Automatic state representation and goal selection in unsupervised reinforcement learning

Nicolas Castanet

Sorbonne Université

Institut des Systèmes Intelligents et de Robotique (ISIR)

Ecole Doctorale Informatique, Télécommunications et Electronique

---

Automatic state representation and goal selection in unsupervised reinforcement learning

---

Author:
Nicolas CASTANET

Under the co-supervision of **Sylvain LAMPRIER** and **Olivier SIGAUD**

A dissertation submitted in partial fulfillment of the degree of
Doctor in Philosophy

Submitted on November 7, 2024, Defended on January 15, 2025.

Jury

Pr. David Filliat	ENSTA Paris	Reviewer
Dr. Alain Dutech	INRIA	Reviewer
Pr. Isabelle Bloch	Sorbonne Université	President
Pr. Matthieu Geist	Cohere & Université de Lorraine	Examinator
Pr. Sylvain Lamprier	Université d'Angers	Supervisor
Pr. Olivier Sigaud	Sorbonne Université	Supervisor

Contents

List of Figures

Remerciements

Je souhaite remercier chaudement les personnes suivantes, sans lesquelles ce travail n'aurait pas été possible.

En premier lieu, je souhaite remercier mes encadrants, Olivier Sigaud et Sylvain Lamprier, qui m'ont accompagné, soutenu et conseillé tout au long de ces trois années. J'ai particulièrement apprécié nos riches discussions, leur implication et leur bienveillance, dans les bons comme les mauvais moments. Olivier m'aura appris la rigueur nécessaire à la recherche scientifique, à une époque où la quantité de publications semble primer sur leur qualité, Olivier a toujours privilégié la pertinence et la compréhension profonde de nos propositions. Sylvain aura apporté une dimension théorique à cette thèse, son implication significative dans les aspects techniques a été déterminante pour mener ce travail à son terme.

Je remercie également mes collègues de l'ISIR, nommément Olivier Serris, Charly Pacqueux et Augustin Chartouny, je suis reconnaissant des discussions scientifiques que nous avons partagées. Je remercie également le personnel administratif qui a toujours été à l'écoute, particulièrement Sylvie Piumi et Awatef Barra, dont la compétence, la gentillesse et la compréhension des différentes situations personnelles m'auront été d'une grande aide.

Je remercie ma famille et mes amis, qui ont toujours été présents et à l'écoute au cours de ces trois dernières années. Particulièrement Alex et Elias, avec qui nous avons traversé cette épreuve du doctorat ensemble, en partageant nos doutes, questions, échecs et succès, cette thèse n'aurait pas été la même sans nos longues discussions. Pour finir, je remercie Lise, qui m'a accompagné et soutenu durant ces trois années, sans qui je n'aurais pu mener ce travail à son terme. Ton soutien a été déterminant, et je t'en suis reconnaissant bien au-delà de ce que tu imagines.

Abstract

In the past few years, Reinforcement Learning (RL) achieved tremendous success by training specialized agents possessing the ability to drastically exceed human performance in complex games like Chess or Go, or in robotics applications.

These agents often lack versatility, requiring human engineering to design their behavior for specific tasks with predefined reward signals, limiting their ability to handle new circumstances. This agent's specialization results in poor generalization capabilities, which make them vulnerable to small variations of external factors and adversarial attacks.

A long term objective in artificial intelligence research is to move beyond today's specialized RL agents toward more generalist systems endowed with the capability to adapt in real time to unpredictable external factors and to new downstream tasks. This work aims in this direction, tackling unsupervised reinforcement learning problems, a framework where agents are not provided with external rewards, and thus must autonomously learn new tasks throughout their lifespan, guided by intrinsic motivations.

The concept of intrinsic motivation arises from our understanding of humans' ability to exhibit certain self-sufficient behaviors during their development, such as playing or having curiosity. This ability allows individuals to design and solve their own tasks, and to build inner physical and social representations of their environments, acquiring an open-ended set of skills throughout their lifespan as a result.

This thesis is part of the research effort to incorporate these essential features in artificial agents, leveraging goal-conditioned reinforcement learning to design agents able to discover and master every feasible goal in complex environments.

In our first contribution, we investigate autonomous intrinsic goal setting, as a versatile agent should be able to determine its own goals and the order in which to learn these goals to enhance its performances. By leveraging a learned model of the agent's current goal reaching abilities, we show that we can shape an optimal difficulty goal distribution, enabling to sample goals in the *zone of proximal development* (ZDP) of the agent, which is a psychological

concept referring to the frontier between what a learner knows and what it does not, constituting the space of knowledge that is not mastered yet but have the potential to be acquired. We demonstrate that targeting the ZDP of the agent's result in a significant increase in performance for a great variety of goal-reaching tasks.

Another core competence is to extract a relevant representation of what matters in the environment from observations coming from any available sensors. We address this question in our second contribution, by highlighting the difficulty to learn a correct representation of the environment in an online setting, where the agent acquires knowledge incrementally as it makes progress. In this context, recently achieved goals are outliers, as there are very few occurrences of this new skill in the agent's experiences, making their representations brittle. We leverage the adversarial setting of *Distributionally Robust Optimization* in order for the agent's representations of such outliers to be reliable. We show that our method leads to a virtuous circle, as learning accurate representations for new goals fosters the exploration of the environment.

Résumé

Au cours des dernières années, l'apprentissage par renforcement a connu un succès considérable en entraînant des agents spécialisés capables de dépasser radicalement les performances humaines dans des jeux complexes comme les échecs ou le go, ou dans des applications robotiques.

Ces agents manquent souvent de polyvalence, ce qui oblige l'ingénierie humaine à concevoir leur comportement pour des tâches spécifiques avec un signal de récompense prédéfini, limitant ainsi leur capacité à faire face à de nouvelles circonstances. La spécialisation de ces agents se traduit par de faibles capacités de généralisation, ce qui les rend vulnérables à de petites variations de facteurs externes.

L'un des objectifs de la recherche en intelligence artificielle est de dépasser les agents spécialisés d'aujourd'hui pour aller vers des systèmes plus généralistes pouvant s'adapter en temps réel à des facteurs externes imprévisibles et à de nouvelles tâches en aval. Ce travail va dans ce sens, en s'attaquant aux problèmes d'apprentissage par renforcement non supervisé, un cadre dans lequel les agents ne reçoivent pas de récompenses externes et doivent donc apprendre de manière autonome de nouvelles tâches tout au long de leur vie, guidés par des motivations intrinsèques.

Le concept de motivation intrinsèque découle de notre compréhension de la capacité des humains à adopter certains comportements autonomes au cours de leur développement, tels que le jeu ou la curiosité. Cette capacité permet aux individus de concevoir et de résoudre leurs propres tâches, et de construire des représentations physiques et sociales de leur environnement, acquérant ainsi un ensemble ouvert de compétences tout au long de leur existence.

Cette thèse s'inscrit dans l'effort de recherche visant à incorporer ces caractéristiques essentielles dans les agents artificiels, en s'appuyant sur l'apprentissage par renforcement conditionné par les buts pour concevoir des agents capables de découvrir et de maîtriser tous les buts réalisables dans des environnements complexes.

Dans notre première contribution, nous étudions la sélection autonome de buts intrinsèques, car un agent polyvalent doit être capable de déterminer ses propres objectifs et l'ordre dans lequel apprendre ces objectifs pour améliorer

ses performances. En tirant parti d'un modèle appris des capacités actuelles de l'agent à atteindre des buts, nous montrons que nous pouvons construire une distribution de buts optimale en fonction de leur difficulté, permettant d'échantillonner des buts dans la *zone de développement proximal* (ZDP) de l'agent, qui est un concept issu de la psychologie signifiant la zone à la frontière entre ce qu'un agent sait et ce qu'il ne sait pas, constituant l'espace de connaissances qui n'est pas encore maîtrisé, mais qui a le potentiel d'être acquis prochainement. Nous démontrons que le fait de cibler la ZDP de l'agent entraîne une augmentation significative des performances pour une grande variété de tâches.

Une autre compétence clé est d'extraire une représentation pertinente de l'environnement à partir des observations issues des capteurs disponibles. Nous abordons cette question dans notre deuxième contribution, en soulignant la difficulté d'apprendre une représentation correcte de l'environnement dans un cadre en ligne, où l'agent acquiert des connaissances de manière incrémentale au fur et à mesure de ses progrès. Dans ce contexte, les objectifs récemment atteints sont considérés comme des valeurs aberrantes, car il y a très peu d'occurrences de cette nouvelle compétence dans les expériences de l'agent, ce qui rend leurs représentations fragiles. Nous exploitons le cadre adversaire de l'*Optimisation Distributionnellement Robuste* afin que les représentations de l'agent pour de tels exemples soient fiables. Nous montrons que notre méthode conduit à un cercle vertueux, car l'apprentissage de représentations correctes pour de nouveaux objectifs favorise l'exploration de l'environnement.

Résumé long en Français

1 Introduction

La recherche en intelligence artificielle (IA) est actuellement engagée dans une course à la conception d'agents autonomes dotés d'une polyvalence croissante dans des domaines aussi variés que la vision par ordinateur, la médecine ou la robotique (Bommasani et al., 2021; Moor et al., 2023; Firoozi et al., 2023).

Un objectif à long terme est de développer des agents généralistes, capables de s'adapter rapidement à de nouvelles tâches en aval sans avoir besoin de supervision humaine, en suivant les traces des récents succès obtenus dans le domaine de la vision par ordinateur (Chen et al., 2020; Radford et al., 2021) et du traitement du langage naturel (Radford et al., 2019; Brown, 2020). Ces succès ont été obtenus en s'appuyant sur des techniques non supervisées très efficaces et indépendantes des tâches à réaliser en aval. L'objectif est de fournir aux humains un assistant multitâche sous la forme d'agents logiciels ou même de robots physiques.

Un tel agent polyvalent devrait présenter un certain nombre de capacités clés :

- **Apprentissage de représentation** : il doit être capable d'acquérir une représentation pertinente de ce qui compte dans son environnement à partir d'observations provenant de tous les capteurs disponibles.
- **Apprentissage automatique de tâches** : il doit être capable de déterminer seul une séquence d'actions futures lui permettant d'accomplir une tâche donnée.
- **Sélection automatique de tâches** : il doit être capable de déterminer ses propres tâches sans supervision, et l'ordre dans lequel apprendre ces tâches afin d'améliorer ses performances (Colas, 2021; Colas et al., 2022).
- **Apprentissage ouvert** : il doit être capable d'apprendre à accomplir de nouvelles tâches dans son environnement tout au long de sa vie, au fur

et à mesure qu'elles se présentent, sans limitation du nombre de tâches (Team et al., 2021; Sigaud et al., 2023).

- **Interactivité** : il doit être capable de comprendre l'intention de son utilisateur à partir de diverses modalités, telles que le langage, l'expression du visage ou la posture du corps, d'interagir ou de collaborer avec cet utilisateur par le biais de modalités similaires, etc. (Akalin and Loutfi, 2021; Sigaud et al., 2022).

Cette liste de capacités n'est certainement pas exhaustive. Dans cette thèse, notre intention n'est pas de construire un agent polyvalent présentant toutes les capacités susmentionnées, s'agissant d'un objectif à long terme qui est actuellement hors de portée. Nous examinons plutôt les trois premières capacités et étudions certains des défis que leur combinaison soulève. Pour rendre compte de l'apprentissage autonome des tâches, nous adoptons le cadre de l'apprentissage par renforcement conditionné par les buts (Kaelbling, 1993; Liu et al., 2022).

Plus précisément, notre travail dans ce sens est divisé en deux étapes, correspondant à deux contributions distinctes.

Dans cette première étape, nous mettons de côté l'apprentissage par représentation et considérons un agent qui a accès à une connaissance précise et peu dimensionnelle de son état. Dans ce contexte, l'apprentissage autonome de tâches nécessite que l'agent découvre lui-même les buts qu'il peut atteindre dans son environnement et qu'il le fasse de manière incrémentale en apprenant à maîtriser chacun des buts découverts. Cette combinaison d'exigences entraîne une tension entre la nécessité de maîtriser les objectifs découverts, qui tend à rendre l'agent conservateur, et la nécessité de découvrir de nouveaux objectifs, qui peut déstabiliser les connaissances acquises jusqu'à présent (Campos et al., 2020). En conséquence, la distribution des objectifs que l'agent aborde à un moment donné devrait évoluer selon une dynamique soigneusement conçue, appelée curriculum (Colas et al., 2018). Notre contribution centrale dans cette première étape est de montrer que l'apprentissage d'un modèle prédictif de la capacité de l'agent à atteindre ou non un objectif donné et le fait de laisser cet agent cibler les objectifs pour lesquels sa prédiction est la plus incertaine est un bon moyen de concevoir un curriculum qui aide beaucoup à aborder l'arbitrage entre découverte d'objectifs et maîtrise d'objectifs. Pour étoffer cette contribution, nous concevons un cadre appelé Stein Variational Goal Generation (SVGG). Dans ce cadre, le modèle prédictif des capacités à atteindre des objectifs est un réseau neuronal entraîné à partir des tentatives de l'agent pour atteindre des objectifs. Pour permettre à l'agent d'échantillonner de nouveaux objectifs, nous nous appuyons sur un outil mathématique appelé Descente du Gradient Variationnel de Stein

(SVGD en anglais) (Liu and Wang, 2016) où la distribution actuelle des objectifs est représentée comme un ensemble de particules qui sont attirées par les objectifs les plus incertains et qui ont tendance à être repoussées les unes des autres, de manière à couvrir l'espace des objectifs de la manière la plus uniforme possible. Nous étudions en détail les propriétés de ce cadre dans des labyrinthes simples à deux dimensions, puis nous montrons qu'il fonctionne bien dans des environnements plus difficiles et de plus grande dimension.

Dans un deuxième temps, nous intégrons l'apprentissage de la représentation dans l'équation et étudions les défis supplémentaires que cette insertion soulève. Au lieu de considérer qu'un agent a accès à une représentation précise de l'état, nous considérons qu'il observe une image de l'environnement et qu'il doit construire sa propre représentation de l'état à partir de cette image. De nombreux travaux dans ce domaine séparent séquentiellement l'étape d'apprentissage de la représentation de l'étape d'apprentissage de la réalisation de la tâche, mais cette approche est difficilement compatible avec les tâches autonomes, où les agents doivent construire leurs propres représentations des tâches et des environnements au fur et à mesure qu'ils les découvrent. Cependant, lorsque les processus d'apprentissage des représentations et des tâches sont menées conjointement, plusieurs nouveaux problèmes se posent, comme le fait que l'apprentissage des tâches est basé sur une représentation latente non stationnaire de l'état, ce qui entraîne une plus grande instabilité dans la maîtrise des tâches, ou le fait qu'une représentation latente peut ne correspondre à aucun objectif possible, ce qui rend difficile la mesure des capacités d'accomplissement des tâches. Plus important encore, nous montrons un cercle vicieux conduisant les agents autonomes à sélectionner des tâches qui possèdent déjà une représentation précise, empêchant toute exploration ultérieure de nouvelles tâches. Pour atténuer cette limitation majeure, nous faisons appel au cadre d'optimisation distributionnellement robuste (DRO en anglais) (Rahimian and Mehrotra, 2019), qui oblige les représentations à être robustes aux changements dans la distribution future des tâches. Nous insérons DRO dans un agent autonome qui apprend en ligne ses propres représentations de l'état, ce qui donne lieu à un cadre appelé Optimisation distributionnellement robuste pour Exploration par image (DROIDE en anglais). Une fois encore, nous étudions ces questions à l'aide de labyrinthes bidimensionnels simples afin de mieux comprendre les phénomènes qui surviennent dans ce contexte et nous montrons que DROIDE permet à l'agent d'anticiper la représentation des tâches non vues et de surmonter ainsi le goulot d'étranglement de l'exploration en les maîtrisant.

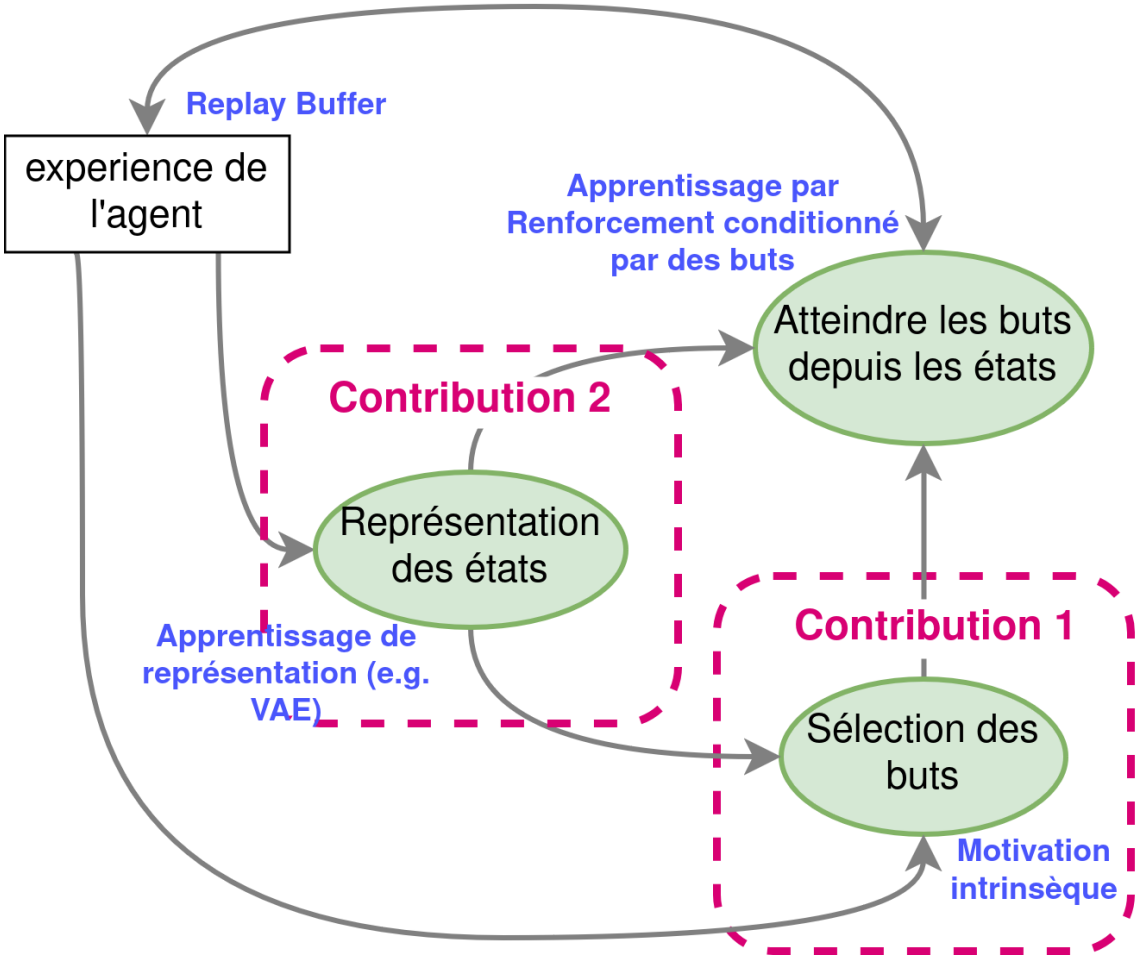

Figure 1: Cette thèse aborde les tâches d'atteinte de buts non supervisées, ce qui impose de concevoir des techniques automatiques en fonction de l'expérience de l'agent avec l'environnement. L'agent doit apprendre à sélectionner des buts sans supervision à travers le prisme du concept de motivation intrinsèque (contribution 1), et apprendre à les atteindre par apprentissage par renforcement. De plus, lorsque l'environnement manque d'états sémantiquement significatifs (par exemple avec des observations de pixels), l'agent doit être capable d'extraire des représentations utiles et compactes (contribution 2).

2 Contexte

2.1 Apprentissage par Renforcement

L'apprentissage par renforcement est un processus de prise de décision séquentiel, dans lequel un agent interagit avec un environnement et doit apprendre à prendre des mesures afin d'améliorer sa performance. L'agent interagit avec un environnement et doit apprendre à agir afin de maximiser une quantité spécifiée appelée récompense. À chaque pas de temps, l'agent se trouve dans un état donné et doit entreprendre une action, l'environnement fournit alors une récompense à l'agent et met à jour son état actuel. Ce processus séquentiel est modélisé par un *Processus de Décision Markovien* ou Processus Décisionnel de Markov (MDP en anglais) Bellman (1966).

Dans cette thèse, nous nous attaquons aux **problèmes de contrôle continu**, car ils offrent un plus grand défi et une plus grande variété d'applications dans des scénarios du monde réel (par exemple, la robotique) contrairement au contrôle discret qui est principalement limité aux jeux vidéo (par exemple, les jeux Atari) et à des programmes logiciels spécifiques. Ce cadre implique que l'espace d'action et l'espace d'état du PDM sont continus. Pour cette raison, nous nous tournons vers des algorithmes basés sur des politiques, alimentés par des réseaux neuronaux qui mettent en correspondance les états et les actions, permettant une modélisation fine du comportement de l'agent, et l'optimisation par des techniques de **gradient de politique**.

En outre, les méthodes de gradient de politique peuvent être classées en deux catégories : **On-Policy** et **Off-Policy**. D'une part, les algorithmes « on-policy » calculent le gradient d'optimisation sur la base des expériences recueillies dans le cadre de la politique comportementale actuelle. D'autre part, les algorithmes hors politique étendent les objectifs du gradient de la politique pour inclure les expériences collectées à partir de différentes politiques dans le processus d'optimisation (en particulier les échantillons collectés à partir de versions antérieures de la politique stockée dans un **tampon de relecture**). Les deux approches présentent des avantages et des inconvénients. En résumé, les méthodes « on-policy » ont tendance à être plus stables, tandis que les méthodes « off-policy » sont plus efficaces en termes d'échantillons ; voir Sutton and Barto (2018) pour un examen plus approfondi de ce sujet.

Plus important encore, les algorithmes off-policy possèdent la propriété attrayante de partager des expériences entre plusieurs politiques, ce qui est un atout majeur pour notre objectif de construire des agents autonomes multitâches avec GCRL. En fait, le fait d'être hors politique est essentiel pour exploiter les trajectoires pour différents objectifs afin de transférer des connaissances et de construire des tremplins, ce qui se traduit par un gain massif

à la fois en termes de performance et d'efficacité de l'échantillon lorsqu'il s'agit d'apprendre des politiques conditionnées par des objectifs (Andrychowicz et al., 2017).

Dans la section suivante, nous ouvrons la voie depuis les bases du gradient de politique jusqu'à un algorithme off-policy de base pour le contrôle continu : Deep Deterministic Policy Gradient (DDPG) Lillicrap et al. (2015), que nous utilisons pour l'apprentissage de politique tout au long de cette thèse.

2.2 Apprentissage par Renforcement conditionné par des buts

L'apprentissage par renforcement conditionné par des buts (Goal-Conditioned Reinforcement Learning, GCRL) Kaelbling (1993); Schaul et al. (2015) est une extension du cadre classique de l'apprentissage par renforcement. Alors que dans le cadre de l'apprentissage par renforcement, l'agent entreprend des actions en fonction de son état actuel s, dans ce nouveau cadre, le comportement de l'agent est conditionné à la fois par son état actuel s et par l'objectif qu'il poursuit g. Le but lui-même peut être généré par l'agent ou provenir de l'environnement. Dans ce dernier cas, le PDM est complété par un *espace d'objectifs* $\mathcal{G}$. L'agent est maintenant représenté par un *politique conditionnée par le but*.

Une *politique conditionnée par les objectifs* (GCP), notée $\pi(.|s, g)$, est une fonction qui fait correspondre la distribution des actions de l'espace d'état $\mathcal{S}$ et de l'espace des objectifs $\mathcal{G}$ du MDP :

$$\begin{aligned} \pi : \mathcal{S} \times \mathcal{G} &\to \mathcal{P}_s \\ (s, g) &\mapsto \pi(.|s, g) = \mathbb{P}(\mathcal{A}_s). \end{aligned}$$

Chaque état de l'environnement correspond à un objectif spécifique. Dans le cas le plus général, la relation entre $\mathcal{S}$ et $\mathcal{G}$ peut être exprimée comme suit :

$$\begin{aligned} f_{\mathcal{G}} : \mathcal{S} &\to \mathcal{G} \\ s &\mapsto f_{\mathcal{G}}(s). \end{aligned}$$

Dans cette thèse, nous adoptons la définition la plus simple de l'espace des objectifs, $f_{\mathcal{G}} = I$, où $\mathcal{G} = \mathcal{S}$, chaque état s correspond à un objectif potentiel à atteindre : $f_{\mathcal{G}}(s) = s$. Pour que la politique conditionnée par les objectifs apprenne à agir de manière à atteindre l'objectif visé, chaque objectif g doit être associé à une fonction de récompense spécifique $\mathcal{R}_g$, qui permet de suivre les progrès accomplis dans la réalisation de l'objectif. Le problème RL classique peut alors être étendu pour gérer des multiples objectifs, ce qui peut être formulé dans le cadre plus général d'un *MDP contextuel* Hallak et al. (2015) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, \mathcal{R}_g, \gamma\}$.

$$\mathcal{R}_g : \mathcal{S} \to \{0, 1\}$$
$$s \mapsto \mathcal{R}_g(s) = \mathbb{1}\Big[||s - g||_2 < \delta\Big],$$

δ étant un hyper-paramètre contrôlant la boule $L2$ centrée sur g à l'intérieur de laquelle nous considérons que l'objectif ciblé doit être atteint. Notons que cette définition n'implique pas l'action entreprise par l'agent ni son état antérieur, l'objectif est contraint d'atteindre le but quelle que soit la manière.

2.3 Apprentissage par Renforcement non supervisé

Malgré l'énorme succès obtenu ces dernières années dans l'apprentissage par renforcement et son application aux jeux vidéo (Mnih et al., 2013; Vinyals et al., 2019; Berner et al., 2019) et à la robotique (Schulman et al., 2017; Akkaya et al., 2019), les agents existants manquent souvent de polyvalence, obligeant les ingénieurs humains à concevoir leur comportement pour des tâches spécifiques, ce qui limite leur capacité à faire face à de nouvelles circonstances. La spécialisation de cet agent se traduit par des politiques dont les capacités de généralisation sont médiocres, ce qui les rend vulnérables aux petites variations des facteurs externes et aux attaques adverses (Gleave et al., 2019).

En outre, dans de nombreux scénarios réels, les tâches à effectuer sont complexes et impliquent diverses manipulations d'objets et des interactions délicates avec l'environnement. Par conséquent, la conception des conditions dans lesquelles l'agent a accompli la tâche peut être alambiquée et prendre du temps, en raison d'interactions inattendues entre les facteurs externes et les spécifications des agents artificiels. Par exemple, en RL, la définition manuelle des fonctions de récompense pour une tâche spécifique est fragile et conduit parfois à un comportement sous-optimal lorsque la politique trouve un moyen de maximiser la fonction de récompense sans achever la tâche prévue. Dans ce cas, le signal de récompense est considéré comme trompeur. Permettre à un agent de concevoir et de poursuivre automatiquement ses propres tâches est donc une caractéristique très précieuse du RL.

Pour ces raisons, il est important d'aller au-delà des agents RL spécialisés d'aujourd'hui vers des systèmes plus généralistes dotés de la capacité de s'adapter à de nouvelles tâches en aval. L'apprentissage par renforcement non supervisé (URL) offre un cadre dans lequel les agents ne reçoivent pas de récompenses externes et doivent donc apprendre de nouvelles tâches de manière autonome tout au long de leur vie, guidés par des motivations intrinsèques. Le champ d'application de cette thèse couvre les environnements RL à *dynamique fixe* Les environnements RL, dans le sens où l'espace d'état $\mathcal{S}$, l'espace d'action

$\mathcal{A}$ et la fonction de transition $\mathcal{T}$ sont constants. La composante changeante du problème, qui permet à l'agent de développer ses compétences par le biais d'un programme de tâches, est le *motivation intrinsèque*, un processus interne qui génère des *récompenses intrinsèques*, sur la base des capacités actuelles de l'agent.

Les agents à motivation intrinsèque multitâches sont un cadre largement utilisé dans l'apprentissage par renforcement pour aborder des contextes non supervisés, qui a émergé du domaine de la robotique développementale (Oudeyer et al., 2007; Oudeyer and Kaplan, 2007), en s'inspirant des mécanismes sous-jacents qui alimentent les comportements autonomes chez l'homme. L'objectif est d'intégrer dans les agents d'apprentissage par renforcement la capacité d'inventer et de poursuivre leurs propres problèmes, en utilisant le retour d'information interne et l'expérience pour évaluer l'achèvement.

Appliqué au RL, le processus de motivation intrinsèque induisant des récompenses internes réside dans la génération automatique d'un curriculum de tâches conditionnant la politique de l'agent, en s'appuyant sur sa propre expérience, comme illustré dans Figure 3.6. Ce cadre a remporté de nombreux succès dans la résolution de problèmes de RL non supervisé au cours des dernières années, en particulier pour les tâches de contrôle et de navigation robotiques (Eysenbach et al., 2018a; Nair et al., 2018; Pong et al., 2019; Pitis et al., 2020). Il convient de noter qu'en plus de fonctionner dans des contextes non supervisés, les agents à motivation intrinsèque multitâches ont été appliqués avec succès pour accélérer l'apprentissage sur des tâches difficiles, même lorsque l'objectif est connu à l'avance (Colas et al., 2018; Ren et al., 2019).

L'ambition des méthodes de motivation intrinsèque multitâches est d'obtenir un agent capable d'explorer et de contrôler son environnement, ce qui implique la capacité d'atteindre n'importe quelle tâche possible grâce à la politique conditionnée par la tâche :

Definition 2.1. La *Politique multitaches optimale* dans le MDP contextuel $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \omega, \mathcal{R}_\omega, \gamma\}$, denoted as π^*, est définie comme:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\omega\sim\Omega, s\sim\rho_0}\Big[\mathbb{E}_{\tau\sim\pi(.|s,\omega)}\big[\sum_{t=0}^{\infty}\gamma^t r_t^\omega\big]\Big],$$

Ou ω est une tâche échantillonnée à partir d'une distribution arbitraire de tâches dans l'environnement Ω, associée à une fonction de récompense conditionnée par la tâche $\mathcal{R}_\omega$, ρ_0 est la distribution d'états initials de l'agent, $\tau \sim \pi(.|s,\omega)$ est une trajectoire obtenue par la politique π avec l'état initial $s \in \mathcal{S}$ and $r_t^\omega = \mathcal{R}_\omega(s_t, a_t, s_{t+1})$.

Le cadre non supervisé impose les contraintes suivantes : la distribution de toutes les tâches valides dans l'environnement Ω, et la fonction de récompense $\mathcal{R}_\omega$ pour $\omega \sim \Omega$ sont inconnues, ce qui constitue une connaissance préalable précieuse de l'environnement. Par conséquent, l'optimisation de cette quantité n'est pas simple.

Toutefois, cet objectif général peut être abordé en deux phases distinctes constituant des objectifs de substitution, qui peuvent être menées conjointement ou séparément, sans supervision : *Exploration* et *Contrôle*. La composante *Exploration* mesure la capacité d'un agent à découvrir de nouvelles tâches :

Definition 2.2. *l'exploration* multitâches en RL est un processus caractérisant l'extension de la distribution des tâches découvertes $\omega \in \Omega$:

$$\max_\pi \Big[\text{explo}(\pi) := H\Big[p_\pi^\omega\Big]\Big],$$

où p_π^ω est la distribution des tâches accomplies par la politique π au cours de l'apprentissage.

L'exploration ne doit pas être confondue avec la capacité à accomplir des tâches de manière fiable et intentionnelle. En effet, certaines tâches peuvent avoir été découvertes accidentellement au cours du processus d'exploration. En outre, des oublis catastrophiques peuvent survenir au cours de l'apprentissage.

Nous appelons *Contrôle* la capacité d'un agent à accomplir de manière fiable les tâches découvertes :

Definition 2.3. Un agent *Controle* une tâche $\omega \in \Omega$ si :

$$\mathbb{P}\Big(\omega \in \tau \mid \tau \sim \pi(.|s,\omega)\Big) = 1,$$

où τ est une trajectoire échantillonnée à partir de la politique $\pi(.|s,\omega)$, l'affirmation $\omega \in \tau$ signifie que la tâche demandée est incluse dans la trajectoire.

Nous définissons l'objectif général de *Control* comme la capacité de l'agent à maximiser la quantité ci-dessus pour les tâches dans p_π^ω :

$$\max_\pi \Big[\text{control}(\pi) := \mathbb{E}_{\omega \sim p_\pi^\omega}\Big[\mathbb{P}\Big(\omega \in \tau \mid \tau \sim \pi(.|s,\omega)\Big)\Big]\Big].$$

Ce cadre général soulève les questions suivantes :

Comment concevoir le processus de motivation intrinsèque ? La distribution de chaque tâche valide dans l'environnement Ω étant inconnue, afin de maximiser *Exploration* et *Contrôle* sans supervision, nous devons définir la distribution de la motivation intrinsèque pour organiser le programme des tâches :

Definition 2.4. La distribution des tâches intrinsèques, désignée par Ω_π, est définit comme le processus qui générant des tâches en fonction de la politique actuelle π :
Nous notons π^Ω la politique résultant de l'entraînement de la politique actuelle π sur la distribution intrinsèque des tâches Ω_π.
Nous définissons l'objectif de la distribution des tâches intrinsèques comme la maximisation de l'exploration et du contrôle de π^Ω :

$$\Omega_\pi{}^* = \underset{\Omega_\pi}{\operatorname{argmax}} \Big[\text{explo}(\pi^\Omega) + \text{control}(\pi^\Omega)\Big]. \tag{1}$$

La définition de Ω^π_λ est au cœur des récents efforts de recherche sur les agents à motivation intrinsèque multitâches.

Comment définir une tâche ? Dans le manuscrit, nous présentons deux paradigmes majeurs d'agents à motivation intrinsèque multitâches, le premier considérant une tâche comme une séquence d'états et d'actions à effectuer, appelé *compétence*, le second considérant une tâche comme un état unique à atteindre, appelé *but*, paradigme que nous utilisons pour ce travail de thèse.

3 Sélection de buts par critère de difficulté optimale

Nous présentons dans cette section la première contribution de cette thèse, visant à concevoir un processus de génération d'objectifs intrinsèques capable de maximiser efficacement l'objectif de couverture du succès (3.7) défini p. 53. Nous présentons l'algorithme Stein Variational Goal Generation (SVGG), qui tire parti des propriétés de Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016) afin d'obtenir une distribution intrinsèque des objectifs combinant des incitations à la fois à l'exploration et au contrôle, et d'être robuste aux changements inattendus de l'environnement. Nous montrons dans nos expériences que SVGG surpasse les méthodes de la littérature sur plusieurs tâches.

Notre objectif avec Stein Variational Goal Generation (SVGG), est d'obtenir

un curriculum pour échantillonner des objectifs de difficulté appropriée pour l'agent RL, où le curriculum est représenté comme une probabilité d'échantillonnage d'objectifs évolutive p_{goals}. Pour ce faire, nous maintenons un modèle des compétences de l'agent - ou de sa capacité à atteindre des objectifs - qui permet de définir une distribution p_{skills}. Cette distribution attribue une masse de probabilité aux zones d'objectifs de difficulté appropriée. En outre, avec un simple SVM à une classe, nous apprenons une distribution de validité p_{valid} qui empêche les particules d'être échantillonnées dans des zones non valides de l'espace des objectifs. Ensuite, nous visons à échantillonner les objectifs d'entraînement à partir de la distribution cible suivante :

$$p_{\text{goals}}(g) \propto p_{\text{skills}}(g).p_{\text{valid}}(g). \tag{2}$$

Comme le calcul de la fonction de partition est difficile à réaliser pour une telle formulation de distribution, nous préférons échantillonner uniformément sur un ensemble de particules $q = \{x_i\}_{i=1}^{m}$, qui sont optimisées par SVGD pour approximer $p_{textgoals}(g)$. En outre, nous sommes intéressés par le suivi des zones utiles pour former l'agent. Dans ce contexte, le traitement d'un ensemble de particules représentant l'état du paysage d'exploration complet semble mieux adapté que les méthodes reposant sur des candidats uniques, telles que MCMC ou la dynamique de Langevin. En effet, ces alternatives produiraient des échantillons très corrélés dans le temps, avec une dynamique instable. Notre choix améliore également l'interprétabilité du processus, en fournissant une image complète de la distribution du comportement actuel au cours de l'apprentissage.

4 Apprentissage de représentation pour l'exploration par image

Avec l'apprentissage par renforcement conditionné par les objectifs, un agent peut apprendre un ensemble varié de comportements dans des environnements complexes en l'absence d'une fonction de récompense prédéfinie. Cependant, avec des entrées visuelles, les informations sémantiques contenues dans les observations ne sont pas explicites et les observations sont hautement dimensionnelles, ce qui rend la génération d'objectifs pour l'exploration ou le calcul de la récompense beaucoup plus difficile. Une approche classique pour atténuer ce problème consiste à exploiter une représentation latente plus compacte de l'espace d'observation qui capture la sémantique de l'espace d'observation et qui est de dimension inférieure. Mais comment passer de l'espace d'observation à l'espace latent ? Une première approche consiste à

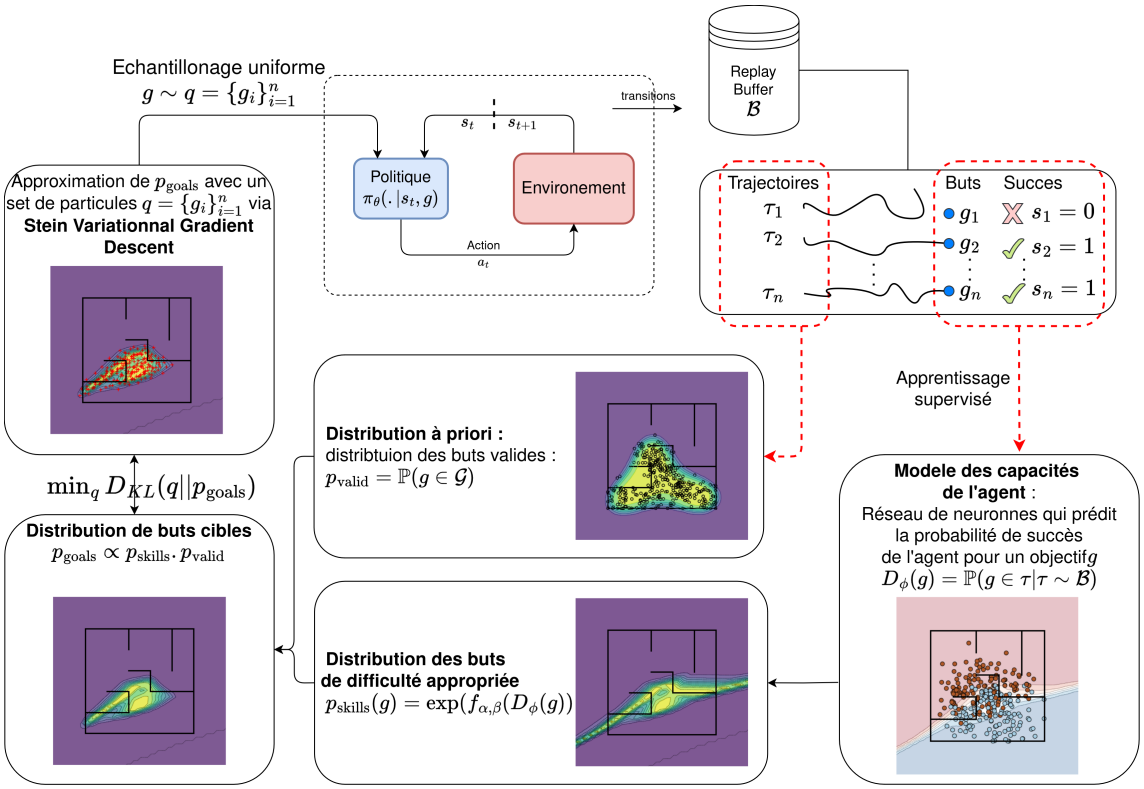

Figure 2: Vue d'ensemble de la méthode SVGG. L'interaction des agents avec leur environnement est stockée dans le *replay buffer* (en haut à droite) et utilisée pour apprendre D_ϕ, un modèle de ses capacités à atteindre des buts (en bas à droite). Nous nous appuyons sur ce modèle pour calculer une distribution d'objectifs de difficulté appropriée p_{skills}, en tirant parti d'une distribution de validité p_{valid} pour rester dans l'espace des objectifs valides. La distribution des de buts cible obtenue p_{goals} est approximée par des particules $\{g_i\}_{i=1}^n$ en utilisant la descente de gradient variationnelle de Stein (SVGD), et l'agent échantillonne un but à partir de ces particules.

apprendre à l'avance une correspondance précise entre l'espace d'observation et l'espace latent, à partir d'un ensemble de données donné dans un contexte hors ligne. Dans le cadre de cette approche, la question est de savoir comment collecter un tel ensemble de données. Si l'ensemble de données couvre tout l'environnement, cela suppose que nous connaissons l'environnement à l'avance, ce qui contredit l'hypothèse selon laquelle l'agent doit l'explorer. Dans l'autre cas, qui correspond à l'environnement en ligne, l'agent collectera l'ensemble de données tout en explorant l'environnement. Dans ce cas, la représentation obtenue se déplace progressivement dans le temps en même temps que la distribution des données provenant séquentiellement de la politique pendant que l'agent explore. Ce changement de distribution – ou distributionnel – est un problème bien connu dans l'apprentissage automatique, il se produit chaque fois qu'un agent est formé à partir d'une distribution donnée de données mais qu'il est soumis à une autre distribution de données lorsqu'il est déployé. La première conséquence du changement progressif de distribution dans notre contexte est que l'espace latent évolue dans le temps, ce qui rend l'apprentissage des politiques plus difficile. La deuxième conséquence est qu'une mauvaise représentation de l'état latent peut empêcher l'agent de découvrir avec succès l'espace d'observation complet, ce qui entraîne un cercle vicieux, comme illustré dans Figure 3.13, page 64. Dans ce travail, comme nous sommes motivés par la construction d'agents autonomes, nous ne pouvons pas nous appuyer sur la première approche qui présuppose une injection importante de connaissances préalables.

Dans ce chapitre, nous abordons donc les questions liées à la deuxième approche. Nous considérons le cas où un agent doit apprendre des représentations significatives à partir de l'observation d'images en ligne, tout en explorant et en apprenant à contrôler l'environnement.

La principale difficulté à laquelle nous sommes confrontés est le changement progressif de la distribution, qui se traduit par une représentation apprise évolutive et potentiellement biaisée. Pour surmonter cette difficulté, nous faisons appel à un outil conçu pour contrer les effets du changement de distribution : le cadre *Optimisation Distributionnellement Robuste*. En le combinant avec un β-VAE classique Higgins et al. (2017a), nous sommes en mesure de biaiser les données provenant de la politique afin d'apprendre une représentation non biaisée pour les agents à motivation intrinsèque conditionnés par des objectifs. Il en résulte un apprentissage plus régulier, une meilleure exploration et enfin une meilleure performance de l'agent.

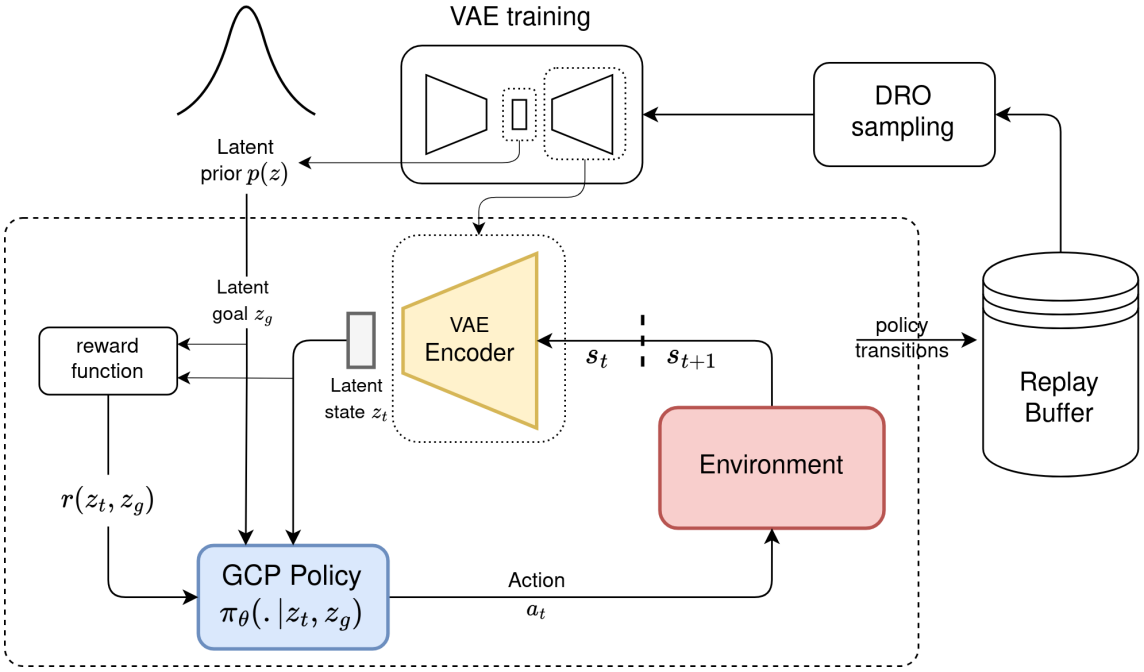

Figure 3: Aperçu de l'exploration avec DROIDE: similaire à Nair et al. (2018), nous apprenons une VAE et utilisons l'encodeur pour obtenir une représentation compacte z_t des états de pixels s_t. Cependant, pour former la VAE, notre innovation clé consiste à échantillonner à partir du tampon de relecture en utilisant DRO, ce qui permet aux représentations d'être robustes aux futurs changements de distribution de l'exploration de la politique.

5 Conclusion

Dans cette thèse, nous soutenons qu'en l'absence de connaissances préalables sur les objectifs à atteindre dans un environnement, l'objectif général des agents à motivation intrinsèque conditionnés par des objectifs devrait être une combinaison d'exploration et de contrôle, que nous proposons de mesurer avec la couverture de succès.

Dans notre première contribution, nous avons proposé Stein Variational Goal Generation (SVGG), une méthode de génération d'objectifs intrinsèques prenant en compte à la fois la difficulté et la nouveauté des objectifs, afin d'aborder à la fois les composantes de contrôle et d'exploration de la couverture de succès, où nous avons remarqué que la plupart des méthodes de la littérature se concentraient simplement sur l'un ou l'autre des deux aspects de l'objectif. À cette fin, nous nous sommes appuyés sur les propriétés uniques de l'algorithme d'inférence variationnelle, Stein Variational Gradient Descent (SVGD), pour approximer notre distribution intrinsèque des objectifs, ce qui

nous permet d'obtenir la « propriété de récupération », qui garantit que l'agent est capable de s'adapter à des changements inattendus dans la topologie de l'environnement. Nous pensons qu'il s'agit d'une caractéristique clé qu'un agent autonome doit présenter dans des environnements non supervisés.

SVGG a montré des améliorations significatives sur des tâches difficiles par rapport aux méthodes concurrentes. Cependant, l'une des principales limites de cette première contribution était l'hypothèse selon laquelle l'agent avait accès à une représentation précise de l'état.

