Automatic Curriculum Learning for Developmental Machine Learners

Rémy Portelas



Automatic Curriculum Learning for Developmental Machine Learners

---

By **Rémy PORTELAS**

Under the co-supervision of **Katja HOFMANN** and **Pierre-Yves OUDEYER**

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

University of Bordeaux

Graduate school of Mathematics and Computer Science

Major in Computer Science

Submitted on December 11, 2021. To be defended on February 11, 2022.

Composition of the jury:

Pr. Ann NOWÉ	Research Director	VUB University	Rapporteur
Pr. Tim ROCKTÄSCHEL	Research Scientist & Associate Pr.	FAIR & UCL	Rapporteur
Pr. Peter STONE	Full Professor	Univ. of Texas	Examinateur
Dr. David HA	Research Scientist	Google Brain	Examinateur
Pr. David FILLIAT	Professor	U2IS Ensta Paris	Président
Dr. Pierre-Yves OUDEYER	Research Director	INRIA	Co-directeur de thèse
Dr. Katja HOFMANN	Senior Principal Researcher	Microsoft Research	Co-directeur de thèse

Génération automatique de curriculum pour apprenants artificiels

Résumé : Un objectif de longue date du domaine de l'*intelligence artificielle* (IA) et de l'*apprentissage machine* (ML) est de concevoir des agents autonomes capables d'interagir efficacement avec notre monde. Dans cette optique, inspirés par le caractère interactif de l'apprentissage humain et animal, plusieurs axes de travaux se sont concentrés sur l'élaboration d'agents de décision incarnés dans des environnements réels ou virtuels. En moins d'une décennie, l'*apprentissage par renforcement profond* (DRL) s'est imposé comme l'un des ensembles de techniques les plus puissants pour créer de tels agents autonomes. Le DRL est basé sur la maximisation de fonctions de récompense définies par des experts, qui guident l'apprentissage d'un agent vers une tâche ou un ensemble de tâches cible prédéfini. En parallèle, le domaine de la *robotique développementale* travaille à la modélisation des théories du développement cognitif et à leur intégration dans des robots réels ou simulés. Un concept central développé dans cette littérature est la notion de *motivation intrinsèque* : les robots développementaux explorent et interagissent avec leur environnement selon des objectifs autosélectionnés, suivant un *apprentissage non borné*.

Récemment, des idées similaires d'auto-motivation et d'apprentissage non borné ont commencé à se développer dans le domaine du DRL, tandis que la communauté de la robotique développementale a commencé à considérer des méthodes DRL dans leurs modèles. Nous proposons d'appeler cette convergence de travaux *apprentissage machine développementale* (Developmental ML). Ce regroupement de travaux porte sur la construction d'agents autonomes incarnés et équipés de mécanismes de motivation intrinsèque façonnant des trajectoires d'apprentissage non borné. Ce manuscrit de thèse se concentre sur l'étude d'un bloc algorithmique de base de tels apprenants : les méthodes de *génération automatique de curriculum* (ACL). Les algorithmes ACL façonnent les trajectoires d'apprentissage des agents en les confrontant à des tâches adaptées à leurs capacités. Ces dernières années, ces méthodes ont été utilisées pour améliorer la vitesse et la qualité d'apprentissage d'agents autonomes, pour organiser l'exploration, pour encourager la généralisation, ou pour résoudre des problèmes à récompenses éparses, entre autres.

Ce manuscrit présente les contributions suivantes dans le domaine de l'ACL. En premier lieu, nous formalisons le problème d'ACL et examinons les méthodes ACL utilisées dans la littérature. Nous présentons ensuite une série d'expériences computationnelles dans des environnements virtuels. Tout d'abords, nous étudions l'application d'une méthode ACL existante (AMB), basée sur la détection empirique du *progrès d'apprentissage* (LP) pour organiser la sélection de buts de manipulation centrés sur un nombre *discret* d'objets. Inspirés par cette première étude, nous présentons un nouvel algorithme basé sur le LP (ALP-GMM), capable d'optimiser la sélection de tâches dans un espace de tâches *continu* pour des agents DRL apprenants des politiques de déplacements robustes. Ensuite, nous identifions que la recherche en ACL est entravée par l'absence d'une plateforme de test standardisé. Pour remédier à ce manque, nous présentons *TeachMyAgent*, une plateforme pour comparer et caractériser facilement les approches ACL existantes. Dans notre prochaine étude, nous considérons le problème suivant : comment guider efficacement l'apprentissage d'un *ensemble* d'apprenants, plutôt qu'un *unique* agent. Nous exposons les limites de l'ACL dans ce cas de figure et proposons le concept de méta-ACL, i.e. des algorithmes cherchant à généraliser la génération de curriculum à plusieurs apprenants. Nous présentons AGAIN, un premier algorithme de méta-ACL. Enfin, alors que les contributions précédentes étudient l'apprentissage de politiques de manipulation ou navigation, l'objectif à long terme du ML est de créer des pairs artificiels *socialement compétents*. Nous proposons d'élargir les recherches actuelles dans ce domaine. Pour ce faire, nous présentons une version préliminaire de *SocialAI*, une suite d'environnements pour étudier l'acquisition d'un large éventail de compétences sociales.

Malgré des succès impressionnants en apprentissage supervisé (e.g. traitement de l'image), l'utilisation à grande échelle et dans le monde réel d'agents artificiels incarnés capable d'apprentissage non borné reste encore à venir. Les travaux présentés dans ce manuscrit visent à contribuer à la création de tels agents en étudiant comment guider leur apprentissage de manière autonome et efficace.

Mots-clés : apprentissage machine développemental, apprentissage par renforcement, apprentissage profond, génération automatique de curriculum, motivations intrinsèques, intelligence artificielle

Unité de Recherche
INRIA, 33000 Bordeaux, France.

Automatic Curriculum Learning for Developmental Machine Learners

Abstract: A long-standing goal of Machine Learning (ML) and AI at large is to design autonomous agents able to efficiently interact with our world. Towards this, inspired by the interactive nature of human and animal learning, several lines of works focused on building decision-making agents embodied in real or virtual environments. In less than a decade, Deep Reinforcement Learning (DRL) established itself as one of the most powerful set of techniques to train such autonomous agents. DRL is based on the maximization of expert-defined reward functions that guide an agent's learning towards a predefined target task or task set. In parallel, the developmental robotics field has been working on modelling cognitive development theories and integrating them into real or simulated robots. A core concept developed in this literature is the notion of *intrinsic motivation*: developmental robots explore and interact with their environment according to self-selected objectives in an open-ended learning fashion.

Recently, similar ideas of self-motivation and open-ended learning started to grow within the DRL field, while the developmental robotics community started to consider DRL methods into their developmental systems. We propose to refer to this convergence of works as *Developmental Machine Learning*. Developmental ML regroups works on building embodied autonomous agents equipped with intrinsic-motivation mechanisms shaping open-ended learning trajectories. The present research focuses on proposing and assessing the performance of a core algorithmic block of such developmental machine learners: *Automatic Curriculum Learning* (ACL) methods. ACL algorithms shape the learning trajectories of agents by challenging them with tasks adapted to their capacities. In recent years, they have been used to improve sample efficiency and asymptotic performance, to organize exploration, to encourage generalization or to solve sparse reward problems, among others.

This thesis makes the following contributions to the ACL field: first, we formalize the ACL problem and survey ACL methods used in Developmental ML works. Then, this manuscript presents experiments on applying a *learning progress*-based ACL method to population-based agents learning tool-use affordances. This work, which focuses on applying ACL to select tasks from a discrete task set (for population-based agents) can be seen as a preliminary step regarding our third contribution, which presents ALP-GMM, an ACL algorithm suited to train DRL agents in continuous task spaces. ALP-GMM is based on the modeling of absolute learning progress with Gaussian mixture models. We showcase the performance advantages of ALP-GMM over other learning progress-based ACL approaches using bipedal locomotion environments with parametric obstacle layouts. One shortcoming of ACL research is that development efforts happens in silos: there is currently no benchmark on which to compare existing approaches. Addressing this lack of a standardized testbed, our next contribution consists in the design and release of *TeachMyAgent*, a benchmark to easily compare and characterize existing and new ACL approaches. We then use *TeachMyAgent* to conduct a comparative study of representative existing approaches. The two previously mentioned contributions focus on ACL approaches which train a *single* agent over a given task space. We argue that this approach is suboptimal if multiple agents are to be trained, as it would imply to repeat a costly tabula rasa exploration of the task space. As such, we introduce the concept of Meta-ACL, i.e. algorithms seeking to generalize curriculum generation to multiple learners. We present AGAIN, a first algorithmic instantiation of Meta-ACL, and showcase its benefits for curriculum generation over classical ACL in parametric locomotion environments. Finally, while previous contributions explored how to efficiently guide learners in pure manipulation or navigation task spaces, which is already challenging, the long-term goal of ML is to create artificial peers able to interact with us, i.e. building *socially proficient agents*. In our last contribution, as a first step, we propose to expand current research to consider a broader set of social skills. To do this, we present *SocialAI*, a suite of grid-world social environments to study the acquisition of social skills for DRL agents.

Despite impressive successes in traditional supervised learning scenarios (e.g. image classification), large-scale and real-world applications of embodied machine learners are yet to come. The present research aims to contribute towards the creation of such agents by studying how to autonomously and efficiently train them up to proficiency.

Keywords: Developmental Machine Learning, Deep Reinforcement Learning, Automatic Curriculum Learning, Machine Learning, Intrinsic Motivation

Équipe Flowers, INRIA, 33000 Bordeaux, France.

Résumé Long

Génération automatique de curriculum pour agents artificiels : vers l'apprentissage machine développemental

Un objectif de longue date dans le domaine de l'*apprentissage machine* (ML) – et plus généralement de l'*intelligence artificielle* (AI) – est de concevoir des agents autonomes capables d'interagir efficacement avec notre monde. Dans cette optique, inspirés par le caractère interactif de l'apprentissage humain et animal, plusieurs axes de travaux se sont concentrés sur l'élaboration d'agents de décision incarnés dans des environnements réels ou virtuels.

En moins d'une décennie, l'*apprentissage par renforcement profond* (DRL) s'est imposé comme l'un des ensembles de techniques les plus puissants pour créer de tels agents autonomes. Le DRL repose sur deux piliers méthodologiques : 1) la maximisation de fonctions de récompense définies par des experts, qui guident l'apprentissage d'un agent vers une tâche ou un ensemble de tâches cible prédéfini et 2) l'utilisation de *réseaux de neurones profonds*, capable de générer des politiques d'action complexes.

En parallèle, le domaine de la *robotique développementale* travaille à la modélisation des théories du développement cognitif et à leur intégration dans des robots réels ou simulés. Un concept central développé dans cette littérature est la notion de *motivation intrinsèque* : les robots développementaux explorent et interagissent avec leur environnement selon des objectifs autosélectionnés, suivant un *apprentissage non borné*.

Récemment, des idées similaires d'auto-motivation et d'apprentissage non borné ont commencé à se développer dans le domaine du DRL, tandis que la communauté de la robotique développementale a commencé à considérer des méthodes DRL dans leurs modèles. Nous proposons d'appeler cette convergence de travaux *apprentissage machine développementale* (Developmental ML). Ce regroupement de travaux porte sur la construction d'agents autonomes incarnés et équipés de mécanismes de motivation intrinsèque façonnant des trajectoires d'apprentissage non borné. Ce manuscrit de thèse se concentre sur l'étude d'un bloc algorithmique de base de tels apprenants : les méthodes de *génération automatique de curriculum* (ACL). Les algorithmes ACL façonnent les trajectoires d'apprentissage des agents en les confrontant à des tâches adaptées à leurs capacités. Ces dernières années, ces méthodes ont été utilisées pour améliorer la vitesse et la qualité d'apprentissage d'agents autonomes, pour organiser l'exploration, pour

encourager la généralisation, ou pour résoudre des problèmes à récompenses éparses, entre autres.

La présente recherche vise à contribuer à la conception d'agents artificiels développementaux compétents. À cette fin, nous proposons de nous inspirer de la littérature sur la robotique développementale –notamment sur l'utilisation de systèmes de motivation intrinsèque – ainsi que des travaux récents sur le ML développemental. Nos contributions portent principalement sur la conception et l'étude d'approches ACL, c'est-à-dire d'algorithmes qui adaptent les expériences d'apprentissage d'un agent donné en fonction de l'évolution des performances de l'agent. Plus précisément, nous dérivons des algorithmes ACL basés sur le concept de progression de l'apprentissage tel que défini dans le domaine de la robotique développementale. Pour tester nos approches de manière pertinente, efficiente et reproductible, nous utilisons des environnements simulés et libre d'accès – que nous concevons ou étendons – permettant d'étudier des agents incarnés évoluant dans des mondes physiques.

Le chapitre 2 débute par une présentation du domaine de l'apprentissage par renforcement (section 2.1), organisé autour d'un cadre computationnel basé sur la maximisation de performance dans des processus de décision markoviens. Nous présentons les notions théoriques fondamentales à ce paradigme d'apprentissage statistique, notamment l'utilisation des méthodes d'apprentissage par renforcement *profond*. Nous dressons ensuite un tableau rapide du paysage de recherche autour du DRL (section 2.1.3). La seconde partie du chapitre est dédié à la présentation des enjeux et sujets de recherche principaux en robotique développementale (section 2.2.1). Nous prendrons une attention particulière à l'étude de systèmes de motivation intrinsèque basé sur le progrès d'apprentissage (section 2.2.2). Enfin, comme dernière partie et première contribution, nous présentons à la fois une formalisation générale du problème ACL (section 2.3) et un état de l'art des approches d'apprentissage automatique du curriculum pour les agents DRL (section 2.3.2).

Notre deuxième contribution est présentée dans le chapitre 3, et concerne l'entraînement d'un agent basé sur l'accumulation d'un ensemble de sous-politiques spécialisés (des réseaux de neurones) à l'aide d'une approche ACL basée sur le *progrès d'apprentissage*. En utilisant un environnement Malmo Minecraft (Johnson et al., 2016) personnalisé basé sur la navigation et la manipulation d'outils, nous montrons que notre agent intrinsèquement motivé est capable d'explorer le monde qui l'entoure et d'apprendre toutes ses interactions possibles. L'analyse détaillée de ces expériences fournit une démonstration convaincante du potentiel des approches ACL pour l'apprentissage de tâches complexes. Ce travail, qui se concentre sur l'application d'ACL pour sélectionner des tâches à partir d'un ensemble de tâches *discret* (pour des agents définis comme un ensemble de sous-politiques) peut être considéré comme une étape préliminaire par rapport à nos autres contributions expérimentales, qui visent à tirer parti de méthodes ACL pour sélectionner des tâches parmi un espace de tâches *continu* (pour des agents DRL).

Dans le chapitre 4, nous étudions comment un algorithme ACL peut permettre à un agent DRL *quelconque* de devenir performant dans l'accomplissement d'un objectif dans un large éventail d'environnements divers. Pour ce faire, nous étudions comment un tel algorithme *enseignant* peut apprendre à générer un curriculum d'apprentissage, c.-à-d. une séquence de paramètres contrôlant une génération procédurale stochastique de tâches. Parce qu'il ne connaît pas au départ les capacités de son *élève* (l'agent DRL), un enjeu

clé pour l'enseignant est de découvrir quelles tâches sont faciles, difficiles ou infaisables, et dans quel ordre les proposer pour maximiser l'efficacité de l'apprentissage. Pour y parvenir, ce problème est transformé en un problème de bandit à N bras *continu*, où l'enseignant sélectionne des tâches afin de maximiser le *progrès d'apprentissage* (LP) absolu de son élève. Nous présentons ALP-GMM, un nouvel algorithme ACL modélisant le LP absolu à l'aide de modèles de mélange Gaussien. Nous comparons ALP-GMM à d'anciennes approches ACL basées sur le LP de robotique développementale, que nous adaptons dans un contexte DRL. En utilisant des variantes paramétriques d'un environnement simulé de déplacement bipède en 2D, nous étudions leur efficacité pour personnaliser un curriculum d'apprentissage pour différents apprenants, leur robustesse à la présence de taches infaisable dans l'espace des tâches, et leur capacité à d'adapter à des espaces de tâches non linéaires et de grande dimension.

Nos travaux précédents nous permettent d'identifier que la recherche en ACL est entravée par l'absence d'une plateforme de test standardisé. Pour remédier à ce manque et soutenir le développement de méthodes ACL pour les agents DRL, notre quatrième contribution expérimentale (chapitre 5) consiste en la conception et la publication de *TeachMyAgent*, une plateforme de test pour comparer et caractériser facilement les approches ACL existantes et à venir. *TeachMyAgent* est composé d'une batterie de tests unitaires spécifiques à différentes dimensions de difficulté dans la génération d'un curriculum d'apprentissage, chacun utilisant des variantes d'un environnement paramétrique de déplacement bipède. *TeachMyAgent* est aussi constitué d'un nouvel environnement de parkour 2D, généré procéduralement, combinant la plupart des dimensions de difficulté ACL, le rendant idéal pour l'évaluation des performances globales de nos algorithmes. Nous utilisons ensuite *TeachMyAgent* pour mener une étude comparative des approches ACL existantes représentatives, que nous ajoutons à notre plateforme en les adaptant à partir de code libre d'accès, ou que nous réimplémentons entièrement.

Le chapitre 4 et le chapitre 5 présentent des contributions expérimentales sur les algorithmes ACL entraînant un *unique* agent sur un espace de tâches donné. Pour proposer un curriculum approprié, ces approches reposent sur l'exploration de l'espace des tâches pour détecter des niches de progrès au fil du temps, ce qui est un processus *tabula rasa* coûteux qui doit être effectué pour chaque nouvel agent à entraîner. Pour remédier à cette limite de l'ACL classique, comme cinquième contribution (chapitre 6), nous introduisons le concept de méta-ACL, c'est-à-dire des algorithmes cherchant à généraliser la génération du curriculum à plusieurs apprenants. Nous présentons ensuite AGAIN, une première instanciation algorithmique de méta-ACL, et présentons ses avantages pour la génération de curriculum par rapport à l'ACL classique dans plusieurs environnements simulés, y compris des environnements de parkour générés de manière procédurale avec des apprenants de morphologies variées. Étonnamment, nous montrons également que AGAIN peut surpasser l'ACL classique lorsqu'il est appliqué à un *unique* apprenant si cet agent peut-être réinitialisé.

