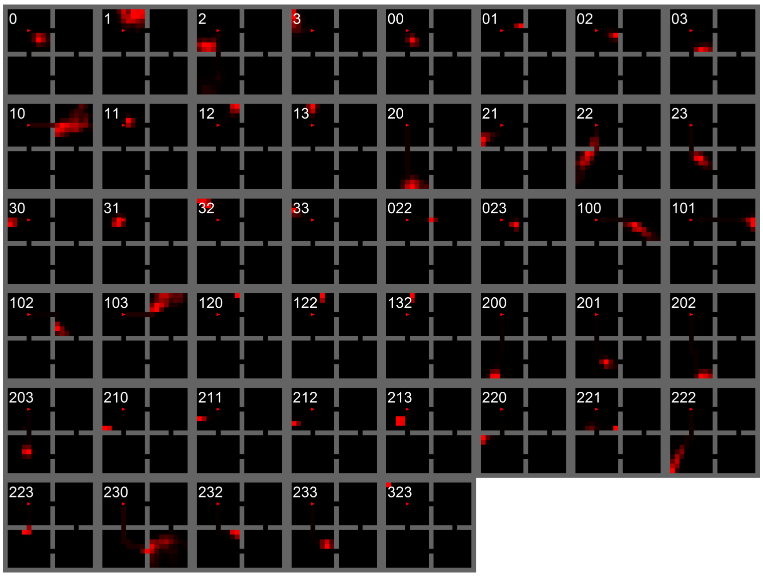


Figure A.1: Full set of skills learned by the agent in an environment composed of four rooms.

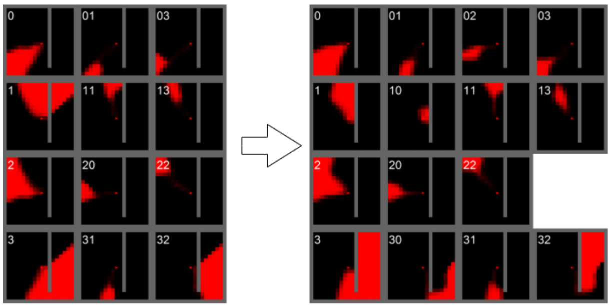


Figure A.2: Probability of achieving each skill in every states (*i.e.* $q(g|s)$).

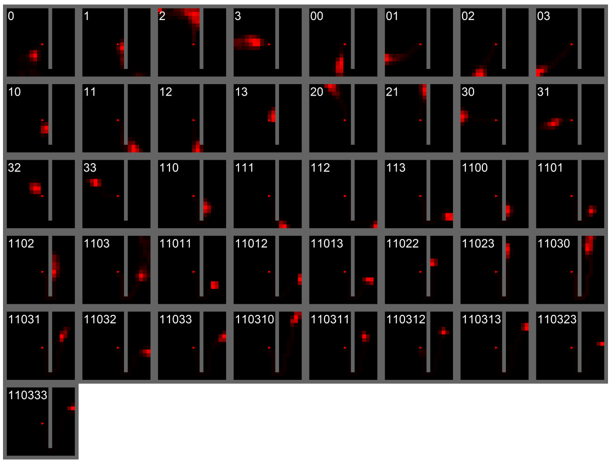


Figure A.3: Skills learned by the agent in an environment with a vertical wall.

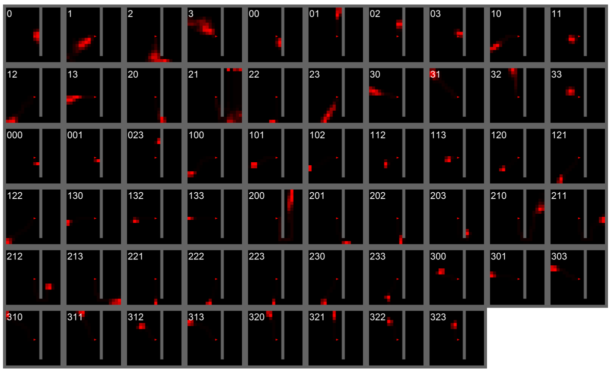


Figure A.4: Skills learned by the agent in an environment with a vertical wall. The agent does not focus on a specific area since there are no rewarding states.