C'est pourquoi, dans notre deuxième contribution, nous avons élargi le champ d'application des agents motivés par des objectifs intrinsèquement conditionnés afin d'intégrer l'apprentissage de la représentation pour les observations de pixels. Constatant que la combinaison de l'apprentissage des représentations et du comportement conduit à un goulot d'étranglement au niveau de l'exploration, nous avons étendu le cadre de l'optimisation distributionnellement robuste (DRO) des paramètres supervisés traditionnels à l'apprentissage non supervisé des représentations avec des auto-codeurs variationnels (VAE). Nous avons montré qu'en apprenant des représentations robustes aux futurs changements de distribution de la politique, notre méthode est capable d'explorer efficacement des labyrinthes à la topologie complexe, contrairement aux approches concurrentes. Contre toute attente, nous avons montré dans des expériences supplémentaires que l'inclusion d'un critère de sélection des objectifs basé sur la difficulté comme dans le SVGG pour gagner en contrôle interfère avec l'apprentissage de représentations correctes, empêchant les gains de couverture de succès. Des expériences supplémentaires sont nécessaires pour trouver un critère de sélection des objectifs adéquat afin de maximiser le contrôle dans ce contexte spécifique.

Chapter 1

Introduction

Research in Artificial Intelligence (AI) is currently engaged into a race to design and exhibit autonomous agents endowed with more and more versatility in domains as varied as computer vision, medicine or robotics (Bommasani et al., 2021; Moor et al., 2023; Firoozi et al., 2023).

A long term objective is to develop generalist agents, able to quickly adapt to new downstream tasks without the need for human supervision, following the footsteps of recent successes reached in Computer Vision (CV) (Chen et al., 2020; Radford et al., 2021) and Natural Language Processing (NLP) (Radford et al., 2019; Brown, 2020). These successes have been obtained by leveraging tasks-agnostic and highly effective unsupervised techniques. The ultimate goal is to provide humans a multi-task assistant under the form of software agents or even physical robots.

Such a versatile agent should exhibit a number of key capabilities:

- **representation learning**: it should be able to acquire a relevant representation of what matters in its environment from observations coming from any available sensors. In particular, a robotic agent should be capable of building such a representation from images acquired with one or several cameras.
- **autonomous task learning**: it should be capable of determining on its own a sequence of actions in the future that let it achieve a given task.
- **autonomous task setting**: it should be able to determine its own tasks without supervision, and the order in which to learn these tasks so as to enhance its performances (Colas, 2021; Colas et al., 2022).
- **open-ended learning**: it should be capable of learning to achieve new tasks in its environment along its lifetime as they come, without any bounds in the number of tasks (Team et al., 2021; Sigaud et al., 2023).

- **interactivity**: it should be capable of understanding the intent of its user from various modalities, such as language, face expression or body posture, to interact or collaborate with this user through similar modalities, etc. (Akalin and Loutfi, 2021; Sigaud et al., 2022).

This list of capabilities is certainly not exhaustive. In this thesis, our intent is not to build a versatile agent exhibiting all the above capabilities, as it remains a long term objective that is currently out of reach. Rather, we consider the first three capabilities and investigate some of the challenges that their combination raises, under the light of rather simplistic environments and tasks. To account for autonomous task learning, we adopt the framework of Goal-Conditioned Reinforcement Learning (GCRL) (Kaelbling, 1993; Liu et al., 2022).

In more detail, our work along this line is split into two stages, corresponding to two separate contributions.

In this first stage, we put representation learning aside and consider an agent that has access to a low dimensional and accurate knowledge of its state. In this context, autonomous tasks learning requires that the agent discovers by itself the goals that can be reached in its environment and that it does so incrementally while learning to master each of the discovered goals. This combination of requirements results in a tension between the need to master the discovered goals, which tends to make the agent conservative, and the need to discover new goals, which may destabilize the knowledge acquired so far (Campos et al., 2020). As a consequence, the distribution of the goals the agent is addressing at a time should evolve according to a carefully designed dynamics, called a curriculum (Colas et al., 2018). In this first stage, we tackle the goal discovery versus goal mastery trade-off by leveraging two components: (1) learning a predictive model of whether the agent can reach a given goal or not and (2) letting this agent target goals for which its prediction is the most uncertain. To flesh out this contribution, we design a framework called Stein Variational Goal Generation (SVGG). In this framework, the predictive model of goal reaching capabilities is a neural network trained from attempts of the agent to reach goals. To let the agent sample new goals, we rely on a mathematical tool called Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016) where the current goal distribution is represented as a set of particles that are attracted by the most uncertain goals, and that tend to be repelled away from each other, so as to cover the goal space as uniformly as possible. We thoroughly investigate the properties of this framework in simple two-dimensional mazes and then show that it performs well in more difficult and higher dimensional environments.

In the second stage, we insert representation learning into the picture and investigate the additional challenges that this insertion raises. Instead of

considering an agent having access to an accurate state representation, we consider it observes an image from the environment and must build its own representation of the state from this image. A lot of works in this domain sequentially separate a representation learning stage from the task achievement learning stage, but this approach is hardly compatible with autonomous tasks setting, where the agents should build its own representations of tasks and environments as they discover them. However, when representation and task learning processes are carried out jointly, several new issues arise, such as the fact that task learning is based on a non-stationary latent state representation, leading to more instability in task mastery, or the fact that a latent representation may not correspond to any possible goal, leading to a difficulty in measuring task achievement capabilities. More importantly, we exhibit a vicious circle leading autonomous agents to select tasks that already possess an accurate representation, preventing further exploration on new tasks. To mitigate this major limitation, we call upon the Distributionally Robust Optimization (DRO) framework (Rahimian and Mehrotra, 2019), which forces the representations to be robust to changes in the future task distribution. We insert DRO into an autonomous agent learning its own representations of the state online, resulting in a framework called Distributionally Robust Optimization for Image-Driven Exploration (DROIDE).

Again, we investigate these issues with simple two-dimensional mazes to get a deeper sense of the phenomena that arise in this context and show that DROIDE allows the agent to learn representation robust to unseen tasks and thus overcoming the exploration bottleneck by mastering them.

As illustrated in Figure 1.1, the thesis is organized into two main parts. The first part gives some background about the topics that are relevant to our contributions. In Chapter 2, we give a broad overview of relevant Reinforcement Learning (RL) concepts, then we turn to multi-task policies and GCRL. After that, we present Unsupervised RL and Intrinsically motivated agents.

The second part presents our contributions. Chapter 4 focuses on the case where the agent has access to a compact and accurate state representation and describes the Stein Variational Goal Generation (SVGG) framework, corresponding to a paper published at ICML 2023. Chapter 5 then addresses the combination of state representation learning from images with intrinsically motivated goal conditioned agents.

In our conclusion, we take a step back to discuss the opportunities provided by autonomous agents in RL regarding real world applications. Then, we outline some limitations of this work and provide a perspective to learn a Riemannian metric in environments adopting a manifold perspective on RL.

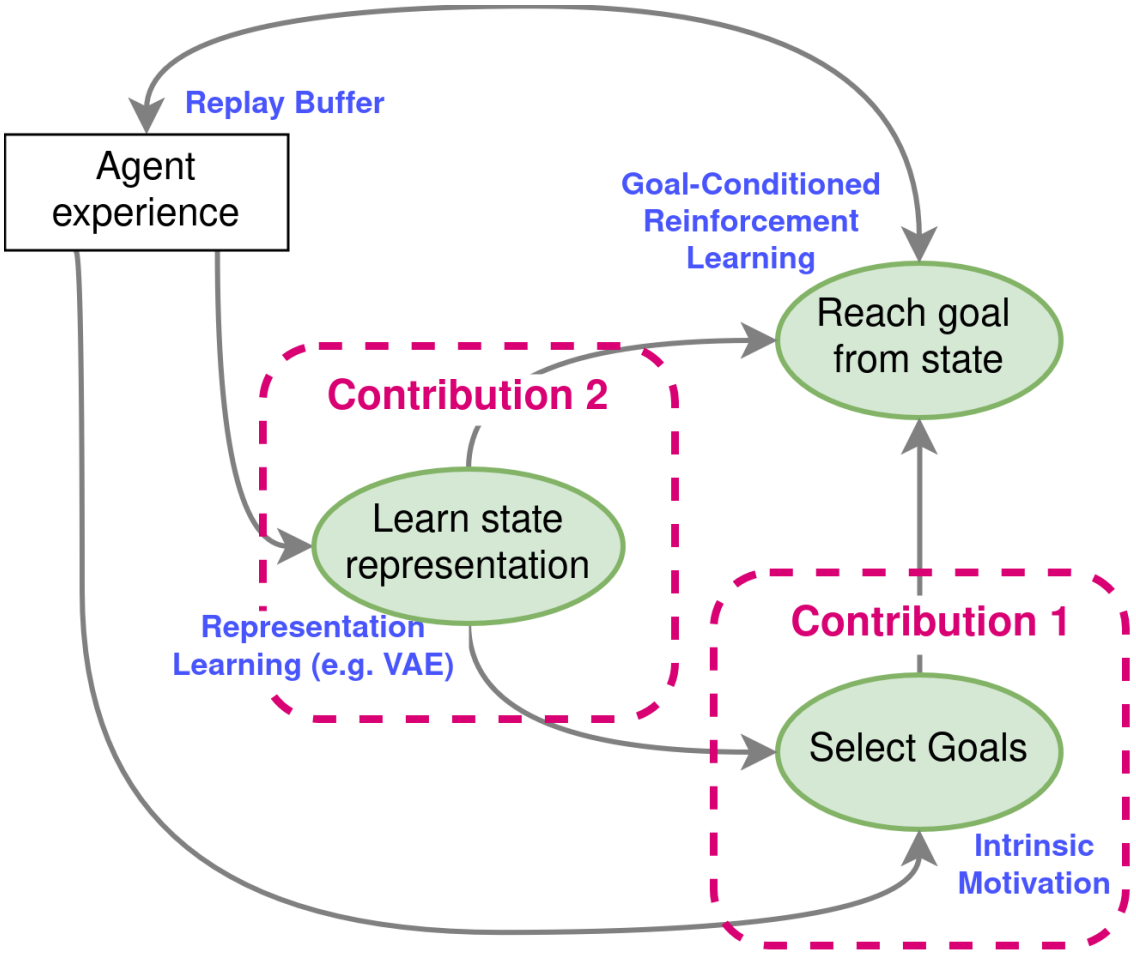

Figure 1.1: This thesis tackles unsupervised goal-reaching tasks. Our setting imposes to design automatic techniques propelled by the agent's experience with the environment. The agent learns to select goals without supervision through the lens of intrinsic motivation (Contribution 1), and learns to reach them via Goal-Conditioned RL. In addition, when the environment lacks semantically meaningful states (e.g. with pixels observations), the agent must extract useful and compact representations (Contribution 2).

Chapter 2

Background

In this chapter, we introduce the necessary tools for presenting our work. First and foremost, we give a glimpse of Reinforcement Learning (Sutton et al., 1999; Sutton and Barto, 2018), from core principles involving notions of Markov Decision Process, reward and behavioral policy, to contemporary algorithms involved in continuous control fueled by Deep Neural Networks. In particular, we give some insight on the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2015), which will be our back-bone policy learning algorithm throughout this Thesis.

Next, we introduce Goal-Conditioned Reinforcement Learning (GCRL) Kaelbling (1993), which is the fundamental framework used in this work to extend classic RL policies to reach multiple goals. We show the limits of classic GCRL methods for learning complex tasks in the absence of prior knowledge on the environment, and then introduce the concept of intrinsic motivation (Oudeyer et al., 2007) as a solution to overcome these limits.

Finally, we address the necessity for representation learning techniques in RL when dealing with high-dimensional observations such as pixels, which is the case in our second contribution. From the incorporation of representation learning in the behavioral policy to more sophisticated and efficient unsupervised learning techniques such as VAE (Kingma and Welling, 2013) or contrastive learning (Oord et al., 2018), we paint the picture of the needs and use of dimensionality reduction techniques in RL, especially related to the framing of GCRL.

Contents

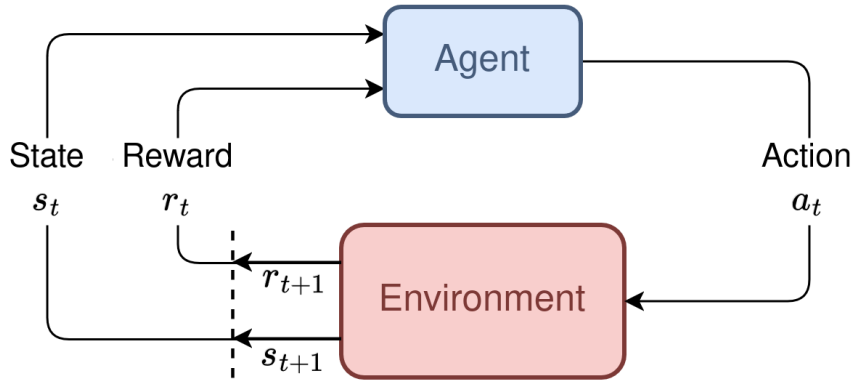

Figure 2.1: Representation of the interaction of the agent with the environment in Reinforcement Learning (adapted from Sutton and Barto (2018))

2.1 Reinforcement Learning

Reinforcement Learning is a sequential decision-making process, where an *agent* interacts with an *environment* and must learn to take actions in order to maximize a specified quantity named *reward*, as pictured in Figure 2.1. At each time step, the agent finds itself in a given state and must take an action, the environment then provides a reward to the agent and updates its current state.

In this section, we first introduce the fundamental frameworks and algorithms of RL related to the contributions of this thesis, from the core principles. Then we present a brief topology of existing RL methods to Deep Reinforcement Learning (DRL) using neural networks for continuous control.

2.1.1 Core principles

We start with the mathematical framework to define a reinforcement learning problem, based on Markov Decision Processes. We then introduce the notions of policies and return, and finally value functions that are central to the quest for optimal policies.

Markov Decision Process

A Markov Decision Process (MDP) (Bellman, 1966; Sutton, 1988) is a mathematical framework used for modeling decision-making problems.

Definition 2.1.1. A *Markov Decision Process* is a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$, where:

- $\mathcal{S}$ is the *state space*, which can be finite or infinite, discrete or continuous. Every element $s \in \mathcal{S}$ is a possible *state* of the environment.
- $\mathcal{A}$ is the *action space*, which represents all possible actions available in the state $s \in \mathcal{S}$. The action space can be constrained to the current state of the MDP, it is then denoted as $\mathcal{A}(s)$. Similarly to the state space S, A can be continuous or discrete.
- T is the *transition function*, defined as a probability distribution over next states s', given a current state s and an action a:

$$\begin{aligned} T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} &\to [0, 1] \\ (s'|a, s) &\mapsto \mathbb{P}(s_{t+1} = s'|a_t = a, s_t = s). \end{aligned}$$

  $T(s'|s, a)$ represents the probability of transitioning to state s' from state s after taking action a.
- $\mathcal{R}$ is the *reward function* and is defined as:

$$\begin{aligned} \mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} &\to \mathbb{R} \\ (s', a, s) &\mapsto r(s_{t+1} = s', a_t = a, s_t = s). \end{aligned}$$

  $r(s', a, s)$ represents the learning signal provided to the agent for taking the action a in the state s and ending up next in the state s'.
- γ is the *discount factor*, it is a parameter between 0 and 1 which determines the importance of future rewards compared to immediate rewards. For instance, a discount factor close to 1 means that future rewards are considered almost as valuable as immediate rewards, leading the agent to value long-term benefits. Note that the discount factor is sometimes considered as a parameter of the agent rather than of the environment.

Return & Policy

In reinforcement learning, ideally, an agent would seek to maximize its long term reward rather than having a short-term vision. Think of it as a tennis game, an intelligent player understands that it is more important to win the game than to win the next point at all costs, and would act accordingly. With

that in mind, the quantity that the agent should maximize is not the next step reward signal but rather the sum of future potential reward, denoted as the *Return*. Among the possible definitions of the return, in this thesis we focus on the *Discounted Return*:

Definition 2.1.2. The *Discounted Return* at timestep t, denoted as G_t, is defined as the sum of future discounted rewards:

$$G_t = r_{t+1} + \gamma r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

In Reinforcement Learning, an agent can be formalized as a *policy*, a mathematical function that maps states to actions or probabilities over actions in order to interact with the environment:

Definition 2.1.3. A *policy*, denoted as π, is a function defined over the state space S of an MDP:

$$\begin{aligned} \pi : \mathcal{S} &\to \mathcal{P}_s \\ s &\mapsto \mathbb{P}(\mathcal{A}_s) \end{aligned}$$

where $\mathcal{A}_s \subset \mathcal{A}$ is a subset of available actions in state s and $\mathcal{P}_s$ is a probability distribution over actions in state s. A policy π can be stochastic or deterministic.

Value Functions

In order to evaluate and improve a policy π, we use the *State-Value* function, which aims to estimate the policy's ability to maximize the reward signal from a given state s:

Definition 2.1.4. The *State-Value* function is defined as the expected Return from state s under policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s].$$

The *Action-Value* function extends the *State-Value* by including the action a taken in the current state s as an additional variable:

Definition 2.1.5. The *Action-Value* function is defined as the expected Return of taking the action a in the state s and then following policy π:

$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a].$$

The action and state value functions are related to one another in the following way:

Result 2.1.1. Relations between V^π and Q^π:

$$\begin{aligned} V^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) Q^\pi(s,a) \\ Q^\pi(s,a) &= \sum_{s'} T(s'|a,s)[r(s,a,s') + \gamma V^\pi(s')] \end{aligned} \tag{2.1}$$

Proof. Development of the expectation in definitions 4 and 5, see Puterman (2014).

Optimal Policies and Bellman equations

The state and action value functions are performance metrics of the current policy that are used to move it toward the optimal policy through a variety of methods, which we discuss later in this section. A policy is defined as optimal if and only if its state and value functions are optimal as well:

Definition 2.1.6. *Optimality Conditions*:

- The *Optimal State Value Function* V^* is defined as the maximum state value function over every possible policies:

$$\forall s \in \mathcal{S}, \quad V^*(s) = \max_{\pi \in \Pi} V^\pi(s).$$

- The *Optimal Action Value Function* Q^* is defined as the maximum action value function over every possible policies:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad Q^*(s,a) = \max_{\pi \in \Pi} Q^\pi(s,a).$$

- The *Optimal Policy* denoted as π^* is defined as the policy achieving the optimal state and action value functions:

$$\pi = \pi^* \iff V^\pi = V^* \iff Q^\pi = Q^*.$$

Brute force calculations of $V^\pi(s)$ and $Q^\pi(s)$ are ineffective if not impossible in case of large state and action space. Bellman's equation gives us a nice recursive formulation of $V^\pi(s)$ and $Q^\pi(s)$, that can be expressed as a function of state and value function of the next potential state $V^\pi(s')$ and $Q^\pi(s')$, following the dynamics of the MDP:

Result 2.1.2. The *Bellman equations* for function V^π and Q^π:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi[r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s] \\ Q^\pi(s,a) &= \mathbb{E}_\pi[r_{t+1} + \gamma \mathbb{E}_{a'\sim\pi} Q^\pi(s_{t+1}, a')|s_t = s, a_t = a]. \end{aligned} \tag{2.2}$$

Proof. Result of the combinations between the two equations in (2.1) in both ways, rewriting $r(s,a,s') = r_{t+1}$.

Optimality conditions of the policy can be further embedded into the Bellman equations:

Result 2.1.3. The *Bellman optimality equations* for function V^* and Q^*:

$$\begin{aligned} V^*(s) &= \max_a \sum_a T(s'|a,s)[r(s,a,s') + \gamma V^*(s')] \\ Q^*(s,a) &= \sum_{s'} T(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q^*(s',a')]. \end{aligned} \tag{2.3}$$

Proof. Application of the Bellman equations (2.2) with the optimal policy π^*.

These equations can be considered as the back-bone of Reinforcement Learning algorithms, used to compute the value functions through dynamic programming, thus obtaining significant computational gains compared to brute force calculation.

The Bellman equations are also employed to define the *temporal difference*:

Definition 2.1.7. The *Temporal difference*, denoted as ΔT is the error at time step t for the transition $(s_t, a_t, s_{t+1}, r_{t+1})$ obtained in the Bellman equations in case of bad estimation of the value functions

$$\begin{aligned} \Delta T(V^\pi) &= r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t) \\ \Delta T(Q^\pi) &= r_{t+1} + \gamma \max_{a'} Q^\pi(s_{t+1}, a') - Q^\pi(s_t, a_t) \end{aligned} \tag{2.4}$$

The minimization of the temporal difference error is used to approximate the value functions at each time step through interactions with the environment, when no knowledge is available about the MDP (i.e., unknown T or R func-

tions). From the value function, one can infer a policy or improve it by a great variety of techniques. We give a brief overview of existing RL approaches in the next section.

2.1.2 Typology of RL Algorithms

Based on the above core principles equations of RL, many methods have been explored in the research community to develop algorithms able to efficiently infer optimal policies. From these efforts comes the crucial question: what RL algorithm shall I use? Many variables must be taken into account, such as the nature of the state and action space, whether it is continuous or discrete and many environment peculiarities. A typology of existing options is given in Figure 2.2. The first key distinction is between *Model-Based* and *Model-Free* RL algorithms, which indicates whether the transition function of the environment T is modeled at training time.

Model-Based RL

Model-Based methods assume the access to the dynamics of the MDP through the transition function, and thus are able to conduct long term simulations and predictions without direct interaction with the environment. When provided with T, a traditional approach is to use dynamic programming with the Bellman equations in order to evaluate and improve the value functions (Sondik, 1971). Another possibility is to use Monte Carlo Tree Search (MCTS) (Coulom, 2006) on entire trajectories to find out the expected long term outcomes of an action in order to act optimally (Silver et al., 2017). When T is unknown, long term planning can still be achieved on a learned model of the environment dynamics T_θ. Widespread implementations of T_θ include Gaussian processes (Deisenroth and Rasmussen, 2011), Neural Networks (Chua et al., 2018; Schrittwieser et al., 2020), or Recurrent State-Space Models (Hafner et al., 2019a).

Model-Free RL

Facing the difficulty of learning an accurate model of the dynamics of the MDP, Model-Free RL algorithm powered by Neural Networks became widely used for their ability of learning complex behavior relying on environment interaction, without long term planning. Model-Free algorithms became popular in the past decades when shown to be easier to train and to outperform significantly their model-based counterparts on many tasks.

Model-Free RL Algorithms can be classified into two categories. First, value-based algorithms focus on learning the value functions, relying on the

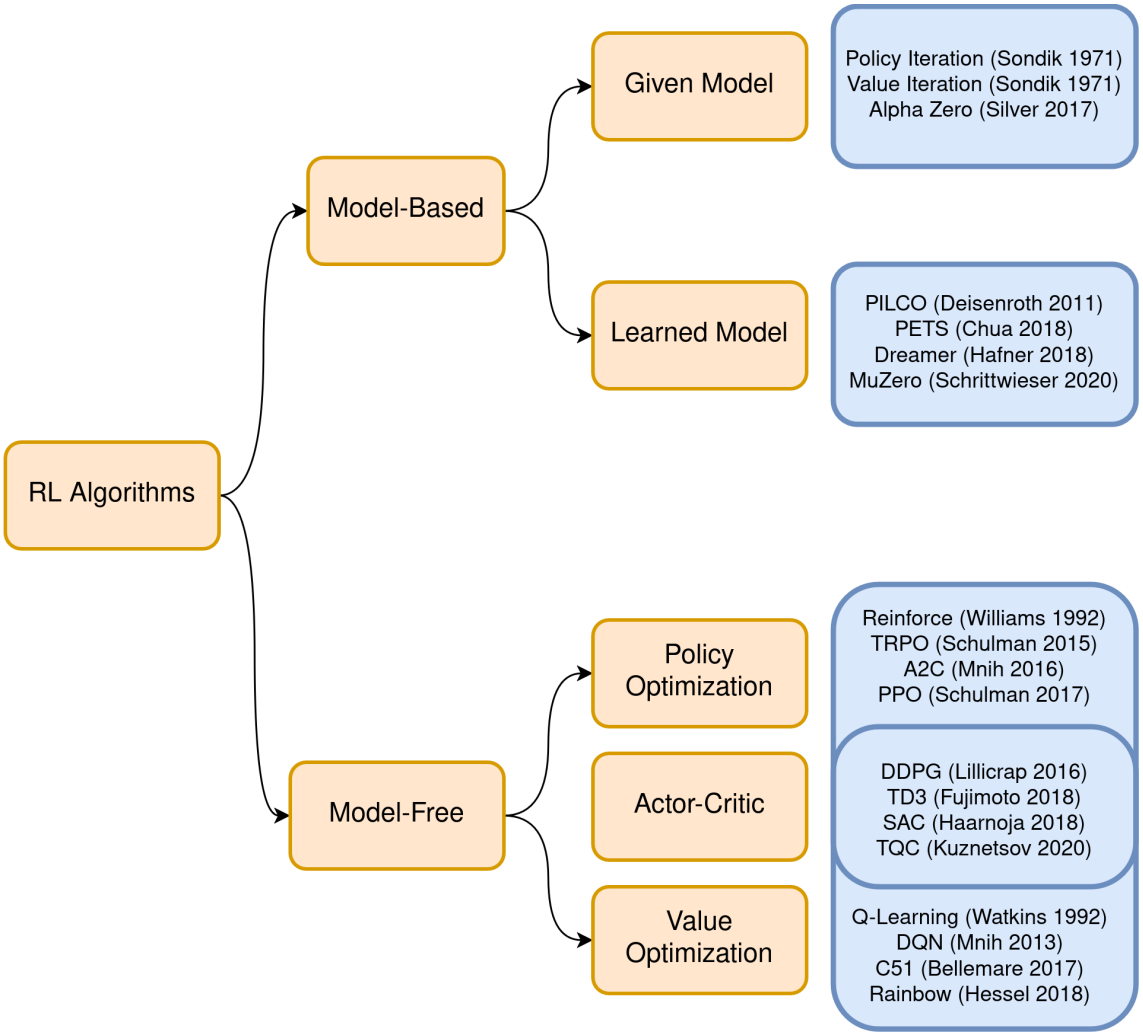

Figure 2.2: Typology of RL Algorithms. Model-based RL relies on a model of the environment (learned or given) to plan actions by predicting future states and rewards, while model-free RL directly learns the optimal policy or value function from interactions without needing an environment model. Actor-Critic algorithms learn both policy and a value function that is commonly referred to as a "critic".

Bellman equations (Watkins and Dayan, 1992; Mnih et al., 2013; Bellemare et al., 2017; Hessel et al., 2018), where the policy is deduced thereafter from the value function. Second, policy-based algorithms, which learn the behavioral policy directly, relying on gradient ascent of the policy's outcomes (Williams, 1992; Schulman, 2015; Mnih, 2016; Schulman et al., 2017). At the intersection of these two categories are actor-critic algorithms, that learn both the value function and the policy in an interactive manner, hoping to have the best of both world: training stability of value based methods and behavioral expressiveness for continuous control of policy-based methods (Lillicrap et al., 2015; Fujimoto et al., 2018; Haarnoja et al., 2018; Kuznetsov et al., 2020).

Our scope

In this thesis, we tackle **continuous control problems**, as they provide a greater challenge and variety of applications in real world scenarios (e.g. robotics) conversely to discrete control that is mainly restricted to video games (e.g. Atari games) and specific software programs. This framing implies that both the action and state space of the MDP are continuous. For this reason, we turn to policy-based algorithms, powered by Neural Networks mapping states to actions, allowing fine-grained modeling of the behavior of the agent, and optimization through **policy gradient** techniques.

Furthermore, policy gradient methods can be classified in two categories: **On-Policy** vs **Off-Policy**. On one hand, on-policy algorithms compute the gradient for optimization on experiences collected from the current behavioral policy. On the other hand, off-policy algorithms extend the objectives in policy gradient to include experiences collected from different policies in the optimization process (in particular samples collected from past versions of the policy stored in a **Replay Buffer**). Both approaches have pros and cons. In summary, on-policy methods tend to be more stable while off-policy methods are more sample efficient, refer to Sutton and Barto (2018) for a deeper dive into this subject.

More importantly, off-policy algorithms possess the appealing property to share experiences between multiple policies, which is a major asset for our objective of building multi-task autonomous agents with GCRL. As a matter of fact, being off-policy is key to exploit trajectories for different goals to transfer knowledge and building stepping stones, which results in a massive gain both in performance and sample efficiency when it comes to learn goal-conditioned policies (Andrychowicz et al., 2017).

In the next section, we pave the way from the basics of policy gradient to a back-bone off-policy algorithm for continuous control: Deep Deterministic Policy Gradient (DDPG) Lillicrap et al. (2015), which we use for policy learning throughout this thesis.

2.1.3 Continuous Control with Policy Gradient: from basics to Deep Deterministic Policy Gradient (DDPG)

In Policy-Gradient algorithms, the policy is defined as a continuous function parametrized by θ, mapping states to actions:

Definition 2.1.8. *Policy parametrized by* θ:

$$\begin{aligned} \pi_\theta : \mathcal{S} &\to \mathcal{P}_s \\ s &\mapsto \pi_\theta(.|s) = \mathbb{P}(\mathcal{A}_s). \end{aligned}$$

The policy can be approximated by any differentiable parametric function, using a Neural Network is common. We measure the performance of the policy using the following objective function:

Definition 2.1.9. The *Objective function* for policy π_θ is defined as the expectation of the state value function in the initial state s_0:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[V^{\pi_\theta}(s_0)].$$

A fundamental result allowing to use gradient ascent from interactions of the policy with the environment for optimization is the policy gradient theorem:

Result 2.1.4. *Policy Gradient Theorem:*

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)]. \tag{2.5}$$

Proof. Refer to Sutton and Barto (2018) Sec. 13.1.

This formulation of policy gradient is denoted as on-policy, as the expectation $\mathbb{E}_{\pi_\theta}$ implies the use of the current policy to compute $\nabla J(\theta)$.

Off-Policy Policy Gradient

The off-policy setting extends the objective function to include data coming from a different policy β called the behavior policy:

Definition 2.1.10. The *Off-Policy Objective function* for policy π_θ is defined as the expectation of $V^{\pi_\theta}(s_0)$ under trajectories sampled from a distinct behavior policy:

$$J_\beta(\theta) = \mathbb{E}_\beta[V^{\pi_\theta}(s_0)].$$

The modification of the objective implies the following changes in the policy gradient theorem, including importance sampling:

Result 2.1.5. *Off-Policy Gradient Theorem:*

$$\nabla J_\beta(\theta) = \mathbb{E}_{s\sim d^\beta, a\sim\beta}\Big[\frac{\pi_\theta(a|s)}{\beta(a|s)} Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\Big]. \tag{2.6}$$

where d^β is the distribution of states reached as the result of policy β's interactions with the environment. *Proof.* Refer to Degris et al. (2012).

Furthermore, Silver et al. (2014) showed that this gradient formulation can be simplified by removing the importance sampling factor using a deterministic policy:

Result 2.1.6. *Deterministic Off-Policy Gradient Theorem:*

$$\nabla J_\beta(\theta) = \mathbb{E}_{s\sim d^\beta}\Big[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\Big] \tag{2.7}$$

Proof. Refer to Silver et al. (2014).

Deep Deterministic Policy Gradient (DDPG)

The DDPG algorithm Lillicrap et al. (2015) operates in continuous actions spaces by modeling the policy as a parametric deterministic function mapping a continuous state to a unique continuous action:

$$\begin{aligned} \pi_\theta : \mathcal{S} &\to \mathcal{A} \\ s &\mapsto \pi_\theta(s). \end{aligned}$$

The use of a deterministic policy infers the following changes into the Bellman equation for the Q-value (2.2):

$$\begin{aligned} Q^{\pi_\theta}(s,a) &= \mathbb{E}_{\pi_\theta}[r_{t+1} + \gamma \mathbb{E}_{a'\sim\pi_\theta} Q^{\pi_\theta}(s_{t+1}, a')|s_t = s, a_t = a] \\ &= \mathbb{E}_{\pi_\theta}[r_{t+1} + \gamma Q^{\pi_\theta}(s_{t+1}, \pi_\theta(s_{t+1}))|s_t = s, a_t = a]. \end{aligned} \tag{2.8}$$

Taking inspiration from the Deep Q-Network algorithm (Mnih et al., 2013), DDPG approximates the Q-value with a Neural Network, trained to minimized the temporal difference induced by the Bellman equations on transitions sampled from a replay buffer:

$$\begin{aligned}\mathcal{L}(\phi) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\Big[\Big(Q^{\pi}_{\phi}(s,a) - \Big(r + \gamma Q^{\pi}_{\phi}(s', \pi_{\theta}(s'))\Big)\Big)^2\Big]. \\ \phi &\leftarrow \phi - \nabla_{\phi}\mathcal{L}(\phi)\end{aligned} \tag{2.9}$$

To update the actor network, we compute the gradient of the Q-value with respect to the actor parameters. The deterministic off-policy gradient theorem (2.7) gives us:

$$\nabla J_{\beta}(\theta) = \mathbb{E}_{s\sim d^{\beta}}\Big[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q^{\pi_{\theta}}_{\phi}(s,a)|_{a=\pi_{\theta}(s)}\Big]. \tag{2.10}$$

We then perform a step of gradient ascent with the following gradient on a batch B of trajectories from the replay buffer:

$$\theta \leftarrow \theta + \nabla_{\theta}\frac{1}{|B|}\sum_{s\in B} Q^{\pi_{\theta}}_{\phi}(s, \pi_{\theta}(s)).$$

In practice, DDPG employs soft updates to the target actor and critic networks (a slowly updated copy of the main actor and critic) by blending the target networks parameters with the current networks parameters. This trick improves the stability and convergence of the learning process, avoiding rapid shifts in target values.

DDPG iteratively improves the policy by directing it toward actions that yield higher expected returns according to the critic's evaluation.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q_\phi(s, a)$ and actor $\pi_\theta(s)$ parametrized by ϕ and θ.
Initialize Q_ϕ and π_θ target network with parameters $\phi^{\text{targ}} \leftarrow \phi$, $\theta^{\text{targ}} \leftarrow \theta$.
Initialize replay buffer $\mathcal{B}$.
for episode $= 1, M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration.
  Receive initial observation state s_0.
  for $t = 0, T$ **do**
    ▶ ***Environment interaction***
    Select action $a_t = \pi_\theta(s_t) + \mathcal{N}_t$ according to the current policy and exploration noise.
    Execute action a_t and observe reward r_t and new state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$.
    ▶ ***Critic/Policy Learning***
    Sample a batch of N transitions $B = \{(s, a, r, s')\}$ from $\mathcal{B}$.
    Set target $y = r + \gamma Q_{\phi^{\text{targ}}}(s', \pi_{\theta^{\text{targ}}}(s'))$.
    Update critic by minimizing the temporal difference:

$$\phi \leftarrow \phi - \nabla_\phi \frac{1}{N} \sum_B (y - Q_\phi(s, a))^2.$$

    Update the actor policy using the sampled policy gradient:

$$\theta \leftarrow \theta + \nabla_\theta \frac{1}{N} \sum_B Q_\phi(s, \pi_\theta(s)).$$

    Update the target networks:

$$\phi^{\text{targ}} \leftarrow \tau\phi + (1 - \tau)\phi^{\text{targ}}.$$

$$\theta^{\text{targ}} \leftarrow \tau\theta + (1 - \tau)\theta^{\text{targ}}.$$

  end for
end for

2.2 Multi-Goal Reinforcement Learning

In this section, we address Multi-Goal Reinforcement Learning. We first describe Goal-Conditioned Reinforcement Learning (GCRL), the standard framework to learn multipurpose goal-directed policies. Then we provide a description of Hindsight Experience Replay (HER), a major breakthrough in off-policy Goal-Conditioned RL, unlocking the possibility to learn from failed attempts, and to share experiences between multiple goals. Finally, we exhibit the limitations of classical RL algorithms such as DDPG when learning complex tasks.

2.2.1 Goal-conditioned Reinforcement Learning

Goal-Conditioned Reinforcement Learning (GCRL) Kaelbling (1993); Schaul et al. (2015) is an extension of the classic RL framework. While in RL the agent takes actions based on the current state s, in this new framework, the agent's behavior is conditioned both by its current state s and the goal it is pursuing g. The goal itself can be generated by the agent, or it can come from the environment. In the latter case, the MDP is augmented with a *goal space* $\mathcal{G}$. The agent is now represented with a *goal-conditioned policy*:

Definition 2.2.1. A *goal-conditioned policy* (GCP), denoted as $\pi(.|s, g)$, is a function that maps the action distribution from the MDP state space $\mathcal{S}$ and goal space $\mathcal{G}$:

$$\begin{aligned} \pi : \mathcal{S} \times \mathcal{G} &\to \mathcal{P}_s \\ (s, g) &\mapsto \pi(.|s, g) = \mathbb{P}(\mathcal{A}_s). \end{aligned}$$

Every state in the environment corresponds to a specific goal. In the most general case, the relation between $\mathcal{S}$ and $\mathcal{G}$ can be expressed as follows:

$$\begin{aligned} f_{\mathcal{G}} : \mathcal{S} &\to \mathcal{G} \\ s &\mapsto f_{\mathcal{G}}(s). \end{aligned}$$

In this thesis, we adopt the most straightforward definition of the goal space, $f_{\mathcal{G}} = I$, where $\mathcal{G} = \mathcal{S}$, every state s corresponds to a potential goal to achieve: $f_{\mathcal{G}}(s) = s$. For the goal-conditioned policy to learn how to act in order to reach the intended goal, every goal g has to be paired with a specific reward function $\mathcal{R}_g$, tracking progress toward the goal. The classic

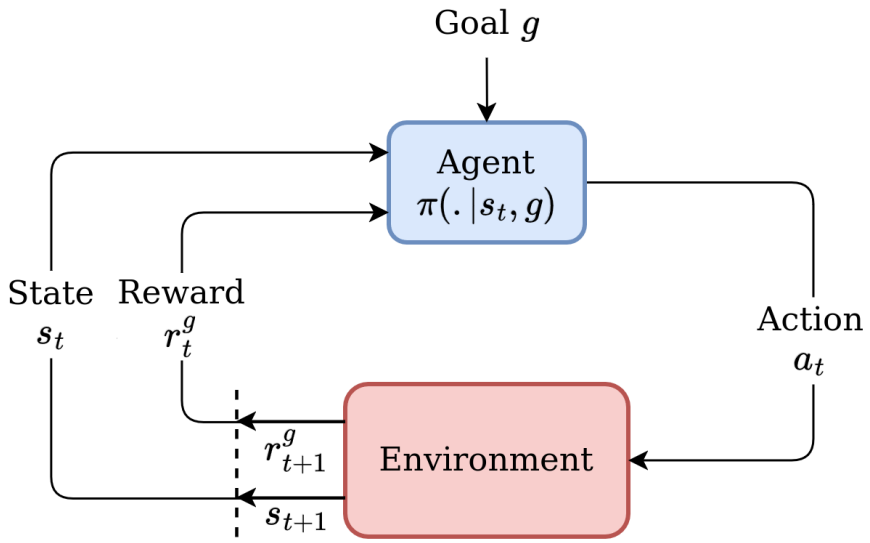

Figure 2.3: The goal conditioned reinforcement learning framework considered in this thesis. The agent takes actions according to its current state and the goal it has to reach. The goal is generated once at the beginning of every episode and the reward is conditioned on the goal being reached.

RL problem can then be extended to handle multiple goals, which can be formulated within the more general framework of a *contextual MDP* Hallak et al. (2015) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, \mathcal{R}_g, \gamma\}$.

Definition 2.2.2. Goal-conditioned *Sparse Reward Function*:

$$\mathcal{R}_g : \mathcal{S} \to \{0, 1\}$$
$$s \mapsto \mathcal{R}_g(s) = \mathbb{1}\Big[||s - g||_2 < \delta\Big],$$

δ being an hyper-parameter controlling the $L2$ ball centered at g inside which we consider the targeted goal to be achieved. Note that this definition does not involve the action taken by the agent nor its previous state, the objective is constrained to reach the goal regardless of the manner.

Other formulations of the goal-conditioned can be used, such as a dense version of the latter function:

Definition 2.2.3. Goal-conditioned *Dense Reward Function*:

$$\begin{aligned} \mathcal{R}_g : \mathcal{S} &\to \{0, 1\} \\ s &\mapsto \mathcal{R}_g(s) = ||s - g||_2. \end{aligned}$$

However, the sparse reward version is far more popular, as a major issue arises when using the dense formulation which is *deceptive reward*. These phenomena arise when the $L2$ direction does not match with the optimal direction to reach the goal, due to specific topology of the environment. A classic example is in maze environments, when the agent has to get away from the goal in the first place to bypass walls in order to eventually reach the goal. We refer to this particular configuration as *goal space discontinuities*.

2.2.2 Relabelling with Hindsight Experience Replay

A major advantage of learning a goal-conditioned policy with respect to using standard RL is that it makes it possible to leverage a specific mechanism called Hindsight Experience Replay (HER) (Andrychowicz et al., 2017). In a few words, the mechanism is often referred to as "learning from failures". When an agent targeting a given behavioral goal g_b fails and ends-up achieving another goal g_a, one can consider that the corresponding trajectory would be successful if the target was in fact g_a. Thus the agent can be rewarded (or can reward itself) for this accidental "success". In practice, HER is used in combination with off-policy RL algorithms that use a replay buffer. The replay buffer contains transitions of the agent in the environment conditioned on the behavioral goal. The HER relabelling mechanism consists in replacing in the samples of the replay buffer the behavioral goal g_b by the achieved goal g_a, and changing the corresponding reward accordingly.

The HER mechanism is particularly helpful in sparse reward contexts where desired goals are hard to reach, as using it results in increasing the density of the reward signal. HER is thus used in many GCRL algorithms.

A key issue in HER is that the replay data can be relabelled using any goal reached along the agent trajectory, not necessarily the goal corresponding to the end of the trajectory, but relabelling with diverse goals can be more or less efficient. Thus, one can design a curriculum over the goals used for relabelling, which is the topic of the Curriculum Hindsight Experience Replay (CHER) paper (Fang et al., 2019b).

Other extensions to HER have been proposed in the literature, such as Dynamic Hindsight Experience Replay (DHER) (Fang et al., 2019a), Competitive Experience Replay (CER) (Liu et al., 2019), Model-based Experience Replay (MHER) (Yang et al., 2021) and many others that are not listed here.

Besides, while the standard HER mechanism requires an off-policy RL algorithm equipped with a replay buffer, one can also apply it to on-policy policy gradient methods, as shown in Rauber et al. (2017).

Algorithm 2 Goal-Conditioned DDPG algorithm with HER

Input: Goal g^*, randomly initialized critic network $Q_\phi(s, g, a)$ and actor $\pi_\theta(s, g)$ parametrized by ϕ and θ, goal reaching threshold parameter δ.
Initialize Q_ϕ and π_θ target network with parameters $\phi^{\text{targ}} \leftarrow \phi$, $\theta^{\text{targ}} \leftarrow \theta$.
Initialize replay buffer $\mathcal{B}$.
for episode $= 1$, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration.
  Receive initial observation state s_0.
  for $t = 0, T$ **do**
    ▶ ***Environment interaction***
    Select action $a_t = \pi_\theta(s_t, g^*) + \mathcal{N}_t$ according to the current policy and exploration noise.
    Execute action a_t, observe new state s_{t+1} and compute reward $r_t = \mathbb{1}\big[||s_{t+1} - g^*||_2 < \delta\big]$.
    Store transition $(g^*, s_t, a_t, r_t, s_{t+1})$ in $\mathcal{B}$.
    ▶ ***Policy Learning***
    Sample a batch of N transitions $B = \{(g^*, s_t, a_t, r_t, s_{t+1})\}$ from $\mathcal{B}$.
    ▷ *HER*
    for $\{(g^*, s_t, a_t, r_t, s_{t+1})\} \in B$ **do**
      Sample a future achieved state from the same trajectory $\tau = \{(g^*, s_0, a_0, r_0, s_1), ...(g^*, s_t, a_t, r_t, s_{t+1}), ..., (g^*, s_{T-1}, a_{T-1}, r_{T-1}, s_T)\}$: $s^{\text{achevied}} = s_{t+t'}$, with $t' \sim \text{Unif}(t, T)$.
      Replace $(g^*, s_t, a_t, r_t, s_{t+1})$ with $(s^{\text{achieved}}, s_t, a_t, r_t, s_{t+1})$
    end for
    ▷ *Optimize θ and ϕ with DDPG (algo 1)*
  end for
end for

2.2.3 Limitations in complex goal reaching tasks

The combination of off-policy algorithms (e.g. DDPG) paired with HER has been a major breakthrough for solving a great variety of continuous tasks, namely in robotic simulation environments (Andrychowicz et al., 2017; Plappert et al., 2018b).

Nevertheless, some complex goal reaching tasks can remain out of reach for several reasons. For instance, the goal space discontinuities are too important or the goal horizon (number of actions required to reach the goal) is high. We

exhibit these limitations on a 2D maze goal reaching task in Figure 2.4.

This figure reports results of experiments we conducted using DDPG+HER on three different maze topologies, one with no walls, a maze with a few walls and another maze with a more complex topology. Each configuration requires a different level of complexity in the behavioral policy to reach the goal, as can be observed with the optimal trajectory. The results demonstrate several points. First and foremost, DDPG+HER fails to reach the goal on the "hard wall" configuration (even after 10 millions training steps). Secondly, the sparse reward function is far more efficient than its dense counterparts, as can be observed with respect to the "easy walls" configuration.

We need Intrinsic Motivations

This limitation of traditional RL algorithms to perform difficult goal reaching tasks has been addressed through the lens of intrinsic motivations (Oudeyer et al., 2007; Colas et al., 2022), allowing the agent to approach the goal iteratively by building stepping stones towards it, in an autonomous manner.

The objective of intrinsic motivation is to generate relevant intrinsic goals g^i for the policy to pursue at training time: $\pi(.|s, g^i)$, allowing the agent to make progress. This stands out from regular GCRL training where the policy is only trained on a static intended goal g^*: $\pi(.|s, g^*)$, which in many cases cannot be achieved as such, like the maze with "hard walls" in our experiments in Figure 2.4.

In addition, in the absence of intrinsic motivations, we presume the access to a goal of interest g^*, which constitutes valuable prior knowledge on the environment, unlikely to be available in real world scenarios. Therefore, the intrinsic motivation framework allows us to work in unsupervised settings, where we make no assumptions on the environment regarding the space of valid goals, nor on the interesting goals to reach. The objective then extends to explore and reach as many goals as possible, which is agnostic of the environment. We discuss Unsupervised RL and Intrinsically motivated agents in Chapter 3.

In the next section, we take a sidestep to discuss representations learning in RL, which is related to our second contribution.