Alors que les chapitres précédents exploraient comment guider efficacement les apprenants dans des espaces de tâches de navigation ou de manipulation pures, qui représentent à la fois des domaines déjà complexes ainsi que des plateformes de test pratique pour le DRL et l'ACL, l'objectif à long terme du ML/AI est de créer des pairs artificiels capables d'interagir avec nous, c'est-à-dire des agents *socialement compétents*. Cet objectif a motivé de nombreux travaux sur l'entrainement d'agents autonomes incarnés capable d'utiliser le

langage naturel. Cependant, dans le chapitre 7, nous soutenons que les travaux actuels se concentrent sur des interactions sociales relativement simples. À la lumière des théories de la psychologie du développement mettant l'accent sur l'importance des interactions socioculturelles pour le développement cognitif, nous soutenons que travailler à l'élaboration d'agents autonomes capable d'interagir efficacement avec des humains nécessite d'étudier un ensemble plus large d'interactions sociales et de compétences sociales. Dans un premier temps, nous proposons un ensemble de compétences sociales importantes à étudier et présentons une première version de *SocialAI*, une suite d'environnements pour évaluer l'acquisition de compétences sociales pour les agents DRL. *SocialAI* propose plusieurs environnements sociaux dans des mondes 2D discrets comprenant des agents sociaux (aux comportements prédéfinis). Nous étudions ensuite les limites d'une approche DRL récente lorsqu'elle est testée dans *SocialAI* et discutons des prochaines étapes importantes vers des agents sociaux compétents. Ce dernier travail peut être considéré comme une étape préliminaire avant d'essayer d'appliquer des méthodes ACL dans des contextes sociaux complexes, que nous laissons pour de futurs travaux.

Notre dernier chapitre propose tout d'abord une synthèse des contributions du manuscrit (section 8.1). Ensuite, à travers l'étude de travaux récents et de travaux en cours faisant suite à cette thèse, nous proposons une discussion des prochaines étapes importantes en ACL, et plus généralement en apprentissage machine développementale.

Malgré des succès impressionnants en apprentissage supervisé (e.g. traitement de l'image), l'utilisation à grande échelle et dans le monde réel d'agents artificiels incarnés capable d'apprentissage non borné reste encore à venir. Les travaux présentés dans ce manuscrit visent à contribuer à la création de tels agents en étudiant comment guider leur apprentissage de manière autonome et efficace.

Acknowledgments

What a pleasant scientific journey! Before diving into the actual content of this manuscript, I would like to dedicate these few lines to express my gratitude to the many people that contributed directly or indirectly into making my PhD an enjoyable and productive scientific ride.

My first thanks go to my supervisors, Pierre-Yves Oudeyer and Katja Hofmann, who both kindly and precisely answered the avalanche of questions I asked them, and helped me find my way within the reinforcement learning and developmental robotics fields.

I am especially grateful to colleagues within the Flowers team. To Sébastien Forestier, my former internship supervisor, with whom I spent a lot of time talking about developmental robotics and many other unrelated subjects (from cryptocurrency to gardening). To Cédric Colas, with whom I spent many hours talking about machine learning, and who made me discover jazz jams. To Benjamin Clément, with whom I share my (unusual?) sense of humor. To Grgur Kovač: our quest on making socially proficient agents is only starting! To William Schueller and his crazy multilingual puns (really, another level). To Théo Segond, who was always there to answer technical questions. To Clément Romac and Paul Germon, the two master students I had the pleasure to work with. To Alexandr Ten, Laetitia Téodorescu, Tristan Karch and Mayalen Etcheverry, also working on their own PhD adventures. I will stop here, but this list could go on: the flowers team is a large family.

Special thanks to Olivier Sigaud, for his great formalization insights, and for uploading very helpful videos about RL.

This work could not have been conducted without the unwavering support of my family. Thank you so much. Likewise, I thank my closest friends. They will recognize themselves: we spend most of our weekend evenings eating at each other's place! Thank you for the laughs, and your help in other aspects of my life.

Finally, I would like to express my deepest gratitude to Camille, with whom I shared wonderful years, and many more to come. Surely, if she did not run away while I was anxiously writing this manuscript, it's meant to last!

Contents

Chapter 1

Introduction

Contents

Since the beginning of the twentieth century and the conceptualization of *robots* in 1920 (fig. 1.1), science-fiction novels and later science-fiction movies fantasized about near futures involving man-made mechanical agents whose motor, social and cognitive abilities were on par with us. A century after the invention of the term robot, although we are definitely surrounded by so-called "smart" devices, we are still waiting for our mechanical *alter ego*. Why ?

Figure 1.1: The term *robot* was invented by Josef Čapek for the theater play *R.U.R* (Rossum's Universal Robots, 1920) created by his brother Karel Čapek. Image: BBC adaptation of *R.U.R*, 1938.

Hand-wiring an adult brain including all its intricacies appears as a daunting task. Perhaps a safer route towards artificial cognition is to craft a simpler agent and equip it with learning mechanisms so that it can grow up to proficiency. This intuition is nothing new. Inspired by both theoretical and computational studies related to this perspective, this thesis aims to contribute towards creating proficient machine learners by pairing them with developmental mechanisms able to autonomously shape their learning trajectories. In this introductory chapter, we will present core theories from developmental sciences and briefly discuss how the developmental robotics field is interested in modeling them. We will also introduce the (deep) reinforcement learning field, which constitutes a more application-oriented path towards designing agents able to learn complex behaviors. We will close this chapter with an outline of our contributions – which lie at the intersection of both

fields – and provide details on the structure of this document. Figure 1.2 synthetically situates the focus of this thesis – building and studying *developmental machine learners* – within related research fields.

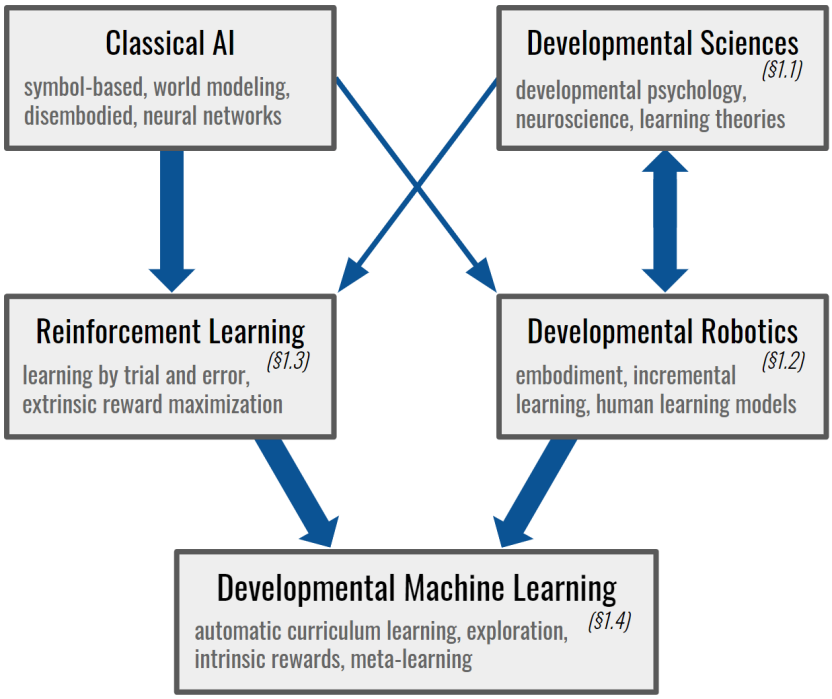


Figure 1.2: **Developmental Machine Learning** refers to a subfield of Artificial Intelligence research which aims to leverage evidence from developmental sciences to design embodied, intrinsically motivated, and incrementally learning agents. Such agents are often implemented as a combination of powerful deep reinforcement learning methods and intrinsic motivation modules to guide the learning process.

1.1 Human Learning and Intrinsic Motivation

Humans are incredible learners. In less than a decade, children go from fully dependent beings, unable to accomplish anything more than crying for care, to intelligent little humans, able to think, move and communicate very efficiently. How do they reach this level of cognition so fast?

While there is currently no holistic theory that reached a scientific consensus, an influential perspective on how such a developmental revolution might unfold is Jean Piaget's foundational theories of cognitive development. Overall, the *Piagetian* perspective on child development influenced countless works in developmental psychology and cognitive science at large, philosophy, education, and artificial intelligence. As of July 2021, more than 1.1 million scientific articles mention Piaget (search done on Google Scholar).

In his proposed 6 stages of a child development, Piaget (1952) exposes a broad and compelling theory of development ranging from the first days of infancy up to early adolescence. For Piaget, the child is a solitary thinker, whose cognitive proficiency is

mainly due to the coupling of biological maturation and their active exploration of their sensorimotor world. The child is a little scientist deciding which experiments to perform to challenge their assumptions and refine their representation of the world. Importantly, Piaget stresses that these experiments are self-selected without the intervention of a social peer, i.e. they are *intrinsically motivated*.

The idea that humans and animals are intrinsically motivated to explore and interact with their environment has been extensively studied in psychology (White, 1959; Berlyne, 1960, 1966; Deci & Ryan, 1985; Loewenstein, 1994; Bazhydai et al., 2021). As exposed by White (1959), this intrinsic motivation is not a mere by-product of primary drives, i.e. a way to find food or to reduce anxiety, but is a parallel cognitive component in itself. White based his theory on the analysis of multiple experimental studies on animal and humans (including works from Piaget). For instance, to assert his claims regarding animals, White discusses multiple interesting lab studies on rhesus monkeys from Butler (1953) and Butler & Harlow (1957). In these experiments, authors showed they could motivate their monkeys to perform color differentiation tasks based on the sole reward of opening a window "which permitted them to look out upon the normal comings and goings of the entrance room to the laboratory". Although these monkeys were usually rewarded with food, pure visual exploration appeared to be another source of pleasure justifying their compliance.

In addition to surveying multiple works on psychology, White perfectly captures this notion of intrinsic motivation for us, humans, in this couple of sentences:

> *"Boredom, the unpleasantness of monotony, the attraction of novelty, the tendency to vary behavior rather than repeating it rigidly, and the seeking of stimulation and mild excitement stand as inescapable facts of human experience and clearly have their parallels in animal behavior. We may seek rest and minimal stimulation at the end of the day, but that is not what we are looking for the next morning. Even when its primary needs are satisfied and its homeostatic chores are done, an organism is alive, active, and up to something."* – White (1959)

For White, this motivation is directed towards *learning to interact efficiently with the environment* at large, e.g. visual exploration, locomotion, object manipulation, language and thinking or producing causal effects on the environment. White appropriately proposed to use the term *competence* to refer to this broad set of activities. For him, Humans and higher animals are intrinsically motivated to improve their *competence*, and it is this mechanism rather than the constant search to satiate our primary drives that is at the heart of our cognitive development and complexity.

At around the same period, The psychologist Berlyne further refined this notion by stating that intrinsic motivation – a.k.a. curiosity – can be seen as a "quest for intermediate arousal potential"(Berlyne, 1960). Humans and higher animals are actively seeking stimulations offering "an optimum amount of novelty, surprisingness, complexity, change, or variety"(Berlyne, 1966). Berlyne's theories are tightly related to Csikszentmihalyi's theory of *flow*, which states that "enjoyment appears at the boundary between

boredom and anxiety, when the challenges are just balanced with the person's capacity to act"(Csikszentmihalyi, 1991).

1.2 From Theory to Practice: Developmental Robotics

At the crossroad between the twentieth and the twenty-first century, the *developmental robotics* field emerged as a multidisciplinary group of researchers from robotics, artificial intelligence and developmental sciences (Weng et al., 2001; Lungarella et al., 2003; Asada et al., 2009; Cangelosi & Schlesinger, 2015). Their objective was two-fold:

- *Building autonomous robots.* Manually designing and programming a robot for a specific task is a labor-intensive operation that must be repeated for each new problem. A core goal of developmental robotics is to avoid such handcrafted approaches and instead work on endowing robots with cognitive maturation systems to foster their autonomous mental development (Weng et al., 2001). By leveraging developmental science theories, their long-term vision was to create learning robots that could be interactively trained to perform any tasks, just as humans do.
- *Studying developmental theories.* Creating such plastic robotic brains requires transforming written theories of development into precise computational models, which can be used as a powerful verification tool by developmental science scholars. Many open-questions around human cognition might be studied based on the behavioral analysis of such autonomous agents.

Towards this, multiple key aspects of human learning have been studied within this literature. In the following paragraphs, we outline three conceptual pillars organizing research in developmental robotics.

Embodied learning

Developmental robotics emphasizes the importance of embodiment as an essential component for the emergence of cognition. Cognition emerges from the development of the brain and the body, which reciprocally influence each other. This point of view departs from the idea that a body is nothing more than a mere vessel waiting for our mind to control, as advocated by traditional symbolic artificial intelligence (Newell & Simon, 1976) and cognitive science (Fodor, 1981).

Intrinsically motivated learning

Of uttermost importance for developmental robotics is the aforementioned notion of intrinsic motivation from developmental psychology (Blank et al., 2005; Oudeyer et al., 2007a). Learning must not be the result of optimizing a singular extrinsic signal, but is rather guided by the objective of improving the robot's *competence* over its world. Intrinsically motivated learning, or curiosity-driven learning, is arguably a prerequisite to the conception of open-ended agents.

Incremental learning

Just as for animals and humans, in developmental robotics, *competence* development must be incremental, by equipping autonomous agents with mechanisms to explore and learn about their environment *step by step*. By following such self-selected objectives that are not too hard nor too simple, the resulting *developmental trajectory* is supposed to simplify the mastering of complex downstream affordances w.r.t directly learning them. This concept has a natural appeal, which has long been thought through, but had yet to be tested. As a testimony to these early thinkers, one can recall a famous – and foundational – excerpt from Alan Turing's *Computing machinery and intelligence* article:

> *"Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain."* – Turing (1950)

1.3 Reinforcement Learning

A parallel path towards training autonomous agents has been studied within the Reinforcement Learning (RL) field (Kaelbling et al., 1996; Sutton & Barto, 2018). Contrary to developmental robotics, which gather works aiming to understand cognitive development, reinforcement learning is more application-oriented. Kaelbling et al. (1996) defines RL as "the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment". The range of considered "dynamic environments" by RL is large, and includes both learning to control mobile entities (e.g. video games, vehicles, robots) and disembodied scenarios (e.g. board games, finance). In each case, experimenters must provide a *reward signal* – extrinsic to the agent – which guides learning towards the desired target behavior (e.g. follow the road, run forward, win the game).

The concept of reinforcement learning dates back to the very beginnings of artificial intelligence (Widrow & Hoff, 1960; Van Der Malsburg, 1986), and was heavily inspired by trial-and-error learning theories from psychology, such as Thorndike's *Law of Effect*:

> *"Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond."* – Thorndike (1911)

Building artificial agents using trial-and-error learning is an old idea. According to Sutton & Barto (2018), one of the earliest attempts was an electromechanical maze

solving machine from Ross (1933), which was able to find its way out of simple mazes and remember its path by using switches as memory. Despite initial successes, RL was somewhat left aside in its infancy, to favor studies such as knowledge-based symbolic reasoning and other supervised learning approaches. Among others, Harry Klopf (1972; 1975; 1982) was one of the main researcher who highlighted the importance of building agents that learn to interact in environments, leading to a resurgence of RL works in early 1980 (Sutton & Barto, 2018).

Deep Reinforcement Learning

In the last decade, after astonishing successes in the realm of image classification (Krizhevsky et al., 2012; LeCun et al., 2015), the now famous *deep* neural networks, i.e. large multi-layered networks, started to be integrated into RL algorithms, enabling the processing of high-dimensional raw observations, such as pixels. The memorable starting point of this rapidly growing *Deep* RL field (DRL) is the development of the Deep Q Network algorithm (DQN) (Mnih et al., 2013, 2015). In these works, Mnih and colleagues showed that replacing the state-action value table of a classical RL method (Q-LEARNING) with approximated values inferred by a deep neural network enabled their agent to learn human-level action policies in multiple Atari 2600 games (Bellemare et al., 2013).

1.4 Towards Developmental Machine Learners

In the recent years, modern DRL approaches established themselves as state of the art autonomous decision-making systems. Although deep learning and hardware innovations (e.g. GPU computing) were decisive factors for this performance leap, we argue that another core ingredient was the integration of algorithmic components inspired from developmental sciences and developmental robotics. Salient examples include the range of works on designing (intrinsic) exploration bonuses (Bellemare et al., 2016; Pathak et al., 2017; Tang et al., 2017; Pathak et al., 2019; Shyam et al., 2019; Burda et al., 2019a; Raileanu & Rocktäschel, 2020), or the recent success of OpenAI to train a robotic hand to manipulate and solve a Rubik's cube, which was done using automatic curriculum learning strategies (OpenAI et al., 2019).

Parallel to this, the developmental robotics field started to integrate deep reinforcement learning models into their experiments (Laversanne-Finot et al., 2018; Kovač et al., 2020; Barros et al., 2020; Kim et al., 2020). Goal-conditioned DRL approaches have recently been successfully trained to learn repertoires of skills (i.e. a form of open-ended learning) using learning progress as intrinsic signals (Colas et al., 2019, 2020a,b).

Besides, the deep reinforcement learning field is currently shifting its focus from traditional single-task learning to problems closer to open-ended scenarios (e.g. generalization, multi-task). The frontier between both fields has become somewhat blurry, and as such we – and other members of the flowers team (Colas et al., 2020b) – propose to denote this intertwined breadth of works jointly as *developmental machine learning*.

1.4.1 Objectives and Contributions

The present research aims to contribute to the design of proficient developmental machine learners. Towards this, we propose to draw inspirations from the developmental robotics literature – along with recent works on developmental ML – and apply them to catalyze the learning of DRL agents.

Our contributions are mainly focused on the design and study of *Automatic Curriculum Learning* (ACL) approaches, i.e. algorithms that adapt the learning experiences of a given agent as a function of the agent's evolving performances. More specifically, we derive ACL algorithms based on the concept of learning progress as defined in the developmental robotics field (Lopes & Oudeyer, 2012a). To test our approaches in relevant, tractable, and reproducible ways, we use open-source simulated environments – that we either design or extend – featuring embodied agents evolving in physical worlds.

Chapter 2 presents important concepts and lines of works from DRL and developmental robotics. As our first contribution, we present both a general formalization of the ACL problem and a survey of existing automatic curriculum learning approaches for DRL agents (section 2.3.2).