A.7 Skill expansion

Figure A.3 shows the complete set of learned skills learned in an environment with a vertical wall. States on the upper right side of the wall give a reward of 1. It completes skills displayed by Figure 4.8.

In Figure A.4, we perform the same simulation as in Figure 4.8 without goals. The *tree-policy* does not focus on the right side of the wall, and thus, gets less controllability than in Figure 4.8.

A.8 *HalfCheetah-Walk*

We introduce a slight modification of *HalfCheetah-v2* to make the creature walk. The reward function used in *HalfCheetah-v2*:

$$R(s, a, s') = Sf(s, s') + C(a) \tag{A.1}$$

where $C(a)$ is the cost for making moves and $Sf(s, s')$ is the speed of the agent. The reward function used in *HalfCheetah-Walk* is:

$$R(s,a,s') = \begin{cases} Sf(s,s') + C(a) & if\, Sf(s,s') > 2 \\ 4 - Sf(s,s') + C(a) & else \end{cases} \tag{A.2}$$

Appendix B

Complements on DisTop experiments

B.1 Ablation study

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 B.1.

B.2 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. [2021b].

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.

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}	5
Minimal number of selection n_{del}	10
Learning rate α	0.001
Neighbors learning rate $\alpha_{neighbors}$	0.000001
Number of updates per batch	~ 32

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

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.

ELSIM: In contrast with our first experiments, 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.

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].

B.3 Hyper-parameters

Table B.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.

B.4 Environment details

B.4.1 Robotic environments

Environments and protocols are as described in Pong et al. [2020]. 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.

B.4.2 Maze environments

These environments are described in Nachum et al. [2019a] and we used the code modified by Li et al. [2021b]. 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 5.10.

Similarly to Li et al. [2021b], 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 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].

B.5 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).

B.6 Examples of skills

Figures B.1, B.2 and B.3 show examples skills learnt in respectively *Visual Door*, *Visual Pusher* and *Ant Maze*. Additional videos of skills are available in supplementary materials. We also provide videos of the topology building process in maze environments. We only display it in maze environments since the 3D-topology is suitable.

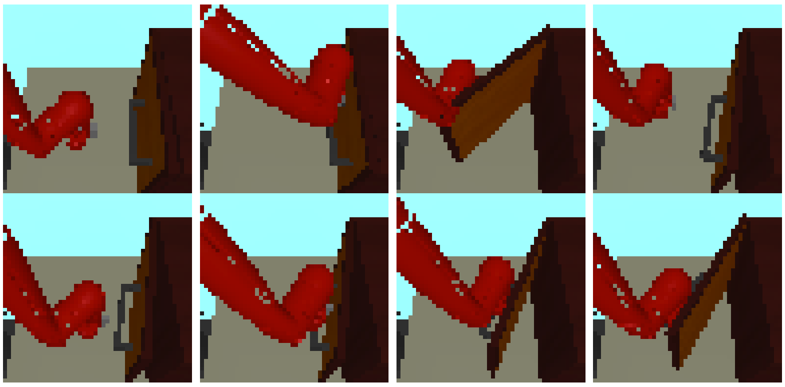


Figure B.1: Examples of 8 skills learnt in *Visual Door*.

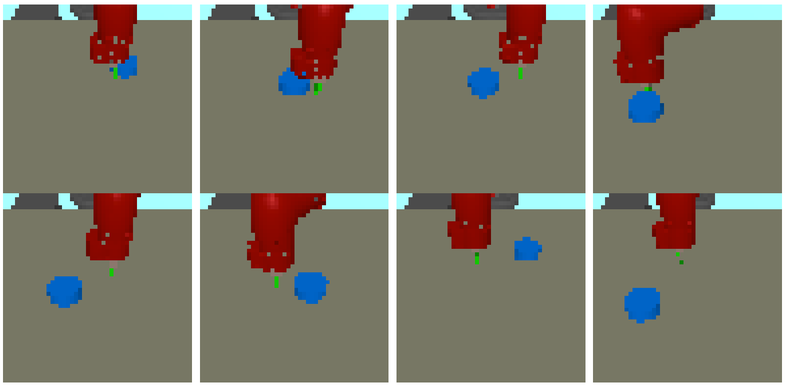


Figure B.2: Examples of 8 skills learnt in *Visual Pusher*.