2.3 Representation Learning for RL

In Reinforcement Learning, real-world environments and complex simulations often produce high-dimensional observations, such as images or sensor data. Directly processing such high-dimensional data poses significant challenges due to the curse of dimensionality, where the exponential growth in volume of

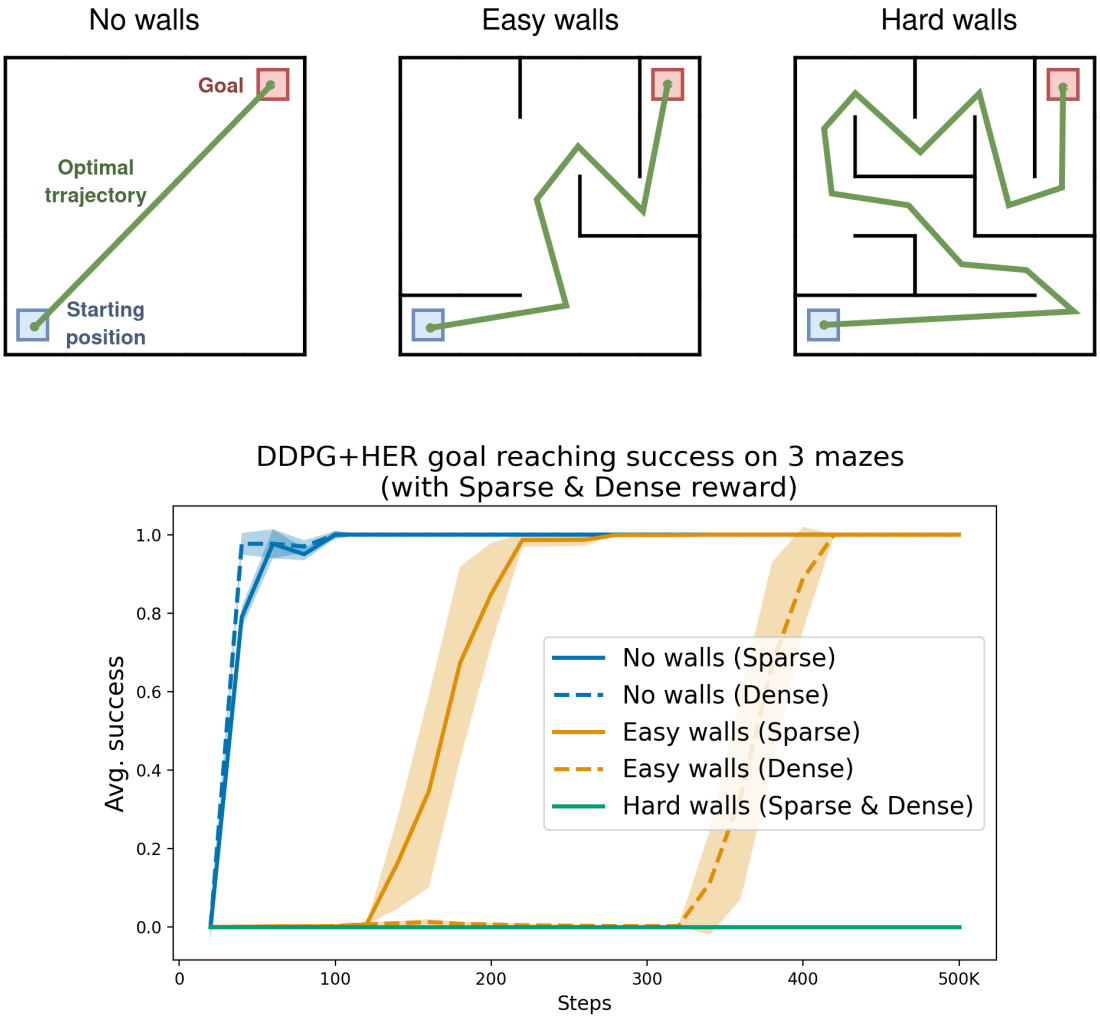

Figure 2.4: Goal reaching experiments for continuous control with DDPG+HER in 2D mazes ($\mathcal{S} = \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^2$, $\mathcal{A} = (\delta x, \delta y) = [0, 1]^2$). Average success for the goal is recorded on three mazes of different topologies, for sparse and dense reward configurations (3 seeds for each run).

the observation space can lead to inefficiencies in learning the policy, difficult reward definition, increased computational requirements, and poor generalization. To address these challenges, it is crucial to learn compact and informative representations of the environment state. Representation learning enables the transformation of high-dimensional observations into lower-dimensional embeddings that capture the main features necessary for decision-making. In this section, we discuss the main representation learning techniques used in RL, and more precisely those used in our Multi-Goal RL setting, which we integrate in the second contribution of this Thesis. For a broader overview on state representation learning in RL, refer to Lesort et al. (2018).

2.3.1 The curse of dimensionality

The curse of dimensionality refers to a set of problems that arise when working with high-dimensional data. As the number of dimensions in a dataset increases, the volume of the space grows exponentially, making the data sparse. This sparsity leads to several challenges:

- **Distance Measures Become Less Informative**: In high dimensions, the difference between the maximum and minimum distances between points becomes negligible, making distance-based methods less effective.
- **Increased Computational Complexity**: The computational cost for algorithms that depend on the number of dimensions (e.g., distance calculations, clustering, etc.) increases significantly.
- **Data Sparsity**: As dimensions increase, the amount of data needed to ensure that there is enough coverage of the space increases exponentially. With limited data, models may struggle to generalize.

Figure 2.5 highlights the curse of dimensionality regarding the behavior of pairwise distances in a dataset: as the data dimension increases, distances between points become less informative as all pairwise distances are restricted to an increasingly limited range. In order to alleviate this issue, many dimension reduction techniques have been developed over the years. These techniques strive to "compress" high dimensional data in order to extract the valuable information embedded in the data that often lie in a very small subspace of the original space of the data. Emblematic dimension reduction techniques include Principal Component Analysis (Wold et al., 1987; Schölkopf et al., 1998), Matrix Factorization (Lee and Seung, 1999), and other modern methods used for visualization as t-Distributed Stochastic Neighbor Embedding (TSNE) (Schubert and Gertz, 2017) and Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018).

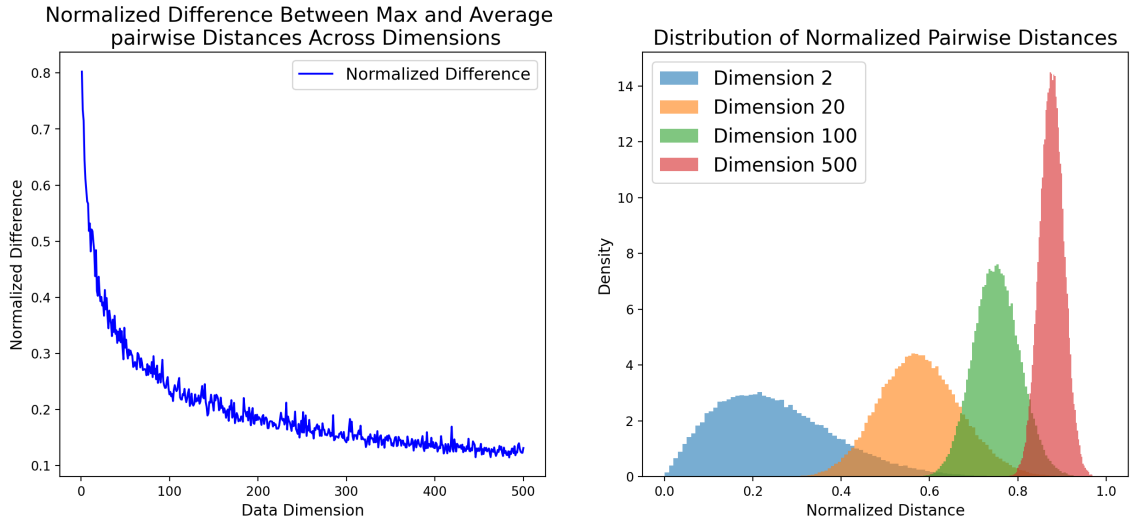

Figure 2.5: Illustration of the curse of dimensionality for a dataset of 500 points sampled from a Normal distribution: **Left:** Evolution of the normalized difference between max and average pairwise difference across dimensions. **Right:** Distribution of normalized pairwise Euclidean distances for dimensions 2, 20, 100 and 500.

How to deal with the curse of dimensionality in RL?

In order to deal with high-dimensional observations in Reinforcement Learning such as screen images in Atari games (Mnih et al., 2015), a standard practice has been to learn latent features jointly with a control policy, in an end-to-end manner, which has been successfully applied to various domains including real-world scenarios (Levine et al., 2016; Kalashnikov et al., 2018), simulated robotics (Laskin et al., 2020b; Kostrikov et al., 2020) and challenging video games (Vinyals et al., 2019). Due to its simplicity, learning features from observations in an end-to-end fashion is appealing. A simple architecture of this process is represented in Figure 2.6.

However, learning latent representations relying on the specific task of making sequential decisions to maximize a reward can be restrictive for several reasons. In our specific GCRL framework, the sparsity of the reward signal can have a negative impact on feature learning. In addition, GCRL is missing a predefined reward function as in Atari games or other environments. As described in Section 2.2.1, the standard reward implementation of goal-reaching task relies on the distance between reached states and goals: $||s_t - g||_2$, which in the case of high dimensional observations is ill-defined due to the curse of dimensionality as described above. Moreover, when learning a Goal-Conditioned Policy, we intend to train it to reach a wide variety of goals,

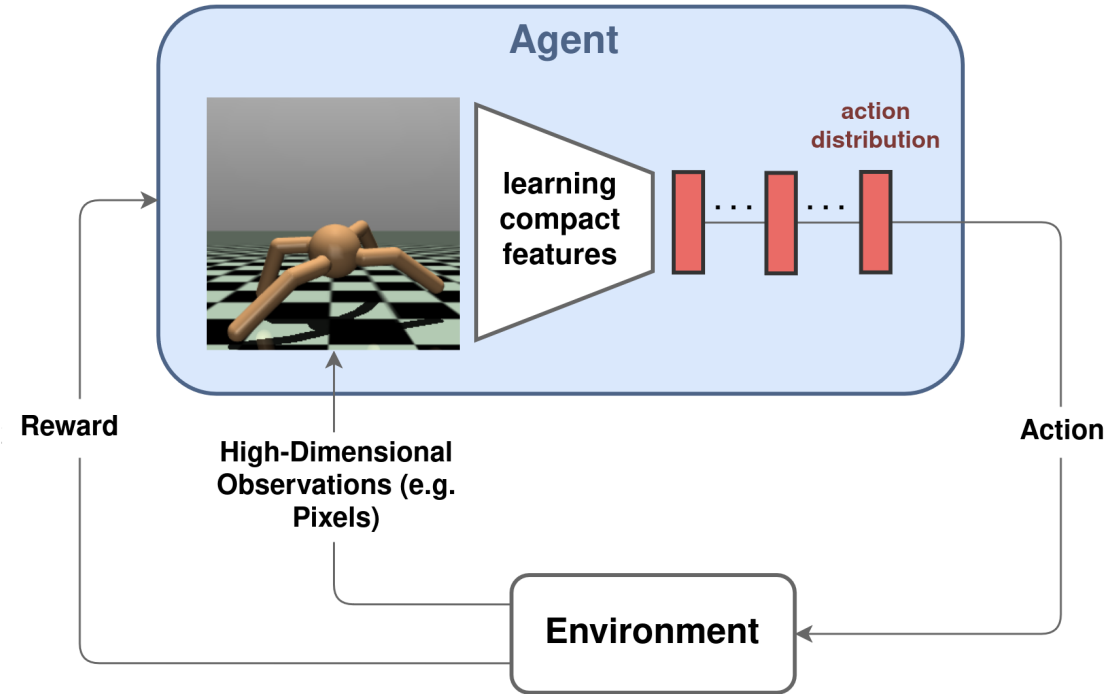

Figure 2.6: End-to-end representation learning of high-dimensional observation in RL: Observation features are embedded in the decision-making process and learned through the reward signal.

which implies that the reward signal in a specific state varies depending on the targeted goal. In that case, relying on the policy to learn latent features is particularly inefficient in the sense that we have to learn representations for the joint space of states and goals $\mathcal{S} \times \mathcal{G}$. For all these reasons, more attention has been given to representation learning techniques, that are agnostic to reward calculation, both in RL (Hafner et al., 2019b; Stooke et al., 2021; Yarats et al., 2021a; Laskin et al., 2020a; Yarats et al., 2021b) and GCRL (Nair et al., 2018; Colas et al., 2018; Pong et al., 2019; Racaniere et al., 2019; Mendonca et al., 2021; Gallouédec and Dellandréa, 2023).

These methods build on the broader interest in the Machine Learning and Computer Vision community for self-supervised/unsupervised learning techniques to build tasks-agnostic representations that are both robust and versatile to diverse downstream tasks. The most widespread unsupervised training objective to learn representations involve data reconstruction (Kingma and Welling, 2013; Higgins et al., 2017a; Van Den Oord et al., 2017; Razavi et al., 2019; Gregor et al., 2019), and contrastive learning (Oord et al., 2018; Henaff, 2020; He et al., 2020; Zbontar et al., 2021). Other methods use a representation learning objective specific to RL, based on the environment dynamics.

2.3.2 Representation Learning through observation reconstruction

Image reconstruction is a standard objective when it comes to learn compact representations, the core idea is to project the original high-dimensional data sample onto a lower-dimensional space and then reconstruct the input from its reduced representation. The quality of the reconstruction reflects how well the latent representation has preserved the important features of the original data point. Auto Encoders (Hinton and Salakhutdinov, 2006) are the most used architectures representative of such an unsupervised approach. They consist of an "encoder" neural network that compresses the input into a lower-dimensional latent space, and a "decoder" neural network whose objective is to reconstruct the input data from the latent representation. The encoder e_ϕ and the decoder d_θ are jointly trained to minimize a reconstruction loss: $\mathcal{L}(x) = ||x - d_\theta(e_\phi(x))||_2$, using gradient descent.

Variational Auto Encoders

Variational Auto Encoders (VAEs) (Kingma and Welling, 2013) are generative models which learn a mapping from the input variable x to a latent variable z. It is a powerful representation learning tool, as its objective is to generate x from a low dimensional variable z, which is a compressed and informative version of x. Assuming the existence of a joint distribution $p_\theta(x, z)$ parameterized by θ, the generative process of x consist of two steps: (1) a latent vector z is sampled from the prior distribution $p_\theta(z)$; (2) a vector x is generated from the conditional distribution $p_\theta(x|z)$. Given a dataset $\{x^{(i)}\}_{i=1}^n$, the objective of VAEs is to maximize the following likelihood (also called the "evidence"), which comes down to maximizing the probability of generating real data:

$$p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}|z)p_\theta(z)dz. \tag{2.11}$$

The integral of the marginal likelihood being intractable in practice, due to the expensive check of all possible z values, the idea behind VAEs is to approximate the true posterior $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$. We learn a variational approximation function $q_\phi(z|x)$, known as the "probabilistic encoder", as it encodes possible values of z from which x could have been generated. On the other hand, $p_\theta(x|z)$ is referred to as the "probabilistic decoder". The model architecture is illustrated in Figure 2.7.

Given a data point x, $q_\phi(z|x)$ outputs a distribution (often Gaussian) over possible values of the latent variable z. Figure 2.8 represents the final graphical model of this probabilistic generative process.

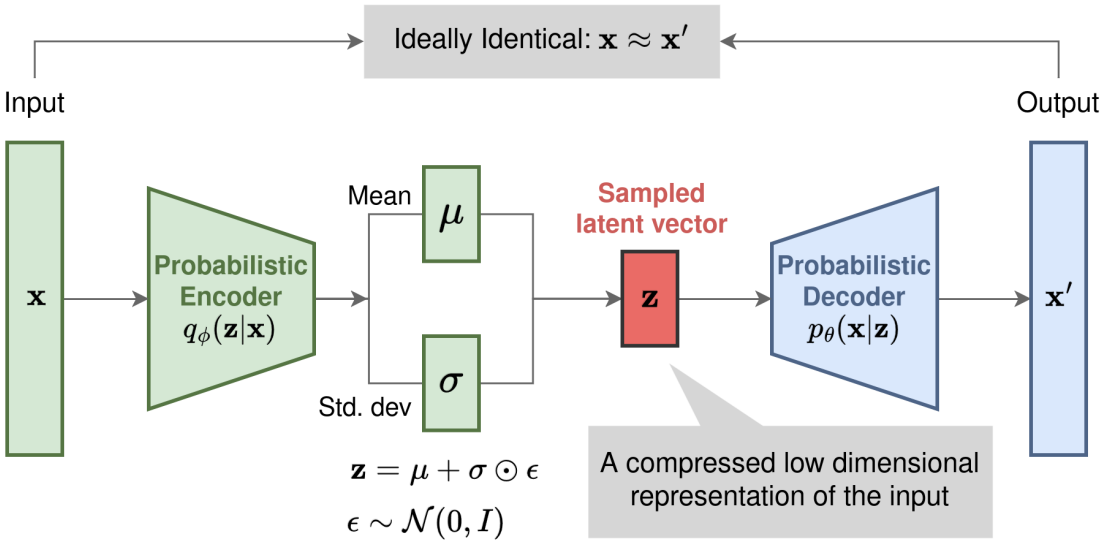

Figure 2.7: Illustration of the Variational Auto Encoder model with multivariate Gaussian posterior distribution $q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma_\phi(x))$. Figure adapted from Weng (2018).

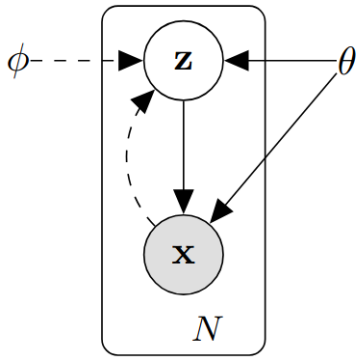

Figure 2.8: VAE graphical model. Solid lines represent the generative process $p_\theta(z)p_\theta(x|z)$, dashed lines represent the variational approximation of the intractable posterior: $q_\phi(z|x) \approx p_\theta(z|x)$.

Variational bound

Back to the VAE objective, which is to maximize the likelihood of a dataset $\{x^{(i)}\}_{i=1}^{n}$, under the aforementioned generative process, we have

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \prod_{i=1}^{n} p_\theta(x^{(i)}) \\ &= \arg\max_{\theta} \sum_{i=1}^{n} \log p_\theta(x^{(i)}).\end{aligned} \tag{2.12}$$

The log-likelihood for a data point x can be decomposed as follows:

$$\log p_\theta(x) = D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x)) + \mathcal{L}_{\theta,\phi}(x), \tag{2.13}$$

with:

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(x) &= \log p_\theta(x) - D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x)) \\ &= -D_{KL}(q_\phi(z|x) \parallel p_\theta(z,x)) \\ &= \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x) \parallel p_\theta(z)).\end{aligned} \tag{2.14}$$

Using the non-negativity of D_{KL}, $\mathcal{L}_{\theta,\phi}(x)$ is a lower bound of $\log p_\theta(x)$. Therefore, in order to maximize $\log p_\theta(x)$, we maximize $\mathcal{L}_{\theta,\phi}(x)$, that is tractable due to the approximate posterior $q_\phi(z|x)$. This quantity is denoted as the Evidence Lower Bound (ELBO), and is a convenient training objective for VAE models:

$$\begin{aligned}\mathcal{L}^{\text{VAE}}_{\theta,\phi}(x) &= \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) \\ \theta^*, \phi^* &= \arg\max_{\theta,\phi} \mathcal{L}^{\text{VAE}}_{\theta,\phi}.\end{aligned} \tag{2.15}$$

In practice, the encoder q_ϕ and the decoder p_θ are neural networks jointly trained by maximizing the ELBO, as shown in (2.15). Back-propagating the gradient to the parameter ϕ becomes possible due to the reparameterization trick:

$$\begin{aligned}&z \sim q_\phi(z|x^{(i)}) = \mathcal{N}(z|\mu^{(i)}, \sigma^{(i)}) \\ &z = \mathcal{T}_\phi(x,\epsilon) = \mu + \sigma \odot \epsilon, \text{where } \epsilon \sim \mathcal{N}(0, I)\end{aligned} \tag{2.16}$$

where the stochastic process is shifted into ϵ and turns $z = \mathcal{T}_\phi(x,\epsilon)$ into a deterministic transform on which it is possible to perform gradient ascent.

Applications in RL

Variational Auto Encoders present appealing properties when it comes to learn latent state representations in RL. With their probabilistic formulation, one can represent the observation space by the latent prior distribution, which enables several operations to take place, such as goal sampling in GCRL (Nair et al., 2018; Pong et al., 2019; Gallouédec and Dellandréa, 2023) and having access of log-likelihood of trajectories for model-based RL and planning (Higgins et al., 2017b; Hafner et al., 2019b; Lee et al., 2020). An implementation of representation learning in RL with VAEs where latent states and the control policy are learned in parallel is illustrated in Figure 2.9.

2.3.3 Contrastive Representation Learning

Contrastive learning is a self-supervised learning technique used to learn useful representations of data by distinguishing between similar and dissimilar pairs of examples. The core idea is to train a model to bring representations of similar data points closer together in the latent space, while pushing representations of dissimilar data points farther apart, as illustrated in Figure 2.10.

Contrastive training objectives

Given a data point $x \in \mathcal{X}$, associated with a positive pair x^+ and a negative pair x^-, one of the most common contrastive training objective is the "Triplet Loss" (Schroff et al., 2015):

$$\mathcal{L}_\theta^{\text{triplet}}(x, x^+, x^-) = \sum_{x \in \mathcal{X}} \max\left(0, \|f_\theta(x) - f_\theta(x^+)\|_2^2 - \|f_\theta(x) - f_\theta(x^-)\|_2^2 + \epsilon\right). \tag{2.17}$$

Another broadly used contrastive training objective is based on Noise Contrastive Estimation (NCE) that uses logistic regression to distinguish between the target data and noise (Gutmann and Hyvärinen, 2010). Building on NCE, more recent works use the categorical cross-entropy to maximize similarity between the positive pairs x and x^+, compared to any other random pair considered to be negative. For example, in Contrastive Predictive Coding (CPC), Oord et al. (2018) proposes the Info-NCE loss:

$$\mathcal{L}_\theta(x) = \log \frac{\exp(f_\theta(x)^T W f_\theta(x_+))}{\exp(f_\theta(x)^T W f_\theta(x_+)) + \sum_{i=0}^{N} \exp(f_\theta(x)^T W f_\theta(x_i))}, \tag{2.18}$$

where the similarity between samples is modeled with the bilinear product $x^T W x_+$. The objective here is that the embedding of x matches with x_+ relatively more than any other random sample x_i in a batch.

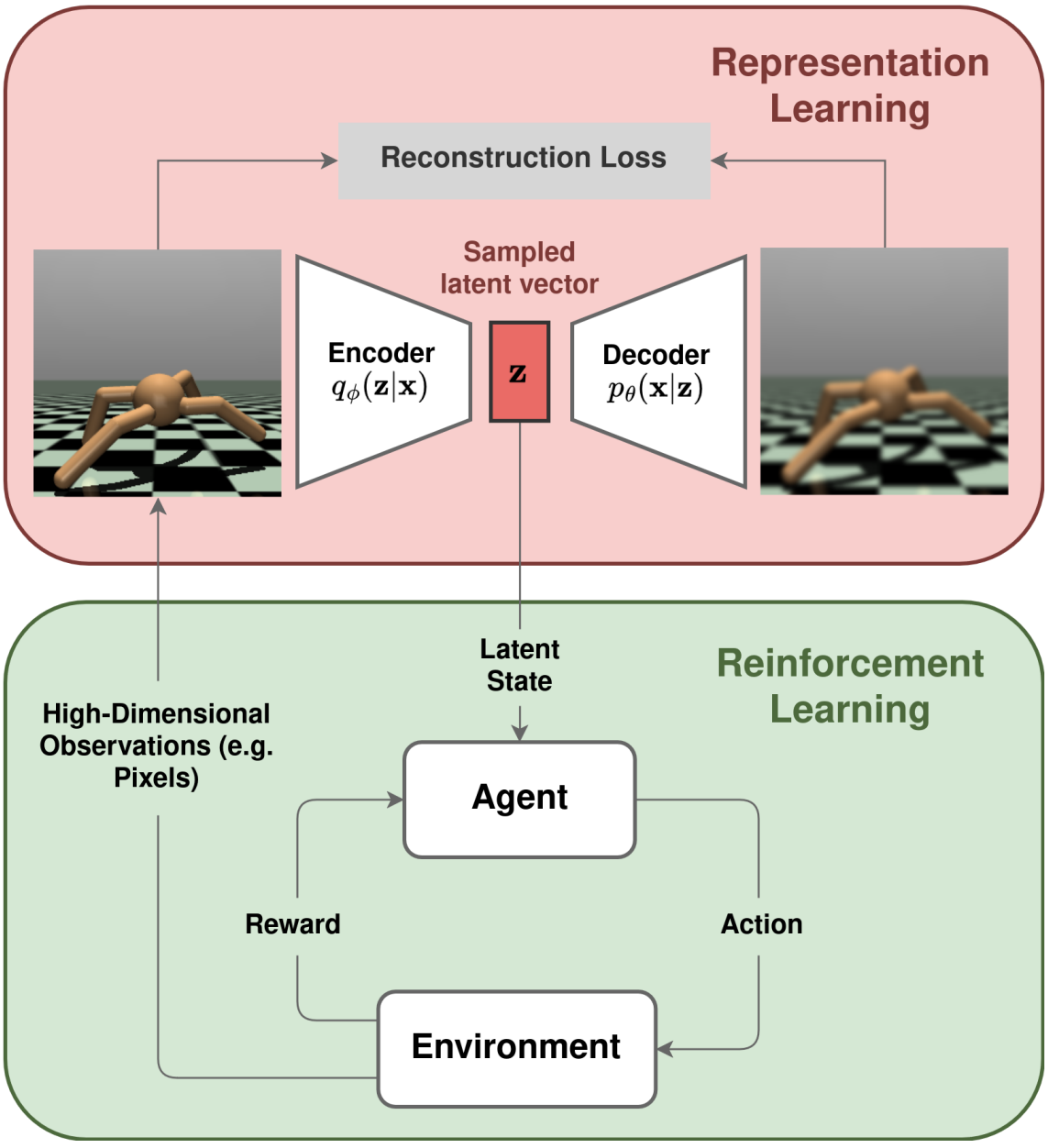

Figure 2.9: Variational Auto Encoders architecture to learn representations in Reinforcement Learning. The VAE is trained to learn a latent representation of high dimensional observations (e.g. pixels). The RL agent receives the latent state as input.

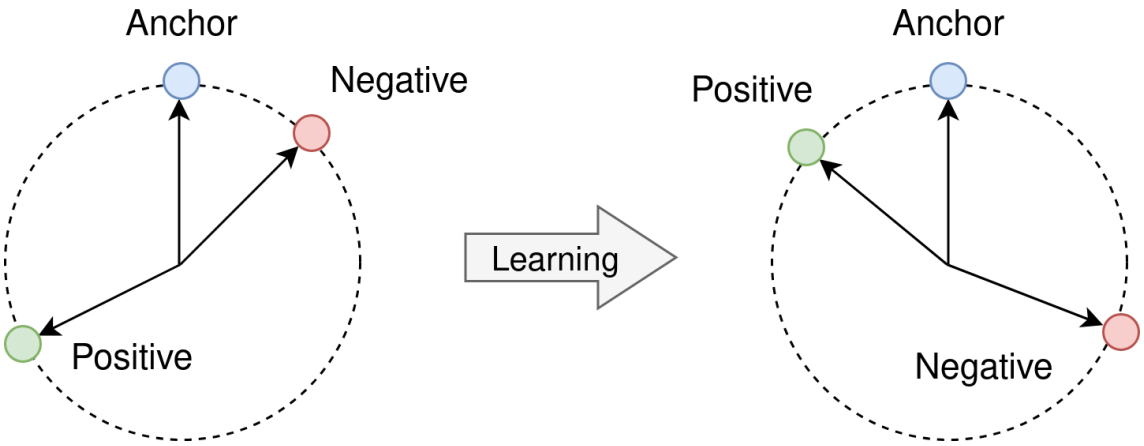

Figure 2.10: General objective of contrastive Learning: learn a representation that minimizes the distance between similar examples (Anchor and positive) and maximizes the distance between dissimilar samples (Anchor and Negative).

Building positive and negative pairs in RL

The data augmentation technique is a key building block of Contrastive Learning, in order to minimize the objective on positive/negative samples. Data augmentation introduces small variations on the input data without modifying its semantic meaning, such as random crop, rotations, horizontal/vertical flip, color distortions and many more.

In RL, another component when it comes to data points similarity is the environment dynamics: two states that belong to the same transitions and are temporally close should also be close in the latent representations. Recent works in GCRL build on the InfoNCE training objective to learn representations and distance metrics that integrate the environment dynamics (Lu et al., 2019; Li et al., 2021; Aubret et al., 2023).

On the other hand, the CURL method illustrated in Figure 2.11 uses contrastive learning on consecutive observation stacks, applying random crop as data augmentation. This technique ensures that close states in time have similar representations, thus capturing temporal features.

2.3.4 Dynamic Aware Representation Learning

Another line of work to learn representations in RL is to design training objectives relying explicitly on the environment dynamics. These methods aim to learn "functional" representations that are not necessarily complete in terms of capturing all factors of variation in the observation space, but rather aim to capture features that are important in decision-making for a particular

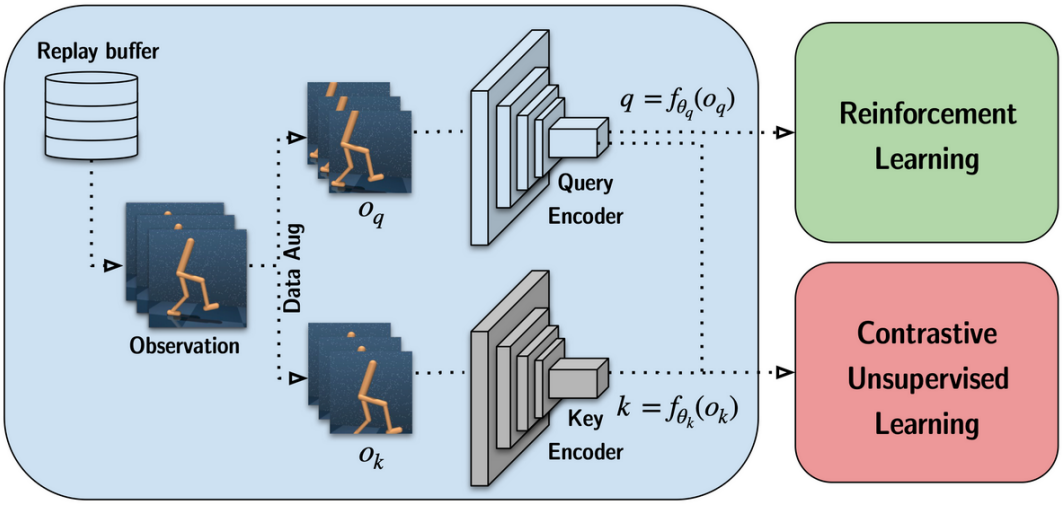

Figure 2.11: The CURL method trains a "Query" encoder to ensure that embeddings of the observation o_q and their data-augmented version o_k (considered as a positive pair) are close, using a contrastive loss. The RL policy is trained with the output embedding of the "Query" encoder, the "Key" encoder is an exponential moving average version of the query. Image source: Laskin et al. (2020a).

task, as illustrated in Figure 2.12.

Pathak et al. (2017) present a method to learn such a feature space using self-supervision to train a neural network with sub-modules ϕ to learn states embeddings, and f_θ to learn inverse dynamics: the objective is to predict the agent's action given a pair of consecutive states s_t and s_{t+1}:

$$\mathcal{L}^{\text{inv.}}_{\theta,\phi}(s_t, a_t, s_{t+1}) = \frac{1}{2}\| a_t - \underbrace{f_\theta(\phi(s_t), \phi(s_{t+1}))}_{\hat{a}_t} \|_2^2. \tag{2.19}$$

Since the neural network is only required to predict the action, it has no incentive to represent within its feature embedding space the factors of variation in the environment that do not affect the decision-making process.

Other works propose to use the forward dynamics prediction specific to model-based RL (Achiam and Sastry, 2017):

$$\mathcal{L}^{\text{for.}}_{\theta,\phi}(s_t, a_t, s_{t+1}) = \frac{1}{2}\| s_{t+1} - \underbrace{f_\theta(\phi(s_t), a_t)}_{\hat{s}_{t+1}} \|_2^2. \tag{2.20}$$

An alternative proposal in Ghosh et al. (2019) consists in learning Action-

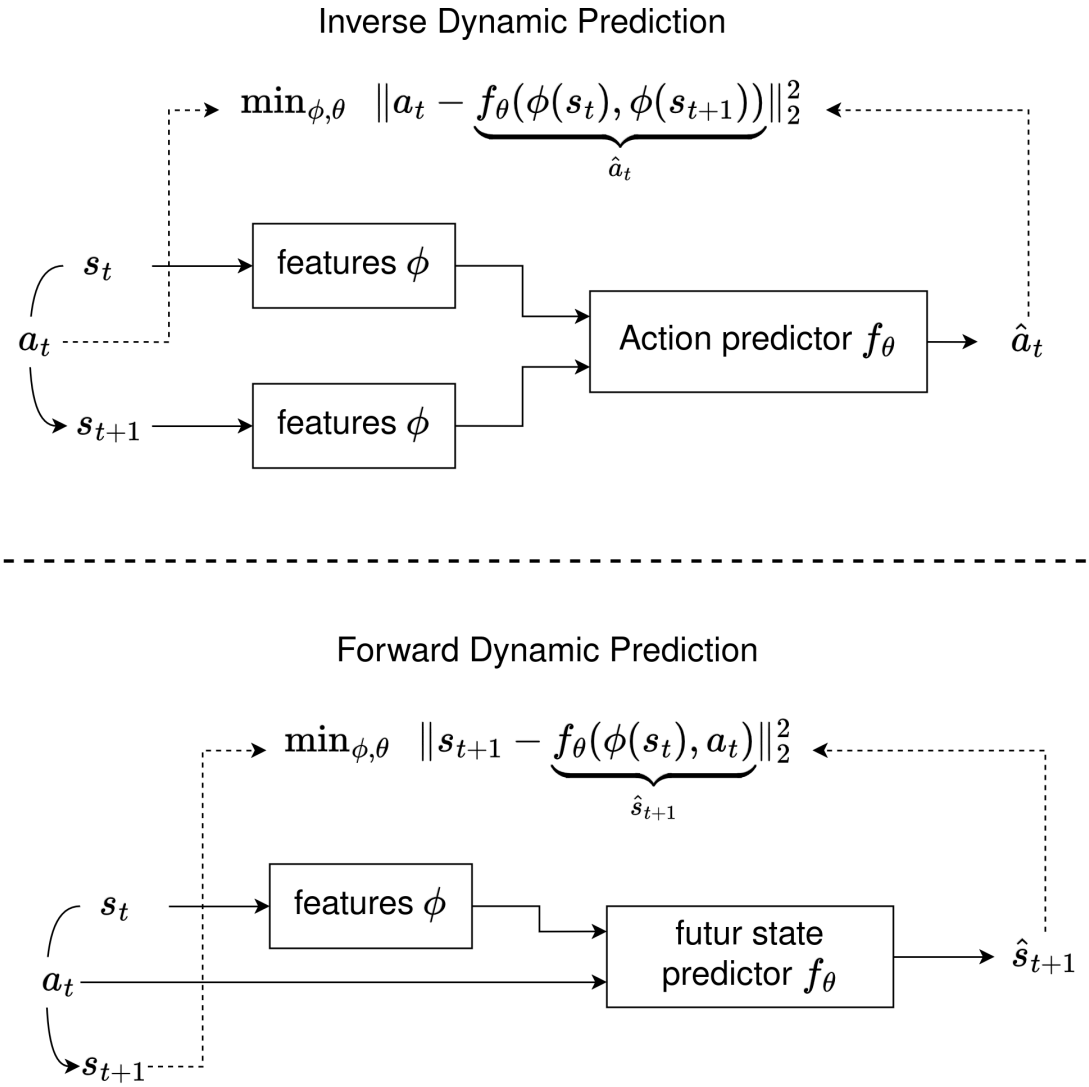

Figure 2.12: Dynamic Aware Representation Learning. Based on transitions (s_t, a_t, s_{t+1}), train end-to-end latent features ϕ and a dynamic predictor f_θ. **Inverse dynamics** aims to predict the action between two consecutive states features $\phi(s_t)$ and $\phi(s_{t+1})$, while **Forward dynamics** aims to predict next states s_{t+1} for taking the action a_t on the state of features $\phi(s_t)$.

able Representations for Control (ARC). Their objective is to design a feature space ϕ governed by what they define as the "actionable distance" between states s_1 and s_2:

$$D_{\text{Act}}(s_1, s_2) = \mathbb{E}_s \left[D_{KL} \left(\pi(a|s, s_1) \| \pi(a|s, s_2) \right) + D_{KL} \left(\pi(a|s, s_2) \| \pi(a|s, s_1) \right) \right]. \tag{2.21}$$

$D_{\text{Act}}(s_1, s_2)$ is large for states that induce different action distributions $\pi(a|s, s_2)$ and $\pi(a|s, s_2)$, and conversely. To ensure that functionally similar states have close representations, ϕ is trained to match D_{Act}:

$$\min_{\phi} \mathbb{E}_{s_1, s_2} \left[\left(\|\phi(s_1) - \phi(s_2)\|_2^2 - D_{\text{Act}}(s_1, s_2) \right)^2 \right]. \tag{2.22}$$

An illustration of this procedure is given in Figure 2.13.

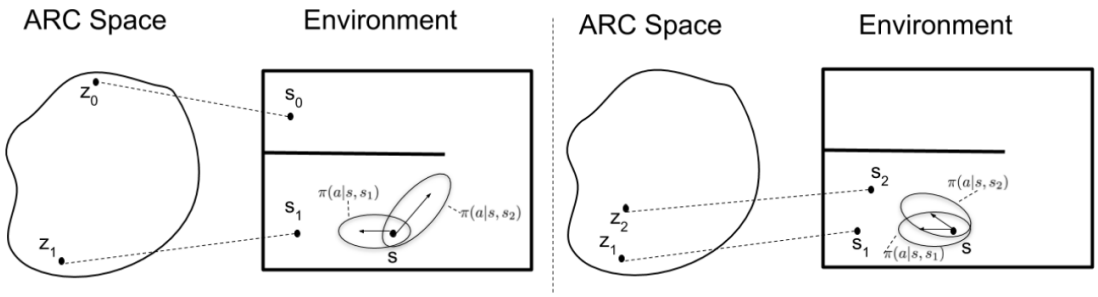

Figure 2.13: Actionable Representation for Control (ARC) Ghosh et al. (2019). Goals that imply close action distributions are close in the ARC space, and conversely.

In the next chapter, we come back to the main research focus of this Thesis, with an extensive review of the application of Intrinsic motivations to solve Unsupervised RL problems.

Chapter 3

Intrinsically Motivated Agents in Unsupervised RL

In this chapter, we introduce the general setting of the work conducted in this thesis, including two components: Unsupervised RL and Intrinsically motivated Agents, that have the ability to learn autonomously to acquire diverse and useful behavior in the absence of supervision, through generic objectives such as *exploration*, *empowerment*, or *control*.

We begin with a description of Unsupervised RL problems, and their importance in regard to the design of generalist and versatile agents, able to adapt and tackle a great variety of tasks. This setting stands out from regular RL problems, where predefined rewards and tasks constraint agents to be specialized in one type of behavior, without exploiting their capacities to adopt diverse behavior. We introduce the concept of intrinsic motivation, which allows an agent to tackle Unsupervised RL problems with either intrinsic rewards or tasks. We then give a brief review of intrinsically rewarded agents.

We continue with multi-task intrinsically motivated agents, which constitute the core of this chapter, as it corresponds to the specific setting adopted in this Thesis. In this section, we start with the formalization of a proximal objective in Unsupervised RL, which we define as a combination of exploration and control. On one hand, exploration stands for the ability to attempt a great variety of tasks, while control refers to the agent's ability to reliably master them.

Applied to goal conditioned RL, intrinsic motivation can be broken down to intrinsic goals. We address the main questions regarding the design of intrinsic goals : first, we give an overview of existing criteria used to prioritize

goals, second, we compare different methods to generate goals, and third, we tackle goal representation learning in case of high dimensional inputs.

This chapter sets up our contributions in Chapters 4 and 5, classifying and comparing methods from the literature, we highlight their main weaknesses and blind spots, that we address with two new methods, named SVGG for goal selection, and DROIDE for goal representation.

Contents

3.1 Unsupervised Reinforcement Learning

Despite the tremendous success achieved in the past years in Reinforcement Learning and its application to video games (Mnih et al., 2013; Vinyals et al., 2019; Berner et al., 2019) and robotics (Schulman et al., 2017; Akkaya et al., 2019), existing agents often lack versatility, requiring human engineers to design their behavior for specific tasks, limiting their ability to handle new circumstances. This agent's specialization results in policies with poor generalization capabilities, which make them vulnerable to small variations of external factors and adversarial attacks (Gleave et al., 2019).

In addition, in many real-world scenarios, tasks to be carried out are complex, involving various object manipulations and tricky environment interactions. As a result, designing conditions under which the agent has completed the task can be convoluted and time-consuming, due to unexpected interactions between external factors and the specifications of artificial agents. For example, in RL, the manual definition of reward functions for a specific task is brittle, and sometimes leads to suboptimal behavior where the policy finds a way to maximize the reward function without completing the intended task. In that case, the reward signal is said to be deceptive. Allowing an agent to automatically design and pursue its own tasks is therefore a very valuable feature in RL.

For these reasons, it is important to move beyond today's specialized RL agents toward more generalist systems endowed with the capability of adapting to new downstream tasks. Unsupervised Reinforcement Learning (URL) offers a framework where agents are not provided with external rewards, and thus must autonomously learn new tasks throughout their lifespan, guided by intrinsic motivations. The scope of this thesis covers *fixed dynamics* RL environments, in the sense that the state space $\mathcal{S}$, the action space $\mathcal{A}$ and the transition function $\mathcal{T}$ are constant. The changing component of the problem, allowing the agent to develop competences through a curriculum of tasks, is the *intrinsic motivation*, an internal process that generates *intrinsic rewards*, based on the current abilities of the agent.

3.1.1 What is intrinsic motivation in RL?

Most of the time, humans are not motivated by external rewards but spontaneously engage into activities to discover and learn about them. We refer to this tendency as *intrinsic motivation*. Hereafter, we propose a short definition of *intrinsic motivation* beyond the scope of Reinforcement Learning:

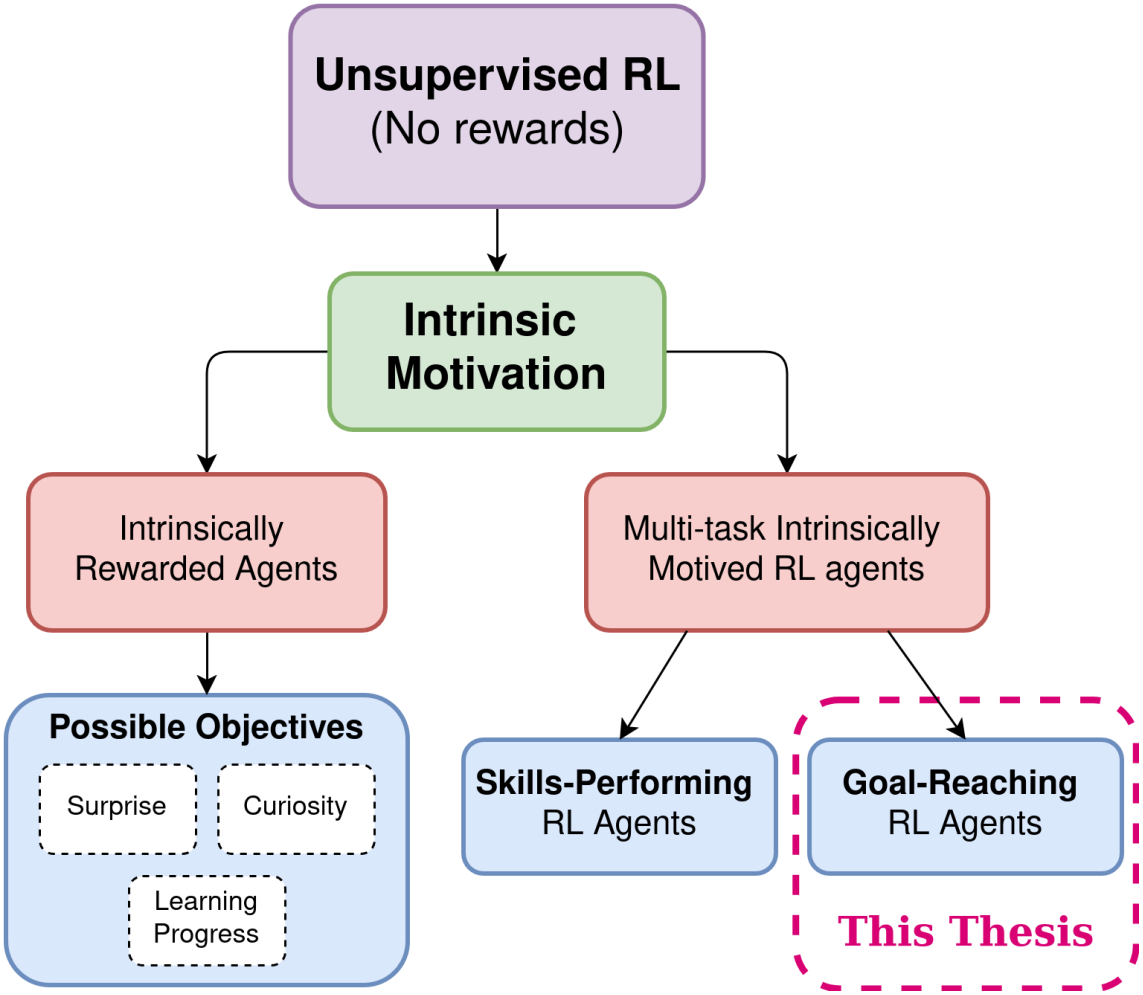

Figure 3.1: A typology of methods tackling the Unsupervised RL problem with intrinsic motivation, classified in two main categories: **Intrinsic Rewards RL agents** generally target a standard MDP with an intrinsic objective combined to the external reward function, whereas **Multi-task Intrinsically Motivated RL agents** aim to train agents to master a diverse set of *goals* or *skills*.

Definition 3.1.1. *Intrinsic motivation* refers to the internal drive to engage in an activity for its own sake, deriving purpose from the activity itself rather than from external rewards or pressures. In Reinforcement Learning, an *intrinsically motivated agent* must be able to set its own objectives and reward signals.

By contrast, *extrinsic motivations* generally refer to the standard design of RL agents where the agent learns to maximize an external reward function.

Definition 3.1.2. *Extrinsic motivation* refers to the incentive to realize a task due to external constraints. External factors can include positive or negative rewards. In Reinforcement Learning, the term *Extrinsic motivation* is used to describe an embedded learning signal within an environment, such as an ad-hoc reward function or a predefined goal distribution.

3.1.2 Intrinsic rewards RL vs Multi-task Intrinsically Motivated RL

Intrinsically Motivated Agents can be categorized in two ways, depending on the RL Agent's specifications and the form of the learning signal:

- **Intrinsic Rewards RL** methods are based on the design of general reward functions that are agnostic to the task or the environment, allowing the agent to make progress autonomously in a generic setting. In these approaches, the intrinsic reward function is generally fixed and designed to target a general and abstract objective such as *Exploration*, *Surprise* or *Learning progress*. This category is also referred to as *Knowledge-Based* Intrinsic Motivation (Oudeyer et al., 2007). Beyond their ability to handle Unsupervised RL, these methods are often applied to environments that already possess a complex embedded reward function, but where it is very hard to maximize the reward by only relying on it. For instance, though most Atari games have built-in mechanisms that negatively or positively reward the agent to varying degrees for taking specific actions in the environment (crossing rooms, climbing a ladder, killing enemies, ...), a few Atari games such as the infamous Montezuma's revenge require additional reward signals, see Figure 3.2. This is more generally the case when the provided reward signal is deceptive or sparse.