Our second contribution is presented in chapter 3, and concerns the design of a population-based agent (using shallow neural networks) trained with an existing learning progress based ACL approach. Using a custom Malmo Minecraft (Johnson et al., 2016) tool-use environment, we show that our intrinsically motivated agent is able to explore its surrounding world and learn all possible interactions. The detailed analysis of these experiments provides a compelling demonstration of the potential of ACL approaches for learning complex tasks. This work, which focuses on applying ACL to select tasks from a *discrete* task set (for population-based agents) can be seen as a preliminary step regarding our other experimental contributions, which aim to leverage ACL to sample tasks in *continuous* task spaces (for DRL agents).

Our third contribution, central to our work, is the design of ALP-GMM, a new ACL algorithm modeling absolute learning progress with Gaussian mixture models, which is presented in chapter 4. We compare ALP-GMM to previous learning progress based approaches from developmental robotics in a DRL context. Using parametric variants of the BipedalWalker environment (Song et al., 2018), we study their efficiency to personalize a learning curriculum for different learners, their robustness to the ratio of learnable/unlearnable environments, and their scalability to non-linear and high-dimensional task spaces.

In chapter 5, to support the development of ACL methods for DRL agents, our fourth experimental contribution consists in the design and release of *TeachMyAgent*, a benchmark to easily compare and characterize existing and new ACL approaches. *TeachMyAgent* includes 1) challenge specific unit-tests using parametric BipedalWalker environments, and 2) a new procedural Parkour environment combining most ACL challenges, making it ideal for global performance assessment. We then use *TeachMyAgent* to conduct a comparative study of representative existing approaches.

Chapter 4 and chapter 5 present experimental contributions about ACL algorithms training a *single* agent over a given task space. To propose an appropriate curriculum, these approaches rely on exploring the task space to detect progress niches over time, which

is a costly tabula rasa process that needs to be performed for each new learning agent. To address this limitation, in chapter 6, as our fifth contribution, we introduce the concept of *meta automatic curriculum learning*, i.e. algorithms seeking to generalize curriculum generation to multiple learners. We then present AGAIN, a first instantiation of meta-ACL, and showcase its benefits for curriculum generation over classical ACL in multiple simulated environments, including procedurally generated parkour environments with learners of varying morphologies. Surprisingly, we also show that AGAIN can outperform classical ACL when applied to a *single* resetable learner.

While previous chapters explored how to efficiently guide learners in pure navigation or locomotion task spaces, which is both an already challenging domain and a convenient testbed for DRL and ACL, the long-term goal of ML/AI is to create artificial peers able to interact with us, i.e. socially proficient agents. This objective motivated multiple works on embodied language use. However, in chapter 7, we argue that current works focus on relatively simple social interactions. In light of developmental psychology theories emphasizing the importance of socio-cultural interactions for cognitive development, we argue that aiming towards human-level AI requires studying a broader set of social interactions and social skills. As a first step towards this, we propose a set of important social skills to study, and present *SocialAI*, a suite of environments to study the acquisition of social skills for DRL agents. *SocialAI* features multiple grid-world environments including (scripted) social agents. We then study the limits of a recent DRL approach when tested on *SocialAI* and discuss important next steps towards proficient social agents. This last work can be seen as a preliminary step before trying to apply ACL in complex social settings, which we leave for future work.

1.4.2 Collaborations

The research projects presented in this manuscript are the result of collaborations with multiple researchers and organizations. Pierre-Yves Oudeyer (INRIA) and Katja Hofmann (Microsoft), my supervisors, were involved in every aspect of it. Sébastien Forestier (Massa Labs), my former supervisor in a previous internship within the team, is the first author of a large research project on intrinsically motivated agents, which is partly composed of the study presented in chapter 3. Cédric Colas (INRIA), a fellow PhD student at this time, is a colleague with whom I worked on the formalization and survey of ACL (section 2.3). Other notable mentions on this project include Lillian Weng (OpenAI), who participated in the writing of the survey, and Olivier Sigaud (ISIR), who provided important insights to refine our formalization. Cédric also participated in the discussion leading to the development of the ALP-GMM algorithm, and was significantly involved in the writing of the associated publication, of which he is co-author. I also had the chance to supervise two master students: Clément Romac and Paul Germon. Clément is the co-first-author of the *TeachMyAgent* benchmark (chapter 5). The subject of Paul's internship was the design of an interactive demonstration of DRL and ACL, which we presented at a science fair (Cap Sciences Bordeaux) in October 2021. Finally, Grgur Kovač and I equally contributed to the development of the *SocialAI* environments (and are currently working on an update).

Most of the experimental studies presented in this paper were carried out using 1) the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université

de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine, 2) the computing facilities MCIA (Mésocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l'Adour, and 3) the HPC resources of IDRIS under the allocation 2020-[A0091011996] made by GENCI. This work was supported by Microsoft Research through its PhD Scholarship Programme.

1.4.3 Publications

Part of the material presented in this manuscript has been presented in the following articles:

Journal

- Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. *Accepted at JMLR 2022 with minor revisions* (Forestier et al., 2017). *Second author.* In chapter 3.

Conferences

- Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments. *CoRL 2019* (Portelas et al., 2019). In chapter 4.
- Automatic Curriculum Learning For Deep RL: A Short Survey. *IJCAI 2020* (Portelas et al., 2020a). In chapter 2.
- TeachMyAgent: a Benchmark for Automatic Curriculum Learning in Deep RL. *ICML 2021* (Romac et al., 2021). *Co-first-author.* In chapter 5.

Workshops / Pre-prints

- Meta Automatic Curriculum Learning. *Pre-print* (Portelas et al., 2020c). In chapter 6.
- SocialAI: Benchmarking Socio-Cognitive Abilities in Deep Reinforcement Learning Agents. *Pre-print* (Portelas et al., 2021). In chapter 7.
- SocialAI 0.1: Towards a Benchmark to Stimulate Research on Socio-Cognitive Abilities in Deep Reinforcement Learning Agents. *ViGiL workshop, NAACL 2021* (Kovac et al., 2021). *Co-first-author.* In chapter 7.
- Trying Again Instead of Trying Longer: Prior Learning for Automatic Curriculum Learning. *BeTR-RL workshop, ICLR 2020* (Portelas et al., 2020b). In chapter 6.

Chapter 2

Background

Contents

Before diving into our experimental contributions, this chapter will introduce core concepts and briefly review both the (deep) reinforcement learning (section 2.1) and developmental robotics (section 2.2) fields. We will then propose a formalization of the automatic curriculum learning problem (section 2.3.1), which will form the basis on which to frame teacher-student interactions in chapters 4, 5 and 6. Finally, we will provide a survey of automatic curriculum learning works applied to deep reinforcement learning agents (section 2.3.2), which are works most related to the present research.

2.1 Reinforcement Learning

This section presents core concepts from the reinforcement learning literature (section 2.1.1 and 2.1.2), up to recent DRL algorithms. We focus especially on theory related to actor-critic agents, a popular learning architecture, used in all our DRL experiments (chapters 4, 5, 6 and 7). We then provide a brief overview of existing DRL approaches and relevant application areas (section 2.1.3).

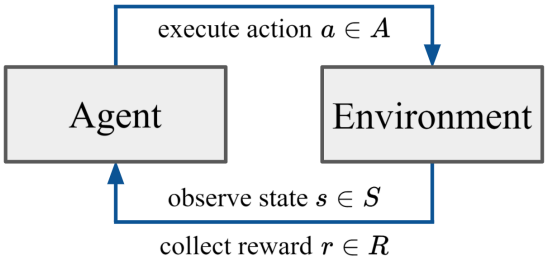


Figure 2.1: Interaction pipeline in a reinforcement learning problem.

2.1.1 Problem Formalization: Markov Decision Processes

Reinforcement learning characterizes a family of problems in which an agent must learn to maximize its cumulative reward collection by refining its behavior over multiple interaction steps with a given environment (see figure 2.1), e.g. learning to escape from a maze. Such problems are usually called *tasks* and are often formalized as Markov Decision Processes (MDPs) of the form $\tau = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0 \rangle$ where:

- $\mathcal{S}$ is the state space.
- $\mathcal{A}$ is the action space.
- $\mathcal{P} : S \times A \times S \rightarrow [0, 1]$ is a stochastic transition function characterizing the probability of switching from the current state s to the next state s' given action a.
- $\mathcal{R} : S \times A \rightarrow \mathbb{R}$ is a reward function.
- ρ_0 is a distribution of initial states.

Within this framework, the behavior of the agent can be formalized as an action *policy* $\pi : S \rightarrow A$. More precisely, as in most DRL works, we consider MDP problems with finite horizon, i.e. *episodic* reinforcement learning problems, in which interactions with the environment are split into multiple independent sequences.

The performance objective of a policy π in such an episodic MDP can be formalized as the maximization of the cumulative sum of expected rewards:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [\gamma^t r(s_t, a_t)], \tag{2.1}$$

with T the maximal number of steps in an episode, and $\gamma \in [0; 1]$ the discount factor, which controls the importance of future reward w.r.t immediate ones. Theoretically, the reward maximization objective of an episodic MDP does not feature a discount factor γ, since the interaction horizon is pre-defined and finite, which makes current and future reward equally important. In practice, episodic tasks considered in the RL and DRL literature, including in the present research, can have variable episode length, either due to abortion rules (e.g. terminate episode if robot falls) or success conditions (e.g. terminate episode if task is completed). Using discounted factors in such scenarios provides a principled way to promote faster task completion.

Terminology: tasks, goals, environments

Here and thereafter, we use the term *task* to denote a learning problem defined as a MDP, for instance a specific maze to solve or a specific robotic manipulation task to perform. We use the term *goal* to refer to a specific objective that must be attained in a given MDP, such as moving a specific robotic arm from position A to position B. More precisely, the reward function of the MDP is conditioned on the chosen goal, which augments the MDP with a goal space $\mathcal{G}$ (Schaul et al., 2015). Note that, given these definitions, a set of *goals* can be understood as a particular set of *tasks* identical in all aspects of their MDPs apart from their goal space $\mathcal{G}$.

The term *environment* is used to refer to an experimental testbed, e.g. RL agents are tested in various environments (e.g. mazes, physics simulation). In addition, we extend the general meaning of this term to more technical grounds: *tasks* only differing by their state space $\mathcal{S}$ – e.g escaping from a set of mazes – will be denoted as *environments*.

2.1.2 Solutions: (Deep) Reinforcement Learning Algorithms

Given full knowledge over the transition and reward functions $\mathcal{P}$ & $\mathcal{R}$ of a given MDP, the optimal policy can be analytically computed using *dynamic programming* methods (Bellman, 1957). Dynamic programming has long been successfully applied to real-world problems, with a predominant area of application being decision-making systems for resource management, e.g. controlling industrial power-grids, water systems, optimal marketing decisions (White, 1969, 1985, 1988). Dynamic programming is not easily applicable to the range of problems related to training autonomous embodied agents. Two important factors accounting for this limitation are 1) assuming access to exact models for embodied environments featuring complex physics and other agents is often unrealistic, and 2) Even when such models are available, Dynamic Programming computation complexity grows exponentially with the number of state variables in a given model, making it quickly intractable.

Reinforcement learning algorithms refer to a collection of methods able to solve MDP problems with unknown transition and reward functions through trial-and-error learning. One can separate RL algorithms in two broad categories: *Model-free* RL methods, which directly learn policies from interacting within a given environment, and *Model-based* RL methods, which learn models of their environment (or use existing ones) to produce policies through combinations of learning and planning mechanisms.

The present research focuses on training developmental learners based on model-free RL approaches. In the recent years, The DRL field predominantly relied on model-free approaches when considering embodied control tasks (Mnih et al., 2015; Lillicrap et al., 2016; Schulman et al., 2017; Haarnoja et al., 2018b). The main reason for this choice is that model-free algorithms are conceptually simpler, as there is no model learning and planning involved. Intuitively, another reason for this success might be that it is easier to learn behavior policies than to learn complex world models on which to derive policies. That being said, when properly setup, model-based approaches can yield impressive results (Schrittwieser et al., 2020), that are often more sample efficient than model-free approaches (Kaiser et al., 2020). Multiple works also showed that combining model

learning to model-free policy learners can yield powerful learning systems (Silver et al., 2016; Pong et al., 2018; Hessel et al., 2021). See Moerland et al. (2020) for a recent survey of model-based RL. Note that, although our experiments feature model-free learners, our proposed algorithmic contributions and benchmarks can easily be applied to model-based agents.

Value-Function methods

Instead of modeling and planning, Model-free RL approaches often rely on learning *value functions* to determine how likely a given *state* (or *action-state*) will lead to downstream rewards. The incremental and empirical (re-)estimation of the expected value of *states* can be expressed as:

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \lambda \cdot \delta_t, \tag{2.2}$$

$$\text{with } \delta_t = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t). \tag{2.3}$$

$V^{\pi} : S \rightarrow \mathbb{R}$ is the so-called *state-value* function of π. Given a state s_t, this function estimates the (γ-discounted) sum of future rewards if following π from state s_t onward (also called the *return* of the policy). δ_t is a retrospective measure of the estimation error of $V(s_t)$ after reaching s_{t+1}, and is commonly referred to as the *Temporal Difference error*, a.k.a. TD-error (Sutton & Barto, 2018). λ is a weighting parameter controlling the speed at which estimations are updated based on new empirical findings, i.e. the *learning rate*.

The aforementioned state-value function update rule is not sufficient to improve over a given policy π if the environment's transition function is unknown: value estimates are useful for knowing which states are interesting in terms of potential reward collection, but they do not inform on *how* to reach such states. Instead of estimating state-value functions, an alternative, from which a policy can be derived, is to use *state-action value functions* $Q^{\pi} : S \times A \rightarrow \mathbb{R}$:

$$Q^{\pi}(s_t, a_t) = \sum_{s^{t+1} \in \mathcal{S}} \mathcal{P}(s_{t+1}|s_t, a_t)\Big[r(s_t, a_t) + \gamma V^{\pi}(s_{t+1})\Big]. \tag{2.4}$$

Q^{π} is often called a *Q-value*. This is for instance what is used in the classical SARSA (Rummery & Niranjan, 1994) and Q-LEARNING (Watkins & Dayan, 1992) algorithms, which are both *value-function* RL approaches.

From RL to Deep RL: Deep Q Network

Instead of using classical tabular methods to approximate Q-values in a Q-LEARNING algorithm, multiple authors proposed to leverage *deep* (i.e. multi-layered) neural networks to deal with large continuous state spaces, such as pixels (Riedmiller, 2005; Lange & Riedmiller, 2010; Mnih et al., 2013, 2015). Mnih et al. (2015) call the resulting approach Deep Q Network (DQN). This article marked the beginning[1] of the deep reinforcement

[1]Note that Mnih et al. (2013) is not the first article to successfully use multi-layered neural networks for RL algorithms: Tesauro (1995) presented TD-GAMMON, a RL algorithm using a multi-layered neural

learning field. In DQN, instead of training their Deep Neural Network (DNN) with a classical supervised learning loss, e.g. squared distance between target value y and the network's prediction, they minimize the squared TD-error from successive Q-value estimations. More precisely, given a batch of N behavioral samples (s_i,a_i,r_i,s_{i+1}), the Q-network's weights θ can be optimized using the following loss function:

$$\begin{aligned} J_Q(\theta) &= \frac{1}{N} \cdot \sum_i (y_i - Q(s_i, a_i|\,\theta))^2, \\ \text{with } y_i &= r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'|\,\hat{\theta}). \end{aligned} \tag{2.5}$$

The target Q-value y_i is a retrospective estimation of what should have been predicted at step i. This estimation is computed as the sum of the reward r_i (obtained upon executing a_i in s_i) and the Q-value estimate on the reached state s_{i+1} if greedily following the Q function estimates for action selection. Note that, to stabilize the learning process and avoid diverging Q-values issues (Baird, 1994), Mnih et al. (2015) propose to maintain a second *target network* $\hat{Q}$, which is a periodically updated copy of Q used to compute the target Q-value estimations. To further stabilize gradient updates and to improve sample efficiency, DQN features an *experience replay buffer* (Lin, 1992), i.e. behavioral samples are stored and randomly selected to generate learning batches.

RL algorithms featuring optimization objectives independent of the policy being updated, as in DQN, are usually referred to as being *off-policy* algorithms. Using a replay buffer, such approaches can update their current policies based on experiences collected from other policies (e.g. previous versions of the current policy). On the contrary, *on-policy* algorithms (such as SARSA) do not use replay buffers and perform gradient updates based on recent experiences collected with the policy that is being updated.

Note that in DQN, and more generally in value-function based RL methods, there is no policy that is explicitly being learned. Instead, the value function is learned, and the policy is implicitly derived from it by greedily executing the action with maximal expected return given a state s. More precisely, for each environment step, the state vector s_t is fed into the Q-network, and the action selected will be the one corresponding to the highest output value of the network. This implies that the action space must be discrete: DQN cannot be directly used for continuous action spaces (one workaround is to discretize the action space).

Policy Gradient methods

Policy gradient methods refer to RL approaches explicitly learning a parametric policy π_ϕ which can output discrete or continuous actions without consulting a value function. Value functions might still be used to train the policy, but action selection is done solely based on the policy (Sutton & Barto, 2018). Policy gradient approaches rely on iterative

network as value function estimator, reaching pro-level at the game Backgammon. This work itself was already inspired from Anderson (1986). One reason why so much time elapsed between these early milestones and the emergence of DRL might be due to concurrent work showing that using multi-layered neural networks in RL can cause diverging issues in value estimations (Baird, 1994).

modifications of their policy parameters using stochastic gradient ascent so as to maximize some empirical estimator of performance, which is algorithm-dependent. For instance, in the REINFORCE algorithm (Williams, 1992), a classical policy gradient method, policy updates are performed after each episode based on the (discounted) empirical return G_t (the sum of obtained rewards from t onward):

$$\theta_{t+1} = \theta_t + \lambda G_t \frac{\Delta \pi_{\phi_t}(a_t|s_t)}{\pi_{\phi_t}(a_t|s_t)}. \tag{2.6}$$

This update rule has an intuitive appeal: If G_t is positive, i.e. following π_ϕ from state s_t was rewarding, the gradient update will increase the probability of the policy to take action a_t given s_t. If G_t is negative, i.e. following π_ϕ from state s_t resulted in negative rewards, the gradient update will reduce the probability of taking a_t given s_t. Additionally, this gradient of the probability of using action a_t in state s_t is inversely proportional to the probability of taking action a_t in state s_t, which prevents actions with high-probability to be overly favored w.r.t actions of lower probability (Sutton & Barto, 2018). Note that REINFORCE is a *Monte-Carlo* approach, i.e. to perform gradient updates it needs to complete entire episodes from which G_t is computed. Such accurate weight updates lead to algorithms with *high sample complexity*, i.e. more training episodes are required w.r.t value-function methods, which can perform gradient updates after each interaction step based on TD-error estimations.