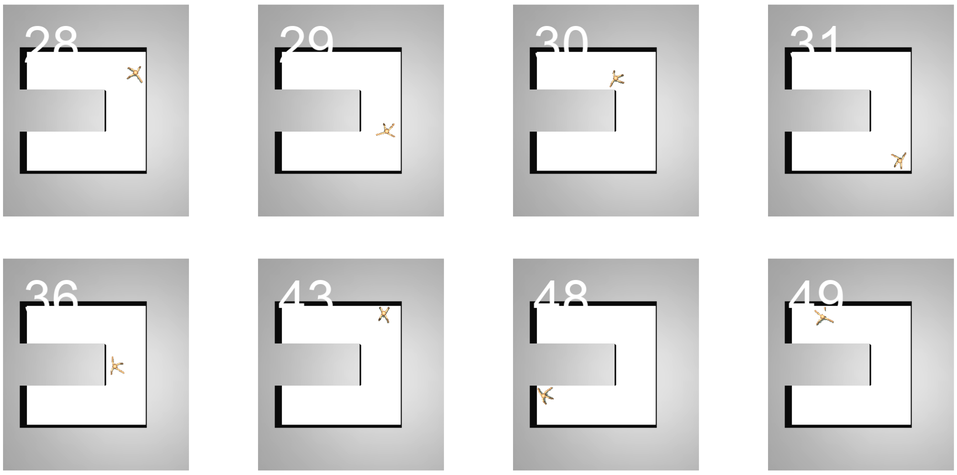


Figure B.3: Examples of 8 skills learnt in *Ant Maze*.

Parameters	Values $RP/RD/MA/MC/SAM/SPM$	Comments
DRL algorithm SAC		
Entropy scale	$0.1/0.1/0.2/0.2/0.1/0.2$	
Hidden layers	$3/3/3/3/4/4$	
Number of neurons	512	Smaller may work
Learning rate	RP: $3 \cdot 10^{-4}$ else $5 \cdot 10^{-4}$	Works with both
Batch size	RP: 256 else 512	Works with both
Smooth update	RP:0.001 else 0.005	Works with both.
Discount factor γ	$0.99/0.99/0.99/0.99/0.996/0.996$	Tuned for mazes
Relabeling	$3/2/1/1/2/2$	3 can replace 2
Representation ϕ_ω		
Learns on B^G	No/No/No/No/Yes/Yes	Works with both
Learning rate	$1 \cdot 10^{-4}$, MA: $5 \cdot 10^{-4}$, MC: $1 \cdot 10^{-3}$	Not tuned on MA, MC
Number of neurons	256 except robotic images	Not tuned
Hidden layers	2 except robotic images	Not tuned
Distortion threshold δ	SPM: 0.01 else 0.1	Tuned on SPM
Distortion coefficient k_c	20	See Appendix B.1
Consistency coefficient β	RD: 0.2 else 2	Not tuned
Smooth update α_{slow}	0.001	Not tuned
Temperature k	$1/1/3/3/3/3$	See Appendix B.1
Topology dimensions d	$10/10/10/3/3/3$	Not tuned
OEGN and sampling		
Creation threshold δ_{new}	RP:0.8 else 0.6	See Appendix B.1
Success threshold $\delta_{success}$	$\infty/\infty/0.2/0.2/\infty/\infty$	
Buffers size	$[8/15/5/5/15/15] \cdot 10^3$	
Skew sampling $1 + \alpha_{skew}$	RD:0.1 else 0	See Appendix B.1
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	$0/0/0.2\alpha_c/0.2\alpha_c/0/0$	Not fine-tuned
Skew selection $1 + \alpha'_{skew}$	$-1/-0,1/0/0/-1/-1/$	See Appendix B.1
Reward temperature t^{ext}	$0/0/50/10/100/100$	

Table B.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.