- **Multi-task Intrinsically Motivated RL** methods rather consist in indirectly providing the agent with reward functions through intrinsic "tasks" generation. In these methods, the agent is trained to acquire

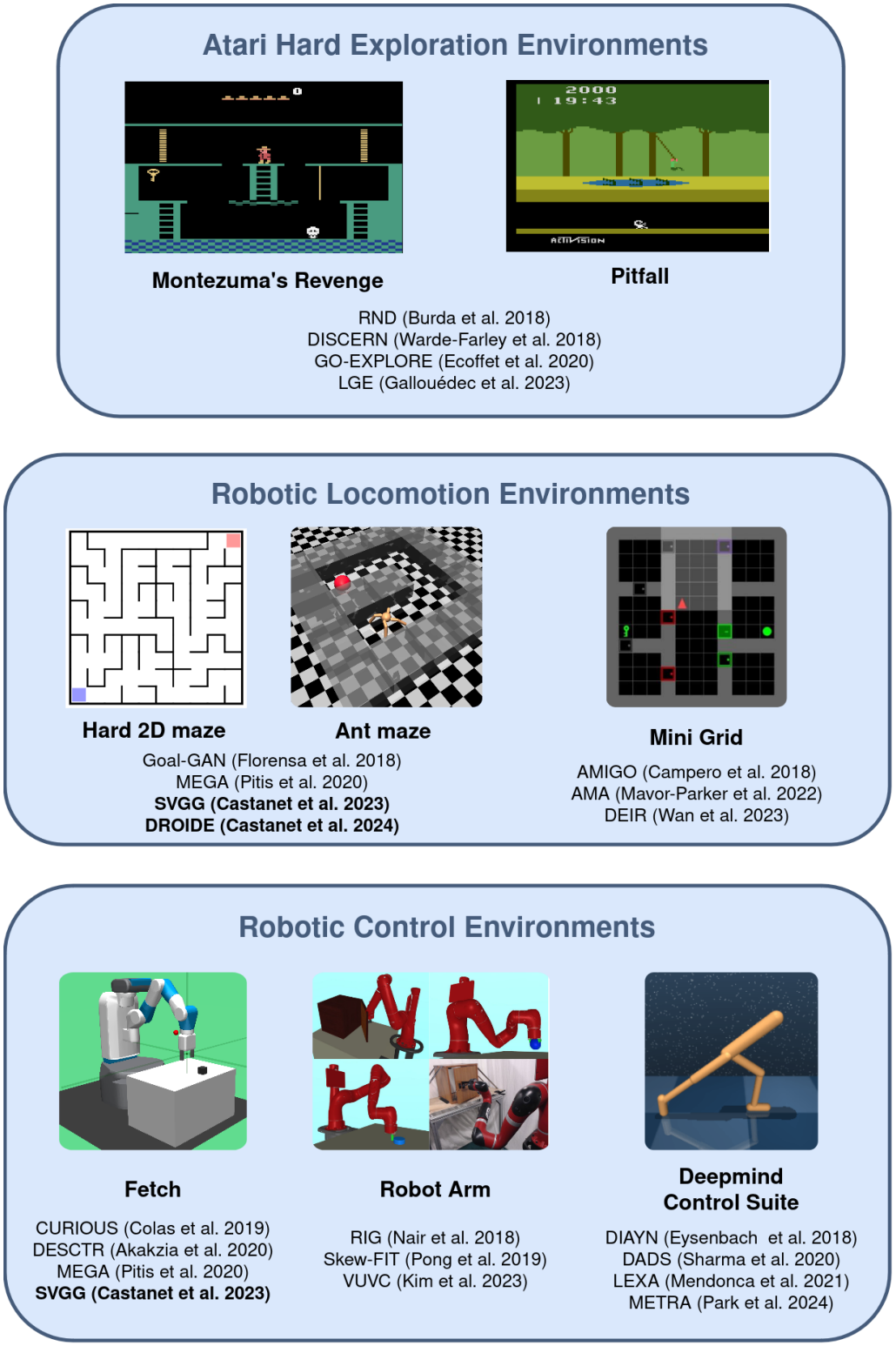

Figure 3.2: A non-exhaustive but representative panel of environments used in Unsupervised RL approaches. This figure is inspired by the categorization adopted in Colas et al. (2022).

a diverse set of behaviors depending on the intended "task". The RL policy is conditioned on its current states and on the task it has to complete. In this context, a task either corresponds to a *skill* to perform or a *goal* to reach. A skill is defined as a sequence of states to reach in the state space $\mathcal{S}$ (and sometimes actions to take), which correspond to a specific and hopefully useful behavior, for example a robot doing a backflip, manipulating objects, painting... On the other hand, a goal corresponds to a single state to achieve, regardless of the way the agent proceeds. This perspective is mainly adopted for robotic manipulation or navigation environments which are not built with an embedded specific reward function, as the only thing that we really care about is to get to a specific configuration of the environment (getting to a position of a maze, grabbing and moving an object to a desired position, ...). Such RL environments can be interpreted as a Goal-Conditioned RL problem, where the goal space $\mathcal{G}$ is either equal to or a subset of the observation space $\mathcal{O}$. It can be the case that every state is a potential task in the sense of a goal to be achieved g, and comes with the corresponding reward function $\mathcal{R}_g$. In this perspective, the agent can be provided with a dynamically changing task distribution, from which it can continually learn to reach new goals, or to re-learn some that it might have forgotten along the way. In addition to having the ability to explore the environment, this approach presents the appealing property of learning a versatile controller through the goal-conditioned policy. This second category is also referred to as *Competence-Based* Intrinsic Motivation (Oudeyer et al., 2007).

3.2 Intrinsically Rewarded Agents

Taking inspiration from the curiosity understanding from psychology and neuroscience (Oudeyer and Kaplan, 2007; Gottlieb et al., 2013), suggesting that some "intrinsic reward" is generated by the brain to acquire new information about the environment without any practical objective, this family of methods aims to solve Unsupervised RL Problems with "intrinsic reward" augmentation. In an unsupervised setting where the reward is sparse or in complete absence of reward in the environment, a common approach is to generate some reward automatically so that the RL Agent can learn to explore its environment. The usual ambition of such methods is to solve *Hard Exploration Problems* which are characterized by a very sparse reward signal, with classic RL policies $\pi(.|s)$. Environments belonging to this category were identified as too difficult for classic deep Reinforcement Learning approaches, achiev-

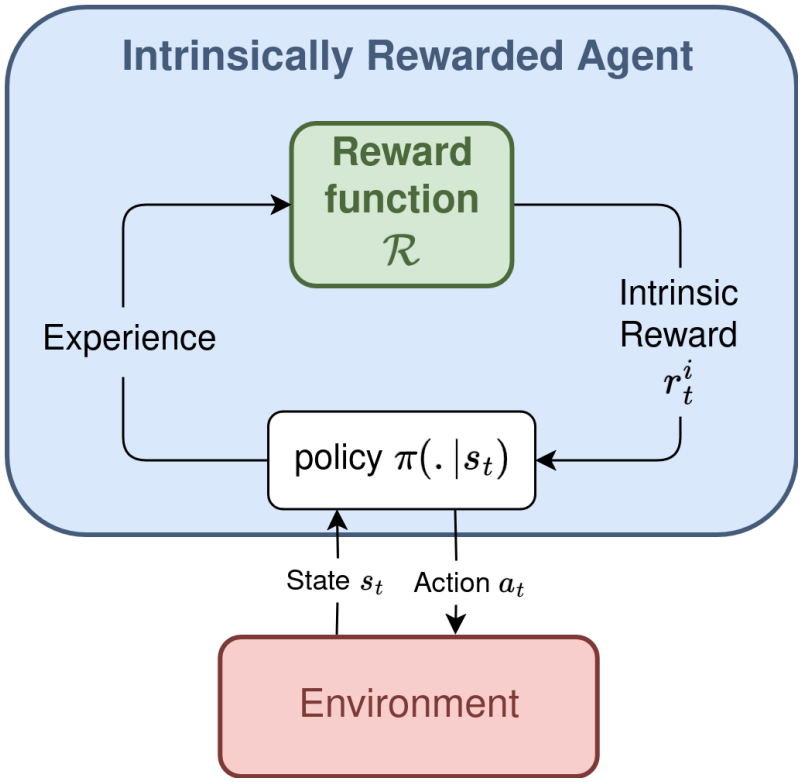

Figure 3.3: Intrinsically rewarded agents use generic functions measuring progresses of the agent, that provide intrinsic reward as a function of the agent's experience. Potential extrinsic reward coming from the environment can be added.

ing very little success compared to average human performance (Mnih et al., 2015). Iconic games falling into Hard Exploration Problems include "Montezuma's Revenge" and "Pitfall", in which the agent must perform a long and complex series of actions (traverse rooms, moving objects, climbing ladders, ...) before receiving any reward signal. As an illustration of the complexity of those environments, ordinary RL algorithms such as DQN usually fail to get out of the first room of Montezuma's Revenge (scoring 400 or lower) and score 0 or lower on Pitfall. Moreover, until 2019, despite considerable research efforts, the SOTA algorithm with augmented intrinsic rewards obtained an average score of 17,500 (compared to 35,000 for a human expert) on Montezuma's Revenge and no algorithm could obtain a score greater than 0 on Pitfall, the human average performance being 5000.

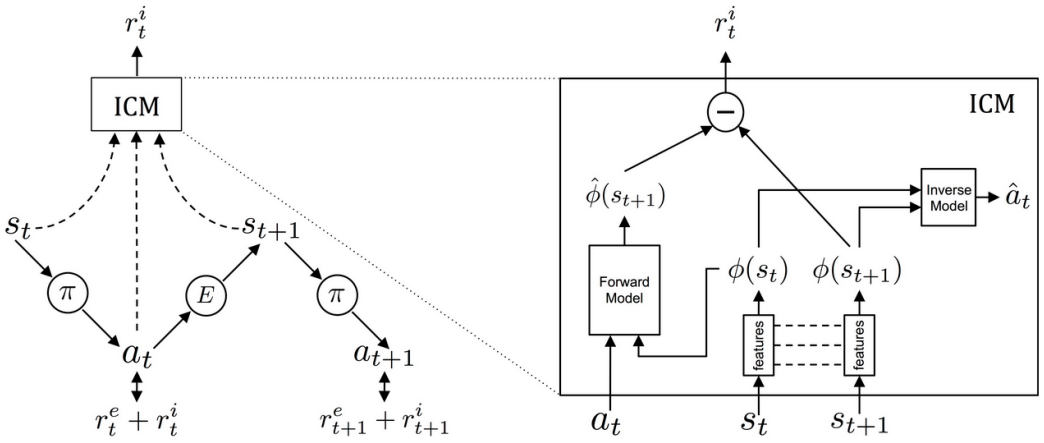

Figure 3.4: Intrinsic Curiosity Module (ICM) in Pathak et al. (2017). Intrinsic rewards are computed as the difference between expected and true dynamic of the environment: the agent is rewarded to reach surprising states.

3.2.1 Intrinsic reward formulations

Intrinsic reward formulations aims to design proximal objectives measuring the "progress" of the agent, being agnostic to the environment characteristic. How to define progress here is the crux of the problem. Existing formulations cover:

- ***Novelty*** where the agent is rewarded for reaching less visited states. For example, count-based methods (Bellemare et al., 2016; Ostrovski et al., 2017) foster exploration by augmenting the classic reward function coming from the MDP with what they call an *Exploration bonus*: $r_t = e_t + i_t$, where e_t and i_t are respectively the *extrinsic* and *intrinsic* reward signals. In the tabular case, where $\mathcal{S}$ is discrete, $i_t = 1/n(s_t)$ is a classical choice where $n(s_t)$ is defined as the visitation count of a given state s. In continuous MDPs, the visitation count can be approximated with some density estimation (Bellemare et al., 2016). These methods have achieved significant success on *Hard Exploration Problems*. For instance, "Random Network Distillation" (RND) was the first method to achieve better than human average performance on Montezuma's Revenge (Burda et al., 2018), using the difference between a random network and a trained network to discover poorly visited states.

- ***Surprise and Curiosity***, where the agent is rewarded for getting a large prediction error on the future based on some model of the environ-

ment. In Deep Learning, prediction errors can be attributed to several factors.

First, stochasticity of the data, that is called *Aleatoric Uncertainty*, refers to the inherent noise in the observations. For example, this is the case of natural randomness in images that lead to class confusion in Deep Learning classification tasks. In Reinforcement Learning, observations can be stochastic, in the sense that the real hidden state may correspond to a distribution of possible observations. For example, some irrelevant random noise can be contained in an image observation depending on the lighting conditions.

Second, prediction errors can be due to the amount of training data, where too few similar examples were seen by the predictor. This is the case for instance for outliers in a data distribution. This kind of prediction errors are due to the *Epistemic Uncertainty* of the model. In practice, they could be avoided by acquiring more training data. Tracking and reducing *Epistemic Uncertainty* is key for outlier detection (Hodge and Austin, 2004) and for avoiding over-confident models (Corbière et al., 2019). Epistemic Uncertainty estimation methods include Bayesian Neural Network (Blundell et al., 2015; Gal and Ghahramani, 2016) and disagreement between multiple models (Lakshminarayanan et al., 2017; Sekar et al., 2020). In Reinforcement Learning and specifically in Hard Exploration Problems, measuring Epistemic Uncertainty is a way to determine which part of the environment has been overlooked the most regarding some model. For instance, Pathak et al. (2017, 2019); Chua et al. (2018) use the uncertainty estimation of a model of the dynamic of the world as an intrinsic motivation to foster exploration. In the Intrinsic Curiosity Module (ICM) method (Pathak et al., 2017), epistemic uncertainty estimation is done through learning representations that filter information that the agent can control via inverse dynamic prediction (as explained in Section 2.3.4). The intrinsic reward is then defined as the difference between the expected next state (predicted by a learned forward model f_θ) and the actual next state (illustrated in Figure 3.4):

$$r_t^i = || \underbrace{f_\theta\big(\phi(s_t), a_t\big)}_{\hat{\phi}(s_{t+1})} - \phi(s_{t+1})||_2. \tag{3.1}$$

This intrinsic reward formulation has proven to be very efficient at pushing the agent to visit surprising but yet controllable unexplored areas of the environment.

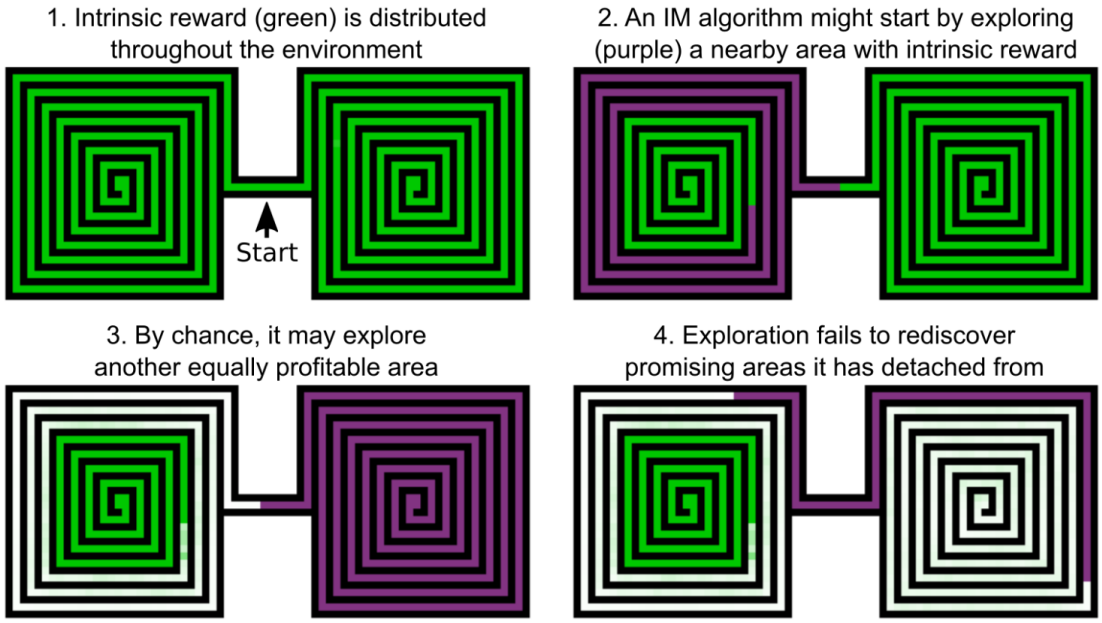

Figure 3.5: Example of detachment in intrinsic motivation (IM) algorithms. Green areas indicate intrinsic reward, white indicates areas where no intrinsic reward remains, and purple areas indicate where the algorithm is currently exploring. Illustration coming from Ecoffet et al. (2019).

3.2.2 Explore, Then Return

A major breakthrough in intrinsically rewarded agents is the understanding of "detachment" in intrinsic motivation (Ecoffet et al., 2021). Detachment is a tendency of RL algorithms to forget how to reach previously visited states. Ecoffet et al. (2019) present a very intuitive illustration of such pitfalls in Figure 3.5: at first, some intrinsic motivation is distributed throughout the environment, and is being consumed by the exploring agent. When the agent comes back to a start position at the end of an episode, the intrinsic motivation that previously led it to some interesting areas worth to continue exploring being consumed, it fails to come back to that area and to discover the remaining promising states. The Go-Explore method (Ecoffet et al., 2019) alleviates this issue by explicitly storing stepping stones for exploration in a buffer, and then goes on exploring after returning to these promising areas, that are both distant from the initial state and hard to reach. Go-explore achieved massive improvement on Montezuma's revenge and Pitfall by outperforming by several orders of magnitude previous state-of-the-art methods.

This family of methods brought a great set of ideas regarding the design of intrinsic motivation processes. These ideas were adapted to the context of

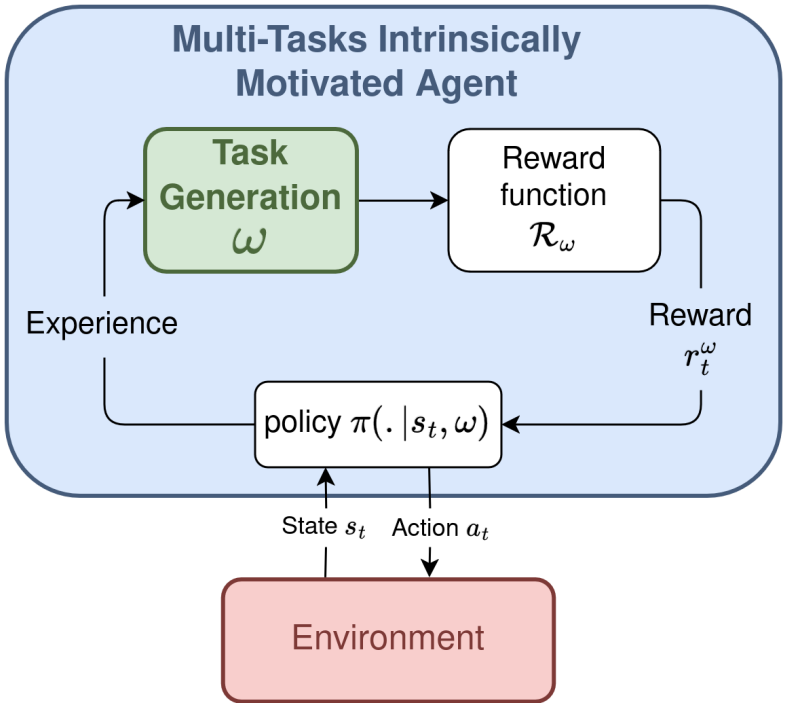

Figure 3.6: multi-task Intrinsically Motivated Agents are embedded with the ability to generate their own tasks from its experience. The policy is rewarded for completing the intrinsic tasks ω through a task-conditioned reward function $\mathcal{R}_\omega$.

multi-task intrinsically motivated agents, which is the main research focus of this work.

3.3 Multi-task Intrinsically Motivated Agents

Multi-task Intrinsically Motivated Agents is a widely used framework in Reinforcement Learning to tackle unsupervised settings, that emerged from the field of developmental robotics (Oudeyer et al., 2007; Oudeyer and Kaplan, 2007), taking inspiration from the underlying mechanisms powering autonomous behaviors in humans. The objective is to embed into Reinforcement Learning agents the ability to invent and pursue their own problems, using internal feedback and experience to assess completion.

Transplanted to RL, the intrinsic motivation process inducing internal rewards lies in automatic generation of a curriculum of tasks conditioning the agent's policy, relying on its own experience, as illustrated in Figure 3.6. This framework achieved many successes tackling Unsupervised RL problems in

the past few years, particularly for robotic control and navigation tasks (Eysenbach et al., 2018a; Nair et al., 2018; Pong et al., 2019; Pitis et al., 2020). It is worth noting that in addition to operating in unsupervised settings, multi-task Intrinsically Motivated Agents have been successfully applied to speedup training on hard tasks even when the goal is known in advance (Colas et al., 2018; Ren et al., 2019).

3.3.1 Exploration & Control

The ambition of multi-task Intrinsic motivation methods is to obtain an agent able to explore and control its environment, which implies the capacity to reach any possible task through the task-conditioned policy:

Definition 3.3.1. The *Optimal multi-task Policy* in the contextual MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \omega, \mathcal{R}_\omega, \gamma\}$, denoted as π^*, is defined as:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\omega\sim\Omega, s\sim\rho_0}\Big[\mathbb{E}_{\tau\sim\pi(.|s,\omega)}\Big[\sum_{t=0}^{\infty}\gamma^t r_t^\omega\Big]\Big],$$

where ω is a task sampled from an arbitrary distribution of tasks in the environment Ω, associated with a task-conditioned reward function $\mathcal{R}_\omega$, ρ_0 is the initial state distribution of the agent, $\tau \sim \pi(.|s,\omega)$ is the trajectory obtained from policy π with the initial state $s \in \mathcal{S}$ and $r_t^\omega = \mathcal{R}_\omega(s_t, a_t, s_{t+1})$.

The unsupervised setting imposes the following constraints: the distribution over every valid tasks in the environment Ω, and the reward function $\mathcal{R}_\omega$ for $\omega \sim \Omega$ are unknown, constituting valuable prior knowledge on the environment. Therefore, optimizing this quantity is not straightforward.

However, this general objective can be approached in two distinct phases constituting surrogate objectives, which can be carried out jointly or separately, without supervision: *Exploration*, and *Control*. The *Exploration* component measures the ability for an agent to discover new tasks:

Definition 3.3.2. *Exploration* in multi-task RL is a process that measures the extension of the distribution of discovered tasks $\omega \in \text{supp}(\Omega)$:

$$\text{explo}(\pi) := H\Big[p_\pi\Big], \tag{3.2}$$

where $H(.)$ is the Shannon entropy (Shannon, 1948) and p_π is the distribution of completed tasks by the policy π over the course of training.

Exploration should not be mistaken for the ability to reliably and intentionally complete tasks. Indeed, some tasks might have been discovered accidentally during the exploration process. In addition, catastrophic forgetting might occur during training.

We call *Control* the capacity for an agent to reliably complete discovered tasks:

Definition 3.3.3. An agent has control on a task $\omega \in \text{supp}(\Omega)$ if:

$$\mathbb{P}\Big(\omega \in \tau \mid \tau \sim \pi(.|s,\omega)\Big) = 1,$$

where τ is a trajectory sampled from the policy $\pi(.|s,\omega)$, the assertion $\omega \in \tau$ means that the requested task is included in the trajectory.

We define the general *Control* objective as the ability for the agent to maximize the above quantity for known tasks in p_π:

$$\text{control}(\pi) := \mathbb{E}_{\omega \sim p_\pi}\Big[\mathbb{P}\Big(\omega \in \tau \mid \tau \sim \pi(.|s,\omega)\Big)\Big]. \tag{3.3}$$

From this general framework, arise the following question:

How to design the intrinsic motivation process? The distribution over every valid task in the environment Ω being unknown, in order to maximize *Exploration* and *Control* without supervision, we must define the intrinsic motivation distribution to organize the curriculum of tasks:

Definition 3.3.4. The intrinsic tasks distribution, denoted as Ω_π, defines the process generating intrinsic tasks ω as a function of the current policy π.
We denote π^Ω the resulting policy after training the current policy π on the intrinsic task distribution Ω_π.
We define the objective of the intrinsic tasks distribution as the maximization of both exploration 3.2 and control 3.3 of π^Ω:

$${\Omega_\pi}^* = \underset{\Omega_\pi}{\text{argmax}} \Big[\text{explo}(\pi^\Omega) + \text{control}(\pi^\Omega)\Big]. \tag{3.4}$$

The definition of Ω_π is at the heart of the recent research effort in multi-task Intrinsically Motivated Agents. We provide in this section an extensive review of the existing proposals and find out that many approaches solely focus on a single aspect of the objective 3.4, being *Exploration* or *Control*.

In GCRL, the set of tasks corresponds to the goal space: $\text{supp}(\Omega) = \mathcal{G}$,

which most of the time correspond to the state space of the environment $\mathcal{G} = \mathcal{S}$ but sometimes is a subset embedded in $\mathcal{S}$, which can contain additional information required to take actions: $\mathcal{G} \subset \mathcal{S}$.

Empirical evaluation metric of exploration and control The intrinsic goals distribution should be designed in order to maximize the combination of exploration and control. This design is a thorny question and has been the subject of many research papers dedicated to the design of this intrinsic goal distribution (Florensa et al., 2018; Colas et al., 2018; Warde-Farley et al., 2018; Racaniere et al., 2019; Pong et al., 2019; Pitis et al., 2020; Gallouédec and Dellandréa, 2023; Kim et al., 2023).

The general objective 3.4 for Ω_π can be used to measure the performance of the intrinsic goals' distribution:

$$\begin{aligned} {\Omega_\pi}^* &= \underset{\Omega_\pi}{\operatorname{argmax}} \Big[\operatorname{explo}(\pi^\Omega) + \operatorname{control}(\pi^\Omega)\Big] \\ &= \underset{\Omega_\pi}{\operatorname{argmax}} \Big[H\Big[p_{\pi^\Omega}\Big] + \mathbb{E}_{g\sim p_{\pi^\Omega}}\Big[\mathbb{P}\Big(g \in \tau \mid \tau \sim \pi^\Omega(.|s,g)\Big)\Big]\Big]. \end{aligned} \tag{3.5}$$

The intrinsic goals distribution that maximizes the policy's exploration achieves:

$$\max_{\Omega_\pi} \operatorname{explo}(\pi^\Omega) = \max_{\Omega_\pi} H\Big[p_{\pi^\Omega}\Big] = H[\mathcal{U}(\mathcal{G}^*)],$$

where $\mathcal{U}(\mathcal{G}^*)$ is the uniform distribution over the set of valid goals $\mathcal{G}^*$, so that:

$$\begin{aligned} {\Omega_\pi}^* &= \underset{\Omega_\pi}{\operatorname{argmax}} \Big[H\Big[\mathcal{U}(\mathcal{G}^*)\Big] + \mathbb{E}_{g\sim\mathcal{U}(\mathcal{G}^*)}\Big[\mathbb{P}\Big(g \in \tau \mid \tau \sim \pi^\Omega(.|s,g)\Big)\Big]\Big]^1 \\ &= \underset{\Omega_\pi}{\operatorname{argmax}} \Big[\underbrace{\mathbb{E}_{g\sim\mathcal{U}(\mathcal{G}^*)}\Big[\mathbb{P}\Big(g \in \tau \mid \tau \sim \pi^\Omega(.|s,g)\Big)\Big]}_{S(\pi^\Omega)}\Big]. \end{aligned} \tag{3.6}$$

We define $S(.)$ as the *success coverage*, which measures the control of any policy π on the entire space of valid goals $\mathcal{G}^*$. Throughout this manuscript, we use this metric for evaluation purposes, using prior knowledge on our environments to build a testing goals set $\hat{\mathcal{G}}$, by splitting $\mathcal{G}^*$ into areas following a regular grid, and then uniformly sampling goals inside each part of the division.

[1]Note that the substitution of p_{π^Ω} with $\mathcal{U}(\mathcal{G}^*)$ within the argmax is possible with the assumption that there exists a single policy which maximizes independently both exploration and control, i.e. if $\max_{\Omega_\pi}[\operatorname{explo}(\pi^\Omega) + \operatorname{control}(\pi^\Omega)] = \max_{\Omega_\pi}[\operatorname{explo}(\pi^\Omega)] + \max_{\Omega_\pi}[\operatorname{control}(\pi^\Omega)]$.

$$S(\pi) = \frac{1}{|\hat{\mathcal{G}}|} \sum_{i=1}^{|\hat{\mathcal{G}}|} \mathbb{P}\big(g \in \tau \mid \tau \sim \pi(.|s, g)\big). \tag{3.7}$$

Equipped with these definitions, we now answer the following questions, in order to find the optimal intrinsic goal distribution design to optimize the above quantity:

1. How to prioritize goals?
2. How to generate goals?
3. How to learn goal representations?

3.3.2 How to prioritize goals?

When designing Intrinsically motivated goal-conditioned agents, a question that naturally arise is: how to prioritize goal selection to maximize exploration and control? Answering this question consists in defining a *curriculum*.

When looking at the definition of the success coverage objective, a natural choice for goal selection during training would be to sample goals from $\mathcal{U}(\mathcal{G})$, the uniform distribution over the goal space. However, two problems occur with this approach. First, it assumes knowledge of the support of the valid goal space which is not trivial in real world scenarios. Ideally, we would prefer to avoid the use of such prior knowledge on the MDP as much as possible in order for the methods to be robust to complex settings. Second and more importantly, goals have to be organized into a curriculum in order to maximize the competence of the agent through an adequate *learning trajectory*. Sampling from $\mathcal{U}(\mathcal{G})$ presumes that the order of tasks to learn does not matter. However, intuitively, some goals are tricky to reach and some stepping stones must be reached before achieving them. In the same way as humans first learn to crawl before they can walk and then run, goals in the IMGC context have to correspond to the current abilities of the agent. As a consequence, the distribution of pursued goals should be updated along the agent's learning trajectory, resulting in a virtuous circle. Obtaining such an efficient curriculum comes down to finding adequate criteria to automatically adapt the goal sampling distribution as a function of the agent performance. Note that *intrinsically rewarded agents* described in the previous section in some cases rely on similar proximal objectives, although these criteria are used to reward the agent for experiencing specific states rather than being applied to goal selection. In the following paragraphs, we investigate common goal selection criteria proposed in the literature.

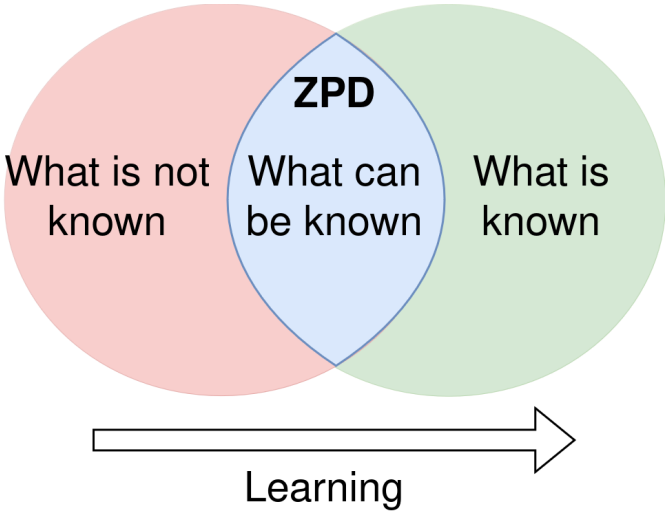

Figure 3.7: Illustration of the concept of "Zone of Proximal Development" (ZPD) from psychologist Vytgosky Vygotsky (2012).

Difficulty criteria

Difficulty criteria are used to determine whether the agent at a certain time possesses the abilities and the interest to target a specific goal. The "interest" to pursue a task has long been studied in psychology through observations of human behavior and how we acquire knowledge through experience. The concept of *Zone of Proximal Development* (Vygotsky, 2012) described as the frontier between what a learner knows and what it does not, constituting the space of knowledge that is not mastered yet but have the potential to be acquired. In Reinforcement Learning, these research findings led to the intuition that focusing on goals that are neither too hard nor too easy will optimally improve the agent's performance. As a matter of fact, in GCRL, targeting goals that are too easy which the agent already controls would yield no changes in the policy parameters. On the other hand, targeting too hard goals would result in systematic failure and therefore no reward. In both cases, there is no improvement on the policy. Based on this observation, Florensa et al. (2018) developed the concept of *Goals of Intermediate Difficulty* (GOIDs):

$$\mathcal{G}^{\pi}_{\text{GOID}} = \{g \in \mathcal{G} | P_{min} < P_{\pi}(g) < P_{max}\},$$

where $P_{\pi}(g)$ is the probability for the policy π to achieve goal g and P_{min} and P_{max} are hyper-parameters between 0 and 1. A Generative Adversarial Network (GAN) G is in charge of generating output goals in $\mathcal{G}^{\pi}_{\text{GOID}}$. This drives the agent to focus on goals that would get the most improvement on

its policy, leading to encouraging results in challenging AntMaze locomotion environments.

Sukhbaatar et al. (2017); OpenAI et al. (2021); Campero et al. (2020) use an adversarial framework where two agents evolve in the same environment but have different roles: Alice will "propose" the task to Bob by doing a sequence of actions and then Bob must repeat them. Bob is rewarded for achieving the tasks Alice proposes, and Alice is rewarded for finding tasks that are just out of reach for Bob, resulting in a sequence of tasks of increasing difficulty along Bob's learning trajectory.

The value function $V^{\pi}(s, g)$ could also provide valuable information on the difficulty for π to reach g from s, as it approximates the expected cumulative reward. Building on this observation, Zhang et al. (2020) use the epistemic uncertainty of V as a goal sampling criterion with the assumption that for goals that are too easy, the value function will confidently assign high values, while confidently assign low values for goals that are too hard.

In the same vein, Racaniere et al. (2019) use a Neural Network trained to predict the probability $P_{\pi}(g)$ and use a generation scheme to train the agent on goals of uniform difficulty.

Novelty criterion

The question: "what is an interesting goal to pursue?" could also be answered as follows: everything new and insufficiently explored in the past could be worth of interest, due to potentially unsuspected positive outcomes. Although behavioral research has shown some evidence of the fear and avoidance of novelty in individuals, an equally diverse evidence for its converse, positive biases toward novelty, and its role in individuals development, has been exhibited Gershman and Niv (2015). This boils down to a curiosity ability that leads humans to constantly learn about new things. Building on the significant results on Hard Exploration Problems that were obtained by count-based Bellemare et al. (2016); Ostrovski et al. (2017) and other novelty oriented methods, some IMGC methods incorporate novelty criteria in their goal selection process to solve complex tasks (Warde-Farley et al., 2018; Pong et al., 2019; Pitis et al., 2020; Kim et al., 2023). In (Pitis et al., 2020), the authors observe that the relabelling strategy (Andrychowicz et al., 2017) alone fails past a certain goal horizon. They propose the Maximum Entropy Goal Achievement (MEGA) objective:

$$J_{\text{MEGA}}(p^t_{ag}) = H(p^t_{ag}), \tag{3.8}$$

where $H(.)$ is the Shannon entropy (Shannon, 1948) and p^t_{ag} is the distribution of achieved goals by the agent at time t. This objective implies that the agent should seek to expand the support of p^t_{ag}. They want to select goals g such

that:

$$g^* = \underset{g \in \mathcal{B}(AG)}{\text{argmax}}\, J_{\text{MEGA}}(p^t_{ag|g'}) = \underset{g \in \mathcal{B}(AG)}{\text{argmax}}\, \mathbb{E}_{g' \sim q(g'|g)} H(p^t_{ag|g'}), \tag{3.9}$$

where $q(g'|g)$ is the distribution of achieved goals by the agent while targeting g. Because $q(g'|g)$ is unknown and in practice very hard to approximate with a learned model, the authors have chosen the useful heuristic $q(g'|g) = \mathbb{1}[g' = g]$, meaning that the agent achieves the goal g. The objective (3.9) then simplifies into:

$$\begin{aligned} g^* &= \underset{g \in \mathcal{B}(AG)}{\text{argmax}} - \log \left[p_{ag}(g) \right] \\ &= \underset{g \in \mathcal{B}(AG)}{\text{argmin}}\, p_{ag}(g). \end{aligned} \tag{3.10}$$

This comes down to a minimum density heuristic under the achieved goals distribution $p_{ag}(g)$, which they approximate with a Kernel Density Estimation (KDE). This simple minimum density objective turns out to be very efficient for the exploration of long horizons goals, obtaining impressive results in navigation tasks.

Learning Progress

Another, completely different perspective about training an agent to reach more and more difficult goals is based on measuring the learning progress of the agent on different goals and letting it train on the goal to which the absolute value of its measure of progress is the greatest. Such an agent will avoid wasting time both on goals that are either too easy for it, as it already masters them, and on goals that are too hard for it, on which it is making no progress at all. But, due to positive transfer between tasks, goals that were previously too hard can become achievable once the agent has made progress on easier tasks. The general idea is formalized in Lopes et al. (2012), and there are many instances of works using this approach, e.g. (Colas et al., 2018; Akakzia et al., 2020).

3.3.3 How to generate goals?

Another important question that arises in IMGC is: "how to generate goals?". Two strategies have been explored in the literature, including sampling goals that have been previously reached, and sampling goals using a leaned generative model. In this paragraph, we discuss the pros and cons of the two strategies.

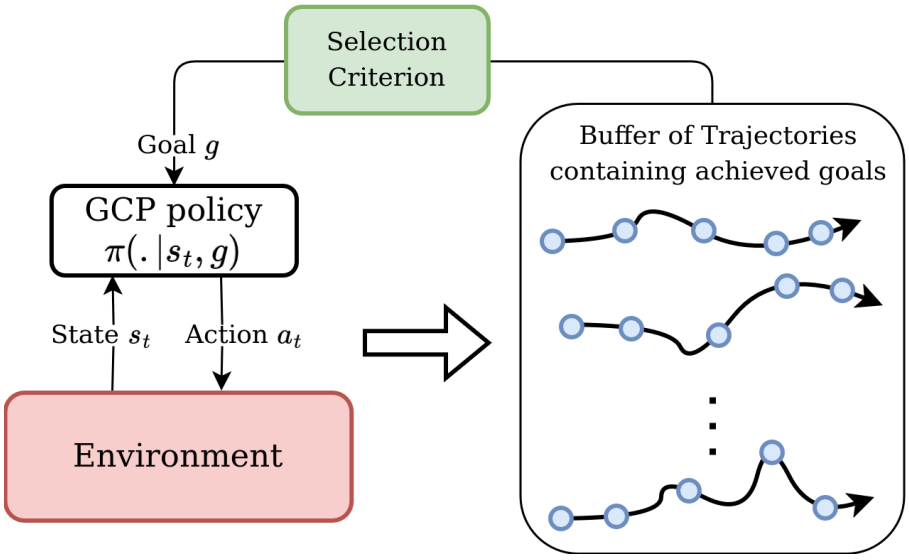

Figure 3.8: Intrinsically motivated agents can select goals directly from the Replay Buffer based on some criterion.

Sampling achieved goals

The most direct way to generate valid goals to pursue without having to worry about the constraints and topology of the goal space is to select goal candidates among the ones that have been achieved from past interactions of the agent with the environment (Warde-Farley et al., 2018; Zhang et al., 2020; Pitis et al., 2020). The main drawback of this approach is that the goal sampling process is biased towards current abilities of the policy, and thus could lead to vicious circles where the agent is stuck in a very restricted area of the environment. Such a goal sampling process would encourage staying in that area rather than getting exploring other goals (Campos et al., 2020). To prevent such an undesirable effect from happening, one could fix the policy bias by sampling achieved goals from another agent. For instance, Mendonca et al. (2021) divide *exploration* and *control* into two distinct policies: the *explorer*, an RL-based autotelic agent whose role is to expand its set of achieved states, and is rewarded by a *surprise* criterion such as in Pathak et al. (2019), and the *achiever*, a Goal-conditioned agent which gets rewarded for achieving goals previously reached by the explorer, that are supposed to be covering the entirety of the goal space $\mathcal{G}$. A GCP agent could also be provided with expert trajectories and be trained to reach the intended expert goals through imitation learning (Ding et al., 2019).

Goal Sampling from a Generative Model

Sampling from a generative model can be very efficient from an exploration point of view as it can generate goals that were never reached by the agent. A model trained to imagine new challenging goals can push the agent past its comfort zone, potentially leading it to far more difficult and interesting areas of the environment than the ones it has already reached. Florensa et al. (2018) use GANs to generate a sequence of Goals Of Intermediate Difficulty (GOIDs): a Generator (G) is trained to output GOIDs while a Discriminator (D) aim to distinguish real GOIDs coming from the environment from the ones coming from G. Racaniere et al. (2019) define a goal generator that they call "Setter", which consists of a Real-Valued Non-Volume Preserving (RNVP) network (Dinh et al., 2016) which has the useful property of providing access to exact log-likelihood of a sample, by relying on an invertible network. Nair et al. (2018); Pong et al. (2019); Nair et al. (2020) learn a compact latent representation of images observations using a VAE (Kingma and Welling, 2013) on the agent's trajectories. This formulation enables goal generation through prior sampling from $p(z)$ in the latent space.

How to deal with invalid goals

Another question that arises when working with generative models is: "how do I ensure that the agent is provided with feasible goals in the environment?". Most Goal-Conditioned RL methods ignore this issue and work with a predefined goal space where all goals are valid (Schaul et al., 2015; Andrychowicz et al., 2017; Plappert et al., 2018b; Ding et al., 2019). Besides, some IMGC methods assume the access to a function that automatically rejects non-valid generated goals (Chen et al., 2021a). As the access to such a function may not be guaranteed in applications that matter, a goal generation model should ideally incorporate a training component that ensures the validity of its output. With their concept of GOIDs, Florensa et al. (2018) ensure that goals g sampled from the GAN generator G matches the following constraint $P_{min} < \mathbb{P}^{\pi}(g) < P_{max}$, therefore G should discard unreachable goals such that $\mathbb{P}^{\pi}(g) = 0$. On the other hand, Racaniere et al. (2019) add a *validity loss* to the Setter with the purpose of reducing the probability of sampling goals that are outliers with respect to $\mathcal{B}^{\pi}(AG)$, the buffer of achieved goals by π.

3.3.4 How to learn goal representations?

In GCRL, most simulation environments are embedded with a low dimensional and meaningful environment state and goal space (Plappert et al.,

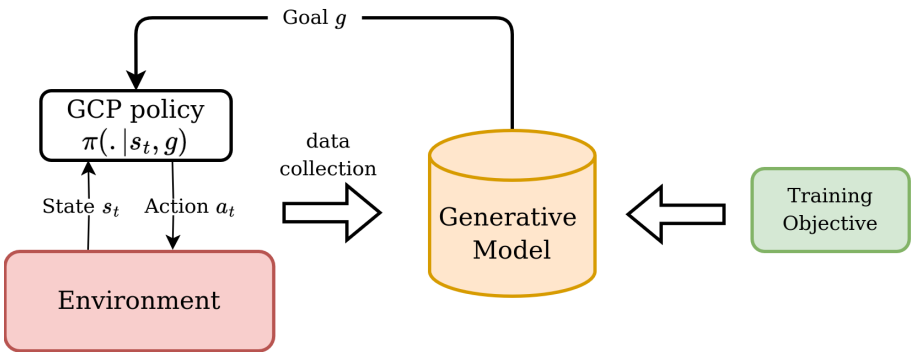

Figure 3.9: Goal for intrinsic motivation can be automatically generated by a Generative Model trained on data coming from the agent interaction with the environment. The training objective is defined by some goal selection criterion.

2018b; Tunyasuvunakool et al., 2020; Gallouédec et al., 2021; Freeman et al., 2021; de Lazcano et al., 2023). This low dimensional space can correspond to a location to reach in a maze, a position of an object to grab for a robot, and many other varieties among a range of plausible tasks. This type of predefined state and goal spaces make the objective of learning policies to achieve goals much easier, as the agent is provided with observations containing the needed information to take actions. However, in real world scenarios, or even in some specific simulations, the access to such semantically meaningful observation space is not guaranteed. For this reason, many methods operate on raw pixels observations of the environment using any dimensionality reduction technique to extract a compact and useful representation of the observation space (Mankowitz et al., 2018; Nair et al., 2018; Racaniere et al., 2019; Pong et al., 2019; Campero et al., 2020; Islam et al., 2022).

The most straightforward method to learn a compact representation of the environment is to use the observations encountered by the policy as training data. We refer to this method as Online Representation Learning: as illustrated in Figure 3.10, the agent and its representation of the observation space are learned jointly, alternating representation learning and policy improvement.

We start by addressing skill discovery with Empowerment, a class of methods that learn to represent the state space with a cluster of latent skills. Then, we discuss latent representation learning with image reconstruction using VAE.

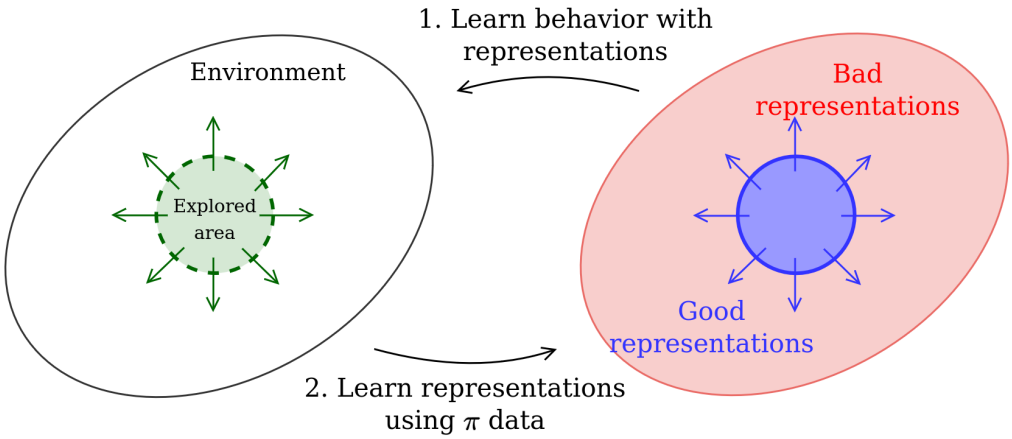

Figure 3.10: Online Representation Learning in RL. Left: current exploration of the policy in the environment. Right: learned representation with the policy data. As the policy expands its behavior in broader areas of the environment (Step 1.), learned representations become accurate in an increasingly larger part of the environment (Step 2.).

Skill discovery with Empowerment

In the context of Skill discovery, the set of tasks is defined as a discrete *skill space*: $\text{supp}(\Omega) = \mathcal{Z}$. Every skill $z \in \mathcal{Z}$ corresponds to a specific set of states to achieve: $\mathcal{S}_z \subset \mathcal{S}$. Therefore, $\mathcal{Z}$ can be seen as a particular clustering of the state space. In the absence of a predefined skill space, we have to design an unsupervised training process to undertake skill discovery and learning. This has been recently approached through the *Empowerment* concept.