Actor-Critic methods

Actor-critic methods refer to a set of RL algorithms at the intersection of policy gradient and value-function techniques. In essence, actor-critic methods learn a parametric behavior policy using TD-error estimates, which allows to train policies able to output continuous actions in a sample efficient manner. More precisely, actor-critic architectures are composed of a value function estimator (the *critic*) and a differentiable parametric policy (the *actor*). Both components are jointly trained through a (*Generalized*) *Policy Iteration* training scheme (Sutton & Barto, 2018), which consists of two simultaneous and interactive steps:

- *policy evaluation*, during which the value estimator (the critic) is updated, e.g. using TD-errors as in equation 2.5.
- *policy iteration*, during which the policy (the actor) is updated based on value function estimates.

Soft Actor-Critic (SAC)

The Soft Actor-Critic algorithm is a State Of The Art (SOTA) off-policy DRL method presented in Haarnoja et al. (2018a,b). It is the DRL agent that we predominantly use in the present research. Like DQN, SAC uses target policies and a replay buffer to stabilize training. Contrary to DQN, SAC is an actor-critic algorithm that can generate continuous actions, which is particularly useful to control complex robotic bodies. To

foster exploration while aiming for performance optimization, SAC modifies the classical RL objective (equation 2.1) by adding an entropy regularization term: i.e. the objective of the policy learning algorithm is to maximize both rewards and action entropy:

$$\begin{gathered}\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(.|s_t))], \\ \text{with } \mathcal{H}(\pi(.|s_t)) = \mathbb{E}_{a\sim\pi(.|s)}[-log(\pi(a|s))].\end{gathered} \tag{2.7}$$

$\mathcal{H}$ is the entropy term. Intuitively, this objective can be seen as pushing SAC agents to learn high-performing policies that are as random as possible. The amount of "randomness" is controlled by the α parameter. Towards this objective, SAC optimizes its policy network π_ϕ using an approximation of the *soft policy iteration* algorithm (Haarnoja et al., 2018b). The soft policy iteration training scheme formalizes the learning of optimal maximum-entropy policies (Ziebart et al., 2008; Haarnoja et al., 2017): "Soft" just means "entropy regularized".

In the following paragraphs, we succinctly present the optimization objectives used to train the actor and critic of the SAC architecture. More details on the approach can be found in Haarnoja et al. (2018a,b), e.g. using two critic networks to improve stability, using the re-parameterization trick to train the actor, using squashed Gaussian distributions for action sampling.

Note that equation 2.7 does not include a discount factor, although in both our experiments and in the following update equations we will consider a discounted scenario. As pointed out by SAC authors (Haarnoja et al., 2018b), writing down the precise discounted objective of SAC is complex (see their appendix A).

Policy evaluation step: updating the critic – To deal with this maximum-entropy MDP objective, SAC uses a modified critic: a *soft* Q-network Q_θ. Given the actor π_ϕ, one can express the soft Bellman update rule of Q_θ (a.k.a. the soft Bellman operator) as:

$$Q_\theta(s_t,a_t) = r + \gamma\mathbb{E}_{a\sim\pi_\phi(.|s_{t+1})}[Q_\theta(s_{t+1},a) - \alpha\log(\pi_\phi(a|s_{t+1}))]. \tag{2.8}$$

Given a replay buffer $\mathcal{D}$ of interaction data, Q_θ can then be updated with stochastic gradient approaches by minimizing the (entropy-modified) squared TD-error using the following loss function:

$$\begin{gathered}J_Q(\theta) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}[(y_t - Q_\theta(s_t,a_t))^2], \\ \text{with } y_t = r(s_t,a_t) + \mathbb{E}_{a\sim\pi_\phi(.|s_{t+1})}[Q_\theta(s_{t+1},a) - \alpha\log(\pi_\phi(a|s))].\end{gathered} \tag{2.9}$$

Policy improvement step: updating the actor – Given Q_θ, the policy π_ϕ can be optimized by greedily biasing its action distribution towards high soft Q-values. SAC implements this as the minimization of the Kullback-Leibler divergence between π_ϕ and exponentiated soft Q-values:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[KL(\pi_\phi(.|s_t)) || \frac{exp(\frac{1}{\alpha} Q_\theta(s_t, .))}{Z_\theta(s_t)} \right], \tag{2.10}$$

with $Z_\theta(s_t)$ a normalizing term to recover a probability distribution. In practice, equation 2.10 is not directly optimized for. By leveraging the differentiable nature of the target density to approach – the Q-network – authors propose to use the re-parameterization trick (see Haarnoja et al. (2018b) for details) to obtain a surrogate objective, which has the same gradient w.r.t. ϕ:

$$J_\pi(\phi) = \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi_\phi(.|s)} [\alpha \log(\pi_\phi(a|s)) - Q_\theta(s, a)] \right]. \tag{2.11}$$

2.1.3 Brief Overview of the DRL Field

As explained in chapter 1, reinforcement learning is a well established field, which dates back to the early days of AI, and has been increasingly studied since the early 1980s (Sutton & Barto, 2018). This long tradition led to foundational theoretical and algorithmic insights, such as the Q-LEARNING algorithm (Watkins & Dayan, 1992), or the policy gradient theorem (Sutton et al., 2000). Classical RL agents have been successfully applied to complex control problems, such as controlling helicopters (Ng et al., 2006), locomotion and manipulation tasks with robots (Kohl & Stone, 2004; Kormushev et al., 2010), or playing complex games at human level, e.g. Backgammon (Tesauro, 1995). The following overview focuses on the subsequent DRL contributions, which heavily (if not entirely) rely on these pioneering RL works. We refer the interested reader to Kaelbling et al. (1996) and Szepesvári (2010) for comprehensive surveys on early RL works. See figure 2.2 for a few examples of testbeds used in the DRL literature.

The aforementioned DQN architecture (Mnih et al., 2015) marks the turning point of the DRL field (Arulkumaran et al., 2017; Wang et al., 2020a) by showing that, with a single learning algorithm and a single network architecture, it was possible to learn human-level policies for Atari 2600 games (Bellemare et al., 2013). RL has long been using games as testbeds, e.g. Backgammon (Tesauro, 1995), Checkers (Samuel, 1959, 1967), Chess (Baxter et al., 2001), *Jeopardy!* (Tesauro et al., 2012, 2013) or Poker (Dahl, 2001). However, Mnih et al. (2015) showed that leveraging DNNs allows to circumvent the usual need for crafted high-level state representations by learning them from high-dimensional raw input data.

Improvements over DQN

Using Atari 2600 games as a benchmark, multiple works proposed improvements over the vanilla DQN algorithm. Van Hasselt et al. (2016) proposed Double-DQN, which uses an additional critic to reduce the overestimation bias inherent to Q-LEARNING (Sutton & Barto, 2018). Schaul et al. (2016) proposed biasing DQN's replay buffer sampling towards transitions with high TD-error (a form of automatic curriculum learning as discussed in section 2.3.2). Noisy-DQN uses stochastic networks to foster exploration (Fortunato et al., 2018). Additional innovations were also proposed in Wang et al. (2016) and Bellemare

et al. (2017). All these algorithmic improvements are complementary. Hessel et al. (2018) combined them into the RAINBOW algorithm, leading to a four-fold performance improvement over vanilla DQN. Subsequent approaches even outperformed RAINBOW, such as APE-X (Horgan et al., 2018), whose main performance improvement factor is due to its use of *distributed* DRL, i.e. using parallel processes to collect experience, allowing to leverage more computing power without additional wall-clock training time.

Dealing with continuous actions

DQN and its successors are DRL methods applicable to discrete action states. As discussed in section 2.1.2, actor-critic architectures are better suited to deal with continuous action spaces. Handling continuous actions is a determinant requirement for learning policies able to pilot complex embodied systems. Note that DQN-like algorithms can be adapted to such application areas, e.g. for robotic arm manipulation tasks (Zhang et al., 2015) or even to control stratospheric balloons in the real world (Bellemare et al., 2020). However in such cases it requires to either hand-craft high-level discrete controllers or discretize the action space (if the action space is not too large and do not require a finely-grained control).

Multiple actor-critic algorithms for continuous action spaces have recently been proposed, such as DDPG (Lillicrap et al., 2016), TD3 (Fujimoto et al., 2018), TRPO (Schulman et al., 2015) or A2C (Mnih et al., 2016). The current two main SOTA actor-critic methods widely used in the literature are SAC (Haarnoja et al., 2018a), described in section 2.1.2, and *Proximal Policy Optimization* (PPO) from Schulman et al. (2017). In our work we predominantly use SAC (chapters 4, 5 and 6) for its simplicity and efficiency, but we also present experiments with PPO in chapters 5 and 7. PPO is an on-policy actor-critic method. Intuitively, its policy optimization strategy is based on taking small gradient steps to improve the policy's performance while avoiding big updates that might lead to a catastrophic deterioration of the policy (i.e. staying in the so-called "trust region"). Preventing such dangerous weight updates is especially important for on-policy methods, as they rely exclusively on their current policy to collect learning data. The design of efficient actor-critic agents is an active field of research: multiple improvements over the vanilla PPO (Vuong et al., 2019; Wang et al., 2019a; Hämäläinen et al., 2020; Cai et al., 2020) and SAC architecture (Ciosek et al., 2019; Wang & Ross, 2019; Sinha et al., 2020) have been proposed, along with other actor-critic methods (Espeholt et al., 2018; Song et al., 2020).

Robotics applications

Several works showed that DRL approaches could be successfully applied to real-world robotics, e.g. for manipulation tasks with gripper-based robotic arms (Levine et al., 2016, 2018; Pham et al., 2018) or dexterous hands (Haarnoja et al., 2018b; OpenAI et al., 2019; Nagabandi et al., 2020), autonomous driving (Kendall et al., 2019), and locomotion with bipedal (Siekmann et al., 2021) or quadrupedal (Haarnoja et al., 2018b, 2019) robots. While training agents in real-world environments is highly desirable, it has proven quite challenging, mainly due to the difficulty of (safely) collecting the massive amounts of

interaction data required to train DRL agents. While it is possible to use multiple robotic setups in parallel to speed up experience collection (Levine et al., 2018), the resulting hardware investment is often prohibitive for most research laboratories. In practice, most of DRL works are conducted in simulation (including the present research), which has greatly accelerated the expansion of the field, as it allows to leverage computing clusters to efficiently conduct large-scale experiments at reasonable costs. Over the last ten years, multiple open-source simulated robotic environments have been released (Todorov et al., 2012; Brockman et al., 2016; Beattie et al., 2016; Ellenberger, 2018–2019; Tassa et al., 2020). Importantly, multiple works showed that it is possible to train agents in simulated environments and then deploy them on real-world robots with few or no additional training (OpenAI et al., 2019; Zhao et al., 2020; Siekmann et al., 2021).

Note that here and in the remainder of this manuscript we refer to "robotics" in its broadest definition, that is, controlling real or simulated controllable agents, which includes classical humanoid robots but also vehicles (e.g. cars, planes, drones), embodied agents in physics engines (e.g. simulated animals, game characters), smart houses/factories, etc.

Actor-critic DRL architectures have been applied to a diversity of simulated control tasks. Typical examples include locomotion with 2D bodies such as cheetahs or bipeds (Schulman et al., 2015; Lillicrap et al., 2016; Schulman et al., 2017; Haarnoja et al., 2018a), or controlling high-dimensional 3D humanoid bodies for locomotion/navigation (Schulman et al., 2017; Peng et al., 2018a; Haarnoja et al., 2018b). Actor-critic algorithms embodied in 3D humanoid bodies have also been shown capable of learning complex high-level policies such as martial arts (Bansal et al., 2018) or football (Kurach et al., 2020). Other works showed that DRL architectures can learn policies for object manipulation tasks with robotic arms (grasping, pushing, stacking) in simulated (Andrychowicz et al., 2017; Colas et al., 2019) and real-world environments (Levine et al., 2016, 2018; Haarnoja et al., 2018b), or to control water cooling systems in (simulated) data centers (Li et al., 2019). Another domain of application is car racing in video games (Lillicrap et al., 2016), and more generally DRL for autonomous driving research (Kiran et al., 2021).

Towards end-to-end learning

Classical RL already had impressive results in controlling complex systems, such as performing helicopter maneuvers (Ng et al., 2006), efficiently learning to flip pancakes with a robotic arm (Kormushev et al., 2010) – a frivolous yet complicated manipulation task – or learning efficient locomotion gates with a quadrupedal robot (Kohl & Stone, 2004). DRL however allows dealing with such approaches in an *end-to-end* manner, i.e. it is not necessary to supply the RL algorithm with carefully chosen representations of the inputs nor to provide high-level action commands. Instead, researchers rely on the versatility of multi-layered networks to gradually and autonomously learn such important building blocks towards successful behaviors.

However, one important drawback of current DRL approaches is their reliance on a careful tuning of all *hyperparameters* of a given model, which can significantly drain both human and compute resources. Hyperparameters refer to the set of learning configurations (e.g. learning rate, entropy coefficient, optimizer) and model configurations (e.g. how many layers, how many neurons per layers) that must be determined before training

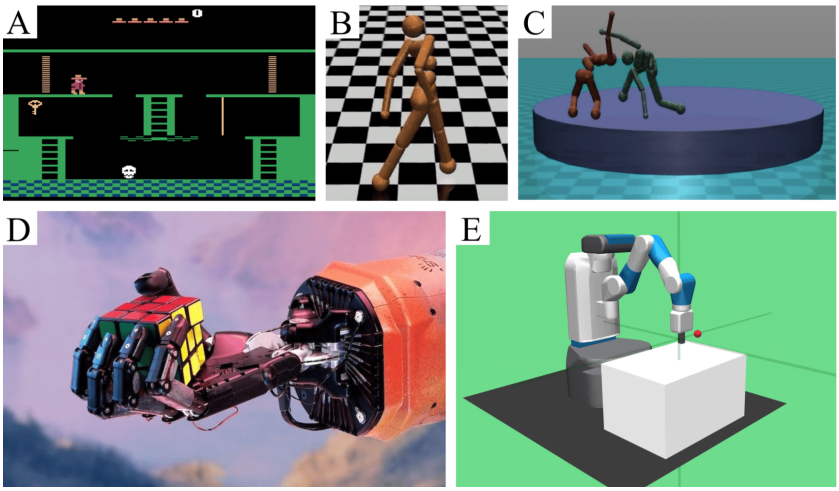


Figure 2.2: Examples of DRL testbeds. **A**: The montezuma revenge environment, an infamous Atari 2600 game (Bellemare et al., 2013).**B**: A simulated humanoid agent from the Mujoco physics simulator (Todorov et al., 2012).**C**: OpenAI sumos environment (Bansal et al., 2018). **D**: OpenAI's real-world robotic hand, operated by their Rubik's cube solving agent (OpenAI et al., 2019). **E**: The fetch environment, a classical mujoco environment to study robotic arm manipulation (Plappert et al., 2018).

time. One potential solution to this problem, generally applicable to any deep learning system, is to rely on additional algorithmic components to automate this hyperparameter selection, i.e. to design Automated Machine Learning (AutoML) approaches (He et al., 2021).

Towards generalization in DRL

Beyond training agents to solve single control problems, DRL researchers have been increasingly interested in finding methods to train generalist agents (Rajeswaran et al., 2017; Zhang et al., 2018b; Vanschoren, 2018; Cobbe et al., 2019; Igl et al., 2019; Kirk et al., 2021), able to act optimally in multiple control scenarios (rather than training on a single task/MDP). Besides, multiple works pointed out that training and testing an agent on a single environment can cause *overfitting* issues (Whiteson et al., 2011; Packer et al., 2018; Zhang et al., 2018b,a), i.e. agents learn brittle policies not able to operate on even minor variations of their training environment.

Recently, Procedural Content Generation (PCG) has been identified as a powerful technique to propose a diversity of training tasks, which elicits the learning of robust multi-purpose policies (Justesen et al., 2018; Hessel et al., 2019; Risi & Togelius, 2019; Cobbe et al., 2019, 2020). If the procedural generation is rich enough, training on PCG environments ensures that the learner never experience the exact same task twice.

OpenAI et al. (2019) present a compelling empirical demonstration of how PCG can promote the learning of robust policies. In this work, authors show that, for a Rubik's Cube manipulation task with a robotic hand, training on a diversity of procedurally generated variants of their simulated environment (a method known as *domain randomization*) was key to learn a robust policy. Authors even showed that their trained agent could

successfully solve a real-world Rubik's Cube with a real-world robotic hand without *fine-tuning* (i.e. without additional training time on the test task). As we will discuss in section 2.3, an important factor to properly leverage their parametric PCG system was the use of an automatic curriculum learning algorithm to guide the parameter sampling procedure.

DRL for Natural Language Processing

The Natural Language Processing (NLP) field encompasses works studying "computational techniques for the automatic analysis and representation of human language"(Cambria & White, 2014). Within this context, DRL methods have been applied to control embodied and disembodied talking agents (Luketina et al., 2019). Representative scenarios studied by this subfield of DRL includes embodied question answering (Das et al., 2018), disembodied text adventure games (Urbanek et al., 2019) or multi-agent emergent communication (Jaques et al., 2019). An extended presentation of such works – in the context of social skill learning – is available in chapter 7.

Solving RL problems without RL algorithms: Evolutionary Algorithms

The list of aforementioned works provides an overview of the diversity of existing (deep) RL solutions that have been applied to a broad range of complex RL problems. However, although effective, iterative gradient descent techniques such as those employed in RL algorithms are not the only solution to obtain high performing policies.

Taking inspiration from natural selection, a large strand of works studies how *Evolutionary Algorithms* (EA) can iteratively create one or several high performing policies (Yu & Gen, 2010). The main difference with DRL is that EA algorithms consider parametric policies as black box functions, i.e. they do not assume any particular structure, e.g. the range of considered parametric policies are not bound to differentiable neural networks. EA methods generate populations of policies, compute a fitness score for each of them, and mutate elite members to create the next population, i.e. the new *generation*. EA algorithms can be divided based on the way policies are "mutated". *Direct encoding* methods generate variants of elite policies by directly perturbing their parameters (changing their "phenotype"), e.g. as in Evolutionary Strategies approaches (Bäck et al., 1991; Li et al., 2020). *Indirect encoding* methods control higher-level parameters encoding the generation of policies (i.e. they mutate the "genotype" of policies). Genetic Algorithms (Stanley et al., 2009; Kumar et al., 2010) are examples of indirect encoding methods. Some EA methods control the selection and mutation of individuals based on both fitness and diversity, e.g. Quality-Diversity algorithms (Pugh et al., 2016), or diversity alone, e.g. Novelty Search algorithms (Lehman et al., 2008). Interestingly, recent work showed that EA approaches could train large DNNs (Risi & Stanley, 2019) with up to millions of parameters. EA mechanisms can also be used as a high-level mechanism to control the diversity and fitness of populations of DRL agents (Wang et al., 2019b, 2020b).