Empowerment is defined as a measure of an agent's awareness and control over its environment based on an Information-Theoretic point of view (Klyubin et al., 2005; Salge et al., 2013). In Reinforcement Learning, methods using this framework to autonomously learn skills are referred to as Information-theoretic skill discovery (Gregor et al., 2016; Eysenbach et al., 2018b; Achiam et al., 2018; Jabri et al., 2019; Sharma et al., 2020; Campos et al., 2020; Kamienny et al., 2021). In the context of skill discovery in RL, empowerment is defined as the maximization of the mutual information $I(Z, S)$ between the set of states reached by the agent S, and a set of skills Z:

$$I(Z,S) = \underbrace{H(S) - H(S|Z)}_{\text{forward}} = \underbrace{H(Z) - H(Z|S)}_{\text{reverse}}, \tag{3.11}$$

where $H(S)$ is the entropy of the states reached by the agent and $H(S|Z)$

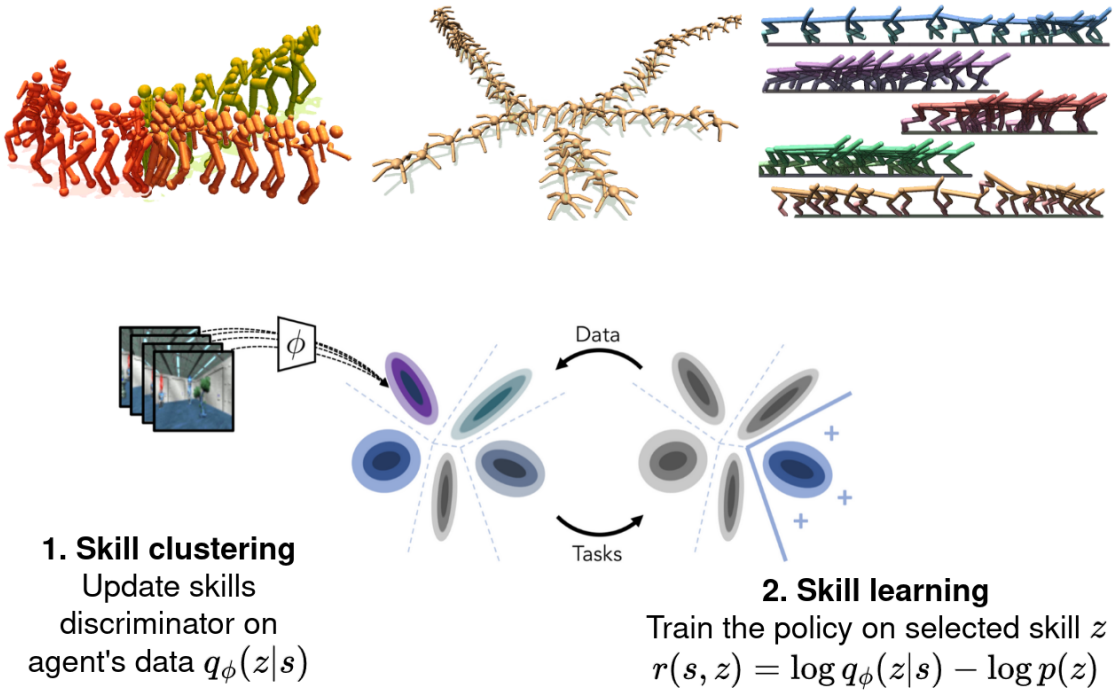

Figure 3.11: Two-step empowerment maximization: (1) clustering the agent's state space into a latent skill space. (2) selecting a latent skill z and training the policy on the appropriate reward function as defined in Equation (3.14). Figure adapted from Jabri et al. (2019).

is the state conditional entropy depending on the skills, defined using the probability $P(S|Z)$ that the state S is reached when asking for skill Z. The formulation of (3.11) has a very appealing form regarding the coupled objective of exploration and control. Indeed, when looking at the *forward* formulation of the above objective, one can see that increasing $H(S)$ would result in expanding the set of states reached by the agent. On the other hand, decreasing $H(S|Z)$ can be viewed as a particular approximation of a skill-performing objective. Indeed, the optimal skill-conditioned policy will deterministically perform the skill, resulting in a conditional entropy of zero $H(S|Z) = 0$. Most methods optimize the reverse formulation of objective (3.11):

$$I(Z, S) = \mathbb{E}_{s,z\sim p(s,z)}[\log p(z|s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)], \tag{3.12}$$

and then, by leveraging the non-negativity property of the KL divergence to compute a variational lower bound Barber and Agakov (2004), we get

$$I(Z,S) \geq \mathbb{E}_{s,z\sim p(s,z)}[\log q_\phi(z|s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)], \tag{3.13}$$

where $q_\phi(z|s)$ is a variational approximation of the true posterior $p(z|s)$, see Tzikas et al. (2008) for more details. In standard implementations, $q_\phi(z|s)$ is a skill discriminator given the current state s of the agent, for example a Neural Network, that is trained online along the agent's progress. The variational lower bound is optimized by training the skill-conditioned policy $\pi(.|s,z)$ using the following reward function:

$$r(s,z) = \log q_\phi(z|s) - \log p(z), \quad z \sim p(z), \tag{3.14}$$

where skills z are sampled at the beginning of every episode from a prior distribution $p(z)$, which can be either fixed or not. This framework offers a very convenient setup to learn a handful of skills without any supervision in an open environment, which can be used for pretraining policies that are fine-tuned afterward on downstream tasks. Empowerment is a very active research field in RL, recent works build on the classic implementation from Gregor et al. (2016); Eysenbach et al. (2018a) to highlight the major issues and drawbacks of these approaches such as lack of skill diversity and mode collapsing. In Campos et al. (2020), the authors suggest that we need to explore the state space before learning skills. Otherwise, the agent lacks an incentive to discover more complex, challenging behaviors (Park et al., 2023a) or suffers from the lack of temporal relation between the state and the skill space (Park et al., 2024). There are strong connections between Information-theoretic skill discovery and Goal-conditioned methods: authors in (Choi et al., 2021) show that the standard goal-reaching objective can be subsumed in the general empowerment objective (3.11) under certain conditions over the goal-reaching reward function. A key limitation of such setting is that it does not directly provide a policy that can control the environment unlike Intrinsically motivated Goal-Conditioned methods, in the sense that the policy conditioning is under a latent skill and not the explicit goal space. However, Achiam et al. (2018) showed that one can use the skill discriminator $q_\phi(z|s)$ to find out the corresponding latent skill z^* of a target state s, next running $\pi(.|.,z^*)$ to achieve s.

Online VAE latent representation

An alternative to skill empowerment for high dimensional data is to assume the existence of a compact latent space, from which each goal and state can be efficiently sampled. In that vein, Nair et al. (2018) introduced the "Reinforcement Learning with Imagined Goals" (RIG) algorithm. They train a VAE

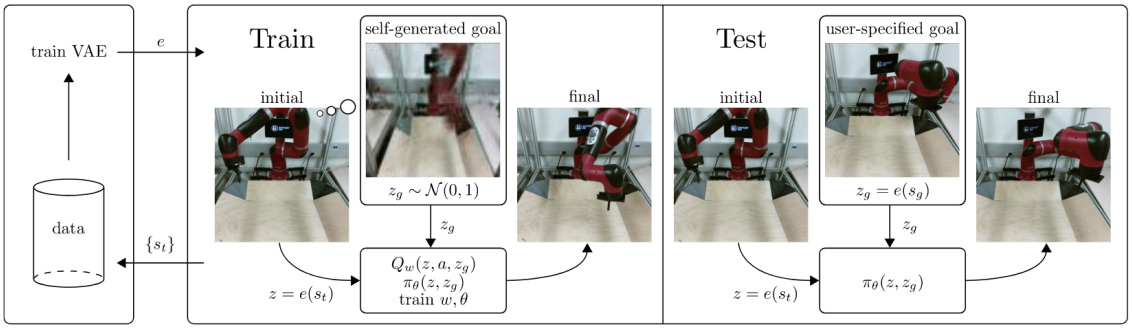

Figure 3.12: Illustration of the RIG method from Nair et al. (2018). The agent is trained on self generated goals in a latent space that corresponds to the prior of a VAE trained on data encountered by the policy.

with sampled data from the policy replay buffer, with a classic pixel reconstruction error. The specificity of this method is that the Goal-conditioned policy is fully trained in the latent space of the VAE, using the encoder to represent states. In addition, intrinsic goals are generated by sampling from the prior distribution, which leads the policy to train on previously unseen goals, as illustrated in Figure 3.12. At every time step t, the policy takes action in the following way:

$$a_t = \pi_\theta(z_t, z_g), \quad z_t = \mu_\phi(s_t), \; z_g \sim p(z), \tag{3.15}$$

where z_g is a latent goal sampled at the beginning of an episode from the prior distribution $p(z)$ and μ_ϕ is the VAE encoder used to compute the latent state z_t from the pixel observation s_t. RIG led to significant improvement over visual robotic tasks, both on simulation and real robot, compared to classical RL algorithms that directly train the GCP on tests tasks without using intrinsic motivations.

Overcoming Exploration Bottleneck in Online Representation Learning

Learning a good representation while exploring and learning to acquire behavior is not an easy task, as it has been observed many times in the literature (Campos et al., 2020; Stooke et al., 2021; Yarats et al., 2021a). This is especially true in environments presenting a bottleneck topology, such as mazes, where, in the absence of oriented exploration, the agent is only able to reach goals in a small subspace of the environment. The RIG method is unable to overcome such exploration bottlenecks. Indeed, learning representation using data coming from a restricted policy and sampling intrinsic goals from $p(z)$

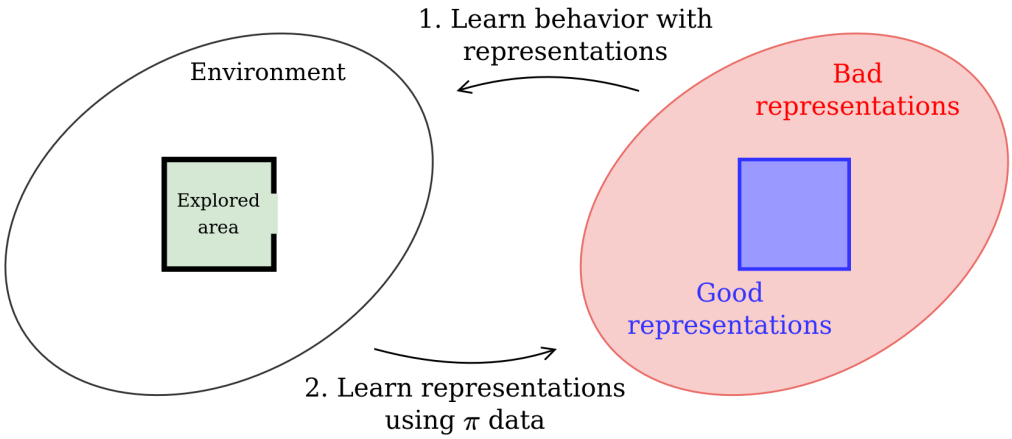

Figure 3.13: Illustration of the Exploration Bottleneck when learning representations in Online RL: (1) the agent learns representations using biased and limited data coming from the policy, (2) The agent uses these poor representations to act in the environment, which results in reinforcing behavior in already covered areas.

results in a vicious circle making learning representations and goal-reaching in broader areas impossible, as illustrated in Figure 3.13.

As a matter of fact, authors in Campos et al. (2020) advocate that the exploration bottleneck might be the major issue in Information-theoretic skill discovery methods, and provide a theoretical analysis to back up this statement. They observed that many existing methods tend to discover skills that provide a poor coverage of the state space, due to a bias toward data coming from the current policy behavior when building the latent skills representations. To alleviate this issue, they propose to split the training pipeline into three distinct phases: *Explore*, *Discover* and *Learn*. The *Explore* and *Discover* phases are dedicated to learning unbiased state covering representations, while the *Learn* phase refers to learning skills from these representations. Building on similar conclusions, decoupling representation and behavior learning in online RL has been an active research topic in the past few years (Stooke et al., 2021; Yarats et al., 2021a).

Starting with the observation that RIG might be susceptible to exploration bottlenecks as we just discussed, Pong et al. (2019) propose the SKEW-FIT method, an extension of RIG where the VAE is leveraged on a skewed distribution of the policy samples, that aims at selecting learning goals of the agent from an uniform distribution over states that the agent can reach. This

distribution is given as $p_{\theta,\phi}(\mathbf{S}) \triangleq \int_{\mathcal{G}} q_\phi(\mathbf{G}) p_\theta(\mathbf{S} \mid \mathbf{G}) d\mathbf{G}$, with $q_\phi(\mathbf{G})$ the distribution of goal selection, and $p_\theta(\mathbf{S} \mid \mathbf{G})$ the probability that state S is reached by the agent when conditioned on G. The objective of skewfit is to fit the generative model q_ϕ on the uniform distribution over the support of $p_{\theta,\phi}$, denoted as $\mathcal{U}_S$. Assuming that the samples from the buffer at step t come from p_{θ,ϕ_t}, this is done via Importance Sampling, to maximize the following maximum likelihood estimation:

$$\begin{aligned} \mathcal{L}(\phi) &= \mathbb{E}_{\mathbf{S}\sim\mathcal{U}_\mathbf{S}} \left[\log q_\phi(\mathbf{S})\right] \\ &= \mathbb{E}_{\mathbf{S}\sim p_{\theta,\phi_t}} \left[\frac{\mathcal{U}_\mathbf{S}(\mathbf{S})}{p_{\theta,\phi_t}(\mathbf{S})} \log q_\phi(\mathbf{S})\right] \\ &= \mathbb{E}_{\mathbf{S}\sim p_{\theta,\phi_t}} \left[w_{t,\alpha} \log q_\phi(\mathbf{S})\right], \end{aligned} \tag{3.16}$$

where $w_{t,\alpha}$ is the IS ratio. As using $w_{t,\alpha} = p_{\theta,\phi_t}(\mathbf{S})^{-1}$ is intractable due to the marginalization stated above, authors of Skew-Fit consider the following approximation, which uses previous generative distribution of goals as a proxy for $p_{\theta,\phi_t}(\mathbf{S})$:

$$w_{t,\alpha} := q_{\phi_t}(\mathbf{S})^\alpha, \quad \alpha < 0, \tag{3.17}$$

where α is a hyper-parameter controlling the skewing importance of the samples. From this, the approach introduces a skewed distribution from which goals can be sampled for training the agent:

$$\begin{aligned} p_{\text{skewed}_t}(\mathbf{s}) &\triangleq \frac{1}{Z_\alpha} w_{t,\alpha}(\mathbf{s}) \delta\left(\mathbf{s} \in \{\mathbf{s}_n\}_{n=1}^N\right), \\ Z_\alpha &= \sum_{n=1}^N w_{t,\alpha}\left(\mathbf{s}_n\right), \end{aligned} \tag{3.18}$$

with $\{\mathbf{s}_n\}_{n=1}^N$ a batch of states sampled from the buffer (or the prior of the VAE). Skew-Fit builds on a resampling of states from this batch using this skewed, batch-normalized, distribution. It led to some significant improvement over RIG, especially in environments with exploration bottlenecks such as mazes.

Chapter 4

Stein Variational Goal Generation

In this chapter, we present the first contribution of this Thesis, aiming to design an intrinsic goal generation process able to efficiently maximize the success coverage objective (3.7) defined p. 53. We introduce the Stein Variational Goal Generation (SVGG) algorithm, leveraging the properties of Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016) in order to obtain an intrinsic goal distribution combining incentives both to explore and control, and to be robust to unexpected environment changes.

We begin with the introduction of SVGD, essential to our method. Then, we provide details about the construction of the criterion shaping the intrinsic goal distribution. This criterion is based on a learned model of the agent's goal reaching capacities and is designed to target goals in the zone of proximal development. In addition, we provide theoretical justification for the ability to handle unexpected environment changes, which we call "recovery property". Finally, we show in our experiments that SVGG outperforms methods from the literature on several tasks.

Contents

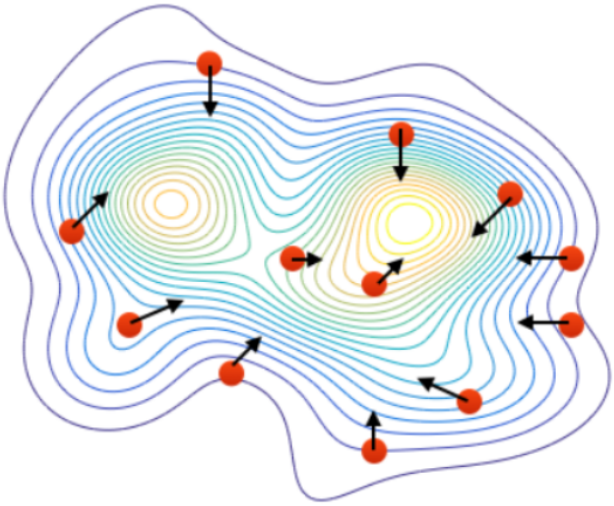

Figure 4.1: Illustration of particles evolution in SVGD: the transform function is akin to a vector field that shapes the particles' direction as a function of their position.

4.1 Stein Variational Gradient Descent

Our method builds on Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016) to approximate the distribution of goals of interest. SVGD is a powerful non-parametric tool for density estimation. It can be applied when the partition function of the target distribution p to approximate is intractable, which is the case when we do not know the support of the valid goal distribution in the environment. It stands as an efficient alternative to Monte Carlo Markov Chain (MCMC) methods, which are proven to converge to the true distribution p but are usually too slow to be used in complex optimization processes. It also stands as an alternative to variational inference of parametric neural distributions q, which are restricted to pre-specified families of distributions (e.g., Gaussians or mixtures of Gaussians) that may not fit target distributions. Instead, it models q as a set of particles $\{x_i\}_{i=1}^{n}$, all belonging to the support of p (as illustrated in Figure 4.1.

The idea behind SVGD is to approximate the target distribution p with q by minimizing their KL-divergence:

$$\min_{q} KL(q||p). \tag{4.1}$$

This objective is reached by iterative deterministic transforms as small perturbations of the identity map, on the set of particles:

$$T(x) = x + \epsilon\phi(x), \tag{4.2}$$

where ϕ is a smooth transform function that indicates the direction of the perturbation, while ϵ is the magnitude. The authors draw a connection between KL-divergence and the *Stein operator*:

$$\mathcal{A}_p\phi(x) = \phi(x)\nabla_x \log p(x)^T + \nabla_x\phi(x) \tag{4.3}$$

by showing that:

$$\nabla_\epsilon KL(q_{[T]}||p)|_{\epsilon=0} = -\mathbb{E}_{x\sim q}[\text{trace}(\mathcal{A}_p\phi(x)], \tag{4.4}$$

where $q_{[T]}$ is the distribution of particles after the transformation T. The KL minimization objective is thus related to the *Stein Discrepancy*, defined as: $\mathbb{S}(q,p) = \max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}[\text{trace}(\mathcal{A}_p\phi(x)]$[1].

Minimizing Stein Discrepancy being intractable as such, Liu et al. (2016) and Chwialkowski et al. (2016) introduce the *Kernelized Stein Discrepancy* (KSD) where the idea is to restrict to projections ϕ that belong to the unit ball of a reproducing kernel Hilbert space $\mathcal{H}$ (RKHS), for which there is a closed form solution. The KSD is defined as:

$$\mathbb{S}(q,p) = \max_{\phi\in\mathcal{H}}\{\mathbb{E}_{x\sim q}[\text{trace}(\mathcal{A}_p\phi(x)], \quad s.t \quad ||\phi||_{\mathcal{H}} \leq 1\}, \tag{4.5}$$

whose solution is given by:

$$\phi^*(.) = \mathbb{E}_{x\sim q}[\mathcal{A}_p k(x,.)], \tag{4.6}$$

where $k(x,x')$ is the positive definite kernel of the RKHS $\mathcal{H}$. The RBF kernel $k(x,x') = \exp(-\frac{1}{h}||x-x'||_2^2)$ is commonly used. Therefore, the steepest descent on the KL-objective is given by the optimal transform:

$$\begin{aligned} &x_i \leftarrow x_i + \epsilon\phi^*(x_i), \quad \forall i = 1\cdots n, \\ &\phi^*(x_i) = \frac{1}{n}\sum_{j=1}^{n}\Big[\underbrace{k(x_j,x_i)\nabla_{x_j}\log p(x_j)}_{\text{attractive force}} + \underbrace{\nabla_{x_j}k(x_j,x_i)}_{\text{repulsive force}}\Big]. \end{aligned} \tag{4.7}$$

The "attractive force" in the update (4.7) drives the particles toward high density areas of the target p. The "repulsive force" pushes the particles away from each other, therefore fosters exploration and avoids mode collapse. Note that if $n = 1$, the update in (4.7) corresponds to a Maximum a Posteriori. The SVGD method has already been successfully explored in the context of RL. The Stein Variational Policy Gradient (SVPG) algorithm (Liu et al., 2017) employs SVGD to maintain a distribution of efficient agents as particles. It strongly differs from our approach, since we consider particles as behavior

[1]Note that $\mathbb{S}(q,p) = 0$ only if $q = p$.

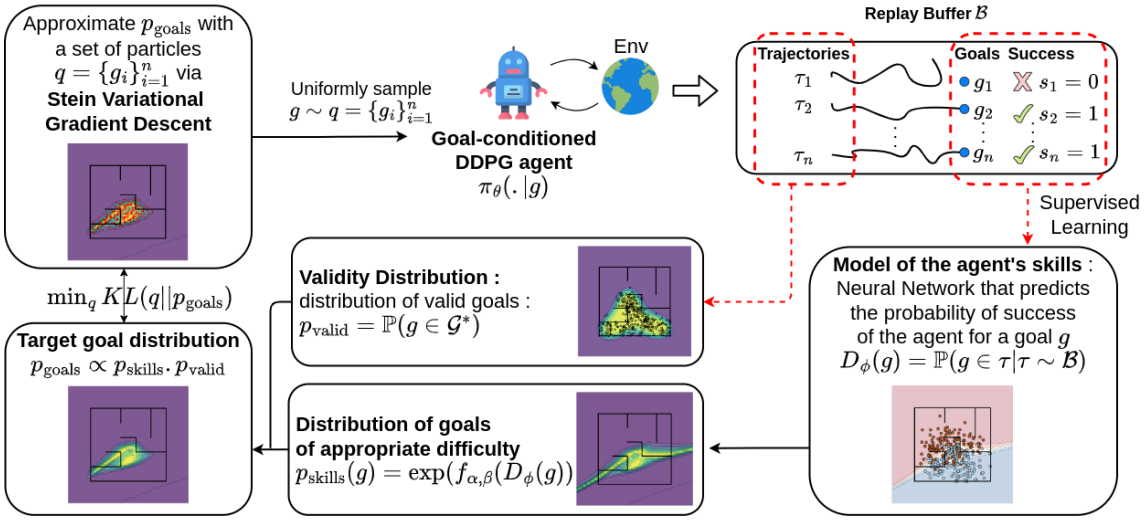

Figure 4.2: Overview of the SVGG method. The interaction of agents with their environment is stored in a replay buffer (top right) and used to learn D_ϕ, a model of its abilities to achieve goals (bottom right). We build on this model to compute a distribution of goals of appropriate difficulty p_{skills}, leveraging a validity distribution p_{valid} to stay inside the space of valid goals. The obtained behavioral goal distribution p_{goals} is approximated with particles $\{g_i\}_{i=1}^n$ using Stein Variational Gradient Descent (SVGD), and the agent samples a goal from these particles.

goal candidates, while SVPG aims at capturing the epistemic uncertainty about policy parameters. Chen et al. (2021b) also rely on SVGD to build a strategy to generate goals for goal-directed agents, but in a very simplified setting without the attractive force from (4.7), which prevents from fully benefiting from this theoretical framework. Notably, such a kind of approach is particularly sensitive to catastrophic forgetting.

4.2 Optimal Difficulty Goal Distribution

In this section we introduce our Stein Variational Goal Generation (SVGG) algorithm. The pseudo-code of SVGG is given in Algorithm 3 (a more detailed version is given in Appendix A.3.2). Our aim is to obtain a curriculum to sample goals of appropriate difficulty for the RL agent, where the curriculum is represented as an evolving goal sampling probability p_{goals}. To do so, we

Algorithm 3 *RL with Stein Variational Goal Generation*

Input: a GCP π_θ, a success predictor D_ϕ, a reachability predictor V_ψ, buffers of transitions $\mathcal{B}$, reached states $\mathcal{R}$ and success outcomes O, a kernel k.
Sample a set of particles $q = \{g^i\}_{i=1}^m$ uniformly from states reached by pre-runs of π_θ;
for n epochs **do**
  ▶ ***Environment interaction***
  Perform Rollouts of π_θ conditioned on goals uniformly sampled from q;
    Store transitions in $\mathcal{B}$, visited states in $\mathcal{R}$ and success outcomes in $\mathcal{O}$;
  ▶ ***Skill model update***
    Update model D_ϕ (e.g., via ADAM), using gradient of $\mathcal{L}_\phi$ (4.9) with samples from $\mathcal{O}$;
  ▶ ***Prior Update***
    Update model R_ψ according to all states in $\mathcal{R}$;
  ▶ ***Particles Update*** (t steps)
    Update particles:
  $g_i \leftarrow g_i + \epsilon \frac{1}{m} \sum_{j=1}^m \Big[\underbrace{k(g_j, g_i) \nabla_{g_j} \log p_{\text{goals}}(g_j)}_{\text{attractive force}} + \underbrace{\nabla_{g_j} k(g_j, g_i)}_{\text{repulsive force}} \Big]$;
  ▶ ***Policy Learning***
    Improve agent with any Off-Policy RL algorithm (e.g., DDPG) using transitions from $\mathcal{B}$;
end for

maintain a model of the agent's skills – or goal reaching capability –, which helps define a distribution p_{skills}. This distribution assigns probability mass to areas of goals of appropriate difficulty. Additionally, with a simple one class SVM, we learn a validity distribution p_{valid} preventing the particles from being sampled from non-valid areas of the goal space. Then, we aim at sampling training goals from the following target distribution:

$$p_{\text{goals}}(g) \propto p_{\text{skills}}(g).p_{\text{valid}}(g). \tag{4.8}$$

Since computing the partition function is intractable for such a distribution formulation, we rather sample uniformly over a set of particles $q = \{x_i\}_{i=1}^m$, that are optimized through SVGD to approximate $p_{\text{goals}}(g)$. Besides, we are centrally interested in tracking useful areas to train the agent. In this context, dealing with a set of particles representing the state of the full exploration landscape appears better fitted than methods relying on single candidates, such as MCMC or Langevin Dynamics. Indeed, these alternatives would

produce samples very correlated in time, with unstable dynamics. Our choice also improves the interpretability of the process, by providing a comprehensive picture of the current behavior distribution along training. Formal definitions of the two components of p_{goals} are given below.

4.2.1 Model of the agent's skills

The probability p_{skills} is modeled as a Neural Network D_ϕ whose parameters ϕ are learned by gradient descent on the following Binary Cross Entropy (BCE) loss:

$$\mathcal{L}_\phi = \sum_{(g^i, s^i) \in O} s^i (\log D_\phi(g^i)) + (1 - s^i)(\log(1 - D_\phi(g^i))), \tag{4.9}$$

where $O = \{g^i, s^i\}_{i=1}^{n_B}$ is a batch of $(goal, success)$ pairs coming from recent trajectories of the agent in the environment. The sampled goals are those whose predicted probability of success is neither too high nor too low, that is we avoid $D_\phi(g) \approx 1$ or $D_\phi(g) \approx 0$.

To build p_{skills} based on the prediction of D_ϕ, we use a beta distribution whose maximum density point mass is determined according to the output of D_ϕ, by two hyper-parameters α and β that shape the distribution and control the difficulty of the addressed goals, as illustrated in Figure 4.3.

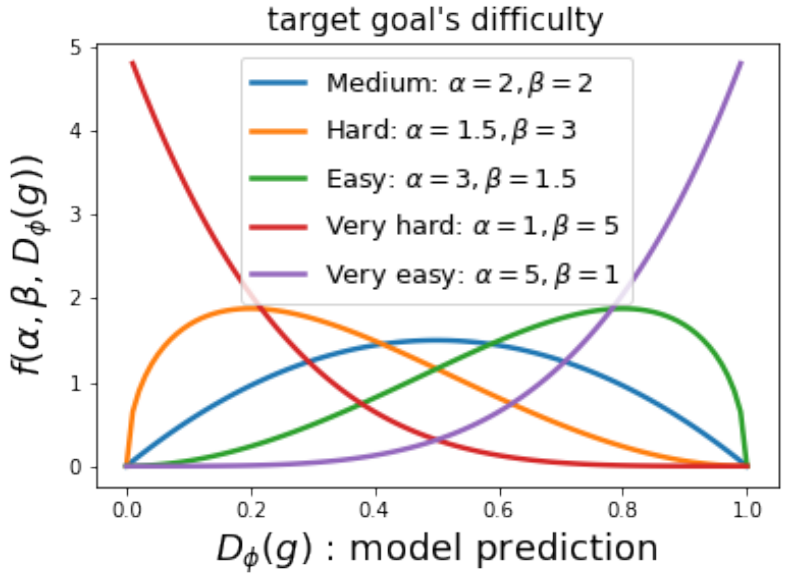

Figure 4.3: Modulation of goal's target difficulty with beta distributions. $D_\phi(g)$ is the model's predicted probability of the agent achieving goal g.

We define the p_{skills} distribution as an energy-based density whose potential is the output of the beta distribution f:

$$p_{\text{skills}}(g) \propto \exp\left(f_{\alpha,\beta}(D_\phi(g)\right). \tag{4.10}$$

In Section 4.4.5, we compare the relative performance of 5 pairs of α and β and show that targeting a *Medium* difficulty works best. We stick to this setting in the rest of the study.

4.2.2 Validity distribution

As outlined in Racaniere et al. (2019), we would like to only sample valid goals. To do so, instead of their *validity loss*, we define a validity distribution which represents the probability that a goal g belongs to the set of valid (reachable) goals $\mathcal{G}^* \subseteq \mathcal{G}$. However, $\mathcal{G}^*$ is not known in advance. To circumvent this difficulty, the states already reached by the agent are stored in an archive $\mathcal{R}$ and we aim at defining the validity distribution as depending on the posterior probability given $\mathcal{R}$: $p_{\text{valid}}(g) \propto \mathbb{P}(g \in \mathcal{G}^*|\mathcal{R})$. We progressively build this distribution with a One Class SVM (OCSVM) (Chen et al., 2001). This model is mainly designed for outlier or novelty detection in the absence of labeled data. Given a dataset $X \in \mathbb{R}^d$, it defines a boundary of the data support in $\mathbb{R}^d$, while keeping a small portion of the data points out of that boundary. With data points being goals from $\mathcal{R}$, we get

$$p_{\text{valid}}(g) \propto V_\psi(g), \tag{4.11}$$

where $V_\psi(g)$ is the output of the OCSVM model trained on $\mathcal{R}$, with parameters ψ. As the agent progresses and expands its set of achieved states through training, it eventually reaches the environment boundary. In this case, we can expect $V_\psi(g) \approx \mathbb{P}(g \in \mathcal{G}^*|g \in \omega)$ for any area $\omega \subseteq \mathcal{G}$.

4.2.3 Recovery property

As demonstrated in Theorem 4.2.1 and empirically validated in Section 4.4.3, SVGG benefits from a useful recovery property: when the environment suddenly changes, the SVGG agent will spontaneously resample goals in the areas that are affected by the change.

Theorem 4.2.1. ***Recovery property****: Let us denote as $\mathcal{G}^+$ the set of goals g such that $V_\psi(g) > 0$ and $\mathcal{C} \in \mathbb{R}^d$ its convex hull. Assume that, at a given iteration l, $D_\phi(g) \approx 1$ for every $g \in \mathcal{G}^+$ (i.e., the whole set $\mathcal{G}^+$ is considered as mastered by $D_\phi(g)$), and that V_ψ is well calibrated on that set: for any area $\omega \subseteq \mathcal{G}^+$ and any goal $g \in \omega$, $V_\psi(g) \approx P(g \in \mathcal{G}^*|g \in \omega)$. Assume*

also that we use a kernel which ensures that the Kernelized Stein Discrepancy $KSD(q, p_{goals})$ *of any distribution* q *with* p_{goals} *is 0 only if* q *weakly converges to* p_{goals}[2]*. Then, with no updates of the models after iteration* l *and a number of particles* $m > 1$*, any area* $\omega \subseteq \mathcal{G}^+ \cap \mathcal{G}^*$ *with diameter* $diam(\omega) \geq \sqrt{d}\frac{diam(\mathcal{C})}{(\sqrt[d]{m}-1)}$ *eventually contains at least one particle, whenever* $KSD(\{x^i\}_{i=1}^n, p_{goals}) = 0$ *after convergence of the particle updates.*

The above theorem (proof in Appendix A.1) ensures that, even if the success model overestimates the capacities of the agent for some area ω (e.g., due to changes in the environment, catastrophic forgetting or success model error), some particles are likely to go back to this area once every goal in $\mathcal{G}^*$ looks well covered by the agent, with an increasing probability for more particles. This way, the process can reconsider overestimated areas, by sampling again goals in them, and hence correcting the corresponding predictions. This leads to attracting attention of p_{skills} back to these difficult areas. Approaches such as MEGA do not exhibit such recovery properties, since they always sample at the boundary of their achievable goal distribution, which is likely to incrementally grow towards $\mathcal{G}^*$. The particles of our approach can be seen as attention trackers which remodel the behavior distribution and mobilize the effort on useful areas when needed. This is much better than uniformly sampling from the whole space of achievable states with small probability, which would also ensure some recovery of forgotten areas but in a very inefficient way. This theoretical guarantee of SVGG is empirically validated by the experiment from Figure 4.7 p. 79, which shows the good recovery property of our approach, after a sudden change in the environment.

4.3 Experiments

Let us now empirically investigate the properties of SVGG. In this section, we discuss the experimental setting used to compare approaches.

4.3.1 Compared Approaches

As baselines, we choose two methods from the literature that span the existing trends in unsupervised RL with goal-conditioned policies, MEGA Pitis et al. (2020) and GoalGan Florensa et al. (2018). In addition, the Random baseline randomly selects the behavior goal among past achieved states.

[2] As assumed for instance in Liu (2017). This is not always true, but gives strong insights about the behavior of SVGD. Refer to Gorham and Mackey (2017) for more discussions about KSD.

We also evaluate two SVGG ablations: a *No Validity Distribution* version, which considers $p_{\text{goals}} \propto p_{\text{skills}}$ and a *Only Validity* version, where $p_{\text{goals}} \propto p_{\text{valid}}$.

Learning to reliably achieve all valid goals can be decomposed in learning to sample appropriate behavioral goals, and learning to achieve these goals. Our focus being on the goal sampling mechanism, all compared approaches learn to achieve goals using DDPG and HER with a mixed strategy described in Appendix A.3. In all versions of SVGG, we use $m = 100$ particles to approximate the target distribution. Implementation details about all considered architectures are given in Appendix A.3.

4.3.2 Questions

To compare the performance of SVGG to baselines, we investigate the following questions:

1. Does SVGG maximize the success coverage metric better than the baselines?
2. Is SVGG more robust than the baselines to catastrophic forgetting that may occur in sudden environment changes?
3. What is the impact of target difficulty (i.e., beta distribution as described above) on success coverage maximization.

4.3.3 Evaluation environments

To answer the above metric-oriented questions, we use a modified version of the FetchReach and FetchPush environments (Plappert et al., 2018a) where we have added obstacles in the workspace of the robot arm to increase the amount of discontinuities between the optimal goal-oriented behaviors. We also compare SVGG to baselines in the U-shaped AntMaze environment (Trott et al., 2019) and in a hard version of FetchPickAndPlace. Additionally, to provide more analysis-oriented visualizations, we use a Point-Maze environment where an agent moves a point without mass within a 2D maze with continuous state and action spaces. As the agent does not perceive the walls, maximizing success coverage in these environments is harder than it seems.

4.4 Results & Analysis

4.4.1 Success coverage evaluations

Figure 4.4 shows that SVGG significantly outperforms all baselines in terms of success coverage. This is especially the case in highly discontinuous goal

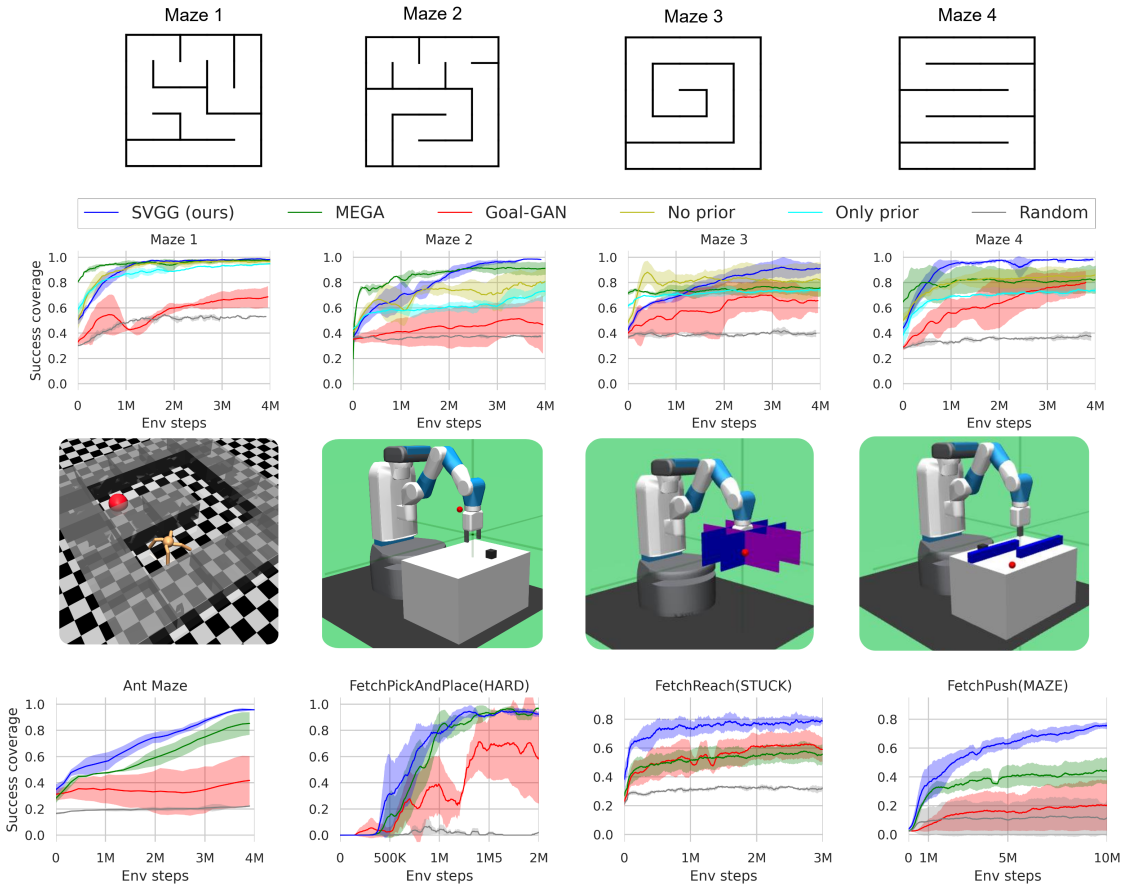

Figure 4.4: Evolution of success coverage over 4 different PointMazes, AntMaze, FetchPickAndPlace (Hard), FetchReach (Stuck) and FetchPush (Maze) (4 seeds each). SVGG outperforms MEGA and GoalGan as well as ablations.

space settings such as in mazes and the modified version on FetchReach and FetchPush, where it efficiently discovers and achieves most of the valid goals. On AntMaze and FetchPickAndPlace, where the goal space is smoother, SVGG obtains comparable results to its competitors.

Due to the known stability issues in GAN training, GoalGan is the least efficient baseline. Another explanation of the failure of GoalGan is that it is likely to generate invalid goals, which is not the case for MEGA or SVGG. MEGA chooses behavior goals from a Replay Buffer of achieved goals, while the validity distribution p_{valid} considered in SVGG keeps particles inside valid areas.

The minimum density heuristic of MEGA efficiently discovers all valid goals in the environment, but our results show that its success plateaus in almost all the considered environments. MEGA's intrinsic motivation only relies on

state novelty. Thus, when the agent has discovered all reachable states, it is unable to target areas that it has reached in the past but that it has not mastered yet.

4.4.2 Visualization of goals

We visualize the evolution of behavioral and achieved goals in Maze 1 (Figure 4.5) and Maze 2 (Figure 4.6), in parallel with the success coverage after training.

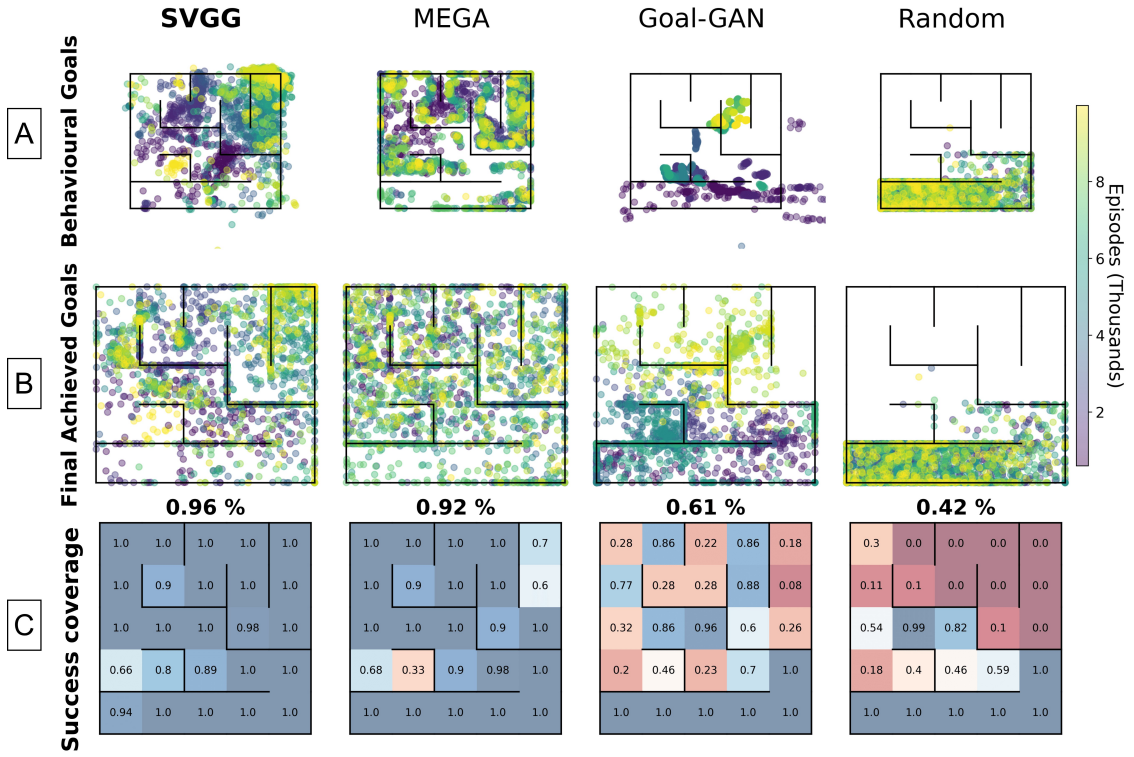

Figure 4.5: Visualization of (A) behavioral goals, (B) achieved goals and (C) success coverage for 1M steps in Maze 1.

The main advantages of our method lie in the capacity to target difficult areas and avoid catastrophic forgetting, which results in nearly optimal success coverage. We observe that MEGA efficiently discovers the environment but fails to master the corresponding goals. This also leads to catastrophic forgetting and a lack of adaptation when the environment changes.

Finally, one can see that the generation of GOIDs in GoalGan is very unstable and tricky in such discontinuous goal spaces, especially as the generator is susceptible to output goals outside the environment boundary, which SVGG avoids with the constraint from the validity distribution.

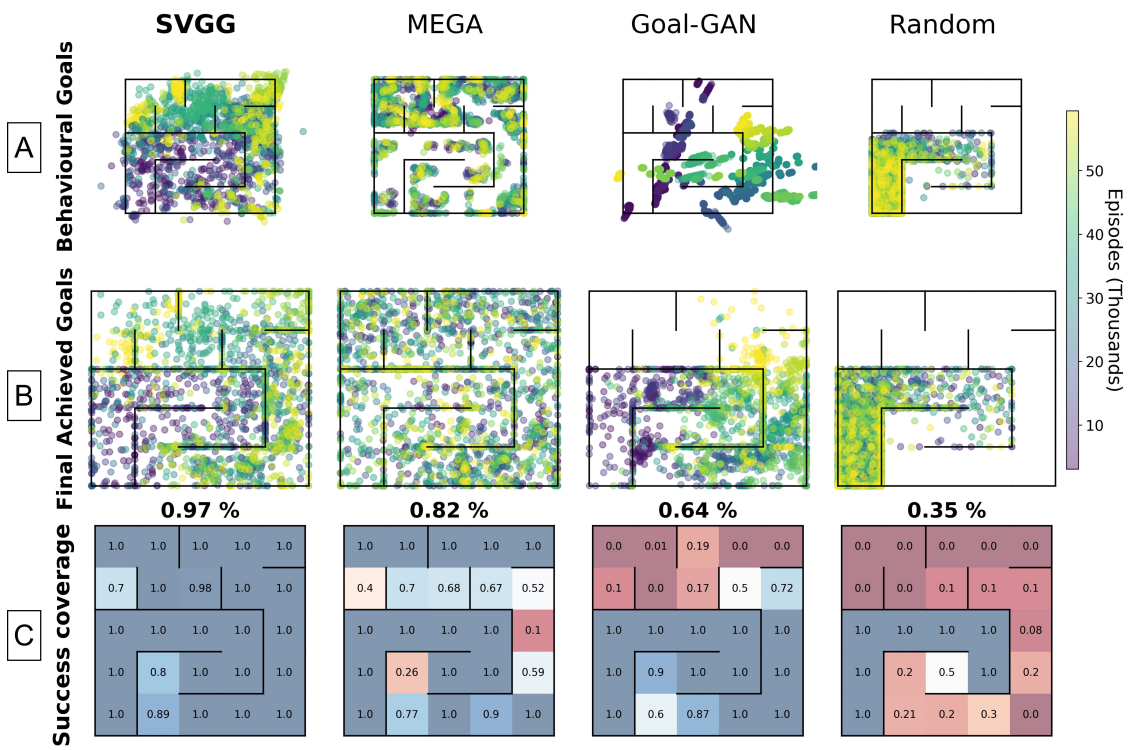

Figure 4.6: Visualization of: (A) behavioral goals, (B) achieved goals and (C) success coverage for 4M steps in Maze 2. SVGG is the only one to achieve nearly perfect success coverage.

4.4.3 Recovery Property

Figure 4.7 shows the advantages of SVGG over MEGA and GoalGan in changing environments. Walls are suddenly added during the training process (dot black line from Figure 4.7), after the methods have reached their peak performance. We see that the performance of MEGA and GoalGan plateaus lower than their peak performance (see Figure 4.4) whereas SVGG discovers new difficult goals resulting from the environment modification and focuses on these goals to finally recover its peak success coverage.

We also observe that the advantages of our method over MEGA in terms of recovery ability are more significant when changes in the environment are more drastic (i.e., when starting from maze B).

4.4.4 Evolution of the particles

To gain further intuition on how SVGG maximizes the success coverage, we show in Figure 4.8 the evolution of the particles throughout training. As the agent progresses and achieves novel and harder goals, the D_ϕ model updates

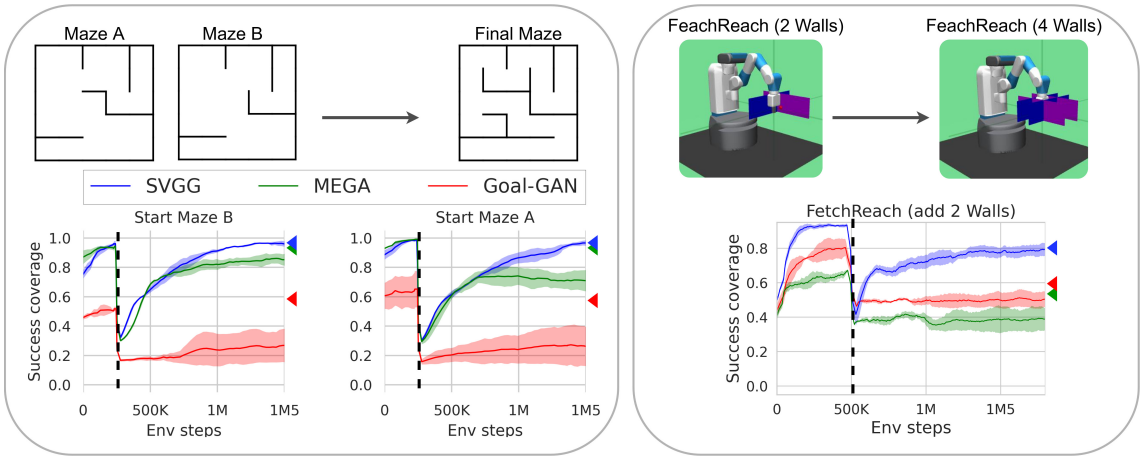

Figure 4.7: Evolution of success coverage in a changing environment for MEGA, GOALGAN and SVGG (4 seeds each). We add walls to the starting mazes A and B (left) and go from 2 to 4 walls in FetchReach (right). The triangles correspond to the peak success coverage of the methods on the final environments (which correspond to Maze 1 and FetchReach (Stuck) Figure 4.4) during regular training.

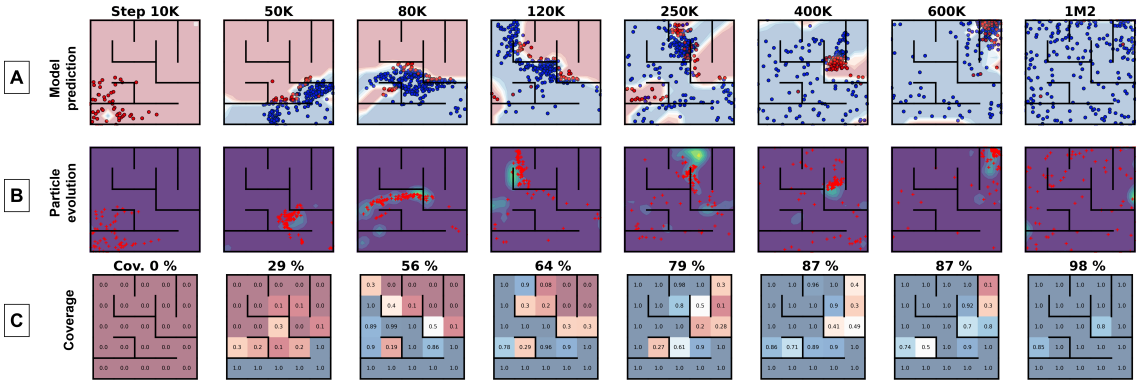

Figure 4.8: Parallel visualization of the model's prediction (A), the particles evolution (B) and of the coverage (C) throughout training in Maze 1.

its predictions. Thus, the target distribution p_{goals} is updated accordingly (background color of the 2^{nd} row of the figure). The particles $q = \{g_i\}_{i=1}^{n}$ are then moved towards new areas of intermediate difficulty through SVGD to minimize $KL(q||p_{\text{goals}})$.

Figure 4.8 also highlights the recovery property of SVGG. When the agent has nothing else to learn in the environment, p_{goals} reduces to p_{valid}, that is at this point uniform over the entire valid goal space. Therefore, the particles

spread uniformly over the goal space and prevent SVGG from catastrophic forgetting, as the model rediscovers areas that the agent has forgotten how to reach (cf. rightmost column in Figure 4.8).

4.4.5 Control of the sampling difficulty

Using beta distributions makes it possible to tune the goal difficulty an agent should aim at to efficiently explore and control an environment. Indeed, by tuning the α and β hyper-parameters of the distribution, one gets a different incentive for goal difficulty. We choose to compare the efficiency of 5 pairs of α and β hyper-parameters, as illustrated in Figure 4.3 p. 72.

In Figure 4.9, we compare the 5 versions of SVGG using different beta distributions from Figure 4.3, on the 4 previously considered mazes. One can observe that extreme targets difficulties are the least effective ones, especially the *Very easy*, which is too conservative to efficiently explore new areas of the space. On the other hand, SVGG performs very well with *Medium* and *Hard* distributions. This suggests that the optimal goal difficulty is somewhere between medium and hard. Performing a more advanced optimization over α and β is left for future work.

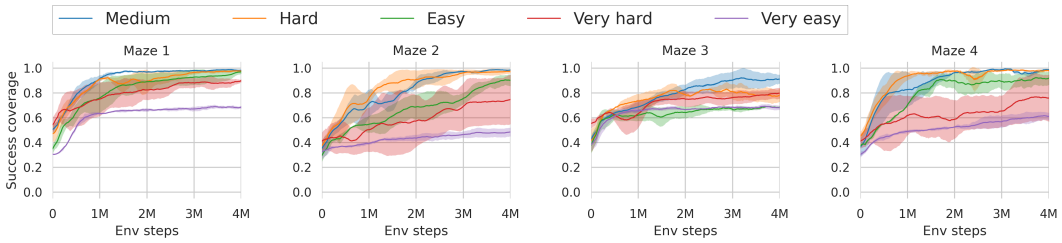

Figure 4.9: Evolution of success coverage for 5 target difficulties as intrinsic motivation for SVGG, on 4 mazes and 4 seeds each.

4.4.6 Target goal distribution ablations

We conduct an ablation which consists in keeping SVGD, but replacing our target goal distribution (corresponding to goals whose prediction of success is close to 0.5) with the target goal distributions of related work such as of SKEW-FIT, MEGA or GOALGAN (resp. the uniform distribution over the support of achieved goals, the region of low density of achieved goals and goals of intermediate difficulty (GOIDs)). For all criteria, the goal target distribution is represented as a set of particles and optimized with SVGD. We compare the average success coverage over our 4 mazes in Figure 4.10 where

we can see that our choice of target distribution is the most effective. We describe below the compared distribution of goals.

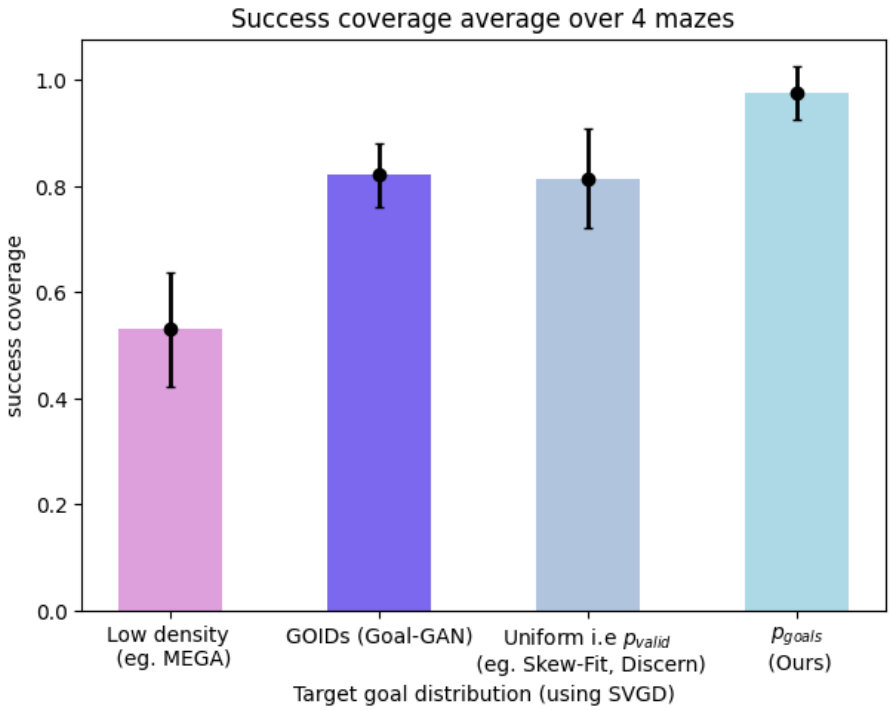

Figure 4.10: Plot of the average success coverage of SVGG over the 4 mazes (3 seeds each for a total of 12 runs) after 4 millions training steps where we replace our target goal distribution with other choices and keep SVGD as sampling method. The error bar shows the standard deviation between all runs.

Uniform distribution over the support of achieved goals (as in SKEW–FIT and DISCERN): This corresponds to our "Only validity" baseline in Figure 4.4. Indeed, the probability for a goal to be valid is learned with a One Class SVM which exactly strives to uniformly cover the distribution of achieved goals. As the results presented in Figure 4.10 show, this baseline performs relatively well, but is unable to target areas where the agent struggles. Therefore, it is not as efficient as our target distribution p_{goals}.

Distribution derived from the region of low density of achieved goals (as in MEGA): We can obtain a distribution of goals very close to the one of MEGA by combining our "Only Validity" ablation with a beta distribution tuned to address the lowest probability region, i.e. taking $p_{goals} \propto f(\alpha, \beta, V_\psi(g)))$, with f a Beta distribution and α and β set to target low den-

sity of V. Due to the absence of a mechanism to avoid sampling unfeasible goals, the particles are attracted to low density areas that mostly correspond to non-valid goals, which makes this baseline very inefficient.

Distribution of GOIDs (as in GoalGan): Our target distribution p_{goals} is very close to the GOID in GoalGan, the main difference is that our Beta distribution is smoother than the indicator function of GoalGan, that labels a goal as of "intermediate difficulty" when the probability of reaching it is between 0.2 and 0.8. So, to move closer to the GoalGan criterion, we replaced the beta-distribution used in SVGG with the crisp distribution (with a generalized Gaussian distribution of skewing parameter $\beta = 6$) which outputs 1 for probabilities between 0.3 and 0.7 and 0 otherwise (which are the parameters that give the best results). Note that while this distribution is differentiable, the gradient is less informative than the SVGG version. As a consequence, approximating this distribution with SVGD is less efficient and gets a lower success coverage.

Ablations of the sampling method To highlight the interest of the SVGD choice to sample goals from our target distribution p_{goals}, we conduct additional experiments where we swap SVGD with some other sampling tools : MCMC (Metropolis Hastings), GANs and direct sampling from the replay buffer.

In Figure 4.11, we can see that SVGD is by far the most efficient way to sample from p_{goals} and thus to maximize the success coverage. We describe below the tested sampling tools.

GANs: We use the same procedure as GoalGan by replacing their GOID criterion by our target distribution p_{goals}. The results are very similar to GoalGan, this can be explained by the proximity of our criterion in terms of intermediate difficulty, except from the fact that we add p_{valid}. We can also conclude that GANs are not the best choice for moving target distributions due to their training instability. We use the same hyperparameters as in the GoalGan baseline.

Buffer sampling: We sample a batch of previously achieved goals in the buffer, and then compute a categorical distribution based on the p_{goals} criterion and sample candidate goals. This method is the least effective baseline, which was expected as the exploration component was provided by SVGD, the goals from the replay buffer are not sufficiently diverse to explore the environment if not combined with a diversity criterion.

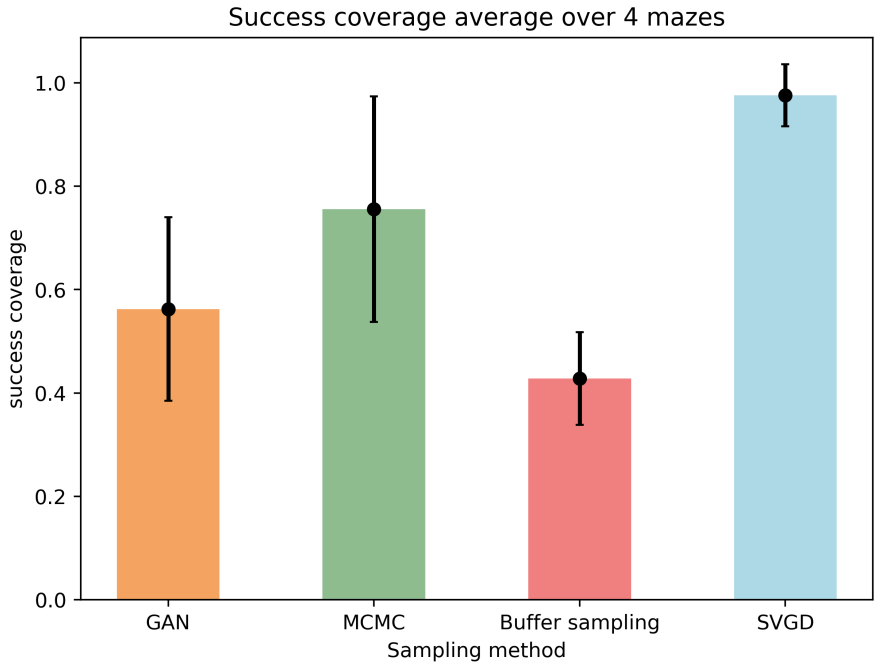

Figure 4.11: Plot of the average success coverage of SVGG over the 4 mazes (3 seeds each for a total of 12 runs) after 4 millions training steps, where we replace SVGD with other sampling methods. The error bar shows the standard deviation between all runs.

MCMC (Metropolis Hastings): At each p_{goals} update, we construct a Markov chain with normal movement (with $\mu = 0$ and $\sigma = 0.5$) then we use classic likelihood ratio rejection sampling to draw goals samples that are provided to the agent. Metropolis-Hastings is a good candidate but is still far from the SVGD performance. It presents some good sampling accuracy at times, but is very slow to converge to the true distribution. Thus, the goal sampling is not always aligned with the current agent skills.

4.5 Conclusion

In this chapter, we introduced SVGG, a method for sampling goals of optimal difficulty leveraging SVGD, and a model of the agent's current abilities. We showed that our method efficiently manages the exploration vs exploitation dilemma in goal sampling, emphasizing diversity or difficulty depending on the detection of goals of optimal difficulty, which are goals belonging to the

zone of proximal development of the agent. We have shown that this process enables to maximize the success coverage. Furthermore, it enables us to detect and adapt new configurations due to external factors such as environment changes.

A key limitation of this contribution is that we assume the access to the internal state of the environment, which is low dimensional and semantically meaningful. This has enabled us to extensively focus on intrinsic goal sampling without having to simultaneously learn representations.

We address this limitation in the next chapter, which is the second contribution of this work, tackling online representation learning for image-based GCRL.

Chapter 5

Online Representation Learning for Image-based GCRL

With Goal-Conditioned Reinforcement Learning, an agent can learn a diverse set of behaviors in complex environments in the absence of a predefined reward function. However, with visual inputs, the semantic information contained in observations is not explicit and observations are high-dimensional, which makes generating goals for exploration or reward calculation significantly harder. A classical approach to mitigate this issue is to leverage a more compact latent representation of the observation space that captures the semantics of the observation space and is of lower dimension. But how can we get the transformation from the observation space to the latent space? A first approach is to learn an accurate mapping from the observation space to the latent space in advance, from a given dataset in an offline setting. With this approach, the question is how do we collect such a dataset? If the dataset covers the whole environment, that presupposes that we know the environment in advance, which contradicts the assumption that the agent has to explore it. In the alternative case, which corresponds to the online setting, the agent will collect the dataset while exploring the environment. In that case, the obtained representation progressively shifts in time together with the distribution of data coming sequentially from the policy while the agent explores. This distribution – or distributional – shift is a well-known issue in machine learning, it happens each time an agent is trained from a given distribution of data but is subject to another distribution of data when deployed. The first consequence of progressive distributional shift in our context is that the latent space evolves through time, making policy learning harder. The second consequence is that a poor latent state representation may prevent the

agent from successfully discovering the full observation space, as illustrated in Figure 3.13, page 64.

In this work, since we are driven by our motivation to build autonomous agents, we cannot rely on pre-trained representation space which presupposes a lot of prior knowledge injection.

Thus, in this chapter, we face the issues of online representation learning. We consider the case where an agent has to learn meaningful representations from images observation in an online fashion, while exploring and learning to control the environment.

The main difficulty we face is the progressive distribution shift, resulting in an evolving and potentially biased learned representation. To overcome this difficulty, we call upon a tool designed to counter the effects of distributional shift: the *Distributionally Robust Optimization* (DRO) framework (Delage and Ye, 2010). By combining it with a classical β-VAE (Higgins et al., 2017a), we are able to skew the data coming from the policy in order to learn an unbiased representation for Intrinsically Motivated Goal-Conditioned agents. This results in steadier learning, better exploration and finally better performance from the agent.

This chapter is organized as follows. First, we describe the DRO framework. We give a technical presentation which focuses on its parametric modeling with likelihood ratios, which we mobilize for our contribution. Thereafter, we introduce our method: Distributionally Robust Optimization for Image-based Exploration (DROIDE). Finally, we evaluate DROIDE experimentally against various baseline agents with representation learning capabilities, and discuss the results.

Contents

5.1 The Distributionally Robust Optimization framework

The distribution or distributional shift issue is central in machine learning. It refers to the fact that an agent may be trained from a given distribution of data but may be subject to another distribution of data when deployed. This is particularly important in real-world applications, where the training data might not perfectly represent the conditions encountered during deployment.

Traditional machine learning models are trained to minimize the average expected loss over the training data, assuming that the future data will follow the same distribution.

Distributionally Robust Optimization (DRO) (Delage and Ye, 2010; Ben-Tal et al., 2013; Duchi et al., 2021) is a machine learning and optimization framework that aims to learn models that perform well under distributional shift. Indeed, DRO aims to minimize the worst-case expected loss over a set of plausible distributions around the training data distribution, which makes the model more robust to variations and shifts in the data distribution.

Given a family of predictive models $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by $\theta \in \Theta$, a loss function $\ell : \mathcal{Y}^2 \to \mathbb{R}$, and an uncertainty set $\mathcal{Q}$ of data distributions, the general DRO problem can be formulated as the problem of finding:

$$\min_{\theta \in \Theta} \left\{ \mathcal{R}(\theta) := \max_{q \in \mathcal{Q}} \mathbb{E}_{(x,y)\sim q} \left[\ell(f_\theta(x), y) \right] \right\}. \tag{5.1}$$

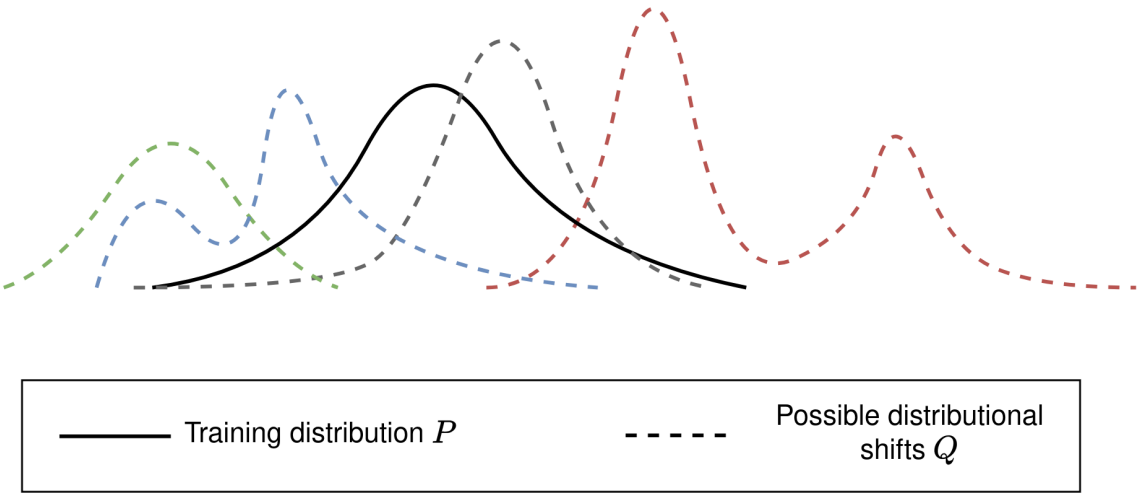

Figure 5.1: Distributional shifts around the training data distribution.

The uncertainty set $\mathcal{Q}$ represents the perturbations that might affect the

data at test time, and can therefore take several forms. While we expect $\mathcal{Q}$ to contain the distribution of test data, leaving too much freedom to q may lead to trivial solutions that degenerate as uniform classifiers (Martinez et al., 2021). To avoid this, the uncertainty set $\mathcal{Q}$ is commonly defined as a ball centered on p using distribution distances or similarities.

In the absence of a predefined uncertainty set $\mathcal{Q}$, DRO methods strive to define such an uncertainty set relying on heuristics. This has been the subject of many research papers, see Rahimian and Mehrotra (2019) for a broad and comprehensive review of these approaches.

Examples include restricting $\mathcal{Q}$ considering KL-divergence (Michel et al., 2022), Chi-square divergence (Hashimoto et al., 2018), maximal Total Variation distance (Wang et al., 2020), Wasserstein distance (Wang et al., 2021) or Jaccard index (Ferry et al., 2022). From the fairness without demographics literature (Duchi et al., 2023), it is known that the maximal allowed divergence is connected to the risk of the smallest component of the training distribution, seen as a mixture of distributions.

In the next section, we describe DRO optimization using parametric likelihood ratios in a standard supervised classification setting, following the implementation of Michel et al. (2022). We then apply this specific formulation to our problem, as it is easy to implement and tune, and also more robust than non-parametric counterparts.

5.2 Distributionally Robust Models with Parametric Likelihood Ratios

A major challenge with formulation (5.1) is that exploring all possible distributions in $\mathcal{Q}$ is infeasible in the general sense. Worse, modeling distribution q directly over the whole feature space as support is very difficult, and usually highly inefficient, even for $\mathcal{Q}$ restricted to distributions close to p. This motivates an adversarial alternative, which relies on importance weighting of training samples from p.

Reformulation of the DRO problem: We therefore restrict $\mathcal{Q}$ to the set of distributions that are absolutely continuous with respect to p[1], inspired by Michel et al. (2022). This allows us to write $q = rp$, with $r : \mathcal{X} \times \mathcal{S} \to \mathbb{R}^+$ a function that acts as a weighting factor. The expectation in Equation (5.1)

[1]In the situation where all distributions in Q are absolutely continuous with respect to p, for all measurable subset $A \subset X \times Y$ and all $q \in \mathcal{Q}$, $q(A) > 0$ only if $p(A) > 0$.

can thus be rewritten, for $q \in \mathcal{Q}$, as:

$$\mathbb{E}_{(x,y)\sim q}\ \ell(f_\theta(x), y) = \mathbb{E}_{(x,y)\sim p}\ \frac{q(x,y)}{p(x,y)}\ell(f_\theta(x), y) = \mathbb{E}_{(x,y)\sim p}\ r(x,y)\ell(f_\theta(x), y), \tag{5.2}$$

which allows algorithms to rely on data sampled from p while anticipating shifts in $\mathcal{Q}$. This is convenient as sampling data points from worst-case distributions in $\mathcal{Q}$ can prove to be particularly challenging. Given the uncertainty set $\mathcal{R} = \{r \,|\, rp \in \mathcal{Q}\}$, the optimization problem thus becomes:

$$\min_{\theta\in\Theta} \max_{r\in\mathcal{R}} \mathbb{E}_{(x,y)\sim p}\ r(x,y)\ell(f_\theta(x), y). \tag{5.3}$$

First, we can note that $r \in \mathcal{R}$ implies that r must respect the following validity constraint: $\mathbb{E}_p r(x,y) = 1$, to ensure that the implicit corresponding q function keeps a valid integration property for a distribution (i.e., $\int_{\mathcal{X},\mathcal{Y}} q(x,y)dxdy = 1$). Additionally, the shape of $\mathcal{Q}$ needs to be controlled. In the following, this is done by ensuring a KL constraint on q regarding p, i.e. with $\mathcal{Q} = \{q|KL(q|p) \leq \epsilon\}$ for a given predefined $\epsilon > 0$. Doing this, the problem in (5.3) is equivalent to:

$$\begin{aligned} \min_{\theta} \max_{r}\ & \mathbb{E}_{(x,y)\sim p}\ r(x,y)\ell(f_\theta(x), y) \\ st\quad & \mathbb{E}_{(x,y)\sim p} r(x,y)\log r(x,y) \leq \epsilon \\ & \mathbb{E}_{(x,y)\sim p} r(x,y) = 1 \end{aligned} \tag{5.4}$$

with the KL constraint rewritten using: $KL(q||p) = KL(pr||p) = \mathbb{E}_{pr}\log\frac{pr}{p} = \mathbb{E}_p r\log r$. Relaxing the KL constraint by the introduction of an hyper-parameter λ, the above problem is reduced to:

$$\min_{\theta}\ \max_{r:\mathbb{E}_p r=1}\ \mathbb{E}_{(x,y)\sim p} r(x,y) l(f_\theta(x), y) - \lambda\mathbb{E}_{(x,y)\sim p} r(x,y)\log r(x,y). \tag{5.5}$$

Given a large enough training dataset $\Gamma = \{(x_i, y_i)\}_{i=1}^N$ sampled from p, the optimization problem can be approximated as (using the law of large numbers):

$$\begin{aligned} \min_{\theta}\max_{r}\ & \frac{1}{N}\sum_{i=1}^N r(x_i,y_i) l(f_\theta(x_i), y_i)) - \lambda\frac{1}{N}\sum_{i=1}^N r(x_i,y_i)\log r(x_i,y_i) \\ & st\quad \frac{1}{N}\sum_{i=1}^N r(x_i,y_i) = 1. \end{aligned} \tag{5.6}$$

From this formulation, we can see that the risk associated to a shift of test distribution can be mitigated by simply associating adversarial weights $r_i := r(x_i, y_i)$ to every sample (x_i, y_i) from the training dataset, respecting $\bar{r} := \frac{1}{N}\sum_{i=1}^N r_i = 1$. This said, r can be viewed as proportional to a categorical distribution defined on the components of the training set.

Analytical solution: Given any function $h : \mathcal{X} \to \mathbb{R}$, we know that the distribution q that maximizes $\mathbb{E}_q[h(x)] + \lambda\mathcal{H}_q$, with $\mathcal{H}_q$ the Shannon entropy of q, is the maximum entropy distribution $q(x) \propto e^{h(x)/\lambda}$. Thus, we can easily deduce that the inner maximization problem of (5.6) owns an analytical solution in $r_i = N\frac{e^{l(f_\theta(x_i),y_i))/\lambda}}{\sum_{j=1}^{N} e^{l(f_\theta(x_j),y_j))/\lambda}}$ (proof in Appendix Section B.1.1, page 153). As so, the spread of $\mathcal{Q}$ is controlled with a temperature weight λ, which can be seen as the weight of a Shannon entropy regularizer defined on discrepancies of q regarding p. In the light of this closed-form solution, we observe that setting λ close to zero comes down to putting extreme attention to samples associated with highest values of the classification loss. On the other hand, higher values for λ favor distributions q that evenly spread over the whole dataset, hence converging towards a classical classification model trained on p.

Solution based on likelihood ratios: While appealing, the use of this analytical solution for r may induce an unstable optimization process in DRO, as weights may vary abruptly for even very slight variations of the classifier outputs. Moreover, it implies individual weights, only interlinked via the outputs from the classifier, while one could prefer smoother weight allocation regarding inputs. This is particularly true for online processes like our RL setting, with new training samples periodically introduced in the learning buffer.

Following Michel et al. (2022, 2021), we rather focus in our contribution in the next section on likelihood ratios defined as functions $r_\psi(x, y)$ parameterized by a neural network f_ψ, where we set:

$$r_\psi(x_i, y_i) = n\frac{\exp^{f_\psi(x_i,y_i)}}{\sum_{j=1}^{n}\exp^{f_\psi(x_j,y_j)}} \quad ,\forall \text{ mini-batch } \{(x_j, y_j)\}_{j=1}^{n}, \tag{5.7}$$

where f_ψ is periodically trained on mini-batches of n samples from the training set, using fixed current θ parameters, according to the unconstrained inner maximization problem of (5.6) for a given number of gradient steps. This parameterization enforces the validity constraint at the batch-level, through batch normalization hard-coded in the formulation of r_ψ. While not fully respecting the full validity constraint from (5.6) in the case of small batches, this proved to perform well for commonly used batch sizes on many classification benchmarks (Michel et al., 2022). Classifiers obtained through the alternated min-max optimization of (5.6) proved to be more robust to distribution shifts than their classical counterparts. Using shallow or regularized networks f_ψ is advised, as strong Lipschitzness of $r(x, y)$ allows to similarly treat similar

samples in the input space, which guarantees better generalization and stability of the learning process. These generalization and stability properties lack to non-parametric versions of DRO, such as a version using the analytical solution for inner-maximization presented above, which could be viewed as the optimal r_ψ based on an infinite-capacity neural network f_ψ. In the next section, we build on this framework to set a representation learning process for RL, that encourages the agent to explore.

5.3 DROIDE: Distributionally Robust Optimization for Image-based Exploration

In our RL setting for hard exploration problems, our aim is to define a representation learning approach that evenly embeds all observed states in a latent space, where the agent can set useful goals and learn corresponding effective policies. We build on RIG-like approaches (presented in Section 3.3.4), based on Variational Auto-encoders, but an important drawback of such approaches is that VAE is designed to model the distribution of training samples $p(x)$, which induces low generation probabilities for states rarely observed in our exploration process. This is problematic when leveraging the learned latent space and the associated state sampling distribution to sample new goals for the agent. Thus, we introduce a new representation learning approach for RL exploration problems, named DROIDE for "DRO for Image-Driven Exploration", where we employ DRO principles to overcome this limitation. The general framework of our architecture is illustrated in Figure 5.2, DRO is used in the "Representation learning" part of the figure, in order to improve the data sampling to train the VAE.

The DRO framework is traditionally used in supervised learning tasks, as presented in the previous section, where there is a well-known loss function to optimize against worst-case distributions. In this context, the goal is to optimize for a particular performance metric under distributional uncertainty. Applied to VAE learning, the training distribution reduces to $p(x)$ instead of $p(x, y)$. Our objective is to give more weight in the representation to samples that are poorly reconstructed, corresponding to rare samples that have not been fairly used during training. Doing so, we force the VAE to expand its representation span of the environment, as illustrated in Figure 5.3.

Additionally, when training VAEs with data coming from the policy in an online fashion, the distribution to model p does not possess clear boundaries and evolves through time together with the agent's progress, which makes it

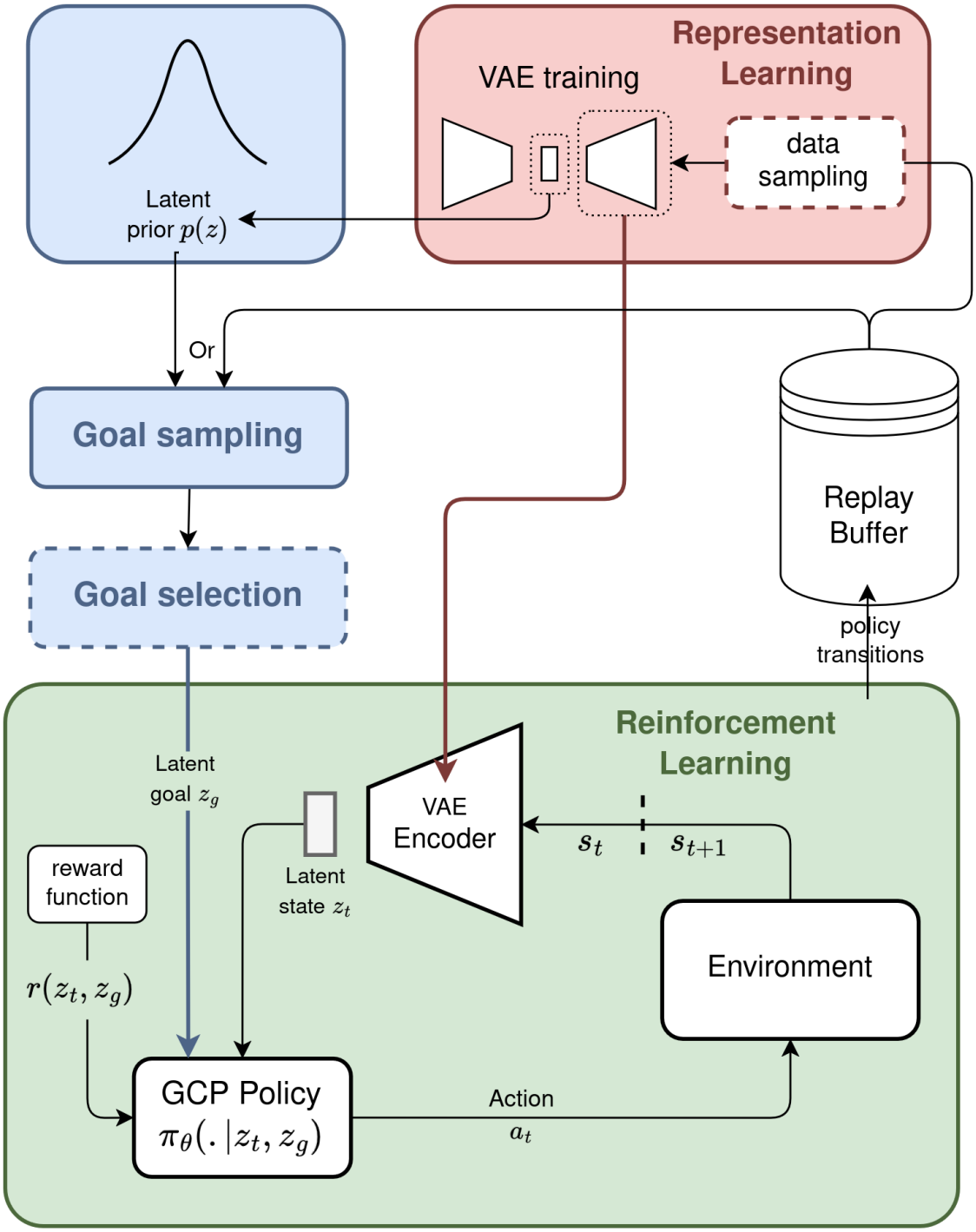

Figure 5.2: General framework of online VAE representation learning in RL. **In green:** RL loop using the VAE encoder to convert high-dimensional states s_t to latent states z_t. **In blue:** latent goal z_g sampling (from prior distribution or replay buffer) and selection. **In red:** Representation Learning with VAE, using data from the replay buffer.

very interesting to model the plausible future distributional shift around p, resulting in better generalization of the agent in new areas of the environment. To do this, we propose to build the weighting distribution of DRO on an approximation of the generative distribution encoded by the VAE architecture, in order to focus attention of encoding-decoding learning on states associated with low probability to be sampled from the model. Please note that, to the best of our knowledge, DRO for VAE has never been considered in the literature yet, for a simple reason: in classical generative modeling tasks, VAE is intended to model $p(x)$, not a broader distribution encompassing it. Applying DRO to VAEs comes down to rather striving to model its support, which is not desirable in classical learning settings, contrary to our dynamic online representation problem which faces exploration issues. Thus, we have to adapt DRO to that type of distributional learning, which we develop in the following.

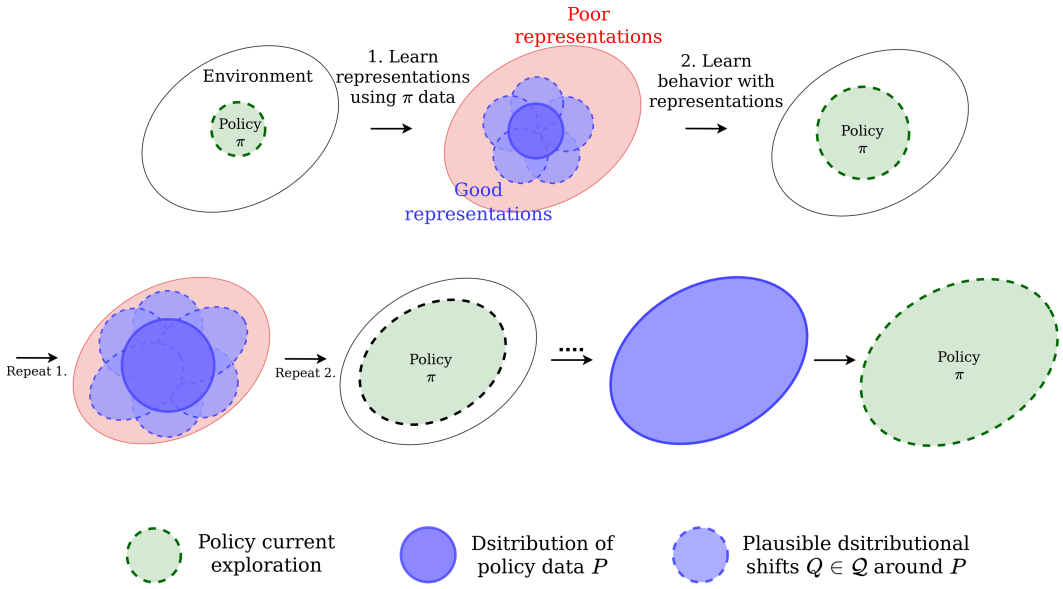

Figure 5.3: Illustration of exploration and representation learning with DROIDE: being robust to future distributional shifts in the policy data facilitates learning accurate representations of states and goals of the environment. From better representations, the policy is then able to explore further, thus overcoming exploration bottlenecks.

Classic VAE learning aims at minimizing the negative log-likelihood:

$$\mathcal{L} = -\mathbb{E}_{x\sim p(x)} \log p_{\theta,\phi}(x),$$

with $p_{\theta,\phi(x)}$ the predictive posterior, that is the probability of generating x via

our model. This predictive posterior $p_{\theta,\phi(x)}$ can be written:

$$p_{\theta,\phi}(x) = \int p(z)p_\theta(x|z)dz,$$

where $p(z)$ is a prior over latent encodings of the data x, commonly taken as $\mathcal{N}(0, I)$, and $p_\theta(x|z)$ is the likelihood of x knowing z and the parameters of the decoding model θ. Given that such a marginalization can be subject to very high variance, the idea is to use an encoding distribution $q_\phi(z|x)$ to estimate this generation probability. For any distribution q_ϕ, such that $q_\phi(z|x) > 0$ for any z such that $p(z) > 0$, we have:

$$p_{\theta,\phi}(x) = \mathbb{E}_{z\sim q_\phi(z|x)} \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)}.$$

In our instance of the DRO framework, we propose to consider the following optimization problem, similar to its classical implementation for classification tasks presented in the previous section:

$$\min_{\theta,\phi} \max_{\xi\in\Xi} -\mathbb{E}_{x\sim\xi(x)} \log p_{\theta,\phi}(x), \tag{5.8}$$

where Ξ is the set of adversarial distributions of our DROIDE approach. As in the previous section, we introduce a weighting function $r : \mathcal{X} \to \mathbb{R}^+$ which aims at modeling $\frac{\xi}{p}$ for a given adversarial distribution $\xi \in \Xi$, and respects both validity (i.e., $\mathbb{E}_p r(x) = 1$) and shape constraints (i.e., $KL(\xi||p) \leq \epsilon$, for a given pre-defined $\epsilon > 0$). Relaxing the KL constraint by the introduction of a λ hyper-parameter, we get the following objective for our DRO-VAE problem:

$$\min_{\theta,\phi} \max_{r:\mathbb{E}_p r=1} -\mathbb{E}_{x\sim p}\ r(x) \log p_{\theta,\phi}(x) - \lambda \mathbb{E}_{x\sim p}\ r(x) \log r(x). \tag{5.9}$$

However, as $\log p_{\theta,\phi}(x)$ is intractable directly, we consider a slightly different objective:

$$\begin{aligned} &\min_{\theta,\phi} -\mathbb{E}_{x\sim p}\ r^*(x) \log p_{\theta,\phi}(x), \\ \text{with} \quad r^* = \arg &\max_{r:\mathbb{E}_p r=1} -\mathbb{E}_{x\sim p}\ r(x)\tilde{\mathcal{L}}_{\theta,\phi}(x) - \lambda \mathbb{E}_{x\sim p}\ r(x) \log r(x), \end{aligned} \tag{5.10}$$

where the only difference is that we consider an approximation $\tilde{\mathcal{L}}_{\theta,\phi}(x) \approx \log p_{\theta,\phi}(x)$ obtained via a Monte-Carlo Importance sampling as: $\tilde{\mathcal{L}}_{\theta,\phi}(x) = \log \sum_{j=1}^{M} \exp(\log p_\theta(x|z^j) + \log p_\theta(z^j) - \log q_\phi(z^j|x)) - \log(M)$ given M samples z^j from $q_\phi(z^j|x)$ for any x, which can be computed accurately (without loss of low log values) using the LogSumExp trick.

This formulation suggests an alternating learning algorithm, which alternates updating the weighting function r and optimizing the VAE.

At each VAE step, the encoder-decoder networks are optimized considering a weighted version of the classical ELBO. Denoting as r the weighting function adapted for current VAE parameters via (5.10), we have:

$$\begin{aligned}
\mathbb{E}_{x\sim p(x)} r(x) \log p_{\theta,\phi}(x) &\geq \mathbb{E}_{x\sim p(x)} r(x) \mathbb{E}_{z\sim q_\phi(z|x)} \log \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} \\
&\approx \sum_{i=1}^{n} \frac{r(x_i)}{n} \mathbb{E}_{z_i\sim q_\phi(z_i|x_i)} \log \frac{p(z_i)p_\theta(x_i|z_i)}{q_\phi(z_i|x_i)} \\
&\approx \sum_{i=1}^{n} \frac{r(x_i)}{n} \left(\frac{1}{m} \sum_{j=1}^{m} \log p_\theta(x_i|z_i^j) - KL(q_\phi(z|x_i)||p(z)) \right) \\
&\triangleq \mathcal{L}^{\text{VAE-DRO}}_{\theta,\phi,r}(\{x_i\}_{i=1}^n),
\end{aligned} \tag{5.11}$$

where the lower-bound $\mathcal{L}^{\text{VAE-DRO}}_{\theta,\phi,r}(\{x_i\}_{i=1}^n)$ is estimated at each step via Monte-Carlo based on mini-batches of n data points $(x_i)_{i=1}^n$ from the training buffer and m latent codes $(z_i^j)_{j=1}^m$ for each data point x_i. Optimization is performed using the reparameterization trick, where each latent code z_i^j is obtained from a deterministic transformation of white noise $\epsilon_i^j \sim \mathcal{N}(0, I)$. Given μ_i and σ_i the Gaussian parameters provided by $q_\phi(.|x_i)$, the latent code is obtained via: $z_i^j = \epsilon_i^j * \sigma_i + \mu_i$.

As discussed in the previous section for classical DRO formulations, the maximization of r in Equation (5.10) owns an analytical solution, with $r_i \propto e^{-\tilde{\mathcal{L}}_{\theta,\phi}(x_i)/\lambda}$. We show in Appendix Section B.1.2 that using this closed-form solution comes down to the Skew-Fit way of skewing the VAE, where VAE training samples are resampled with replacement based on their p_{skewed} distribution, introduced in Section 3.3.4. As mentioned in the previous section, the DRO literature for supervised classification reported that such an analytical way for setting the weighting function is subject to high instability. We claim that such instability is supposed to be amplified in our online RL setting, where the sampling distribution p depends on the behavior of a constantly evolving RL agent. Our droide method thus considers the parametric version of the weighting function, as defined in Equation (5.7), trained periodically for a given number of gradient steps on the inner maximization problem of (5.10):

$$r_\psi(x_i) = n \frac{\exp^{f_\psi(x_i)}}{\sum_{j=1}^{n} \exp^{f_\psi(x_j)}} \tag{5.12}$$

In our experiments we use for f_ψ a similar CNN architecture as the encoder of the VAE, but with a greatly smaller learning rate for stability (as it induces a regularizing lag behind the encoder, and hence enforces a desirable smooth weighting w.r.t. the input space). The full pseudo-code of our approach is given in Algorithm 4.

Algorithm 4 *Distributionally Robust Exploration*

Input: a GCP π_θ, a VAE: encoder $q_\phi(z|x)$, decoder $p_\theta(x|z)$, latent prior $p(z)$, DRO Neural Weighter r_ψ, buffers of transitions $\mathcal{B}$, reached states $\mathcal{R}$, train size N, batch-size n, number of Monte Carlo samples m, temperature λ, number of optimization steps $nsteps$, frequence of VAE optimization $freqOpt$.
while not stop **do**
  ▷ *Data Collection (during $freqOptim$ steps):* Perform rollouts of $\pi_\theta(.|z_t, z_g)$ in the latent space, conditioned on goals sampled from prior $z_g \sim p(z)$ or the buffer (with possible resampling depending on the goal selection strategy), and latent state $z_t = q_\phi(x_t)$, with x_t a pixel observation;
  Store transitions in $\mathcal{B}$, visited states in $\mathcal{R}$;

  ▷ *Learning Representations with VAE*
  for $nsteps$ epochs **do**
    Sample a train set of N states Γ from $\mathcal{R}$
    for every mini-batch $\{x_i\}_{i=1}^n$ from Γ **do**
      ▷ *DRO Weighter Update*
        Update weighter by one step of Adam optimizer, for the maximization problem from (5.10) with temperature λ.
    end for
    for every mini-batch $\{x_i\}_{i=1}^n$ from Γ **do**
      ▷ *Weighted VAE Update*
        Update encoder q_ϕ and decoder p_θ by one step of Adam optimizer on $\mathcal{L}^{\text{VAE-DRO}}_{\theta,\phi,r_\psi}(\{x_i\}_{i=1}^n)$, as defined in (5.11), with m sampling noises $(\epsilon_i^j)_{j=1}^m$ for each x_i.
    end for
  end for

  ▷ *Agent Improvement*
    Improve agent with any Off-Policy RL algorithm (e.g., DDPG) using transitions from $\mathcal{B}$;
end while

5.4 Experiments

Our experiments seek to highlight the differences between DROIDE and several baselines in regard to the two main components of the considered training procedure, which are representation learning through VAE modeling of the

pixel input, and the intrinsic motivation component that defines the goal sampling process.

In the first step of our experiments, we leave aside the intrinsic motivation component and consider sampling from the learned prior of the VAE (i.e., $z_g \sim \mathcal{N}(0, I)$), a straightforward choice adopted in RIG, which consists in sampling goals according to the distribution of states encoded by the VAE, i.e states that have been encountered by the agent during training. We compare different representation learning strategy, first our DRO-based learning method that we described in 5.3, that we name DROIDE, to RIG, which consists in regular VAE-training from the agent distribution of states, and SKEW-FIT, which attempts to skew the distribution of achieved states by importance sampling to foster exploration.

In a second step, we investigate the intrinsic motivation component with different goal selection criteria from IMGC methods described in Chapter 2, added on top of DROIDE for the representation learning aspect.

In a nutshell, we investigate the following questions:

- **Representation Learning strategy**: How to overcome exploration bottlenecks and to learn good representations over the whole environment?
- **Latent goal sampling strategy**: Can additional goal selection criteria be beneficial to maximize the success coverage, or is sampling from the learned prior enough?

Throughout this section, we evaluate the different algorithms over the 4 continuous point maze environments described in Chapter 4, page 76 as they are difficult to explore given each maze's specific topology. We turn 2D xy observations and goals into a pixel top-down view of the maze with a red dot highlighting the position of the agent. Every image is of size 82x82.

5.4.1 Representation Learning strategy

We compare three different representation learning strategies, each corresponding to a specific training distribution for the VAE:

• RIG uses the regular distribution of states x visited by the policy: $p_{visited} \propto p_{\pi_\theta}(x)$,

• SKEW-FIT weights this distribution by the inverse probability of a state being reached, forming the training distribution $p_{skewed} \propto p_{\pi_\theta}(x)^\alpha$, which we discussed earlier in Section 3.3.4 with Equation (3.18), page 65.

• Finally, our DROIDE strategy, weights training samples for the VAE by the likelihood ratio 5.12 $r_\psi(x)$: $p_{dro} \propto r_\psi(x) p_{\pi_\theta}(x)$.