2.2 From Developmental Robotics to Learning Progress

The profoundly interdisciplinary Developmental Robotics field (DevRob) gathers a wide range of computational approaches aiming to model various stages of human development (Brooks et al., 1998; Asada et al., 2001, 2009; Cangelosi & Schlesinger, 2015). Formalizing all such works into a coherent whole, as in the MDP-solving paradigm of RL and DRL, would require many unpleasant theoretical contortions. Instead, this section directly starts with a brief overview of developmental robotics at large. It then focuses on a subfield of DevRob interested in the design and study of *Intrinsic Motivation Systems* (IMS) (Oudeyer et al., 2007a), which studies how self-motivated exploration and development can be hard-wired into robotic agents. We will then expand on the notion of *Learning Progress* (LP): a popular motivation mechanism from the IMS literature, central to the present research, which motivates learners towards improving competence over their world. Among other benefits (discussed in details in section 2.3), endowing agents with a motivation towards LP allows them to autonomously shape efficient learning curricula by preventing them from focusing on distracting/unfeasible dimensions of their environment, for which no LP is detected.

2.2.1 Overview of Developmental Robotics Research

In this brief overview, we will present some of the main facets of development that are studied by the DevRob community: *sensorimotor development*, *maturational constraints*, *social interactions*, and finally, *intrinsic motivation systems*, which are works most related to the present research. For a more in depth introduction to developmental robotics, we refer the interested reader to both Lungarella et al. (2003), which provides a comprehensive (although outdated) survey of the field, and Cangelosi & Schlesinger (2015), a standard DevRob textbook, which contains more up-to-date references.

Sensorimotor development

A core topic of DevRob is the study of sensorimotor development, i.e. works modeling the acquisition of motor and/or visual skills.

For instance, inspired by neuroscience works suggesting an intricate coupling of brain areas corresponding to vision and manipulation in human/primate brains (Ungerleider, 1982; Kovács, 2000), Metta & Fitzpatrick (2003) proposed a computational experiment to study *why* such a cognitive architecture could be advantageous. Using *Cog*, a 22-DoF robotic upper-torso humanoid robot (see figure 2.3A), authors were able to show that learning simple object-poking manipulation behaviors could bootstrap downstream vision tasks such as object segmentation. This belief on a tight coupling between motor and visual/sensor learning is a main factor for DevRob's emphasis on the importance of embodiment (Lungarella et al., 2003).

In Ugur et al. (2015), as an attempt to characterize the kind of mechanisms at play in human infants' gradual sensorimotor skill learning, authors proposed a three-staged developmental system. Using a robotic arm initialized with simple reaching behaviors, they show that their agent is able to explore its motor space and learn control primitives

(stage 1), later reused to learn object manipulation skills (stage 2) and, finally, to bootstrap imitation learning with a cooperative social peer (stage 3).

Beyond learning *object affordances* (learning the range of interactions an object can afford to a given embodied agent), other works focused on tool-use problems, i.e. learning object-object affordances. To better understand the emergence of this pivotal human skill, multiple works proposed computational studies featuring learning systems embodied in robotic arms or full humanoid robots, that learned to manipulate sticks, hooks and rattles to move out-of-reach objects (Stoytchev, 2005; Tikhanoff et al., 2013; Gonçalves et al., 2014). In Forestier & Oudeyer (2017), inspired by previous cognitive science works indicating a strong link between tool-use and speech learning (Gibson et al., 1993), authors presented a developmental agent able to synergistically learn various tool-use and early vocalization skills (using a simulated robotic arm and vocal tract). Using kinesthetic demonstrations, other works showed that complex tool-use behaviors could be acquired in real-world scenarios, e.g. water pouring tasks (Kroemer et al., 2012) or manipulating tackers and electric drills (Li & Fritz, 2015). See figure 2.3 for visual examples of some of these robotic setups.

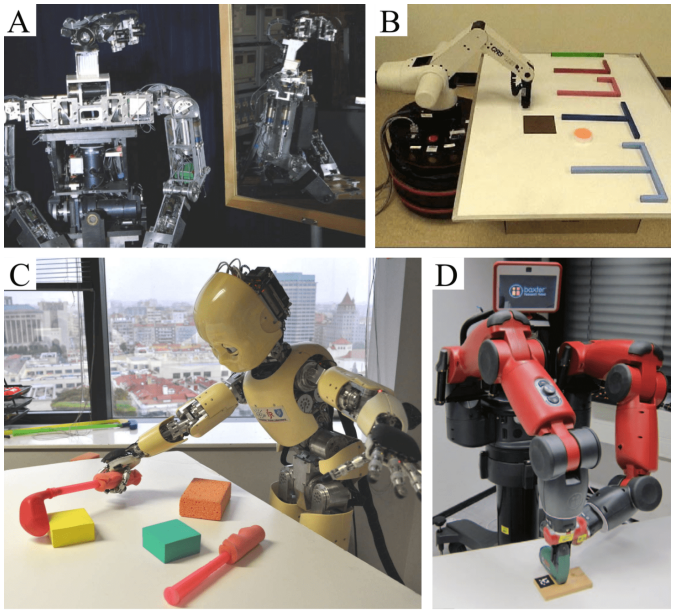


Figure 2.3: Robotic environments used to study sensorimotor development. **A:** *Cog*, the robotic humanoid from Metta & Fitzpatrick (2003). **B:** Early tool-use experiments with a robotic arm (Stoytchev, 2005). **C:** The *iCub* humanoid, learning tool-use strategies to retrieve out-of-reach toys. **D:** A *Baxter* robot which learned to use an electric tacker from kinesthetic demonstrations (Li & Fritz, 2015).

Studying embodied sensorimotor development is often seen in DevRob as an essential component for category learning (Lakoff et al., 1999; Pfeifer & Scheier, 2001), e.g. food/not food, safe/dangerous, which is a fundamental cognitive ability (Edelman, 1987; Rosch, 1999). Multiple DevRob works proposed category learning mechanisms based on visual sensors for robotic systems (Scheier & Lambrinos, 1996; Pfeifer & Scheier, 2001; Krichmar & Edelman, 2002; Te Boekhorst et al., 2003).

In contrast with industrial robotic systems which rely on precise models of their environment to operate, multiple application-oriented DevRob works studied how to learn and adapt approximate kinematic models such that robots can adapt their motor policies

to changing environments, e.g. using classical ML regression techniques (Nguyen-Tuong & Peters, 2008; Sigaud et al., 2011; Nguyen-Tuong & Peters, 2011).

Social Interaction

Multiple psychologists highlighted the importance of social interactions for the cognitive development of humans (Vygotsky & Cole, 1978; Whiten, 2000; Meltzoff & Prinz, 2002). As such, a significant branch of DevRob is interested in creating computational models able to interact in social settings (Fong et al., 2003; Breazeal et al., 2016). Although not the main research subject of our work, we acknowledge the central importance of social interactions in cognitive development. In chapter 7, we expand on this social perspective, discuss works on DRL and social interactions, and propose the *SocialAI* environment suite to seed further research in socially proficient agents.

Many aspects of human's socio-cognitive abilities have been studied within the developmental robotics literature. Some works focused on low-level mechanisms, such as modeling and studying early vocal development (Markey, 1994; Kröger et al., 2009; Warlaumont, 2012; Warlaumont et al., 2013; Moulin-Frier et al., 2014). For instance, in Moulin-Frier et al. (2014), authors hypothesized that intrinsic motivation is an important driver for early vocal development. To validate their perspective, they conducted computational experiments with a simulated vocal tract controlled by a learning system equipped with intrinsic motivation mechanisms. They were able to showcase that such intrinsically motivated agents could gradually develop vocalization skills akin to what is observed is human infants: from phonation, to unarticulated sounds, to babbling with articulated proto-syllables, up to imitation of sounds coming from (simulated) social peers.

To study higher-level aspects of sociality, multiple works conducted experiments with both robots and humans. In Breazeal & Aryananda (2002), using the *Kismet* robotic head (see figure 2.4 A & B), authors studied how to recognize and express emotions. Other works focused on modeling systems able to perform joint visual attention tasks with social peers (Scassellati, 2001; Kozima & Yano, 2001; Nagai et al., 2002), e.g. detecting which object a caregiver is visually attending to, which is an important cognitive skill. Related to joint attention, multiple works studied social referencing scenarios, in which a caregiver instructs a learner about the status of specific entities (Breazeal et al., 2004; Thomaz et al., 2005). In Thomaz et al. (2005), authors use the *Leonardo* robot (see figure 2.4 C & D) to study the importance and interconnection of three mechanisms that might allow humans to perform social referencing: emotional empathy (through facial imitation), joint attention, and an affective memory system. They were able to show that their agent could successfully and efficiently learn to exhibit positive or negative affective states when confronted to objects that were previously presented by a human caregiver.

DevRob works on social robotics is strongly linked to the Human-Robot Interaction (HRI) field, which regroup works "dedicated to understanding, designing, and evaluating robotic systems for use by or with humans."(Goodrich & Schultz, 2008). In such scenarios, developmental mechanisms can be used to create fluid interactions between robots and humans, e.g. by self-assessing its motor skills to autonomously require human demonstrations (Maeda et al., 2017), imitating social peers (Dautenhahn & Billard, 1999; Breazeal & Scassellati, 2002) and perform collaborative tasks (Busch et al., 2018).

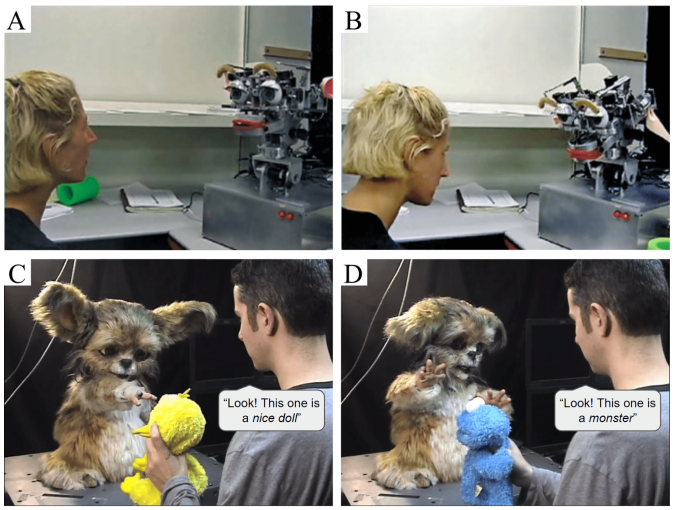


Figure 2.4: DevRob works on social interactions. **A,B:** The *Kismet* robotic head, able to recognize and express emotions (Breazeal & Aryananda, 2002). **C,D:** The *Leonardo* robot, learning object status through social referencing (Thomaz et al., 2005).

Maturational constraints

As framed by Lungarella et al. (2003), "the central tenet of embodied cognition is that cognitive and behavioural processes emerge from the reciprocal and dynamic coupling between brain, body and environment". So far, we presented works focused on studying learning scenarios, i.e. how embodied synthetic *brains* aggregate and refine knowledge through interaction within *environments*. Another facet of development which is studied and modeled in DevRob is the study of brain-body interactions, i.e. how body maturation dynamics affect development (Schlesinger et al., 2008; Baranes & Oudeyer, 2011). Such works are heavily motivated by the joint observations that 1) humans reach high levels of general cognition (higher than any other primate), although 2) human babies learn early locomotion skills at much slower speeds than most other primates. Based on this, multiple psychologists posited that the slow locomotion development of human babies might allow them to develop other skills, such as object manipulation, tool use or social interaction (Schlesinger et al., 2008; Baranes, 2011). More generally, rather than holding back development, the global immaturity of human infants has been seen as a key to human's unique cognitive development (Turkewitz & Kenny, 1982; Bjorklund, 1997). Immaturity is thought to be positive for development because it 1) fosters cognitive plasticity, 2) biases towards social interactions, and 3) protects against over-stimulation (Baranes, 2011).

Based on these observations, robotic experiments were performed to assess how the gradual release of *maturational constraints* could enhance the learning abilities of robotic systems. Perceptual maturation mechanisms have been shown to improve learning abilities. Examples of such works include experiments which gradually improved vision sensors from embodied robotic systems (French et al., 2002; Dominguez & Jacobs, 2003; Nagai et al., 2006), or language processing abilities, so far only done in disembodied scenarios (Elman, 1993). Other works focused on motor constraints, mostly through the gradual release of DoF for robotic bodies, showing that using motor maturation schedules results in better performances than training without them (Berthouze & Kuniyoshi, 1998; Metta et al., 1999; Lungarella & Berthouze, 2002a,b; Baranes & Oudeyer, 2010b, 2011).

Works studying how to leverage maturational constraints for learning can be seen as implementing forms of *internal* curriculum learning approaches, as they modulate the range of possible interactions an agent can experience.

Intrinsic Motivation

As mentioned in chapter 1, multiple psychologists observed and acknowledged the importance of *intrinsic motivation* mechanisms in shaping the learning of humans and animals (White, 1959; Berlyne, 1960, 1966; Deci & Ryan, 1985; Loewenstein, 1994; Benson, 2020). As such, the development of intrinsic motivation systems, to promote a self-organized and self-motivated development of robotic agents, has been an important focus of the DevRob community (Blank et al., 2005; Oudeyer et al., 2007a; Oudeyer & Kaplan, 2007; Baldassarre & Mirolli, 2013; Bazhydai et al., 2021). More precisely, such approaches are based on the computation of internal incentives pushing artificial agents towards seeking interactions providing an intermediate level of "novelty", "surprise", or "challenge". The range of possible IMS implementations lies in the many ways to operationalize such words into empirical and tractable measurements (Oudeyer et al., 2007a). To do so, works on IMS often frame their considered learning situations into the MDP-solving paradigm of RL problems (explained in section 2.1.1), in which internal incentives takes the form of intrinsic rewards. In Oudeyer & Kaplan (2007), authors propose two main categories to regroup intrinsic motivation systems: *knowledge-based* works and *competence-based* works.

Knowledge-based IMS regroups approaches rewarding agents based on measurements of similarity (or dissimilarity) between an agent's representation of the world w.r.t. new interaction data. For instance, novelty-seeking robotic systems can have their behavior reinforced towards discovering new sensor readings (Weng, 2002; Huang & Weng, 2002, 2004; Barto et al., 2004), and more generally new *behavioral outcomes* (Benureau & Oudeyer, 2016). In such cases, intrinsic rewards are often computed based on measuring the difference between empirical observations w.r.t predictions from a learned model (Kaplan & Oudeyer, 2003; Barto et al., 2004; Marshall et al., 2004; Singh et al., 2005). In Oudeyer et al. (2007a), authors propose to use a form of learning progress to compute intrinsic rewards. Agents are rewarded for focusing interaction on sensorimotor regions for which outcome prediction errors are decreasing, which indicates progress. Knowledge-based IMS works bear many similarities with pioneering RL works on intrinsic motivations from (Schmidhuber, 1991; Thrun, 1995; Herrmann et al., 2000). For instance, Schmidhuber (1991) showed that RL agents could efficiently learn world-models through "artificial curiosity", i.e. by rewarding them to focus on parts of the environment for which the expected derivative of state prediction error is high.

Competence-based IMS characterizes a more recent branch of IMS works, which considers robotic agents able to self-select their own learning experiences, e.g. their own goals. Given this setup, intrinsic rewards based on the evolving goal-reaching competence of the agent can be computed to guide learning. Such approaches have been formalized as Intrinsically Motivated Goal Exploration Processes, a.k.a. IMGEPs (Forestier et al., 2017). Over the last fifteen years, multiple *learning progress*-based IMGEP approaches were

successfully applied to various DevRob domains, from guiding vocal development (Moulin-Frier et al., 2014; Forestier & Oudeyer, 2017), to various tool-use learning scenarios (Forestier & Oudeyer, 2016a,b,c; Forestier et al., 2017). Because of the central importance of early IMGEP works for the present research, in the following section, we expand on this literature and more generally on the notion of learning progress. A core objective of this thesis is precisely to adapt such LP-based IMS for DRL learners (chapter 4 and 6).

Note that, parallel to the present research, multiple recent works studied how to leverage competence-based IMS for goal-conditioned DRL agents instead of population-based learners (Laversanne-Finot et al., 2018; Colas et al., 2019, 2020a). See Colas et al. (2020b) for a review. Many of these works will be discussed from an ACL perspective in section 2.3.2.

2.2.2 IMGEPs and Learning Progress

Learning Progress

In Kaplan & Oudeyer (2007), inspired by a panel of early theories on the intrinsically motivated nature of human and animal development – e.g. White (1959); Berlyne (1960); see chapter 1 – authors formulated the hypothesis that humans use *learning progress* to organize the exploration and learning of their world. Humans are intrinsically pushed towards focusing on experiences in which their competence improvement is maximized. Interestingly, beyond theoretical considerations, very recent work showed for the first time evidence that humans actually monitor LP to self-organize their learning curriculum (Ten et al., 2021) .

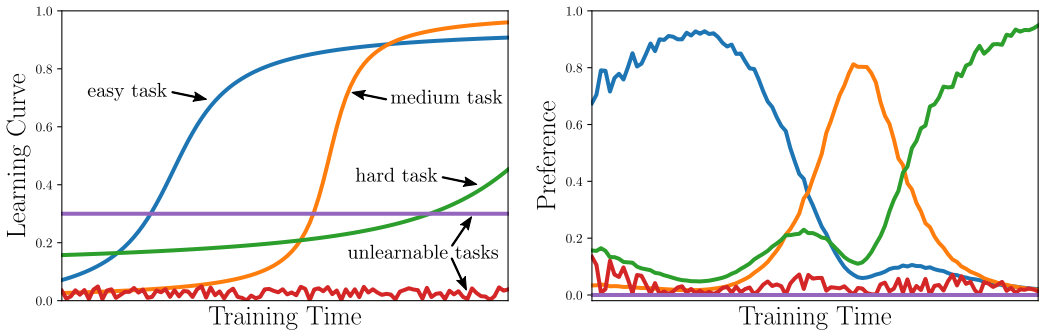
As previously mentioned, multiple computational works within the DevRob literature proposed to derive intrinsic rewards based on this notion of learning progress. LP has also been used within the RL literature, e.g. to accelerate the training of LSTMs and neural turing machines (Graves et al., 2017), and in education research, to personalize sequences of exercises for children in educational technologies (Clément et al., 2015).