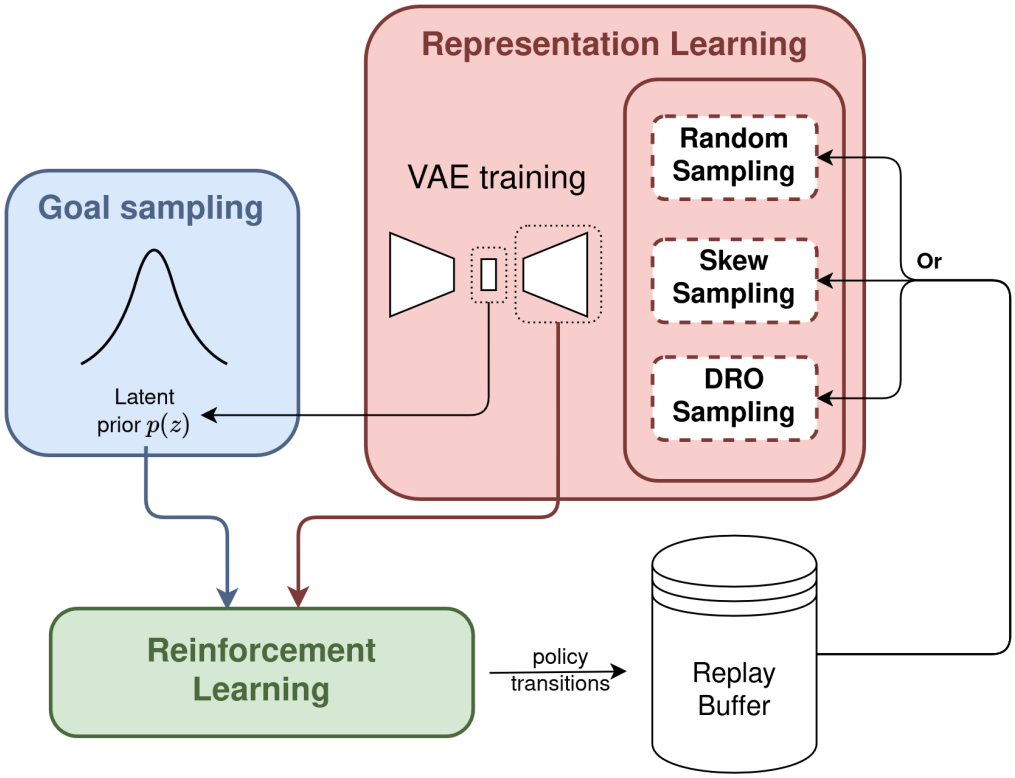

Figure 5.4: Representation Learning strategy experiments.

Methods	Representation Learning distribution	Goal distribution
RIG Nair et al. (2018) SKEW-FIT Pong et al. (2019) DROIDE (ours)	$p_{visited} \propto p_{\pi_\theta}(x)$ $p_{skewed} \propto p_{\pi_\theta}(x)^\alpha$ $p_{dro} \propto r_\psi(x) p_{\pi_\theta}(x)$	$p_{\text{prior}}(z_g) = \mathcal{N}(z_g \vert 0, I)$

Table 5.1: Comparison of Representation Learning methods in GCRL.

Success Coverage evaluation

Results in Figure 5.5 show the success coverage evolution over 4 millions steps. We see that our method significantly outperforms RIG and SKEW-FIT. These results corroborate the observations we made in Section 3.3.4, about how online representation learning with RIG is unable to overcome an exploration bottleneck. Therefore, a RIG agent is only able to explore and control a very small part of the environment. The results show that the success coverage of RIG is systematically capped to a certain value. SKEW-FIT is often able to overcome the exploration bottleneck but suffers from high instability, which indeed corresponds to the main drawback of non-parametric DRO, highlighted in Michel et al. (2021). Therefore, SKEW-FIT is not able to reliably maximize

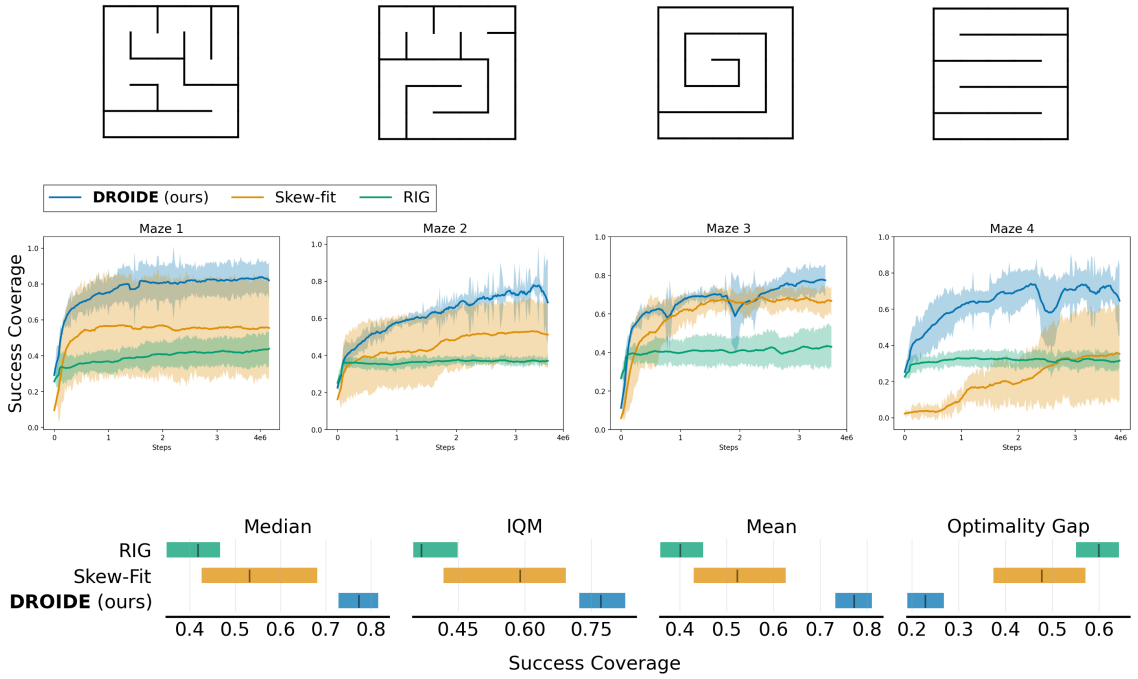

Figure 5.5: Success coverage over 4 different PointMazes (6 seeds each) for 4 millions steps (shaded areas correspond to standard deviation), from top-down view **pixel observations** of size (82x82). Real goal and state coordinates xy have only been used to evaluate the success coverage. **Third row:**, Median, Interquartile Median, Mean and Optimality Gap of the success coverage metric across the 4 mazes after 4 millions training steps. We compute/plot these metrics and confidence intervals using the Rliable library (Agarwal et al., 2021).

the success coverage. On the other hand, DROIDE is more stable due to the use of parametric likelihood ratios, and then is able to explore and control the environment to maximize the success coverage.

Learned Latent Representations visualization

Figure 5.6 presents our methodology to study latent representation learning. We uniformly sample data points in the maze and process them iteratively from 2D points to pixels, then from pixels to the latent code of the VAE. By using the same color for the source data points and the latent code, this process allows us to visualize the 2D latent representation of the VAE over the environment (Figure 5.6 left and middle). In addition, to get a sense of what part of the environment is encoded in the latent prior, we sample latent codes from $p(z)$ and plot the 2D coordinates of the decoded observations

using $p_\theta(x|z)$, which corresponds to the red dots. The distribution in blue corresponds to a KDE estimation fitted on the red dots.

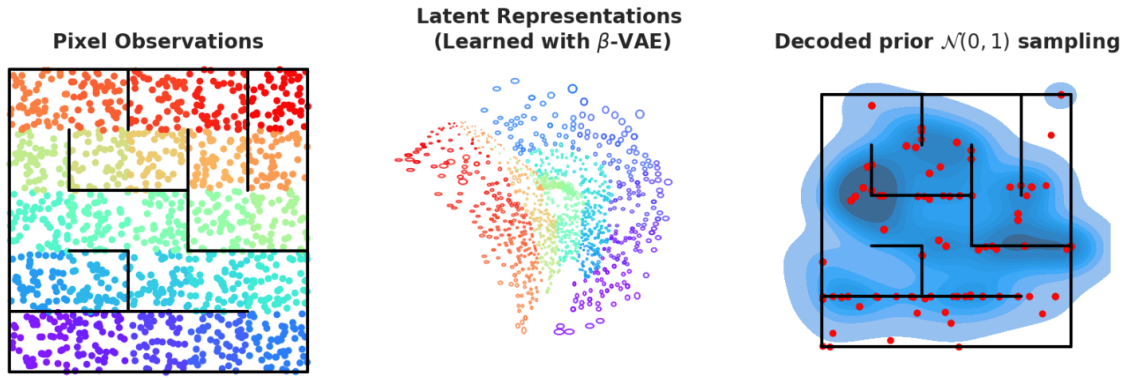

Figure 5.6: Learned Representation of DROIDE after 1 million training steps in Maze 0. **Left**: every colored dot corresponds to the pixel observation x of its specific xy coordinates. **Middle**: Every pixel observation x on the left is processed by the VAE encoder to get the learned latent posterior distribution $q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma_\phi(x))$. Colored ellipsoids correspond to these 2-dimensional Gaussian distributions. **Right:** we sample latent goals from the latent prior $z \sim p(z) = \mathcal{N}(0, I)$ and we decode the corresponding pixel observations $p_\theta(x|z)$ (red dots correspond to the xy coordinates of the pixel observations).

In order to gain a deeper insight into the performance of RIG, SKEW-FIT and DROIDE, we show in Figure 5.7 the parallel evolution of the prior sampling $z \sim \mathcal{N}(0, I)$ and the corresponding learned representations.

One can clearly see that RIG is stuck in an exploration bottleneck (which in this case corresponds to the first U-turn of the maze): the VAE is unable to learn meaningful representations of poorly explored areas (red part of the maze in Figure 5.7). As a consequence, the prior distribution $p(z)$ only encodes a small subspace of the environment. On the other hand, SKEW-FIT and DROIDE manage to escape these bottlenecks and incorporate an organized representation of nearly every area of the environment, with the difference that DROIDE is more stable and therefore reliably learns well organized representations.

In order to quantify the evolution of latent representations to highlight the differences in terms of latent distribution dynamics between RIG, SKEW-FIT and DROIDE, we introduce the following measurement:

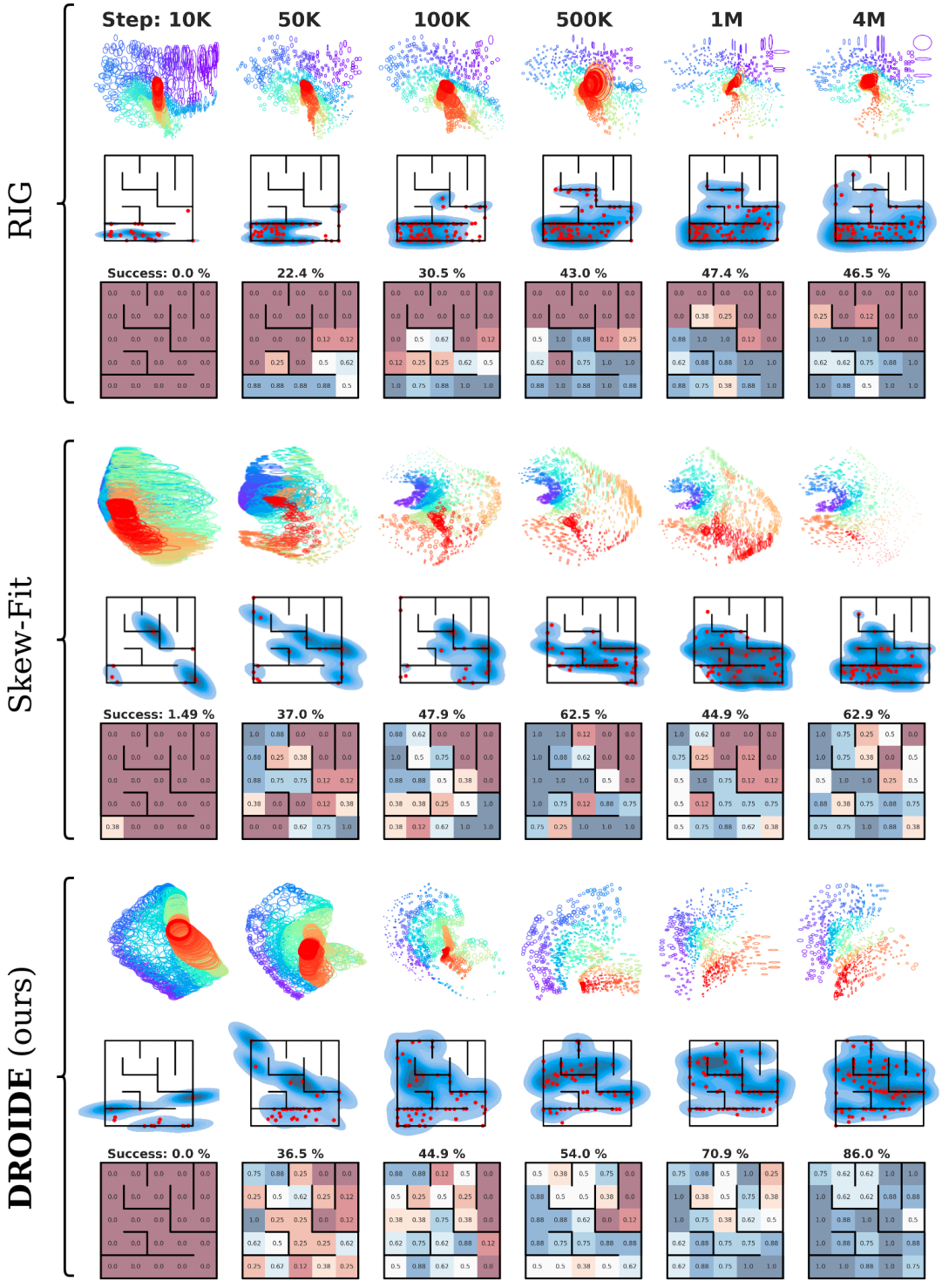

Figure 5.7: **First row** of each method: evolution of learned representations. **Second row** of each method: evolution of the intrinsic goal distribution when sampling from the latent prior $p(z) = N(0, 1)$. **Third row** of each method: success coverage evolution. (See Figure 5.6 for details on how we obtain these plots).

$$\forall t = 1...T, \quad d_t(\mathbf{x}) \triangleq \frac{1}{N} \sum_{i=1}^{N} \parallel \mu_{\phi^t}(x_i) - \mu_{\phi^{t-1}}(x_i) \parallel, \tag{5.13}$$

where $\mathbf{x} = \{x_i\}_{i=1}^{N}$ is a batch of pixel observations uniformly sampled from the environment state space using prior knowledge (only for evaluation purposes). With this metric, we measure the embedding evolution of every point x_i, using the movement of the expectation $\mu_\phi(x_i)$ from the latent posterior distribution $q_\phi(z|x_i) = \mathcal{N}(z|\mu_\phi(x_i), \sigma_\phi(x_i))$, throughout the updates of VAE parameters ϕ.

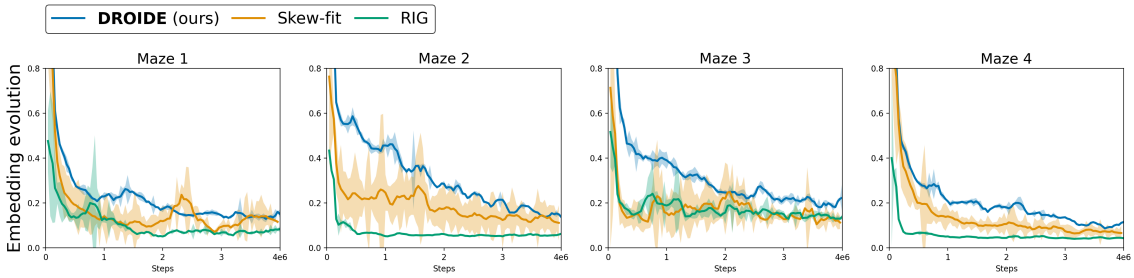

Figure 5.8: Embedding evolution over 4 different PointMazes (6 seeds each) for 4M steps (shaded areas correspond to standard deviation). Every point corresponds to the shift of representation between step t and step $t+1$ of VAE training: $\frac{1}{N}\sum_{i=1}^{N} \parallel \mu_{\phi^{t+1}}(x_i) - \mu_{\phi^t}(x_i) \parallel$. For every pixel observation x_i and timestep t, we have $q_{\phi^t}(z|x_i) = \mathcal{N}(z|\mu_{\phi^t}(x_i), \sigma_{\phi^t}(x_i))$. We compute representation shifts between t and $t+1$ every 40.000 training steps.

Figure 5.8 shows that the embedding movement d of DROIDE is higher and less variable across seeds, which indicates that the learned representations evolve more consistently. Meanwhile, the VAE training process of SKEW-FIT is prone to variability, and the embedding evolution in RIG is close to null after a certain number of training steps.

5.4.2 Latent Goal selection strategy

In the second phase of our experiments, in order to study the impact of goal selection on the maximization of success coverage, we investigate several goal selection criteria from the literature, on top of DROIDE to learn the representation. In the latter experiences, displayed in Figure 5.5, we sample intrinsic goals from the latent prior $\mathcal{N}(0, I)$ in order to give a fair comparison of the different representation learning schemes used in RIG, SKEW-FIT and DROIDE. Here, we fix the representation learning method to be DROIDE,

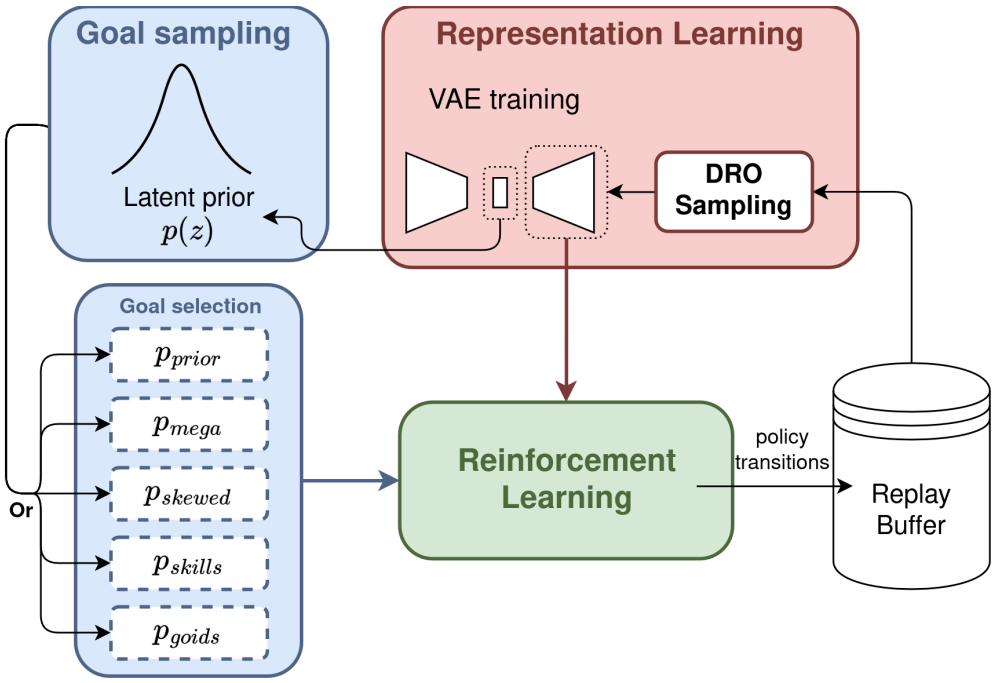

Figure 5.9: Latent Goal selection strategy experiments.

proven to be the most effective, and compare different goal selection criterion as follows: first, we sample a set of candidate goals from the latent prior $\mathcal{N}(0, I)$, then we resample, among such pre-sampled candidates, the final intrinsic goals given the following criteria:

- **Control of goals Difficulty Distribution** from Stein Variational Goal Generation (SVGG) (Castanet et al., 2022):

$$p_{\text{skills}}(z_g) \propto \exp\left(f_{\alpha,\beta}(D(z_g)\right),$$

  where D is a success prediction model trained on previous experiences, and $f_{\alpha,\beta}$ is a beta distribution controlling the target goal difficulty.

- **Goals of Intermediate Difficulty selection** from GoalGan (Florensa et al., 2018):

$$\text{GOIDs} = \{z_g \sim p_{\text{prior}}(z_g) | P_{min} < D(z_g) < P_{max}\},$$

$$p_{\text{goid}}(z_g) = \mathcal{U}(\text{GOIDs})$$

  where $D(z_g)$ is also a success prediction model, and P_{min}, P_{max} two hyperparameters between 0 and 1. p_{goid} uniformly samples goals in the set GOIDs.

- **Minimum density Goal selection** from MEGA (Pitis et al., 2020):

$$p_{\text{mega}}(z_g) = \delta \underset{z_g \sim p_{\text{prior}}(z_g)}{\text{argmin}} p_{\pi_\theta}(z_g),$$

  p_{mega} is a Dirac distribution centered at the minimum density goal according to p_{π_θ}.

- **Diversity distribution** from SKEW-FIT, also similar to DISCERN (Warde-Farley et al., 2018):

$$p_{\text{skewed}}(z_g) \propto p_{\pi_\theta}(z_g)^\alpha,$$

  with p_{π_θ} the distribution of achieved goals by the current policy, and α the skewing parameter between 0 and 1.

Methods	Representation Learning distribution	Goal distribution
DROIDE + MEGA	$p_{dro} \propto r_\psi(x) p_{\pi_\theta}(x)$	$p_{\text{mega}}(z_g)$
DROIDE + SKEW-FIT		$p_{\text{skewed}}(z_g)$
DROIDE + GOALGANS		$p_{\text{goid}}(z_g)$
DROIDE + SVGG		$p_{\text{skills}}(z_g)$
DROIDE		$p_{\text{prior}}(z_g)$

Table 5.2: Comparison of Goal Sampling methods in GCRL.

Interestingly, we observe in Figure 5.10 that the addition of a goal selection criterion on top of DROIDE does not improve performance. Actually, it is quite the opposite, especially for curriculum criteria based on the goal reaching difficulty like GOALGAN and SVGG, that significantly lower the final success coverage of DROIDE. We believe that the biased goal selection criterion might interfere with the representation learning process, additional experiments to exhibit this phenomenon or to find adequate goal selection criterion are necessary.

5.5 Conclusion & Perspectives

In this second contribution, we introduced DROIDE, an algorithm leveraging Distributional Robust Exploration, to learn representation from pixel observations in online RL. We showed that by taking advantage of the DRO framework, we are able to overcome exploration bottlenecks in environments with discontinuous goal spaces, setting us apart from previously existing methods like RIG and SKEW-FIT.

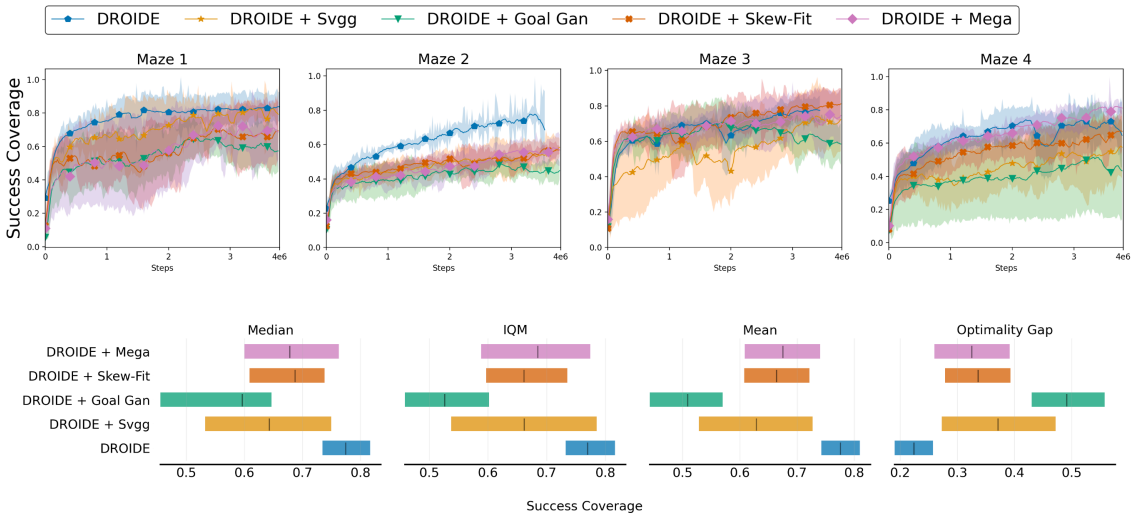

Figure 5.10: **Fist row:** Success coverage over 4 different PointMazes (6 seeds per run) for 4 millions steps (shaded areas correspond to standard deviation), from top-down view **pixel observations** of size (82x82). **On the second row**, we plot the Median, Interquartile Median, Mean and Optimality Gap of the success coverage metric across the 4 mazes after 4 millions training steps. We compute/plot these metrics and confidence intervals using the library Rliable (Agarwal et al., 2021).

Beyond our work, some further investigation on the reward function definition in the latent space are needed. As the standard sparse reward function for goal-conditioned RL works fairly well, we left this point out of the scope of this contribution. In the next paragraph, we deliver some insight on possible future work on this specific issue.

5.5.1 Perspective on learned prior for VAE

Traditionally in VAE, we fix the prior distribution $p(z)$ to be a normal distribution, for the sake of simplicity and training stability. However, this natural choice leads to undesirable behavior regarding latent prior sampling, known as "VAE holes", suggesting that some high density areas of the prior might not correspond to any plausible data sample (Chen et al., 2018; Tomczak and Welling, 2018; Davidson et al., 2018; Rezende and Viola, 2018).

In our context, we might sample goals in areas of the latent space that do not correspond to any feasible goals embedding in the environment, which could significantly slow down or even prevent policy training.

The "VAE holes" phenomena can be highlighted by looking closer at the

VAE objective, which is to maximize the ELBO over a training data distribution $p(x)$:

$$\mathcal{L}^{\text{ELBO}} = \mathbb{E}_{p(x)}\Big[\mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x) \parallel p(z))\Big] \tag{5.14}$$

The development of the regularization term involving the latent prior distribution $p(z)$ gives us:

$$\begin{aligned}
Reg &:= -\mathbb{E}_{p(x)}\Big[D_{KL}(q_\phi(z|x) \parallel p(z))\Big] \\
&= \mathbb{E}_{p(x)}\Big[\mathbb{E}_{q_\phi(z|x)}\big[\log p(z) - \log q_\phi(z|x)\big]\Big] \\
&= \int_x p(x) \int_z q_\phi(z|x)\big[\log p(z) - \log q_\phi(z|x)\big]\,\mathrm{d}z\,\mathrm{d}x \\
&= \int_x \int_z \sum_{n=1}^{N} \delta(x - x_n) q_\phi(z|x)\big[\log p(z) - \log q_\phi(z|x)\big]\,\mathrm{d}z\,\mathrm{d}x \\
&= \int_z \sum_{n=1}^{N} q_\phi(z|x_n)\big[\log p(z) - \log q_\phi(z|x_n)\big]\,\mathrm{d}z \\
&= \int_z q_\phi(z) \log p(z)\,\mathrm{d}z - \int_z \sum_{n=1}^{N} q_\phi(z|x_n) \log q_\phi(z|x_n)\,\mathrm{d}z \\
&= -H\big[q_\phi(z), p(z)\big] + H\big[q_\phi(z|x)\big].
\end{aligned} \tag{5.15}$$

From line 3 to 4 we used the Dirac formulation of the empirical distribution: $p(x) = \sum_{n=1}^{N} \delta(x - x_n)$ and from line 4 to 5 we used the following Dirac property: $\int \delta(x - y) f(x)\,\mathrm{d}x = f(y)$. Interestingly, the regularization term of the ELBO loss can be written as the difference between a cross entropy term $H\big[q_\phi(z), p(z)\big]$ and the entropy $H\big[q_\phi(z|x)\big]$. Maximization of the regularization term implies the minimization of the cross entropy term, which means that $q_\phi(z)$ should match the prior $p(z)$. The term $q_\phi(z)$ is denoted as the *aggregated posterior* (Makhzani et al., 2015; Hoffman and Johnson, 2016). It is defined as follows:

$$q_\phi(z) := \sum_{n=1}^{N} q_\phi(z|x_n), \tag{5.16}$$

which correspond to the mixture of variational posteriors over the training data distribution. In practice, it is very difficult for the aggregated posterior to adequately match a fixed shape prior $p(z)$, resulting in "holes" in the prior distribution, corresponding to regions of the latent space where the aggregated posterior assigns low probability while the prior assigns relatively high probability, as illustrated in Figure 5.11.

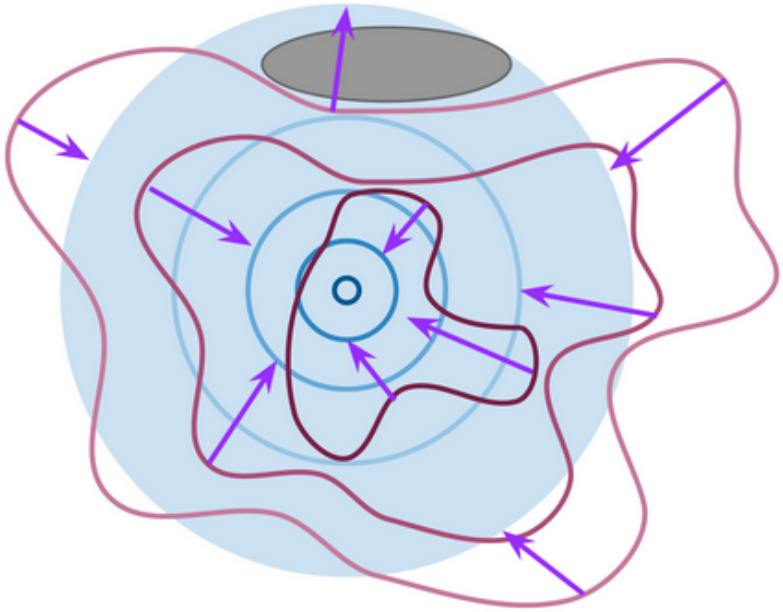

Figure 5.11: Illustration of the "hole problem" in VAE prior: cross-entropy optimization with a non-learnable prior normal prior. The aggregated posterior (purple contours) tries to fit the non-learnable prior (in blue). The purple arrows indicate the optimization process of the aggregated posterior. An example of a hole is presented as a dark gray ellipse, sampling in that zone from the prior would result in an out of distribution data sample. Image source: Tomczak (2021).

This observation motivates the consideration of flexible learnable prior distributions, where both the aggregated posterior and the prior try to match each other. A trivial solution is to use a mixture of Gaussian priors with learnable means and standard deviation:

$$p_\gamma(z) = \sum_{i=1}^{K} w_k \mathcal{N}(z|\mu_k, \sigma_k^2), \tag{5.17}$$

where $\gamma = \{(w_k, \mu_k, \sigma_k)\}_{k=1}^K$. More sophisticated formulations of learnable priors include the VampPrior (Tomczak and Welling, 2018), Implicit Optimal Priors (Takahashi et al., 2019), Hierarchical priors (Klushyn et al., 2019), Flow-based priors (Gatopoulos and Tomczak, 2021), Riemannian priors (Arvanitidis et al., 2021) and energy-based priors (Yu et al., 2023).

In online representation learning for goals in GCRL, working with flexible priors could have the potential to provide more flexible exploration. As the agents discover new areas, the VAE optimization process sometimes struggles

to incorporate new data to its latent space. When a new area of the environment is discovered, a learnable prior such as a mixture of Gaussian would have the ability to form new clusters in the latent space for the new embeddings, without having to adjust the representation of former discovered states.

Chapter 6

Conclusion

I will use the first person for this short paragraph ahead of the conclusion, as I would like to express personal feelings and thoughts on the work I have conducted with my coworkers, without involving them in statements that they might not share.

A PhD thesis is a long journey. Just like any research project, it has been full of twists and turns, unexpected ideas, findings, but also dead ends and difficulties. All these ups and downs along the way have shaped this work as it is today.

This journey has taught me the fundamental human aspect of the process of making science, made of intuitions, discussions, encounters, shared perspectives and personal takes. This process is certainly not only made of indisputable facts and laws, as the old inductivist perspective on science persists in the public eye. Nevertheless, I hold science in high esteem, and I must say that its intrinsic human nature makes it all the more beautiful, making us fight against our fundamental irrational nature, full of bias and limited perspective, hoping to get closer to the very brittle notion of "truth".

Given the chaotic nature of this process, it is a dreaded task to find a comprehensive meaning of a three years long work. However, I believe it to be a necessary effort to get a deeper sense of what we are actually doing. For this reason, throughout this manuscript, I have done my best to reproduce the thought process that led us to our contributions, taking our readers to where we arrived in a comprehensive and meaningful manner.

It is now time to close this manuscript. In this final chapter, we start by highlighting the main contributions of our work. Then we outline a few of the limitations of the studies we performed. Finally, we present some perspectives that our research has made possible. We conclude the manuscript with a more personal perspective on the work we have accomplished.

Contents

6.1 Summary of contributions

We argued in this Thesis, that in absence of prior knowledge on the goals to reach in an environment, the general objective for Intrinsically motivated Goal-Conditioned agents should be a combination of exploration and control, which we propose to measure with the success coverage.

In our first contribution, we proposed Stein Variational Goal Generation (SVGG), a method for intrinsic goal generation taking into account both goal difficulty and novelty, in order to tackle both control and exploration components of the success coverage, where we noticed that most methods from the literature were merely focused on either one of the two aspects of the objective. To this end, we built on the unique properties of the variational inference algorithm, Stein Variational Gradient Descent (SVGD) (Liu et al., 2017) to approximate our intrinsic goal distribution, that enables us to obtain the "recovery property", which guarantees that the agent is able to adapt to unexpected changes in the environment topology. We believe it is a key feature that an autonomous agent should exhibit in unsupervised settings.

SVGG showed significant improvements on challenging tasks over concurrent methods. However, a key limitation of this first contribution was the assumption that the agent had access to an accurate state representation.

Therefore, in our second contribution, we expanded the scope of Intrinsically motivated Goal-Conditioned agents to incorporate representation learning for pixel observations. Noticing that the combination of learning representations and behavior leads to an exploration bottleneck, we extended the Distributionally Robust Optimization (DRO) framework from traditional supervised settings to unsupervised representation learning with Variational Auto Encoders (VAE). We showed that by learning representations robust to future distributional shifts of the policy, our method is able to efficiently explore mazes with a complex topology, unlike concurrent approaches.

6.2 Limitations & Perspectives

In this section, we address some limitations and perspectives on our work. To begin with, we compare experimental results between Chapters 4 and 5, describing interferences between intrinsic motivation and representation learning. Then, we propose an intrinsic motivation criterion to tackle goal-conditioned stochastic environments. Finally, we give a perspective on reward formulation from a manifold point of view on RL.

6.2.1 Interferences between intrinsic motivation and representation learning

In our contributions in Chapters 4 and 5, we aimed to maximize the overall agent's capacity to reach goals in challenging environments, through intrinsic motivations. Even if we would have liked to give a comprehensive response on how to design the intrinsic motivation process in a general framework, it appears that each setting adopted in Chapters 4 and 5 faces issues that should be address in a specific manner.

In Chapter 4, we used a predefined environment representation, and extensively focused on the intrinsic motivation, i.e the goal selection component. Our experiments in Section 4.3 demonstrate that, in order to maximize the success coverage in this setting, it is necessary to include in the goal selection process a difficulty criterion as in SVGG, see Section 4.2 page 70.

However, our experiments about the latent goal sampling strategy in Section 5.4.2 suggest otherwise. Indeed, sampling from the latent prior or using a pure exploration strategy as in Skew-Fit or MEGA seems to be far more efficient than methods involving a goal selection based on difficulty like SVGG or GoalGan. Some additional experiments with different goal selection criteria might be necessary to confirm this trend. However, we present hereafter an hypothesis on why criteria based on difficulty might not be adequate when mixed with representation learning.

A difficulty-based criterion tends to attract the agents toward narrow areas of the environment where it struggles, keeping it there until it masters the corresponding region, for as long as it takes. We believe this feature prevents learning proper representations, as those narrow areas monopolize the latent codes in the VAE prior. As a result, the environment encoding needs significant reorganization as soon as new areas are discovered, which either slows down or prevents adequate learning.

On the other hand, diversity-based criteria grasp as many areas as possible without bothering to master the corresponding goals. While this trait comes with drawbacks that we addressed in Chapter 4, it has the advantage of embedding a broader part of the environment in the latent representation at an early stage of training.

Now, how can we maximize the success coverage just like in SVGG, while learning representations online using DROIDE? This amounts to merging diversity with difficulty criteria, which can be obtained by leveraging our measure $d_t(\mathbf{x})$ (5.13) of the latent space evolution to put more emphasis either on exploration or on exploitation. For example, above some threshold of the value of $d_t(\mathbf{x})$, the latent space is still evolving fast, indicating that new areas are still getting discovered, which makes diversity criteria interesting to fully

explore the rest of the environment. On the other hand, under some threshold of the value of $d_t(\mathbf{x})$, the latent space has reached a certain stability, indicating that the agent has fully explored the environment. In this latter context, a difficulty criterion would efficiently target areas on which the agent still struggles.

6.2.2 Stochastic environments

Our work in this Thesis focuses on classic navigation and manipulation environments that are usually deterministic, which means that the transition function T is also deterministic: $T(s_t, a_t) = s_{t+1}$. Our methods could be further extended to tackle stochastic settings where $T(s_t, a_t) = \mathbb{P}(s_{t+1}|a_t = a, s_t = s)$.

This case is particularly challenging for intrinsic motivation methods, as many formulations of intrinsic goals or rewards rely on predictions of the future, which is complicated in non-deterministic settings. The extreme case is known as the "Noisy TV" problem, if the agent receives images as state inputs a television screen displaying white noise, every state will be novel, and it would be impossible to predict the value of any pixel in the future. This case has been addressed in intrinsically rewarded agents by learning representations that only contain what can be controlled by the agent (for example in Pathak et al. (2019)), a class of methods we described earlier in Section 3.2, page 44.

In our goal-conditioned framework, we could add a stochastic component to the environment with a random variable controlling the presence or absence of a wall in a maze. This would be akin to a door being open or close, which further determines the trajectory that must be adopted to reach a specific goal, as illustrated in Figure 6.1.

This setting requires that intrinsic motivation methods must be designed in order to distinguish between aleatoric and epistemic uncertainty, i.e distinguish between goals reached randomly because of the aleatoric component inherent to the environment, and goals reached randomly due to the agent's lack of skills. Intrinsic motivation methods should focus on the latter.

The goal selection process in SVGG 4.2 would be attracted to uncontrollable goals, as the particles in SVGG are attracted to areas where the model of the agent's skills is most uncertain: $D_\phi(g) \approx \frac{1}{2}$, regardless of the source of this uncertainty.

In order to capture the epistemic uncertainty of the environment, we could adapt the deep ensemble uncertainty prediction (Lakshminarayanan et al., 2017) to GCRL, using a set of k RL agents parametrized by θ_k, each paired with a success model $D^k_\phi(g)$. The variance between agents of the predicted

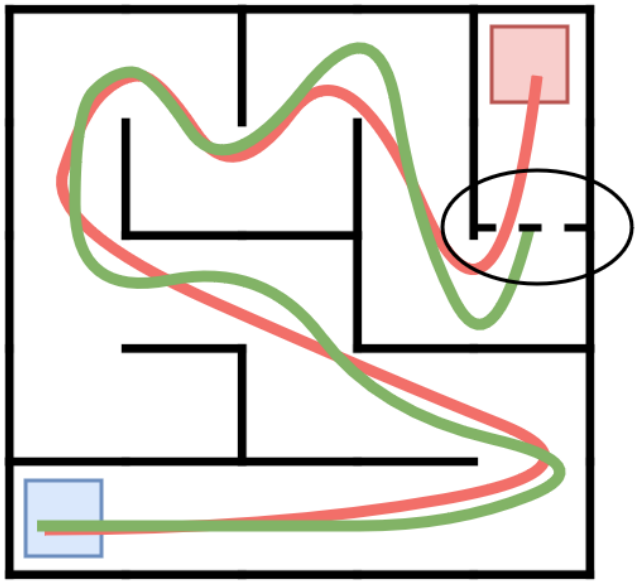

Figure 6.1: Example of stochastic maze, where the dashed line corresponds to a wall whose presence is random, which means that reaching the goal behind this wall is also random, even for an optimal GCP policy. The red trajectory was achieved in the absence of the wall, while the green one was blocked by its presence.

success for a goal g would be able to capture the epistemic uncertainty:

$$p(g) \propto \text{Var}_{\theta_k}\{D_\phi^k(g)\}.$$

For uncontrollable areas where optimal policies converge to $D_\phi^k(g) \approx \frac{1}{2}$, we get $\text{Var}_{\theta_k}\{D_\phi^k(g)\} = 0$. On the other hand, goals that are mastered by a few agents (e.g $D_\phi^k(g) = 1$ for half the agents and $D_\phi^k(g) = 0$ for the others) would maximize the variance and indicate that the area is controllable with additional training.

6.2.3 Reward formulation

By working on Intrinsically Motivated Goal Conditioned Agents, we were able to provide some contextual answers and proposals to the following questions: How to generate intrinsic goals? How to learn goal representations for high-dimensional observations? One key element of the problem that we decided not to tackle is the answer to the question: **How to measure if a goal has been achieved**?

For the sake of simplicity, and to obtain some meaningful insights about the questions addressed in this thesis, we adopted during all of our experiments the sparse reward setting: $r_t = \mathbb{1}\Big[||s_t - g||_2 < \delta\Big]$, widely used in GCRL to avoid deceptive rewards with dense reward functions such as: $r_t = ||s_t - g||_2$. One can easily understand that the problem in the dense reward formulation comes from the use of an Euclidean metric, unsuited to capture the underlying topology of the environment, thus, misleading the agent about the right direction. Finding the right metric to use dense rewards would be a breakthrough, as it would enable us to come back to the dense reward formulation.

Manifold perspective on RL A key observation is that the interactions between the agent and the environment are constrained by several factors including the environment topology (e.g. walls in a maze, environment's boundaries), the agent design and capabilities (e.g. degrees of freedom of a robot arm) and many training variables specific to RL, conditioning the agent's behavior (e.g. policy learning algorithm, intrinsic goals...). All these constraints imply that the potential space of states achieved by the agent constitutes a subset embedded into the observation's Euclidean space: $S_\pi \subset \mathbb{R}^d$ (where observations are d-dimensional). Therefore, S_π can be considered as a manifold, that is locally Euclidean, but can globally possess a more complex topology (e.g a sphere or a plane in $\mathbb{R}^3$). A manifold $\mathcal{M}$ is endowed with an intrinsic coordinate system denoted as a *chart*, intuitively, a chart is a function assigning a data point on the manifold from a lower-dimensional coordinate system (as illustrated in Figure 6.2):

$$g : \mathcal{Z} \to \mathcal{M} \subset \mathbb{R}^d, \tag{6.1}$$

where $\mathcal{Z}$ is a lower-dimensional space containing the intrinsic coordinate system. Recent work has exposed a geometrical perspective in generative models (Hauberg, 2018; Arvanitidis et al., 2020; Chadebec and Allassonnière, 2022; Park et al., 2023b), and shown that any generator function g (e.g. the generator of a GAN or the decoder of a VAE) modeled on a data distribution can be seen as an approximation of the chart of the training data manifold (Arvanitidis et al., 2017; Hauberg, 2018):

$$g : \mathcal{Z} \to \hat{\mathcal{M}} \approx \mathcal{M} \subset \mathbb{R}^d. \tag{6.2}$$

This geometrical interpretation of a generator function implies that the latent space $\mathcal{Z}$ should not be seen as a linear Euclidean space, but rather as a curved space that captures the local distortions on the manifold. Then we can measure the distance between z and $z + \delta$ on the manifold using Taylor's theorem:

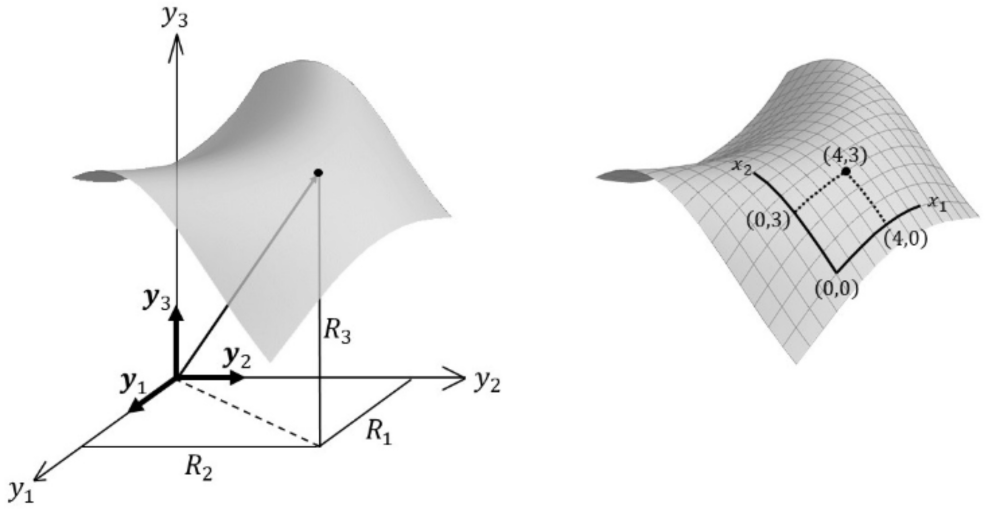

Figure 6.2: Example of a 2-dimensional curved manifold embedded in $\mathbb{R}^3$, with a 2-dimensional intrinsic coordinate system.

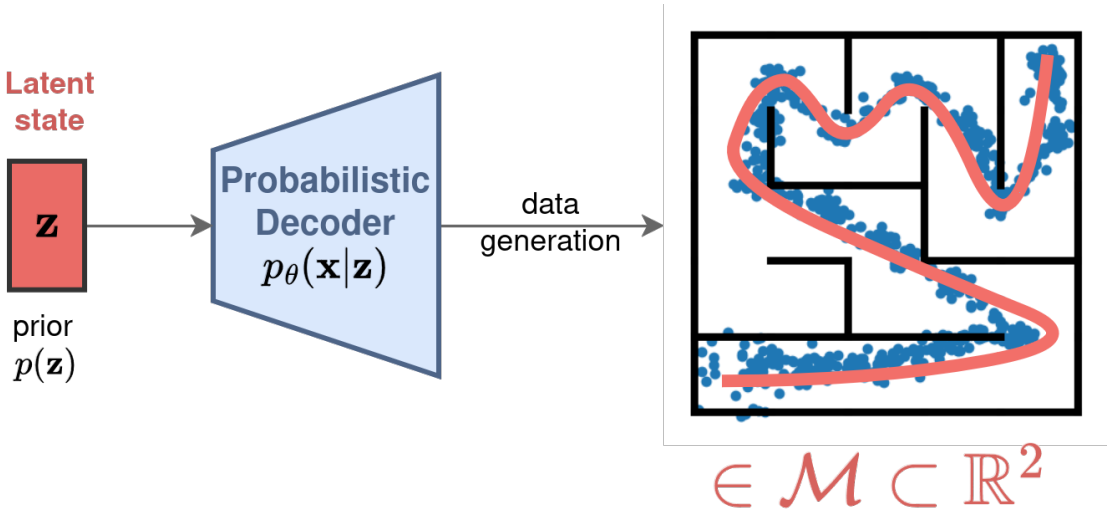

Figure 6.3: Example of VAE learned to represent observations from a 2-dimensional pointmaze. Data points generated from the decoder lie in a locally 1-dimensional curved manifold.

$$\begin{aligned} \|g(z) - g(z+\delta)\|^2 &= \|g(z) - (g(z) + J_z^\top \delta)\|^2 \\ &= \delta^\top J_z^\top J_z \delta, \end{aligned} \tag{6.3}$$

where J_z is the Jacobian of g at z. This implies that $\mathbf{M}_z := J_z^\top J_z$ is a Riemannian Metric, known as the *pull-back* metric, that defines a local inner product under which we can define curve lengths on the manifold through integration Gallot et al. (1990), known as *geodesics*:

$$\text{Length}(c) = \int_a^b \| \partial_t g(c_t) \| \, dt = \int_a^b \sqrt{\dot{c}_t^\top \mathbf{M}_z \dot{c}_t^\top} \tag{6.4}$$

The capacity to compute *geodesics* that follows the topology of the environment from data (Arvanitidis et al., 2019) would unlock further perspectives in intrinsically motivated Goal-conditioned agents:

1. Compute dense rewards with geodesics to speed up learning and maximize the potential of Intrinsically Motivated Goal-conditioned methods.

2. Generate goals that maximize the geodesic distance to foster exploration.

3. Using geodesic discretization through curve parametrization (Yang et al., 2018) to compute subgoals on the manifold respecting the environment topology.

6.3 Final word

In this final section, we take a step back to discuss ongoing ideas relative to the field of AI in general and Reinforcement Learning in particular. We start by discussing Anthropomorphism in the conception of AI algorithm, and explain how, although it has been successful in some regards, it could be counterproductive regarding RL. Then we discuss the Offline data-driven perspective on RL and the limitations we perceive, as well as its possible future combination we envision with the field of this Thesis, intrinsic motivations in Online RL.

6.3.1 Anthropomorphism is AI design

Throughout this manuscript, we aimed to propose efficient unsupervised procedures for GCRL, through the lens of intrinsic motivations. A determining factor in this Thesis was the purpose to design generalist training objectives,

in order for our methods to be agnostic to as many external variables as possible.

From our perspective, this is a crucial step toward making RL algorithms functional in real world applications. Training generalist autonomous agents is not only about performance gains. More importantly, we argue that being generalist and autonomous is an essential feature to operate complex tasks in hardly predictable environments. The real world is full of sources of uncertainty, an agent must be able to adapt to unexpected changes in the environment (e.g. sensory inputs with changes of light, moved object, ...). When fulfilling a task, as humans, we have a pretty good understanding on what to anticipate and what constitutes the challenging part of the task which will require our maximum effort. We argue that this comprehension, and the proximity of RL to decision-making activities typically associated with humans, might lead to counterproductive anthropomorphism in AI development.

Anthropomorphism is defined as the attribution of human-like feelings, mental states and behavioral characteristics to inanimate objects, animals, and in general to natural phenomena (Epley et al., 2008). Although inspiration from human-like characteristics like vision, language or decision-making has been successfully applied to design AI and particularly RL algorithms (e.g. with the concept of reward and intrinsic motivations) (Salles et al., 2020), we believe that for complex tasks and environment interactions, we could be misled about what constitutes the real challenge for the agent. The first risk is to hand design representations and mechanisms that are specific to a given task, which may not generalize well to other tasks. The second, more subtle risk is to design generic mechanisms based on some understanding of our own functioning, but which may turn out to be inadequate for an agent with a different embodiment and different information processing priors. For example, an AI researcher might be tempted to hand design stepping stones toward a final task for an agent to complete based on their understanding of the task. However, we struggle to understand the interactions between the sensory inputs and the optimization process that shapes the behavioral characteristics of an agent, mainly due to the "black-box" nature of neural networks and the optimization algorithms. Thus our concepts of "easy" and "hard" might completely differ from the agent's perspective. The agents have to be embedded with intrinsic mechanisms based on metrics relevant to their implementation, thus allowing them to assess their situation and to adapt to new circumstances.

Consequently, it is important to dedicate significant research effort into the conception of unsupervised objectives, allowing to establish and to focus on what is important, in an autonomous manner. This is specially the case in RL where, in our sense, the definition of agents mainly designed to operate in

narrow supervised settings drastically undermines their potential in real world applications. Though it is modest, we hope this work is a useful contribution toward this objective.

6.3.2 Offline Data-driven vs Online Reinforcement learning

In the pursuit of building generalist and versatile agents, Reinforcement Learning has recently witnessed an increasing popularity of data-driven methods based on prior large scale training. It has become clear that building skilled and useful agents from scratch on real world scenarios is a dreaded task, for example us humans are embedded with a handful of prior knowledge on our environment, whether innate or acquired at an early stage of life, in terms of language, social and physical representation of our world, which are essential to acquire diverse new skills. Propelled by the advances and impressive performances of large scale models trained on heavy datasets, such as Large Language Models (LLM), offline RL that learns a policy from a collection of pre-existing and diverse environments interactions has gained traction. For instance, D4RL (Fu et al., 2020) or Open X-Embodiment O'Neill et al. (2023) are efforts to gather a sufficiently large dataset of previously collected agents trajectories on standard RL environments, on which offline RL methods aim to extract optimal policies. Even further, facing the limited number of agent-environment interactions compared to text, Nair et al. (2022) and Ma et al. (2022) suggest casting human videos as goal-conditioned RL, aiming to extract universal goal oriented representations and learn generalist policies from data available on the internet. As an example of this conceptual perspective on RL, the Decision Transformer method (Chen et al., 2021c) considers modeling entire trajectories with GPT-like architectures (Radford et al., 2018), where actions states and rewards are equally framed as tokens. These architectures show promising results as the model is able to capture the dynamics of the environment and seems to get a sense on what should be done to get a specific reward.

Our work fits the Online conception of RL, we aimed to train generalist agents from scratch in an end-to-end fashion. One could argue that the lack of prior knowledge provided by external data constrains us to rather limited applications, however we believe that unsupervised training in the online RL setting remains a core building block toward autonomous generalist agents. Indeed, despite the impressive performances of large scale models such as LLMs, recent work exhibit their brittleness to outliers, suggesting poor generalization capabilities for non-trivial knowledge (Huang et al., 2023), more akin to memorization. As it stands, offline RL agents would periodically need

additional fine-tuning to handle new circumstances. The contributions of this Thesis in terms of autonomous goal selection in online learning could be embedded into large scale data-driven agents in order to benefit from the best of both worlds: extensive prior knowledge provided by large scale models, and the autonomous capacity to pursue new challenging tasks and knowledge that either consist of outliers or previously non-mastered tasks. Some attempts to combine large scale models with online RL setting have recently been introduced, for example with LLMs in textual sequential decision-making tasks (Huang et al., 2022; Carta et al., 2023; Du et al., 2023; Colas et al., 2023; Abdelghani et al., 2024). We believe this specific research field to be the key to generalist agents able to understand and interact with complex environments.

Bibliography

Rania Abdelghani, Yen-Hsiang Wang, Xingdi Yuan, Tong Wang, Pauline Lucas, Hélène Sauzéon, and Pierre-Yves Oudeyer. Gpt-3-driven pedagogical agents to train children's curious question-asking skills. *International Journal of Artificial Intelligence in Education*, 34(2):483–518, 2024.

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning, 2017. URL https://arxiv.org/abs/1703.01732.

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms, 2018. URL https://arxiv.org/abs/1807.10299.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Decstr: Learning goal-directed abstract behaviors using pre-verbal spatial predicates in intrinsically motivated agents. *arXiv preprint arXiv:2006.07185*, 2020.

Neziha Akalin and Amy Loutfi. Reinforcement learning approaches in social robotics. *Sensors*, 21(4):1292, 2021.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. *arXiv preprint arXiv:1707.01495*, 2017.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017.

Georgios Arvanitidis, Soren Hauberg, Philipp Hennig, and Michael Schober. Fast and robust shortest paths on manifolds learned from data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1506–1515. PMLR, 2019.

Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, 2020.

Georgios Arvanitidis, Bogdan Georgiev, and Bernhard Schölkopf. A prior-based approximate latent riemannian metric. *arXiv preprint arXiv:2103.05290*, 2021.

Arthur Aubret, Laetitia Matignon, and Salima Hassas. Distop: Discovering a topological representation to learn diverse and rewarding skills. *IEEE Transactions on Cognitive and Developmental Systems*, 15(4):1905–1915, 2023.

David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16(320):201, 2004.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/blundell15.html.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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

Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

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, 2020. URL https://arxiv.org/abs/2002.03647.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

Nicolas Castanet, Sylvain Lamprier, and Olivier Sigaud. Stein variational goal generation for adaptive exploration in multi-goal reinforcement learning. *arXiv preprint arXiv:2206.06719*, 2022.

Clément Chadebec and Stéphanie Allassonnière. A geometric perspective on variational autoencoders. *Advances in Neural Information Processing Systems*, 35:19618–19630, 2022.

Jiayu Chen, Yuanxin Zhang, Yuanfan Xu, Huimin Ma, Huazhong Yang, Jiaming Song, Yu Wang, and Yi Wu. Variational automatic curriculum learning for sparse-reward cooperative multi-agent problems. *Advances in Neural Information Processing Systems*, 34:9681–9693, 2021a.

Jiayu Chen, Yuanxin Zhang, Yuanfan Xu, Huimin Ma, Huazhong Yang, Jiaming Song, Yu Wang, and Yi Wu. Variational automatic curriculum learning for sparse-reward cooperative multi-agent problems, 2021b. URL https://arxiv.org/abs/2111.04613.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021c.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 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.

Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class svm for learning in image retrieval. In *Proceedings 2001 international conference on image processing (Cat. No. 01CH37205)*, volume 1, pages 34–37. IEEE, 2001.

Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. Variational empowerment as representation learning for goal-based reinforcement learning. *arXiv preprint arXiv:2106.01404*, 2021.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018. URL https://arxiv.org/abs/1805.12114.

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2606–2615, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/chwialkowski16.html.

Cédric Colas. *Towards Vygotskian Autotelic Agents: Learning Skills with Goals, Language and Intrinsically Motivated Deep Reinforcement Learning*. PhD thesis, Université de Bordeaux, 2021.

Cédric Colas, Pierre Fournier, Olivier Sigaud, and Pierre-Yves Oudeyer. CURIOUS: intrinsically motivated multi-task multi-goal reinforcement learning. In *ICML*, 2018.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74: 1159–1199, 2022.

Cédric Colas, Laetitia Teodorescu, Pierre-Yves Oudeyer, Xingdi Yuan, and Marc-Alexandre Côté. Augmenting autotelic agents with large language models. In *Conference on Lifelong Learning Agents*, pages 205–226. PMLR, 2023.

Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2023. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.

John Duchi, Tatsunori Hashimoto, and Hongseok Namkoong. Distributionally robust losses for latent covariate mixtures. *Operations Research*, 71(2):649–664, 2023.

John C Duchi, Peter W Glynn, and Hongseok Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. *Mathematics of Operations Research*, 46(3):946–969, 2021.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

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

Nicholas Epley, Adam Waytz, Scott Akalis, and John T Cacioppo. When we need a human: Motivational determinants of anthropomorphism. *Social cognition*, 26(2):143–155, 2008.

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

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018b. URL https://arxiv.org/abs/1802.06070.

Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse RL: hindsight inference for policy improvement. *CoRR*, abs/2002.11089, 2020. URL https://arxiv.org/abs/2002.11089.

Meng Fang, Cheng Zhou, Bei Shi, Boqing Gong, Jia Xu, and Tong Zhang. DHER: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations*, 2019a.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32, 2019b.

Julien Ferry, Ulrich Aivodji, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. Improving fairness generalization through a sample-robust optimization method. *Machine Learning*, pages 1–62, 2022.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/gal16.html.

Sylvestre Gallot, Dominique Hulin, Jacques Lafontaine, et al. *Riemannian geometry*, volume 2. Springer, 1990.

Quentin Gallouédec and Emmanuel Dellandréa. Cell-free latent go-explore. In *International Conference on Machine Learning*, pages 10571–10586. PMLR, 2023.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. *arXiv preprint arXiv:2106.13687*, 2021.

Ioannis Gatopoulos and Jakub M Tomczak. Self-supervised variational auto-encoders. *Entropy*, 23(6):747, 2021.

Samuel J Gershman and Yael Niv. Novelty and inductive generalization in human reinforcement learning. *Topics in cognitive science*, 7(3):391–415, 2015.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies, 2019. URL https://arxiv.org/abs/1811.07819.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. In *International Conference on Machine Learning*, pages 1292–1301. PMLR, 2017.

Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences*, 17(11):585–593, 2013.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control, 2016. URL https://arxiv.org/abs/1611.07507.

Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder, 2019. URL https://arxiv.org/abs/1806.03107.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/gutmann10a.html.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

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, 2019b.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, pages 1929–1938. PMLR, 2018.

Søren Hauberg. Only bayes should learn a manifold (on the estimation of differential geometric structure from data). *arXiv preprint arXiv:1806.04994*, 2018.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020.

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

Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017a.

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, 2017b.

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

Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22:85–126, 2004.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1(2), 2016.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Riashat Islam, Hongyu Zang, Anirudh Goyal, Alex M Lamb, Kenji Kawaguchi, Xin Li, Romain Laroche, Yoshua Bengio, and Remi Tachet des Combes. Discrete compositional representations as an abstraction for goal conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:3885–3899, 2022.

Allan Jabri, Kyle Hsu, Ben Eysenbach, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning, 2019. URL https://arxiv.org/abs/1912.04226.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

Pierre-Alexandre Kamienny, Jean Tarbouriech, Alessandro Lazaric, and Ludovic Denoyer. Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. *CoRR*, abs/2110.14457, 2021. URL https://arxiv.org/abs/2110.14457.

Seongun Kim, Kyowoon Lee, and Jaesik Choi. Variational curriculum reinforcement learning for unsupervised discovery of skills. *arXiv preprint arXiv:2310.19424*, 2023.

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

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in vaes. *Advances in neural information processing systems*, 32, 2019.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pages 128–135. IEEE, 2005.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics, 2020. URL https://arxiv.org/abs/2005.04269.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017. URL https://arxiv.org/abs/1612.01474.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 13–18 Jul 2020a. URL https://proceedings.mlr.press/v119/laskin20a.html.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020b.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model, 2020. URL https://arxiv.org/abs/1907.00953.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Siyuan Li, Lulu Zheng, Jianhao Wang, and Chongjie Zhang. Learning subgoal representations with slow dynamics. In *International Conference on Learning Representations*, 2021.

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.

Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. Competitive experience replay. *arXiv preprint arXiv:1902.00528*, 2019.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Qiang Liu. Stein variational gradient descent as gradient flow. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/17ed8abedc255908be746d245e50263a-Paper.pdf.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *arXiv preprint arXiv:1608.04471*, 2016.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 276–284, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/liub16.html.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. *Advances in neural information processing systems*, 25, 2012.

Xingyu Lu, Stas Tiomkin, and Pieter Abbeel. Predictive coding for boosting deep reinforcement learning with sparse rewards. *arXiv preprint arXiv:1912.13414*, 2019.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Daniel J Mankowitz, Augustin Žídek, André Barreto, Dan Horgan, Matteo Hessel, John Quan, Junhyuk Oh, Hado van Hasselt, David Silver, and Tom Schaul. Unicorn: Continual learning with a universal, off-policy agent. *arXiv preprint arXiv:1802.08294*, 2018.

Natalia L Martinez, Martin A Bertran, Afroditi Papadaki, Miguel Rodrigues, and Guillermo Sapiro. Blind pareto fairness and subgroup robustness. In *International Conference on Machine Learning*, pages 7492–7501. PMLR, 2021.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models, 2021.

Paul Michel, Tatsunori Hashimoto, and Graham Neubig. Modeling the second player in distributionally robust optimization, 2021.

Paul Michel, Tatsunori Hashimoto, and Graham Neubig. Distributionally robust models with parametric likelihood ratios, 2022.

Volodymyr Mnih. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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):529–533, 2015.

Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, and Pranav Rajpurkar. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956): 259–265, 2023.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *arXiv preprint arXiv:1807.04742*, 2018.

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. In *Conference on Robot Learning*, pages 530–539. PMLR, 2020.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation, 2022. URL https://arxiv.org/abs/2203.12601.

Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open X-embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique P d O Pinto, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*, 2021.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:108, 2007.

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.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery, 2023a. URL https://arxiv.org/abs/2302.05103.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction, 2024. URL https://arxiv.org/abs/2310.08887.

Yong-Hyun Park, Mingi Kwon, Jaewoong Choi, Junghyo Jo, and Youngjung Uh. Understanding the latent space of diffusion models through the lens of riemannian geometry. *Advances in Neural Information Processing Systems*, 36:24129–24142, 2023b.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/pathak17a.html.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7750–7761. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/pitis20a.html.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018a.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018b.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Sebastien Racaniere, Andrew Lampinen, Adam Santoro, David Reichert, Vlad Firoiu, and Timothy Lillicrap. Automated curriculum generation through setter-solver interactions. In *International Conference on Learning Representations*, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Juergen Schmidhuber. Hindsight policy gradients. *arXiv preprint arXiv:1711.06006*, 2017.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation, 2019. URL https://arxiv.org/abs/1906.04279.

Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment – an introduction, 2013. URL https://arxiv.org/abs/1310.1863.

Arleen Salles, Kathinka Evers, and Michele Farisco. Anthropomorphism in ai. *AJOB neuroscience*, 11(2):88–95, 2020.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320. PMLR, 2015.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10 (5):1299–1319, 1998.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

Erich Schubert and Michael Gertz. Intrinsic t-stochastic neighbor embedding for visualization and outlier detection: A remedy against the curse of dimensionality? In *Similarity Search and Applications: 10th International Conference, SISAP 2017, Munich, Germany, October 4-6, 2017, Proceedings 10*, pages 188–203. Springer, 2017.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

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

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.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills, 2020. URL https://arxiv.org/abs/1907.01657.

Olivier Sigaud, Ahmed Akakzia, Hugo Caselles-Dupré, Cédric Colas, Pierre-Yves Oudeyer, and Mohamed Chetouani. Toward teachable autotelic agents. *IEEE Transactions on Cognitive and Developmental Systems*, 15 (3):1070–1084, 2022.

Olivier Sigaud, Gianluca Baldassarre, Cedric Colas, Stéphane Doncieux, Richard Duro, Nicolas Perrin-Gilbert, and Vieri-Giuliano Santucci. A definition of open-ended learning problems for goal-conditioned agents. *arXiv preprint arXiv:2311.00344*, 2023.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Edward Jay Sondik. *The optimal control of partially observable Markov processes*. Stanford University, 1971.

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.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33(01), pages 5066–5073, 2019.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents, 2021. URL https://arxiv.org/abs/2107.12808.

Jakub Tomczak and Max Welling. Vae with a vampprior. In *International conference on artificial intelligence and statistics*, pages 1214–1223. PMLR, 2018.

Jakub M Tomczak. Why deep generative modeling? In *Deep Generative Modeling*, pages 1–12. Springer, 2021.

Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards, 2019. URL https://arxiv.org/abs/1911.01417.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Dimitris G. Tzikas, Aristidis C. Likas, and Nikolaos P. Galatsanos. The variational approximation for bayesian inference. *IEEE Signal Processing Magazine*, 25(6):131–146, 2008. doi: 10.1109/MSP.2008.929620.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Lev S Vygotsky. *Thought and language*. MIT press, 2012.

Serena Wang, Wenshuo Guo, Harikrishna Narasimhan, Andrew Cotter, Maya Gupta, and Michael Jordan. Robust optimization for fairness with noisy

protected groups. *Advances in neural information processing systems*, 33: 5190–5203, 2020.

Yijie Wang, Viet Anh Nguyen, and Grani A Hanasusanto. Wasserstein robust classification with fairness constraints. *arXiv preprint arXiv:2103.06828*, 2021.

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

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018. URL https://lilianweng.github.io/posts/2018-08-12-vae/.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Rui Yang, Meng Fang, Lei Han, Yali Du, Feng Luo, and Xiu Li. Mher: Model-based hindsight experience replay. *arXiv preprint arXiv:2107.00306*, 2021.

Tao Yang, Georgios Arvanitidis, Dongmei Fu, Xiaogang Li, and Søren Hauberg. Geodesic clustering in deep generative models. *arXiv preprint arXiv:1809.04747*, 2018.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11920–11931. PMLR, 18–24 Jul 2021a. URL https://proceedings.mlr.press/v139/yarats21a.html.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35 (12), pages 10674–10681, 2021b.

Peiyu Yu, Yaxuan Zhu, Sirui Xie, Xiaojian Shawn Ma, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Learning energy-based prior model with diffusion-amortized mcmc. *Advances in Neural Information Processing Systems*, 36: 42717–42747, 2023.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *arXiv preprint arXiv:2006.09641*, 2020. doi: 10.48550/ARXIV.2006.09641. URL https://arxiv.org/abs/2006.09641.

Appendix A

Stein Variational Goal Generation

A.1 Proof of Theorem 1

Theorem 1. **Recovery property**: Let us denote as $\mathcal{G}^+$ the set of goals g such that $V_\psi(g) > 0$ and $\mathcal{C} \in \mathbb{R}^d$ its convex hull. Assume that, at a given iteration l, $D_\phi(g) \approx 1$ for every $g \in \mathcal{G}^+$ (i.e., the whole set $\mathcal{G}^+$ is considered as mastered by $D_\phi(g)$), and that, on that set, V_ψ is well calibrated: for any area $\omega \subseteq \mathcal{G}^+$ and any goal $g \in \omega$, $V_\psi(g) \approx P(g \in \mathcal{G}^* | g \in \omega)$. Assume also that we use a kernel which ensures that the Kernelized Stein Discrepancy $KSD(q, p_{\text{goals}})$ of any distribution q with p_{goals} is 0 only if q weakly converges to p_{goals}[1]. Then, with no updates of the models after iteration l and a number of particles $m > 1$, any area $\omega \subseteq \mathcal{G}^+ \cap \mathcal{G}^*$ with diameter $diam(\omega) \geq \sqrt{d} \frac{diam(\mathcal{C})}{(\sqrt[d]{m}-1)}$ eventually contains at least one particle, whenever $KSD(\{x^i\}_{i=1}^n, p_{\text{goals}}) = 0$ after convergence of the particle updates.

Proof. In settings where the KSD of a distribution μ from a target p is 0 only if μ weakly converges to p, Liu (2017) previously proved that, for any compact set, the empirical measure μ^n of μ, computed on a set of n particles, converges weakly towards the target p when a sufficient number of particles is used. Thus, under our hypotheses, the set of particles of SVGG appropriately represents p_{goals} after a sufficient number of steps of Stein transforms.

Now, we know that particles cannot leave $\mathcal{G}^+$ since $V_\psi = 0$ outside, and so does p_{goals}. Since V_ψ is well calibrated on $\mathcal{G}^+$, we also know that $V_\psi = 1$ on

[1] As assumed for instance in Liu (2017). This is not always true, but gives strong insights about the behavior of SVGD. Please refer to Gorham and Mackey (2017) for more discussions about KSD.

every area of $\mathcal{G}^+$ only containing achievable goals. Thus, since $p_{\text{skills}} = 1$ in $\mathcal{G}^+$, p_{goals} is maximal and constant for any area $\omega \in \mathcal{G}^+ \cap \mathcal{G}^*$. This implies that the concentration of particles in any area of $\mathcal{G}^+ \cap \mathcal{G}^*$ is greater than if particles were uniformly spread over $\mathcal{C}$. In that case, for any particle x from the set $\{x_i\}_{i=1}^m$, we know that $P(x \in \omega | KSD(q = \{x_i\}_{i=1}^m, p_{\text{goals}}) = 0) \geq P(x \in \omega | KSD(q = \{x_i\}_{i=1}^m, U(\mathcal{X})) = 0)$, with $\mathcal{X}$ an hypercube of d dimensions with side length equal to $diam(\mathcal{C})$ and $U(\mathcal{X})$ the uniform distribution over $\mathcal{X}$.

Next, if $KSD(q = \{x_i\}_{i=1}^m, U(\mathcal{X})) = 0$, we know that particles are spread as a grid over each dimension of $\mathcal{X}$. Thus, in each dimension of $\mathcal{X}$ any particle is separated from its closest neighbor by at most a difference of $diam(\mathcal{C})/(\sqrt[d]{m} - 1)$ in the worst case. Thus, any area ω with diameter greater than $\sqrt{d}\frac{diam(\mathcal{C})}{(\sqrt[d]{m}-1)}$ is guaranteed to contain a particle in that case, which concludes the proof. □

A.2 Additional experiments details

A.2.1 Environments

Pointmaze We use a 2D pointmaze environment where the state space and goal space are (x, y) coordinates (the agent cannot see the walls), and the agent moves according to its action, which is a 2-dimensional vector with dimensions constrained to $[-0.95, 0.95]$. All methods are tested on four 5×5 mazes with different shapes and a highly discontinuous goal space. The maximum number of steps in one trajectory is set to 30. We argue that the difficulty of these environments does not lie in their size, but rather in the complexity of their goal space and thus of the trajectories adopted by the agent to achieve these goals.

Antmaze An Ant has to move around in a U-shape hallway maze of size 20x4, the goal space remains (x, y) coordinates of the Ant as in Pointmaze. However, the exploratory behavior is much simpler than in the considered pointmazes environments, as the maze is simply U-shaped. The difficulty lies in the control of the Ant dynamics, as it must move its legs with a 8-dimensional action space and the observation space is a 30-dimensional vector corresponding to the angles of the legs.

FetchPickAndPlace (Hard version) We also perform comparisons on a hard version of FetchPickAndPlace-v1 from OpenAI gym Brockman et al. (2016). The agent controls a robotic arm which must pick and place a block to a 3D desired location. In the hard version, the behavioral goals are all between

20 and 45cm in the air, while in the original version, 50% of behavioral goals are on the table and the remaining ones are from 0 to 45cm in the air.

While this environment presents a greater difficulty in terms of control (the action dimension is 4 and the state dimension is 24 which correspond to the various positions of the robot joint), the goal generation component is significantly easier: as the 3-dimensional goal space is smooth, there is no obstacle for the agent to bypass. As Figure 4.4 shows, SVGG solves the environment within 1.5M steps, but does not significantly outperform MEGA.

We argue that the interest of our goal generation algorithm resides in environments with highly discontinuous goal space as the action control is largely supported by the RL algorithm (e.g. DDPG). Therefore, the smaller difference between MEGA and SVGG in this environment was expected, as SVGG is mainly designed to target non-smooth goal spaces, and to avoid pitfalls such as catastrophic forgetting.

FetchReach (Stuck version) We compare SVGG to baselines with a modified version of the FetchReach environment, where the gripper is initially stuck between four walls and has to carefully navigate between them to reach the goals. The observations are 9-dimensional vectors, the actions as well as the goals are 3-dimensional. The success coverage is computed on uniformly sampled goals with a target range fixed to 0.2, which corresponds to the maximal L2 distance between the goal and the initial gripper position. The initial task is solved within 20.000 steps by all baselines, whereas with the modified version, only SVGG achieves a 0.8 success rate in 3M steps.

FetchPush (Maze version) We add a maze structure with walls on the table to the FetchPush benchmark to once again add discontinuities in the goal space. The observations are 24-dimensional vectors, the actions as well as the goals are 3-dimensional. The success coverage is computed on goals uniformly sampled on the table, and the block's initial position is sampled behind the first wall on the robot side. The standard FetchPush task was solved by SVGG and MEGA in less than 1M steps. The maze version is much harder: SVGG achieves a 0.8 success coverage rate within 10M steps, whereas MEGA barely reaches 0.5. The distance to reach the goals in all environments is $\delta = 0.15$.

A.3 Methods details

We first focus on the behavioral goal sampling strategy. We use DDPG with HER to learn how to reach a goal in SVGG, MEGA and GoalGan. DDPG is

parameterized as in Table A.1. All algorithms are run using the same number of training epochs n.

Table A.1: DDPG parameters

DDPG Hyper-Parameters	**Value**
Batch size for replay buffer	2000
Discount factor γ	0.99
Action L2 regularization	0.1
(Gaussian) Action noise max std	0.1
Warm up steps before training	2500
Actor learning rate	1e-3
Critic learning rate	1e-3
Target network soft update rate	0.05
Actor & critic networks activation	Gelu
Actor & critic hidden layers sizes	512^3
Replay buffer size	nb training steps

A.3.1 Hindsight Experience Replay

The original goal relabeling strategy introduced in HER Andrychowicz et al. (2017) is the *future*, which consists in relabeling a given transition with a goal achieved on the trajectory later on. This is very effective in sparse reward setting to learn a GCP. However, many works suggested that relabeling transitions with goals outside the current trajectory helps the agent generalize across trajectories. For example, one can use inverse RL to determine the optimal goal to relabel a given transition Eysenbach et al. (2020). We use a naive version of this method. As in Pitis et al. (2020), we relabel transitions for DDPG optimization using a mixed strategy. All methods presented in this work use the same strategy.

10% of real experience are kept while 40% of the transitions are relabeled using the *future* strategy of HER. We relabel the remaining 50% transitions with goals outside of their trajectory, with randomly sampled goals among the past behavioral and achieved goals. The latter part of the strategy helps share information between different trajectories that often contains similar transitions.

A.3.2 SVGG

SVGG is described in the main paper, in this section, we go through implementation details and hyper-parameters.

SVGG Hyper-Parameters	Symbol	Value
SVGD		
Number of particles	m	100
Optimization interval (in steps)		20
Nb of particle moves per optim.	$k^{(p)}$	1
RBF kernel $k(.,.)$ bandwidth	σ	1
Learning rate	ϵ	1e-3
Agent's skill model D_ϕ		
Hidden layers sizes		(64, 64)
Gradient steps per optimization	K	10
Learning rate		1e-3
Training batch size	$l^{(m)}$	100
Training history length (episodes)		1000
Optimization interval (in steps)		4000
Nb of training steps	$k^{(m)}$	100
Activations		Gelu
OCSVM validity distribution		
RBF kernel $k(.,.)$ bandwidth	σ	1
Optimization interval (in steps)		4000

A.3.3 MEGA

The authors of MEGA Pitis et al. (2020) train a GCP with previously achieved goals from a replay buffer. Their choice of goals relies on a minimum density heuristic, where they model the distribution of achieved goals with a KDE. They argue that aiming at novel goals suffices to efficiently discover and control the environment. We use the original implementation of the authors, the pseudocode is given in Algorithm 6 and specific hyper-parameters are in Table A.2.

Table A.2: MEGA parameters

MEGA Hyper-parameters	Symbol	Value
RBF kernel bandwidth	σ	0.1
KDE optimization interval (in steps)		1
Nb of state samples for KDE optim.		10.000
Nb of sampled candidate goals	N	100
Q-value cutoff	c	-3

A.3.4 GoalGAN

GoalGan Florensa et al. (2018) uses a procedural goal generation method based on GAN training. As our SVGG, it aims at sampling goals of intermediate difficulty, which they define as $\mathcal{G}_{\text{GOID}} = \{g | P_{min} < P_\pi(g) < P_{max}\}$, $P_\pi(g)$ being the probability for the policy π to achieve goal g, P_{min} and P_{max} are hyper-parameters. A Discriminator D_{θ_d} is trained to distinguish between goals in $\mathcal{G}_{\text{GOID}}$ and other goals, while a generator G_{θ_g} is trained to output goals in $\mathcal{G}_{\text{GOID}}$ by relying on the discriminator outputs. They optimize G_{θ_g} and D_{θ_d} in a manner similar to the Least-Squares GAN (LSGAN) with the following losses:

$$\begin{aligned} \min_{\theta_d} V(D_{\theta_d}) =& \mathbb{E}_{g\sim\mathcal{R}}\Big[y_g(D_{\theta_d}(g)-1)^2 \\ &+ (1-y_g)(D_{\theta_d}(g)+1)^2\Big] \\ \min_{\theta_g} V(G_{\theta_g}) =& \mathbb{E}_{z\sim\mathcal{N}(0,I)}\Big[D_{\theta_d}(G_{\theta_g}(z))^2\Big], \end{aligned} \tag{A.1}$$

where y_g is the label that indicates whether g belongs to $\mathcal{G}_{\text{GOID}}$ or not. In Florensa et al. (2018), the authors use Monte-Carlo sampling of the policy to estimate y_g. For efficiency reasons, we use a learned model of the agent's capabilities as in SVGG. The pseudocode is given in Algorithm 7 and specific hyper-parameters are in Table A.3.

A.3.5 Ressources

Every seed of each experiment was run on 1 GPU in the following list {Nvidia RTX A6000, Nvidia RTX A5000, Nvidia TITAN RTX, Nvidia Titan Xp, Nvidia TITAN V}. The total training time is close to 4.000 hours, most of it executed in parallel, as we had full time access to 12 GPUs.

Table A.3: GoalGan parameters

GoalGAN Hyper-parameters	**Symbol**	**Value**
Gaussian prior dimension		4
Generator hidden layers sizes		(64, 64)
Discriminator hidden layers sizes		(64, 64)
Optimization interval (in steps)		2000
GAN training batch size	$l^{(g)}$	200
Nb of GAN optimization steps	$k^{(g)}$	100
GAN Learning rate		1e-3
Minimum GOID probability	P_{min}	0.1
Maximum GOID probability	P_{max}	0.9

Algorithm 5 *RL with Stein Variational Goal Generation*

Input: a GCP π_θ, a success predictor D_ϕ, a reachability predictor R_ψ, buffers of transitions $\mathcal{B}$, reached states $\mathcal{R}$ and success outcomes O, a kernel k, hyper-parameters $t^{(r)}$, $t^{(m)}$, $t^{(p)}$ standing for number of rollouts, model updates and particles updates respectively, and hyper-parameter b standing for the batchsize of the model update.
Sample a set of particles $q = \{x^i\}_{i=1}^m$ uniformly from states reached by pre-runs of π_θ;
for n epochs **do**
  for $t^{(r)}$ iterations **do** ▷ *Data Collection*
    Sample a goal g from q;
    $\tau \leftarrow$ Rollout $\pi_\theta(.|., g)$;
    Store all $(s_t, a_t, s_{t+1}, r_t, g)$ from τ in $\mathcal{B}$ and every s_t from τ in $\mathcal{R}$;
    Optionally (HER): Store relabeled transitions from τ;
    Store outcome $(g, I(s_{|\tau|} \approx g))$ in O;
  end for
  for $t^{(m)}$ iterations **do** ▷ *Model Update*
    Sample a batch $\{(g^i, s^i)\}_{i=1}^b$ from O;
    Oversample the minority class w.r.t. s to get a balanced success/-failure dataset;
    Update model D_ϕ (e.g., via ADAM), with gradient of $\mathcal{L}_\phi$ (4.9));
  end for
  Update model R_ψ according to all states in $\mathcal{R}$; ▷ *Prior Update*
  for $t^{(p)}$ iterations **do** ▷ *Particles Update*
    Compute the density of the target p_{goals} for the set of particles q using 4.8;
    Compute transforms: $\phi^*(x_i) = \frac{1}{m}\sum_{j=1}^m \left[k(x_j, x_i)\nabla_{x_j} \log p_{\text{goals}}(x_j) + \nabla_{x_j} k(x_j, x_i)\right]$;
    Update particles $x_i \leftarrow x_i + \epsilon\phi^*(x_i), \quad \forall i = 1 \cdots m$;
  end for
  Improve agent with any Off-Policy RL algorithm ▷ *Agent Improvement*
  (e.g., DDPG) using transitions from $\mathcal{B}$;
end for

Algorithm 6 *MEGA*

Input: a GCP π_θ, buffer of reached states $\mathcal{R}$, a KDE model P_{as} of the achieved states, numbers c, N, $t^{(r)}$ and $l^{(r)}$.
Initialize $\mathcal{R}$ with states reached by pre-runs of π_θ;
for n epochs **do** ▷ *Data Collection*
  for $t^{(r)}$ iterations **do**
    Sample a batch $\{g^i\}_{i=1}^{N}$ uniformly from $\mathcal{R}$;
    Eliminate unachievable candidates if their Q-values are below cutoff c;
    Choose goals $\{g^i\}_{i=1}^{l^{(r)}} = \text{argmin}_{g_i} P_{as}(g_i)$
    for i from 1 to $l^{(r)}$ **do**
      $\tau \leftarrow$ Rollout $\pi_\theta(.|., g = g^i)$;
      Store all $(s_t, a_t, s_{t+1}, r_t, g^i)$ from τ in $\mathcal{B}$ and every s_t from τ in $\mathcal{R}$;
      Optionally (HER): Store relabeled transitions from τ;
      Store outcome $(g^i, I(s_{|\tau|} \approx g^i))$ in O;
    end for
  end for
  Update KDE model P_{as} with uniform sampling from $\mathcal{R}$; ▷ *Model Update*
  Improve agent with any Off-Policy RL algorithm ▷ *Agent Improvement*
  (e.g., DDPG) using transitions from $\mathcal{B}$;
end for

Algorithm 7 *GoalGAN*

Input: a GCP π_θ, a goal Generator G_{θ_g}, a Discriminator D_{θ_d}, a success predictor D_ϕ, buffers of transitions $\mathcal{B}$, reached states $\mathcal{R}$ and success outcomes O, numbers $t^{(r)}$, $l^{(r)}$, $l^{(m)}$, $k^{(m)}$, $l^{(g)}$ and $k^{(g)}$.
Initialize G_{θ_g} and D_{θ_d} with pre-runs of π_θ;
for n epochs **do** ▷
Data Collection
  for $t^{(r)}$ iterations **do**
    Sample noise $\{z_i\}_{i=1}^{l^{(r)}} \sim \mathcal{N}(0, I)$
    generate $\{g^i\}_{i=1}^{l^{(r)}} = G_{\theta_g}(\{z_i\}_{i=1}^{l^{(r)}})$
    for i from 1 to $l^{(r)}$ **do**
      $\tau \leftarrow$ Rollout $\pi_\theta(.|., g = g^i)$;
      Store all $(s_t, a_t, s_{t+1}, r_t, g^i)$ from τ in $\mathcal{B}$ and every s_t from τ in $\mathcal{R}$;
      Optionally (HER): Store relabeled transitions from τ;
      Store outcome $(g^i, I(s_{|\tau|} \approx g^i))$ in O;
    end for
  end for
  Sample a batch $\{(g^i, s^i)\}_{i=1}^{l^{(g)}}$ from O; ▷
  GAN training
  Label goals (GOID or not) with model D_ϕ : $\{y_{g_i}\}_{i=1}^{l^{(g)}} = \{P_{min} < D_\phi(g_i) < P_{max}\}_{i=1}^{l^{(g)}}$
  for $k^{(g)}$ iterations **do**
    Update G_{θ_g} and D_{θ_d} (e.g. with ADAM) with gradients of LSGAN losses; (A.1)
  end for
  for $k^{(m)}$ iterations **do** ▷
  Model Update
    Sample a batch $\{(g^i, s^i)\}_{i=1}^{l^{(m)}}$ from O;
    Update model D_ϕ (e.g., via ADAM), with gradient of $\mathcal{L}_\phi$ (agent model loss described in the paper);
  end for
  Improve agent with any Off-Policy RL algorithm ▷
  Agent Improvement
  (e.g., DDPG) using transitions from $\mathcal{B}$;
end for

Appendix B

Distributionally Robust Optimization for Image Driven Exploration

B.1 Skew-Fit is a non-parametric DRO

B.1.1 Non-parametric solution of DRO

We start from the inner maximization problem stated in (5.6), for a given fixed θ:

$$\max_{r} \frac{1}{N}\sum_{i=1}^{N} r(x_i, y_i) l(f_\theta(x_i), y_i)) - \lambda \frac{1}{N}\sum_{i=1}^{N} r(x_i, y_i) \log r(x_i, y_i) \qquad \text{(B.1)}$$
$$st \quad \frac{1}{N}\sum_{i=1}^{N} r(x_i, y_i) = 1.$$

From this formulation, we can see that the risk associated to a shift of test distribution can be mitigated by simply associating adversarial weights $r_i := r(x_i, y_i)$ to every sample (x_i, y_i) from the training dataset, respecting $\bar{r} := \frac{1}{N}\sum_{i=1}^{N} r_i = 1$. This can be viewed as an infinite capacity function r, able to over-specify on every training data point. Equivalently to (B.1), we thus have:

$$\max_{(r_i)_{i=1}^{N}} \frac{1}{N}\sum_{i=1}^{N} r_i l_i - \lambda \frac{1}{N}\sum_{i=1}^{N} r_i \log r_i \qquad \text{(B.2)}$$
$$st \quad \frac{1}{N}\sum_{i=1}^{N} r_i = 1,$$

where $l_i := l(f_\theta(x_i), y_i))$. The Lagrangian corresponding to this constrained maximization is given by:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} r_i l_i - \lambda \frac{1}{N}\sum_{i=1}^{N} r_i \log r_i - \gamma(\frac{1}{N}\sum_{i=1}^{N} r_i - 1) \tag{B.3}$$

where γ is an unconstrained Lagrangian coefficient.

Following the Karush-Kuhn-Tucker conditions applied to the derivative of the Lagrangian function $\mathcal{L}$ of this problem in r_i for any given $i \in [[1, N]]$, we obtain:

$$\frac{\partial \mathcal{L}}{\partial r_i} = 0 \Leftrightarrow l_i - \lambda(\log r_i - 1) - \gamma = 0 \Leftrightarrow r_i = e^{\frac{l_i - \gamma}{\lambda} - 1} = z e^{\frac{l_i}{\lambda}} \tag{B.4}$$

with $z := e^{\frac{-\gamma}{\lambda} - 1}$.

The KKT condition on the derivative in γ gives: $\frac{\partial \mathcal{L}}{\partial \gamma} = 0 \Leftrightarrow \frac{1}{N}\sum_{i=1}^{N} r_i = 1$. Combining these two results, we thus obtain:

$$\frac{1}{N}\sum_{i=1}^{N} r_i = \frac{1}{N}\sum_{i=1}^{N} z e^{\frac{l_i}{\lambda}} = 1 \Leftrightarrow z = \frac{N}{\sum_{i=1}^{N} e^{\frac{l_i}{\lambda}}}$$

Which again gives, reinjecting this result in Equation (B.4):

$$r_i = N \frac{e^{\frac{l_i}{\lambda}}}{\sum_{j=1}^{N} e^{\frac{l_j}{\lambda}}}$$

This leads to the form of a Boltzmann distribution, which proves the result.

B.1.2 Application to VAE and Relation with Skew-Fit

As introduced in section 3.3.4, Skew-Fit resamples training data points from a batch $\{x_i\}_{i=1}^n$ using a skewed distribution defined, for any sample x in that batch, as:

$$\begin{aligned} p_{\text{skewed}}(\mathbf{x}) &\triangleq \frac{1}{Z_\alpha} w_{t,\alpha}(\mathbf{x}), \\ Z_\alpha &= \sum_{i=1}^{n} w_{t,\alpha}(\mathbf{x}_i), \end{aligned} \tag{B.5}$$

where $w_{t,\alpha}$ is an importance sampling coefficient given as:

$$w_{t,\alpha}(x) := p_{\theta,\phi}(x)^\alpha, \quad \alpha < 0, \tag{B.6}$$

with $p_{\theta,\phi}(x)$ the generative distribution of samples x given current parameters (θ, ϕ).

Applied to a generative model defined as a VAE, we have:

$$p_{\theta,\phi}(x) = \mathbb{E}_{z\sim q_\phi(z|x)} \frac{p(z)p_\theta(x|z)}{q_\phi(z|x)} dz,$$

with where $p(z)$ is the prior over latent encodings of the data x, $p_\theta(x|z)$ is the likelihood of x knowing z and $q_\phi(z|x)$ the encoding distribution of data points. As stated in Section 5.3, this can be estimated on a set of m samples for each data point using the log-approximator: $\tilde{\mathcal{L}}_{\theta,\phi}(x) = \log \sum_{j=1}^{M} \exp(\log p_\theta(x|z^j) + \log p_\theta(z^j) - \log q_\phi(z^j|x)) - \log(M)$.

Thus, this is equivalent as associating any i from the data batch with a weight r_i defined as:

$$\begin{aligned} r_i := p_{\text{skewed}}(\mathbf{x_i}) &= \frac{1}{Z_\alpha} e^{\alpha \tilde{\mathcal{L}}_{\theta,\phi}(x_i)}, \\ Z_\alpha &= \sum_{j=1}^{n} e^{\alpha \tilde{\mathcal{L}}_{\theta,\phi}(x_j)}, \end{aligned} \tag{B.7}$$

for any $\alpha < 0$. Setting $\alpha = -\frac{1}{\lambda}$, we get $p_{\text{skewed}}(\mathbf{x_i}) \propto e^{-\tilde{\mathcal{L}}_{\theta,\phi}(x_i)/\lambda}$, for any temperature $\lambda > 0$. Reusing the result from Section B.1.1, this is fully equivalent to the analytical closed-form solution of DRO when applied to $-\tilde{\mathcal{L}}_{\theta,\phi}(x_i)$ as we use in our DRO-VAE approach. Using p_{skewed} with $\alpha = -\frac{1}{\lambda}$ for weighting training points of a VAE thus exactly corresponds to the non-parametric version our DROIDE algorithm.

B.2 Methods details

B.2.1 VAE training schedule

During the first 100k steps of the agent, we train the VAE every 5k steps on 50 minibatch of 100 examples, afterward, we train it every 10k steps.

The following other schedules have been experimented, each getting worse results:

1. During the first 100k steps of the agent, train the VAE every 10k step on 50 minibatch of 100 states, afterward, train it every 20k steps.
2. During the first 50k steps of the agent, train the VAE every 5k step on 50 minibatch of 100 states, afterward, train it every 10k steps.
3. During the first 50k steps of the agent, train the VAE every 2k step on 50 minibatch of 100 states, afterward, train it every 5k steps.

B.2.2 Hyperparameters

The hyper-parameters of our DROIDE algorithm are given in Table B.1. RIG and SKEW-FIT share the same hyperparameters, except for the skewing temperature in SKEW-FIT, which we set to $\alpha = -1$, following the grid search: $\alpha \in [-50, -10, -5, -1, -0.5, -0.1]$.

DROIDE Hyper-Parameters	**Symbol**	**Value**
β-**VAE**		
Latent dim	d	[maze env :2]
Prior distribution	$p(z)$	$\mathcal{N}(0, I_d)$
Regularization factor	β	2
Learning rate	ϵ	2e-3
CNN channels		(3, 32, 64, 128, 256)
Dense layers		(512,128)
Activation Function		ReLu
Kernel size		4
Training batch size	n	100
Optimization interval (in agent steps)		10e3
Nb of training steps		50
DRO Weighter r_ϕ		
Learning rate	ϵ	2e-6
Convolutional layers channels		(3, 32, 64, 128, 256)
Dense layers		(512,128)
Activation Function		ReLu
Temperature	λ	-1.0
Optimization interval (in agent steps)		10e3
Nb of training steps		10

Table B.1: DDPG parameters

DDPG Hyper-Parameters	**Value**
Batch size for replay buffer	2000
Discount factor γ	0.98
Action L2 regularization	0.1
(Gaussian) Action noise max std	0.1
Warm up steps before training	2500
Actor learning rate	1e-3
Critic learning rate	1e-3
Target network soft update rate	0.05
Actor & critic networks activation	ReLu
Actor & critic hidden layers sizes	512^3
Replay buffer size	nb training steps