LP is a simple concept: given a task A to repeatedly train on, one can intuitively define the learning progress of an agent on task A as the derivative of its competence on A. If considering multiple tasks to learn, using LP allows agents to derive efficient learning curricula by dynamically shifting training focus on tasks with high learning progress. See the following box on learning progress for a schematic example. However, complexity arises when researchers try to implement this simple idea into tractable intrinsic motivation mechanisms. First, exact competence evaluation is not possible in non-trivial scenarios. Instead, LP-based approaches rely on computing empirical and approximate performance evaluations. Additional approximations are necessary when considering open-ended environments containing an infinite number of possible tasks, as in the present research. In such cases, aggregated LP estimations must be computed.

LP-based IMGEPs

In Baranes & Oudeyer (2009), authors present RIAC, an algorithm using learning progress signals to organize the collection of interaction data such that forward motor

Learning Progress: an Illustrative Example



This pair of figures (adapted from Forestier et al. (2017)) represents a schematic example of training dynamics for an agent with LP-based intrinsic rewards.

Evolution of competence – The left figure illustrates the schematic evolution of competence for such an agent when learning to master 5 – color differentiated – hypothetical tasks (a competence of 1 means the task is completely mastered). Over training time, one can see that the agent is quickly able to improve over the blue task, followed by the yellow one. The flat purple curve indicates that the agent did not manage to improve over the purple task.

Evolution of training focus – The right figure illustrates the schematic task selection preference of the same agent across training. By using a learning progress heuristic, such as the derivative of competence, the agent is able to efficiently allocate its training time into a coherent learning curriculum. For instance, the agent initially focuses its training on the blue task, for which high learning progress is initially experienced, while marginally training on other tasks to maintain exploration. After reaching a performance plateau on the blue task, i.e. a low-LP situation, our hypothetical learner shift its focus towards more promising tasks, i.e. tasks with higher LP. Organizing training focus based on learning progress allows agents to avoid wasting time on unlearnable tasks that can have fixed (purple) or randomly changing (red) outcomes.

dynamics predictors can be efficiently learned. Learning a "forward model" amounts to learning to predict motor positions $m^{(t+1)}$ when given a motor action a_m and an initial motor position m^t. Using multiple simulated 2D robotic arm experiments, authors show that using forward models learned with RIAC for downstream control tasks leads to agents with more efficient control policies compared to using random or previous forward model learning approaches (Oudeyer et al., 2007a). RIAC is a LP-biased parameter sampling approach. In their work, authors use it to sample target motor configurations to reach, i.e. RIAC is used in the motor space. To organize sampling, the core idea of RIAC is to split the motor space in hyperboxes (called regions) according to their respective LP value. LP is defined as the difference in prediction error between the oldest and newest motor parameters sampled in the region (see figure 2.5). If the forward dynamic model is getting better at predicting motor positions from a given region, the LP of the region

will be positive. New target motor parameters are then sampled within regions selected proportionally to their LP score, which allows focusing learning on feasible subspaces.

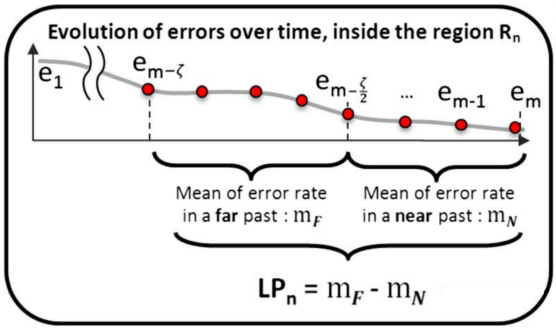


Figure 2.5: Example of aggregated and empirical LP computation in RIAC for a given region. Figure adapted from Baranes & Oudeyer (2009).

In the aforementioned experiments, RIAC is used as a knowledge-based IMS, i.e. it is helping to learn a predictive model. However, in subsequent work, Baranes & Oudeyer (2010a) showed that the very same LP-biased sampling procedure could be applied in a competence-based fashion, i.e. to control the self-selection of skills to train on for developmental robots. More precisely, instead of applying RIAC at the motor space level, authors propose to use it directly to sample high-level *goals*, e.g. target end-effector positions or object manipulation objectives. Authors argue that using RIAC or similar approaches at the motor space level can lead to poor exploratory behaviors, as agents can waste time learning to reach many motor positions with equivalent outcomes, e.g. "learning 10 ways to push a ball forwards, instead of learning to push a ball in 10 different directions"(Baranes & Oudeyer, 2010a). The motor space is *redundant*. Besides, another presented limitation of using the motor space is that it can quickly become high-dimensional, especially if considering complex robots, while it is often possible to craft tractable goal spaces (e.g. the x and y position of a ball to push). Given this, authors propose the SAGG-RIAC architecture, which combines a goal-space version of RIAC to guide the learning of a lower-level goal-reaching procedure learning inverse motor dynamics (i.e. learning to generate motor actions given a target goal position). Authors showcase the effectiveness of their architecture to enable robotic agents to autonomously organise the open-ended learning of efficient control policies for a 15-DoF simulated robotic arm.

In Baranes & Oudeyer (2013), the same authors showcase the effectiveness of the SAGG-RIAC archicture on real robotic environments, including the control of a 8-DoF robotic arm, the learning of locomotion policies with a quadrupedal robot, and even the learning of fishing rod control policies. Interestingly, additional experiments showed that the SAGG-RIAC architecture could be efficiently integrated with robotic systems endowed with maturational constraints (Baranes & Oudeyer, 2011). In Moulin-Frier et al. (2014), inspired by the SAGG-RIAC architecture, authors propose to apply a similar LP-based IMS to model the autonomous development of vocalization in infants using a simulated vocal tract. To do so, they propose a more principled variant of RIAC using Gaussian Mixture Models. The ALP-GMM algorithm, presented in chapter 4, can be seen as an adaptation of these aforementioned works to the context of autonomously shaping the training of deep reinforcement learning agents.

In Forestier & Oudeyer (2016b), authors observe that approaches such as SAGG-RIAC have so far only been successfully applied to relatively simple, low-dimensional task spaces. To extend such methods to more structured and higher-dimensional sensorimotor spaces, they propose the Modular Active Curiosity-driven mOdel Babbling architecture (MACOB). The key insight of MACOB is to consider a modular goal space, i.e. to rely on the pre-definition of distinct and meaningful goal spaces, and to learn one inverse model per goal space, a.k.a. module. This is especially useful when considering learning situations in environments featuring multiple objects: one goal space is created per object (e.g. 2D goal spaces corresponding to the end-position of each object). Then, LP-estimations can be performed discretely, at the level of modules. For each new training episode, a goal space is sampled based on LP-estimates, and a goal is sampled on the selected goal space (at random in their experiments). This dynamic creates an automatic curriculum which allows the learning agent to focus on adapted goal spaces throughout its training. Authors show that an agent trained with MACOB in a simulated tool-use environment is able to autonomously learn a diversity of efficient tool-use control policies to manipulate its surrounding world w.r.t non-modular learning systems.

In Forestier et al. (2017), authors present the IMGEP framework, which unifies previously mentioned competence-based IMS. IMGEP agents autonomously shape their developmental trajectory by self-selecting goals to perform in their environment and self-monitoring the evolution of their competence over these goals. Authors propose several algorithmic improvements over the MACOB architecture, and showcase the performance of the resulting approach on both the same simulated tool use environment and on a more complex real-world robotic environment featuring a Poppy torso robot (Lapeyre et al., 2014). See figure 2.6 for visualizations of these environments.

Chapter 3 presents a complementary experimental study, now featured in Forestier et al. (2017). In these experiments, we showcase for the first time that such modular IMGEP approaches can be efficiently used with neural networks as expressive controllers.

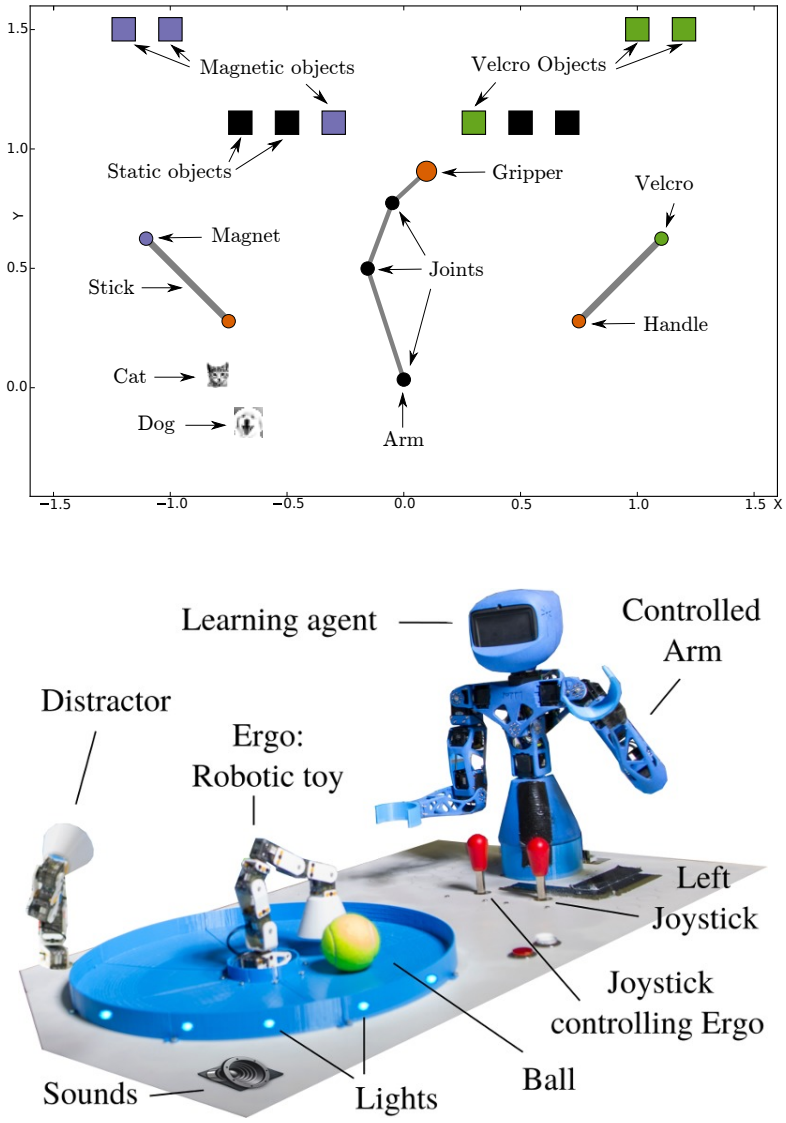


Figure 2.6: Experimental robotic environments from (Forestier et al., 2017).
Top: *2D Simulated Tool-Use Environment.* A simulated robotic arm with a gripper can grab sticks and move toys. The gripper has to close near the handle of a stick to grab it. One magnetic toy and one Velcro toy are reachable with their corresponding stick. Other toys cannot be moved (static or too far away). The cat and the dog are distractors: they move randomly, independently of the arm.
Bottom: *Robotic Tool-Use Environment.* a Poppy Torso robot (the learning agent) is mounted in front of two joysticks that can be used as tools to act on other objects: a Poppy Ergo robotic toy and a ball that can produce light and sound. An additional Ergo robot is featured in the environment and act as distractors that move randomly, independently of the agent.

2.3 Automatic Curriculum Learning

Human learning is organized into a curriculum of interdependent learning situations of various complexities. Homer sure learned to formulate words before he could compose the Iliad. This idea was first transferred to machine learning in (Selfridge et al., 1985), where authors designed a *learning scheme* to train a cart pole controller: first training on long and light poles, then gradually moving towards shorter and heavier poles. In the following years, curriculum learning was applied to organize the presentation of training examples or the growth in model capacity in various supervised learning settings (Elman, 1993; Krueger & Dayan, 2009; Bengio et al., 2009). In parallel, as discussed in the previous section, the developmental robotics community proposed *learning progress* as a way to automatically organize the *developmental trajectories* of learning agents (Kaplan & Oudeyer, 2007). Inspired by these earlier works, the deep reinforcement learning community developed a family of mechanisms called *automatic curriculum learning*, which we propose to define as follows:

Automatic Curriculum Learning *for DRL is a family of mechanisms that automatically adapt the distribution of training data by adjusting the selection of learning situations to the capabilities of learning agents.*

In this section, we present our first two contributions: a general formalization of automatic curriculum learning and a survey of the ACL literature. The ambition of the survey is dual: 1) to present a compact and accessible introduction to the automatic curriculum learning literature, and 2) to draw a bigger picture of the current state of the art in ACL to encourage the cross-breeding of existing concepts and the emergence of new ideas.

2.3.1 ACL Formalization

The concept of ACL can be simply understood as characterizing the interplay between a *student* algorithm and a *teacher* algorithm. Both algorithms are *learning* processes. The student learns to improve its action policies on sequentially presented tasks. The teacher – i.e. the ACL method – learns to select which tasks to present to its student. The teacher objective is to generate a sequence of training tasks – as short as possible – that will enable his student to learn a policy able to reach high-performances on a set of target tasks. In other words, as depicted in figure 2.7, the objective of this automatic generation of a task curriculum is to provide a *sample efficient* training leading to high *asymptotic performances* on the target task set.

More precisely, our formalization makes the following set of assumptions:

1. *Single student* – We consider a single autonomous action policy learner π (the student).
2. *MDP setup* – π is trained in tasks defined as episodic-MDPs (as presented in section 2.1.1).
3. *ACL algorithm* – Let $\mathcal{D}$ be the parametric *learning* algorithm in charge of generating the task curriculum of agent π.

4. *Single training session* – The ACL method must adapt its curriculum through a single training session with π: the student's knowledge state is *non-resetable*. In other words, it is not possible to simply restart training in the advent of catastrophic policy updates, which implies that curriculum generation must be done carefully.

5. *Budgeted training* – Training time is finite: ACL must be performed within a budget of E tasks (i.e. E episodes) presented to π.

Given this, an automatic curriculum learning mechanism can be formalized as proposing to learn a task selection function $\mathcal{D}(\mathcal{H}) \rightarrow \tau \in \mathcal{T}$ where $\mathcal{H}$ can contain any information about past interactions, and $\mathcal{T}$ is a task set (discrete case) or a task space (infinite case). To measure performance, experimenters define the objective of such approaches as the maximization of a behavior metric P computed over a distribution of target tasks $\mathcal{T}_{target}$. Assuming that $\mathcal{T}$ is a continuous task space, we can express this objective as

$$\begin{aligned} Obj : \max_{\mathcal{D}} \int_{\tau \sim \mathcal{T}_{target}} P_{\tau}^{\pi} \mid \mathcal{T}_{train}^{\pi} \, \mathrm{d}\tau, \\ \text{with } \mathcal{T}_{train}^{\pi} = \left[\tau_1, \tau_2, ..., \tau_E\right]_{\tau_i \sim \mathcal{D}} \end{aligned} \tag{2.12}$$

The term $P_{\tau}^{\pi} \mid \mathcal{T}_{train}^{\pi}$ quantifies the agent's behavior on a target task τ (P_{τ}^{π}) *given* a previous training on a set of E tasks ($\mathcal{T}_{train}^{\pi}$) selected by the ACL function $\mathcal{D}$. Cumulative reward or exploration measures are typical examples of used behavior metrics P.

Figure 2.7 provides an illustration of the expected benefits of using ACL over randomly proposing tasks. In practice, the objective presented in equation 2.12 is not directly optimized for, as it is intractable to optimize w.r.t. downstream performance over a target task set. Besides, this target task set is not always known to the researcher, e.g. when aiming to train open-ended agents. As we will see in the following survey, ACL methods rely on the optimization of surrogate objectives, e.g. based on empirical learning progress maximization.

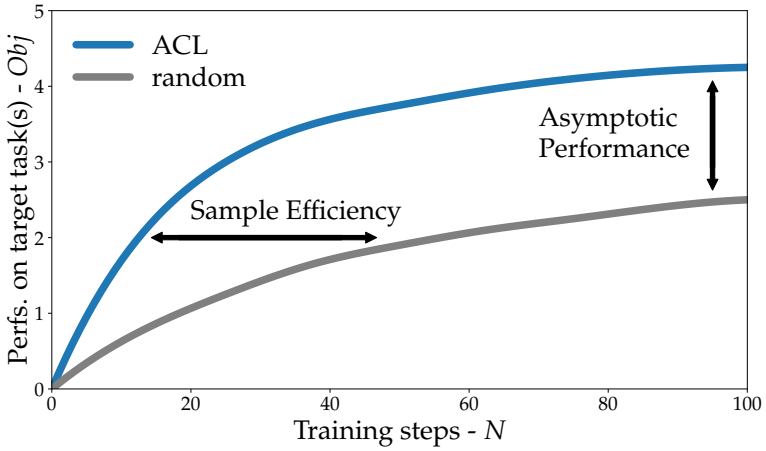


Figure 2.7: The objective of ACL methods is to adapt the presentation of learning experiences to a given policy learner such that its training is more *sample efficient* and ultimatly, leads to trained agents with higher *asymptotic performances* than when following a random curriculum. Figure inspired from Narvekar et al. (2020).

Related formalizations

Our ACL formalization is conceptually close to the *strategic student problem* (Lopes & Oudeyer, 2012a), which formalizes a DevRob setting where an agent has to sequentially select tasks to train on to maximize its average competence over the whole set of tasks after a given number of interactions. However, the strategic student problem do not define the nature of considered tasks, while we propose to focus on episodic MDPs (and consequently, on policy learning students). Additionally, they assume that the training task set is *discrete* (e.g. as in Matiisen et al. (2017)), while our formalization accommodates for both discrete and infinite settings (i.e. continuous task spaces). Finally, we do not assume that the target task set is necessarily part of the training task set.

Concurrently to our work, Narvekar et al. (2020) proposed a formalization (and survey) of curriculum learning in RL[2]. In their work, authors present a broad formalization of what is curriculum learning and frame multiple forms of curriculum generation (Single-task Curriculum, Task-level Curriculum, Sequence Curriculum). Compared to their work, we provide a single, compact formalization of ACL, encompassing a wide range of existing approaches, as presented in section 2.3.2, focusing more specifically on ACL algorithms that sample tasks *presented* to the policy learner. As we will see in the following survey, there are also other forms of ACL methods that are based on selecting *already collected experience transitions* from which π learns from (i.e. "data exploitation" ACL), which is also discussed in Narvekar et al. (2020).

2.3.2 Survey of Automatic Curriculum Learning for DRL Agents

Related fields – ACL shares many connections with other fields. For example, ACL can be used in the context of *transfer learning* where agents are trained on one distribution of tasks and tested on another (Taylor & Stone, 2009). *continual learning* trains agents to be robust to unforeseen changes in the environment while ACL assumes agents to stay in control of learning scenarios (Lesort et al., 2019). *policy distillation* techniques (Czarnecki et al., 2019) form a complementary toolbox to target multi-task RL settings, where knowledge can be transferred from one policy to another (e.g. from task-expert policies to a generalist policy).

Scope – This short survey proposes a typology of ACL mechanisms when combined with DRL algorithms and, as such, does not review population-based algorithms implementing ACL (e.g. Forestier et al. (2017), Wang et al. (2019b)). ACL refers to mechanisms *explicitly* optimizing the automatic organization of training data. Hence, they should not be confounded with *emergent curricula*, by-products of distinct mechanisms. For instance, the on-policy training of a DRL algorithm is not considered ACL, because the shift in the distribution of training data *emerges* as a by-product of policy learning. To keep this survey relatively short, we do not present the details of every particular mechanism. As

[2]Interestingly, their preprint version (`https://arxiv.org/abs/2003.04960`) was released *the very same day* as ours (`https://arxiv.org/abs/2003.04664`), which illustrates the general need to provide high-level perspectives on CL/ACL.

the current ACL literature lacks theoretical foundations to ground proposed approaches in a formal framework, this survey focuses on empirical results. For a complementary perspective, we refer to Narvekar et al. (2020), which is a concurrent work providing a broader survey of curriculum learning (not necessarily automatic) for RL domains (not necessarily with DRL agents).

ACL Typology – We propose a classification of ACL mechanisms based on three dimensions:

1. *Why use ACL?* We review the different objectives that ACL has been used for.
2. *What does ACL control?* ACL can target different aspects of the learning problem (e.g. environments, goals, reward functions).
3. *What does ACL optimize*? ACL mechanisms usually target surrogate objectives (e.g. learning progress, diversity) to alleviate the difficulty to optimize the main objective *Obj* directly.

Why use ACL?

ACL mechanisms can be used for different purposes that can be seen as particular instantiations of the general objective defined in equation 2.12, thereafter referred to as *Obj*.

Improving performance on a restricted task set – Classical RL problems are about solving a given task, or a restricted task set (e.g. which vary by their initial state). In these simple settings, ACL has been used to improve sample efficiency or asymptotical performance (Schaul et al., 2016; Horgan et al., 2018).

Solving hard tasks – Sometimes the target tasks cannot be solved directly (e.g. too hard or sparse rewards). In that case, ACL can be used to pose auxiliary tasks to the agent, gradually guiding its learning trajectory from simple to difficult tasks until the target tasks are solved (Matiisen et al., 2017; Florensa et al., 2017; Riedmiller et al., 2018; Ivanovic et al., 2018; Salimans & Chen, 2018). Another line of work proposes to use ACL to organize the exploration of the state space so as to solve sparse reward problems (Bellemare et al., 2016; Pathak et al., 2017; Shyam et al., 2019; Pathak et al., 2019; Burda et al., 2019b). In these works, the performance reward is augmented with an intrinsic reward guiding the agent towards uncertain areas of the state space.

Training generalist agents – Generalist agents must be able to solve tasks they have not encountered during training (e.g. continuous task spaces or distinct training and testing set). ACL can shape learning trajectories to improve generalization, e.g. by avoiding unfeasible task subspaces, as we will see in chapter 4. ACL can also help agents to generalize from simulation settings to the real world (Sim2Real) (OpenAI et al., 2019; Mehta et al., 2019) or to maximize performance and robustness in multi-agent settings via *self-play* (Silver et al., 2017; Pinto et al., 2017; Bansal et al., 2018; Vinyals et al., 2019; Baker et al., 2020).

Training multi-goal agents – In multi-goal RL, agents are trained and tested on tasks that vary by their goals. Because agents can control the goals they target, they learn a behavioral repertoire through one or several goal-conditioned policies. The adoption of ACL in this setting can improve performance on a testing set of pre-defined goals (Andrychowicz et al., 2017; Sukhbaatar et al., 2018; Zhao & Tresp, 2018a; Fournier et al., 2018; Florensa et al., 2018; Zhao & Tresp, 2018b; Racaniere et al., 2020; Cideron et al., 2019; Fang et al., 2019; Colas et al., 2019).

Organizing open-ended exploration – In some multi-goal settings, the space of achievable goals is not known in advance. Autonomous agent must discover achievable goals as they explore and learn how to reach them. For this problem, ACL can be used to organize the discovery and acquisition of repertoires of robust and diverse behaviors (Eysenbach et al., 2019; Lair et al., 2019; Jabri et al., 2019; Pong et al., 2020; Colas et al., 2020a).

What does ACL control?

While *on-policy* DRL algorithms directly use training data generated by the current behavioral policy, *off-policy* algorithms can use trajectories collected from other sources. This practically decouples *data collection* from *data exploitation*. Hence, we organize this section into two categories: one reviewing ACL for data collection, the other ACL for data exploitation.

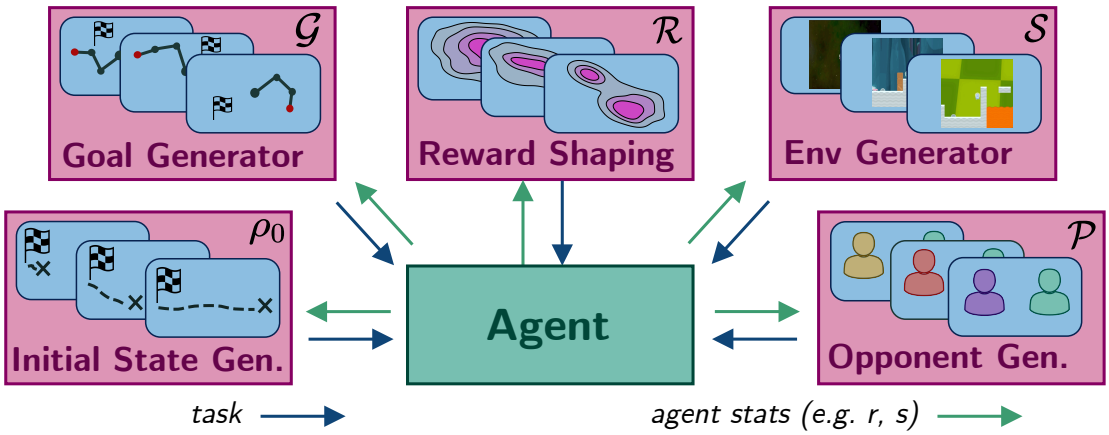


Figure 2.8: **ACL for data collection.** ACL can control each element of task MDPs to shape the learning trajectories of agents. Given metrics of the agent's behavior like performance or visited states, ACL methods generate new tasks adapted to the agent's abilities.

ACL for Data Collection – During data collection, ACL organizes the sequential presentation of tasks as a function of the agent's capabilities. To do so, it generates tasks by acting on elements of task MDPs (e.g. $\mathcal{R}, \mathcal{P}, \rho_0$, see Fig. 2.8). The curriculum can be designed on a discrete set of tasks or on a continuous task space. In single-task problems, ACL can define a set of auxiliary tasks to be used as stepping stones towards the

resolution of the main task. The following paragraphs organize the literature according to the nature of the control exerted by ACL:

Initial state (ρ_0) - The distribution of initial states ρ_0 can be controlled to modulate the difficulty of a task. Agents start learning from states close to a given target (i.e. easier tasks), then move towards harder tasks by gradually increasing the distance between the initial states and the target. This approach is especially effective to design auxiliary tasks for complex control scenarios with sparse rewards (Florensa et al., 2017; Ivanovic et al., 2018; Salimans & Chen, 2018).

Reward functions ($\mathcal{R}$) - ACL can be used for automatic reward shaping: adapting the reward function $\mathcal{R}$ as a function of the learning trajectory of the agent. In curiosity-based approaches especially, an internal reward function guides agents towards areas associated with high uncertainty to foster exploration (Bellemare et al., 2016; Pathak et al., 2017; Shyam et al., 2019; Pathak et al., 2019; Burda et al., 2019b). As the agent explores, uncertain areas –and thus the reward function– change, which automatically devises a learning curriculum guiding the exploration of the state space. In Fournier et al. (2018), an ACL mechanism controls the tolerance in a goal reaching task. Starting with a low accuracy requirement, it gradually and automatically shifts towards stronger accuracy requirements as the agent progresses. In Eysenbach et al. (2019) and Jabri et al. (2019), authors propose to learn a skill space in unsupervised settings (from state space and pixels respectively), from which are derived reward functions promoting both behavioral diversity and skill separation.

Goals ($\mathcal{G}$) - In multi-goal DRL, ACL techniques can be applied to order the selection of goals from discrete sets (Lair et al., 2019), continuous goal spaces (Sukhbaatar et al., 2018; Florensa et al., 2018; Pong et al., 2020; Racaniere et al., 2020) or even sets of different goal spaces (Colas et al., 2019). Although goal spaces are usually pre-defined, recent work proposed to apply ACL on a goal space learned from *pixels* using a generative model (Pong et al., 2020).

Environments ($\mathcal{S}, \mathcal{P}$) - ACL has been successfully applied to organize the selection of environments from a discrete set, e.g. to choose among Minecraft mazes (Matiisen et al., 2017) or Sonic the Hedgehog levels (Mysore et al., 2018). A more general –and arguably more powerful– approach is to leverage parametric *Procedural Content Generation* (PCG) techniques (Risi & Togelius, 2019) to generate rich task spaces. In that case, ACL allows to detect relevant niches of progress, as with our ALP-GMM algorithms (see chapter 4) and other related works (OpenAI et al., 2019; Mehta et al., 2019).

Opponents ($\mathcal{S}, \mathcal{P}$) - Self-play algorithms train agents against present or past versions of themselves (Silver et al., 2017; Bansal et al., 2018; Vinyals et al., 2019; Baker et al., 2020). The set of opponents directly maps to a set of tasks, as different opponents results in different transition functions $\mathcal{P}$ and possibly state spaces $\mathcal{S}$. Self-play can thus be seen as a form of ACL, where the sequence of opponents (i.e. tasks) is organized to maximize performance and robustness. In single-agent settings, an adversary policy can be trained to perturb the main agent (Pinto et al., 2017).

ACL for Data Exploitation – ACL can also be used in the data exploitation stage, by acting on training data previously collected and stored in a *replay memory*. It enables the agent to "mentally experience the effects of its actions without actually executing them", a technique known as *experience replay* (Lin, 1992). At the data exploitation level, ACL can exert two types of control on the distribution of training data: *transition selection* and *transition modification*.

Transition selection ($\mathcal{S} \times \mathcal{A}$) - Inspired from the *prioritized sweeping* technique that organized the order of updates in planning methods (Moore & Atkeson, 1993), Schaul et al. (2016) introduced *prioritized experience replay* (PER) for model-free RL to bias the selection of transitions for policy updates, as some transitions might be *more informative* than others. Different ACL methods propose different metrics to evaluate the importance of each transition (Schaul et al., 2016; Zhao & Tresp, 2018a,b; Colas et al., 2019; Lair et al., 2019; Colas et al., 2020a).

Transition modification ($\mathcal{G}$) - In multi-goal settings, *Hindsight Experience Replay* (HER) proposes to reinterpret trajectories collected with a given target goal with respect to a different goal (Andrychowicz et al., 2017). In practice, HER modifies transitions by substituting target goals g with one of the outcomes g' achieved later in the trajectory, as well as the corresponding reward $r' = R_{g'}(s, a)$. By explicitly biasing goal substitution to increase the probability of sampling rewarded transitions, HER shifts the training data distribution from simpler goals (achieved now) towards more complex goals as the agent makes progress. Substitute goal selection can be guided by other ACL mechanisms, e.g. by favoring diversity (Fang et al., 2019; Cideron et al., 2019).

What Does ACL Optimize?

Objectives such as the average performance on a set of testing tasks after N training episodes can be difficult to optimize directly. To alleviate this difficulty, ACL methods use a variety of surrogate objectives.

Reward – As DRL algorithms learn from reward signals, rewarded transitions are usually considered as more informative than others, especially in sparse reward problems. In such problems, ACL methods that act on transition selection may artificially increase the ratio of high versus low rewards in the batches of transitions used for policy updates (Narasimhan et al., 2015; Jaderberg et al., 2017; Colas et al., 2020a). In multi-goal RL settings where some goals might be much harder than others, this strategy can be used to balance the proportion of positive rewards for each of the goals (Colas et al., 2019; Lair et al., 2019). Transition modification methods favor rewards as well, substituting goals to increase the probability of observing rewarded transitions (Andrychowicz et al., 2017; Cideron et al., 2019; Lair et al., 2019; Colas et al., 2020a). In data collection however, adapting training distributions towards more rewarded experience leads the agent to focus on tasks that are already solved. Because collecting data from already solved tasks hinders learning, data collection ACL methods rather focus on other surrogate objectives.

Intermediate difficulty – A more natural surrogate objective for data collection is *intermediate difficulty*. Intuitively, agents should target tasks that are neither too easy (already solved) nor too difficult (unsolvable) to maximize their learning progress. Intermediate difficulty has been used to adapt the distribution of initial states from which to perform a hard task (Florensa et al., 2017; Salimans & Chen, 2018; Ivanovic et al., 2018). This objective is also implemented in GOAL-GAN, where a curriculum generator based on a Generative Adversarial Network is trained to propose goals for which the agent reaches intermediate performance (Florensa et al., 2018). Racaniere et al. (2020) further introduced a *judge network* trained to predict the feasibility of a given goal for the current learner. Instead of labelling tasks with an intermediate level of difficulty as in GOAL-GAN, their SETTER-SOLVER model generates goals associated to a random feasibility uniformly sampled from $[0, 1]$. The type of goals varies as the agent progresses, but the agent is always asked to perform goals sampled from a distribution balanced in terms of feasibility. In Sukhbaatar et al. (2018), tasks are generated by an RL policy trained to propose either goals or initial states so that the resulting navigation task is of intermediate difficulty w.r.t. the current agent. Intermediate difficulty ACL has also been driving successes in Sim2Real applications, where it sequences *domain randomizations* to train policies that are robust enough to generalize from simulators to real-world robots (Mehta et al., 2019; OpenAI et al., 2019). OpenAI et al. (2019) trains a robotic hand control policy to solve a Rubik's cube by automatically adjusting the task distribution so that the agent achieves decent performance while still being challenged.

Learning progress – The *Obj* objective of ACL methods can be seen as the maximization of a *global learning progress*: the difference between the final score and the initial score:

$$Obj \Leftrightarrow \max_{\mathcal{D}} \int_{\tau \sim \mathcal{T}_{target}} \left[P_\tau^\pi \mid \mathcal{T}_{train}^\pi\right] - \left[P_\tau^\pi \mid \emptyset\right] \mathrm{d}\tau.$$

This global learning progress is difficult to optimize as the impact of each task selection may not be easily traced to the final score after E training episodes, especially when E is large. Instead, one can use measures of competence learning progress localized in space and time, as in aforementioned developmental robotics works (section 2.2.2). This follows the intuition that maximizing LP here and now (modulo some exploration required to measure LP) will eventually result in maximizing global, long-term LP. In multi-task or multi-goal settings, the agent first focuses on tasks/goals where it learns the most and then moves towards more difficult tasks after the earlier tasks have been mastered (i.e. when $LP \to 0$). *Intermediate difficulty* can be seen as a proxy for an expected LP, but might get stuck in areas of the task space where the agent achieves intermediate scores but cannot improve.

LP maximization is usually framed as a multi-armed bandit problem where tasks are arms and LP measures are associated values. Maximizing LP values was shown optimal under the assumption of concave learning profiles (Lopes & Oudeyer, 2012a). Both Matiisen et al. (2017) and Mysore et al. (2018) measure LP as the estimated derivative of the performance for each task in a discrete set (Minecraft mazes and Sonic the Hedgehog levels, respectively) and apply a multi-armed bandit algorithm to automatically build a curriculum for their learning agents. In a similar way, CURIOUS (Colas et al., 2019) uses LP to select goal spaces to sample from in a simulated robotic arm setup. There, LP is

also used to bias the sampling of transition used for policy updates towards high-LP goals. Our proposed ALP-GMM algorithm uses LP to organize the presentation of procedurally-generated Bipedal-Walker environments sampled from a continuous task space through a stochastic parameterization (chapter 4). Based on pairs of task parameters and their associated LP scores previously collected, ALP-GMM fits a Gaussian mixture model and samples task parameters from a Gaussian selected proportionally to its mean LP. LP can also be used to guide the choice of accuracy requirements in a reaching task (Fournier et al., 2018), or to train a *replay policy* via RL to sample transitions for policy updates (Zha et al., 2019).

Diversity – Some ACL methods choose to maximize measures of diversity (also called novelty or low density). In multi-goal settings for example, ACL might favor goals from low-density areas either as targets (Pong et al., 2020) or as substitute goals for data exploitation (Fang et al., 2019). Similarly, Zhao & Tresp (2018b) biases the sampling of trajectories falling into low density areas of the trajectory space. In single-task RL, *count-based* approaches introduce internal reward functions as decreasing functions of the state visitation count, guiding agent towards rarely visited areas of the state space (Bellemare et al., 2016). Through a variational expectation-maximization framework, Jabri et al. (2019) propose to alternatively update a latent skill representation from experimental data (as in Eysenbach et al. (2019)) and to meta-learn a policy to adapt quickly to tasks constructed by deriving a reward function from sampled skills. Other algorithms do not optimize directly for diversity but use heuristics to maintain it. For instance, ALP-GMM maintains exploration by using a residual uniform task sampling and Bansal et al. (2018) sample opponents from past versions of different policies to maintain diversity.

Surprise – Some ACL methods train transition models and compute intrinsic rewards based on their prediction errors (Pathak et al., 2017; Burda et al., 2019b) or based on the disagreement (variance) between several models from an ensemble (Shyam et al., 2019; Pathak et al., 2019). The general idea is that models tend to give bad prediction (or disagree) for states rarely visited, thus inducing a bias towards less visited states. However, a model might show high prediction errors on stochastic parts of the environment – i.e. the noisy-TV problem (Pathak et al., 2017) – a phenomenon that does not appear with model disagreement, as all models of the ensemble eventually learn to predict the (same) mean prediction (Pathak et al., 2019). Other works bias the sampling of transitions for policy update depending on their temporal-difference error, i.e. the difference between the transition's value and its next-step bootstrap estimation (Schaul et al., 2016; Horgan et al., 2018). Whether the error computation involves value models or transition models, ACL mechanisms favor states related to maximal *surprise*, i.e. a maximal difference between the expected (model prediction) and the truth.

Energy – In the data exploitation phase of multi-goal settings, Zhao & Tresp (2018a) prioritize transitions from *high-energy* trajectories (e.g. kinetic energy) while Colas et al. (2019) prioritize transitions where the object relevant to the goal moved (e.g. cube movement in a cube pushing task).

Adversarial reward maximization – Self-Play is a form of ACL which optimizes

agents' performance when opposed to current or past versions of themselves, an objective that we call *Adversarial Reward Maximization (ARM)* (Hernandez et al., 2019). While agents from Silver et al. (2017) and Baker et al. (2020) always oppose copies of themselves, Bansal et al. (2018) train several policies in parallel and fill a pool of opponents made of current and past versions of all policies. This maintains a diversity of opponents, which helps to fight catastrophic forgetting and to improve robustness. In the multi-agent game Starcraft II, Vinyals et al. (2019) train three main policies in parallel (one for each of the available player types). They maintain a *league* of opponents composed of current and past versions of both the three main policies and additional adversary policies. Opponents are not selected at random but to be challenging (as measured by winning rates).

Conclusion

In this survey, we unify the wide range of ACL mechanisms used in symbiosis with DRL under a common framework. ACL mechanisms are used with a particular goal in mind (e.g. organizing exploration, solving hard tasks, etc., § *Why use ACL?*). It controls a particular element of task MDPs (e.g. $\mathcal{S}, \mathcal{R}, \rho_0$, § *What does ACL control?*) and maximizes a surrogate objective to achieve its goal (e.g. diversity, learning progress, § *What does ACL optimize?*). Table 2.1 organizes the main works surveyed here along these three dimensions. This survey presents what has been implemented in the past, and thus, by contrast, highlight potential new avenues for ACL & DRL. We refrain from expanding on potential future work, and leave this discussion to chapter 8. A core objective of the present work is to contribute to the ACL literature, e.g. by proposing LP-based ACL methods for black-box learners (chapter 4), by presenting a standardized testbed for ACL (chapter 5), and by proposing to study *meta*-ACL methods to efficiently train *multiple* students (chapter 6).

Algorithm	Why use ACL?	What does ACL control?	What does ACL optimize?
ACL for Data Collection:			
ALP-GMM (chapter 4)	Generalization	Environments ($\mathcal{S}$) (PCG)	LP
ADR (OpenAI) (OpenAI et al., 2019)	Generalization	Environments ($\mathcal{S}, \mathcal{P}$) (PCG)	Intermediate difficulty
ADR (Mila) (Mehta et al., 2019)	Generalization	Environments ($\mathcal{P}$) (PCG)	Intermediate diff. & Diversity
RC (Florensa et al., 2017)	Hard Task	Initial states (ρ_0)	Intermediate difficulty
1-demo RC (Salimans & Chen, 2018)	Hard Task	Initial states (ρ_0)	Intermediate difficulty
BaRC (Ivanovic et al., 2018)	Hard Task	Initial states (ρ_0)	Intermediate difficulty
Asym. SP (Sukhbaatar et al., 2018)	Multi-Goal	Goals ($\mathcal{G}$), initial states (ρ_0)	Intermediate difficulty
GOAL-GAN (Florensa et al., 2018)	Multi-Goal	Goals ($\mathcal{G}$)	Intermediate difficulty
Setter-Solver (Racaniere et al., 2020)	Multi-Goal	Goals ($\mathcal{G}$)	Intermediate difficulty
RgC (Mysore et al., 2018)	Generalization	Environments ($\mathcal{S}$) (DS)	LP
TSCL (Matiisen et al., 2017)	Hard Task	Environments ($\mathcal{S}$) (DS)	LP
Acc-based CL (Fournier et al., 2018)	Multi-Goal	Reward function ($\mathcal{R}$)	LP
SKEW-FIT (Pong et al., 2020)	Open-Ended Explo.	Goals ($\mathcal{G}$) (from pixels)	Diversity
DIAYN (Eysenbach et al., 2019)	Open-Ended Explo.	Reward functions ($\mathcal{R}$)	Diversity
CARML (Jabri et al., 2019)	Open-Ended Explo.	Reward functions ($\mathcal{R}$)	Diversity
RARL (Pinto et al., 2017)	Generalization	Opponents ($\mathcal{P}$)	ARM
AlphaGO Zero (Silver et al., 2017)	Generalization	Opponents ($\mathcal{P}$)	ARM
HIDE&SEEK (Baker et al., 2020)	Generalization	Opponents ($\mathcal{P}$)	ARM
AlphaStar (Vinyals et al., 2019)	Generalization	Opponents ($\mathcal{P}$)	ARM & Diversity
Competitive SP (Bansal et al., 2018)	Generalization	Opponents ($\mathcal{P}$)	ARM & Diversity
Count-based (Bellemare et al., 2016)	Hard Task	Reward functions ($\mathcal{R}$)	Diversity
RND (Burda et al., 2019b)	Hard Task	Reward functions ($\mathcal{R}$)	Surprise (model error)
ICM (Pathak et al., 2017)	Hard Task	Reward functions ($\mathcal{R}$)	Surprise (model error)
Disagreement (Pathak et al., 2019)	Hard Task	Reward functions ($\mathcal{R}$)	Surprise (model disagreement)
MAX (Shyam et al., 2019)	Hard Task	Reward functions ($\mathcal{R}$)	Surprise (model disagreement)
CURIOUS (Colas et al., 2019)	Multi-goal	Goals ($\mathcal{G}$)	LP
LE2 (Lair et al., 2019)	Open-Ended Explo.	Goals ($\mathcal{G}$)	Reward & Diversity
ACL for Data Exploitation:			
HER (Andrychowicz et al., 2017)	Multi-goal	Transition modification ($\mathcal{G}$)	Reward
HER-curriculum (Fang et al., 2019)	Multi-goal	Transition modification ($\mathcal{G}$)	Diversity
Language HER (Cideron et al., 2019)	Multi-goal	Transition modification ($\mathcal{G}$)	Reward
PER (Schaul et al., 2016)	Performance boost	Transition selection ($\mathcal{S} \times \mathcal{A}$)	Surprise (TD-error)
Curiosity Prio. (Zhao & Tresp, 2018b)	Multi-goal	Transition selection ($\mathcal{S} \times \mathcal{A}$)	Diversity
En. Based ER (Zhao & Tresp, 2018a)	Multi-goal	Transition selection ($\mathcal{S} \times \mathcal{A}$)	Energy
CURIOUS (Colas et al., 2019)	Multi-goal	Trans. select. & mod. ($\mathcal{S} \times \mathcal{A}, \mathcal{G}$)	LP & Energy
LE2 (Lair et al., 2019)	Open-Ended Explo.	Trans. select. & mod. ($\mathcal{S} \times \mathcal{A}, \mathcal{G}$)	Reward
IMAGINE (Colas et al., 2020a)	Open-Ended Explo.	Trans. select. & mod. ($\mathcal{S} \times \mathcal{A}, \mathcal{G}$)	Reward

Table 2.1: **Classification of the surveyed papers.** The classification is organized along the three dimensions defined in the above text. In *Why use ACL*, we only report the main objective of each work. When ACL controls the selection of environments, we precise whether it is selecting them from a discrete set (*DS*) or through parametric Procedural Content Generation (*PCG*). We abbreviate *adversarial reward maximization* by *ARM* and *learning progress* by *LP*.

2.4 Chapter Summary

This chapter presented important theoretical and experimental background for the present research on building developmental machine learners.

Section 2.1 formalized the definition of tasks as episodic MDPs, in which RL agents learn behavior policies aiming to maximize reward collection. We discussed how RL and *deep* RL algorithms have been successfully used for a variety of control tasks (section 2.1.3), e.g. in discrete and continuous action scenarios, for real and simulated robotics applications. Beyond single task learning, an important challenge in DRL is to enable agents to *generalize* their policy to multiple tasks. Section 2.2 presented an overview of the developmental robotics field, which seeks to understand and model human learning through various perspectives, such as sensorimotor development, social interactions or maturational constraints (section 2.2.1). Common to these areas of research is the study of *intrinsic motivation*, i.e. how it enables or improves learning abilities in developmental robotic systems. We particularly focused on the notion of *learning progress*, an efficient form of intrinsic motivation, and how it can be used to organize autonomous learning and exploration in robotic scenarios (section 2.2.2). Finally, at the crossroad of both RL and DevRob, section 2.3 formalized the notion of *automatic curriculum learning*, i.e. the problem of efficiently adapting the training of a given learning student based on its evolving abilities. We then proposed a survey of recent works leveraging ACL methods for DRL agents (section 2.3.2).

In the following chapters, we propose computational experiments building up on these core concepts and existing literature. More precisely:

- Chapter 3 presents computational experiments to further study how LP can guide population-based learners in complex settings.
- Inspired by the demonstrated advantages of LP-based IMS in DevRob scenarios, chapter 4 presents a new LP-based ACL algorithm well suited for DRL agents learning within continuous task spaces.
- Based on our survey, chapter 5 identifies a lack of comparative analysis in ACL works and proposes a new test platform to easily characterize existing and future teacher algorithms.
- Chapter 6 presents and formalizes a more general form of ACL, i.e. meta-ACL, aiming to efficiently organize curriculum generation for *multiple* students.
- Finally, given the importance of social interactions for cognitive maturation and the relative lack of DRL works considering such scenarios, chapter 7 presents an invitation to study the acquisition of complex social skills around a suite of simulated social environments.

Chapter 3

Automatic Curriculum Learning for Population-Based Agents: a Case-Study

Contents

Intrinsically motivated spontaneous exploration is a key enabler of autonomous lifelong learning in human children. It enables the discovery and acquisition of large repertoires of skills through self-generation, self-selection, self-ordering and self-experimentation of learning goals (see section 1.1). As discussed in section 2.2.2, multiple computational models trying to emulate such learning dynamics, formalized as Intrinsically Motivated Goal Exploration Processes (IMGEPs), have been proposed. In this chapter, we present experiments on Active Model Babbling (AMB), a particularly efficient form of IMGEP proposed in Forestier et al. (2017). AMB relies on the empirical estimation of Learning Progress (LP) between multiple objects – i.e. multiple *goal spaces* – to organize learning. As in Forestier et al. (2017), this work will apply AMB to a population-based agent such that it can autonomously grow a set of policy experts able to cover a wide range of behaviors. This work can be seen as a preliminary step before transferring similar DevRob algorithmic ideas into deep reinforcement learning frameworks.

More precisely, this chapter focuses on experiments done on a simulated tool-use environment designed using the Malmo platform (Johnson et al., 2016), which allows researchers to train agents on Minecraft environments (Minecraft[1] is a popular sandbox 3D video game). Contributions described in this chapter are part of a larger research project presented in Forestier et al. (2017), which features a mathematical formalization of the IMGEP framework along with two additional experimental environments: The *2D*

[1] https://www.minecraft.net/about-minecraft

Simulated Tool-Use Environment, and the *Robotic Tool-Use Environment* (see section 2.2.2 and figure 2.6).

Main contributions:

- We present *Malmo Mountain Cart*[2], an open-source Minecraft environment featuring tool-use and nested interactions, which provides a challenging environment to study and assess the performance of IMGEPs.
- Using this environment, we compare several variants of IMGEP algorithms in terms of sample efficiency to discover a diversity of behavioral features. Importantly, as an illustration of the versatility of IMGEP algorithms, we showcase that such approaches can successfully leverage neural networks as controllers, rather than more classical multi-step closed-loop controllers.

3.1 AMB: a Modular Population-based IMGEP algorithm

In this section, after formalizing the interaction setup considered in our experiments, we present the AMB algorithm, with a focus on its LP computation procedure.

Agent interaction pipeline

Our experiments can be framed as an under specified episodic MDP problem, in which there is no extrinsic reward function. Given such an environment, we consider an agent that executes continuous actions $a \in \mathcal{A}$ in continuous states $s \in \mathcal{S}$. This agent is *population-based*: instead of corresponding to a single parametric policy optimized over training, this agent is iteratively growing a collection of sub-policies π^i, and uses a meta procedure to decide which sub-policy to use for each new episode. To foster exploration, such approaches usually rely on applying parameter mutations to used policies, such that new behaviors can be potentially discovered. We denote $o \in \mathcal{O}$, and call *outcome*, a vector of measurements (also called descriptors or features) characterizing the behavior of the agent and of the environment during one episode. The descriptors in the outcome vector o characterize properties of the agent's behavioral trajectory (e.g. the end-of-episode positions of objects in the scene).

We assume that this MDP is goal-conditioned: the agent is given access to a goal space $\mathcal{G}$, and is able to compute episodic fitness measures $f_g(o)$, i.e. an agent-specific assessment of the agent's ability to reach goal g. We assume that given the outcome o of an interaction episode, the agent can compute $f_g(o)$ for any $g \in \mathcal{G}$. Given this, the IMGEP agent explores the environment by sampling goals in $\mathcal{G}$ and searching for good solutions to those goals, and expands its population of sub-policies towards the ability of reaching any goal from any state.

[2] https://github.com/rPortelas/malmo_mountain_cart

Active Model Babbling

The Active Model Babbling learning algorithm (Forestier et al., 2017) assumes that both the outcome space $\mathcal{O}$ and goal space $\mathcal{G}$ can be decomposed into multiple lower-dimensional (sub-)spaces – $\mathcal{G} = \bigcup_k \mathcal{G}_k$ and $\mathcal{O} = \bigcup_k \mathcal{O}_k$ – i.e. modules corresponding to the sensory feedback of specific objects. More precisely, we assume that $\forall k, \mathcal{G}_k = \mathcal{O}_k$, i.e. goals are specific object-centered outcomes to attain such that a fitness measure $f_g(o_k)$ is maximized. In our case we consider end-positions of objects as goals (e.g. the 2D end position of the agent in our environment), but Forestier et al. (2017) also features experiments in which goals are episode-long object trajectories. Because of this object-centered perspective, AMB is considered a *modular* IMGEP approach, which associates each object with one independent learning module. This allows to compute empirical per-object learning progress estimates, used to create efficient learning curricula. For each new interaction episode, The AMB exploration and learning procedure can be described as follows (see support figure 3.1):

(1): Select object to explore (using LP estimates), corresponding to a goal space $\mathcal{G}_k$, and sample (at random) a goal $g \in \mathcal{G}_k$ to reach.

(2): Select a policy π_θ^i to reach g. 80% of episodes, mutate its parameters, and execute the new policy $\pi_{\theta'}^i$. In 20% of episodes, directly execute π_θ^i.

(3): Compute episode outcome o^{new} from the agent's behavioral trajectory.

(4*a*): If the policy was mutated, update population of policies with $\pi_{\theta'}^i$ (with its associated outcome o^{new}). *End.*

(4*b*): If the policy was not mutated, update the object's estimated LP value based on $f_g(o_k^{new})$. In our experiments we compute this fitness measure using the object-specific Euclidean distance between reached end-position (o_k^{new}) and target goal (g). *End.*

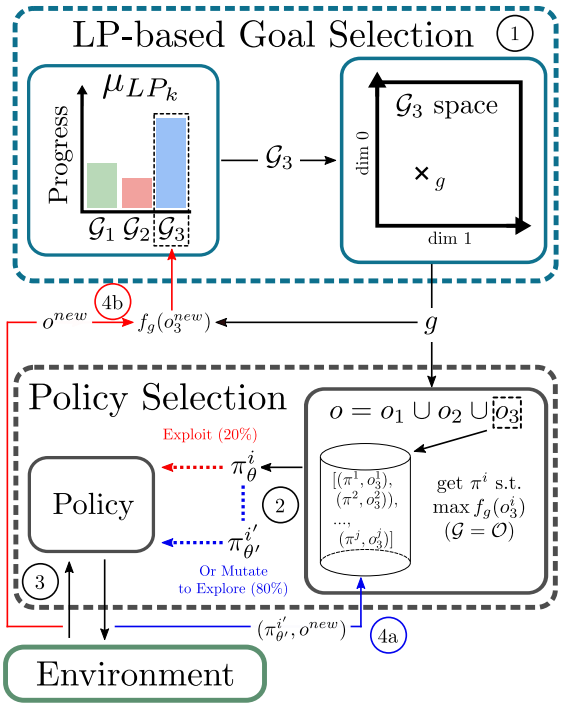


Figure 3.1: Interaction pipeline of AMB

LP estimation and goal sampling – To estimate the empirical learning progress LP_k made to reach the current goal $g \in \mathcal{G}_k$, the agent compares the object-specific outcome o_k with the outcome o'_k obtained for the previous goal g' most similar (Euclidean distance) to g: $LP_k = f_g(o_k) - f_g(o'_k)$. Note that here we assume that the initial state

of the learner is identical across each new episode[3]. Such LP estimates can be used for goal sampling using a non-stationary bandit algorithm in which an arm is a distinct goal space. We use an approach similar to the EXP4 algorithm (Auer et al., 2002). Our bandit keeps track of a running average μ_{LP_k} of the intrinsic rewards LP_k associated to the current goal space $\mathcal{G}_k$. With probability 20%, it samples a random goal space $\mathcal{G}_k$ (for exploration), and with probability 80%, the probability to sample $\mathcal{G}_k$ is proportional to (the exponential of) LP_k.

Policy selection – Population-based agents iteratively grow a set of expert policies. Our policy selection approach is based on recording for each policy π_θ in our database the associated outcome o they produced in the environment. Given a new goal $g \in \mathcal{G}_k$ to attain, which correspond to a specific object-outcome o_k to produce, we re-use for interaction the policy π_θ^i whose associated object-outcome o_k^i maximizes the agent's fitness measure $f_g(o_k^i)$. In our experiments, we use Euclidean distances between a goal and an outcome (the closer, the better) to measure fitness. This nearest neighbor procedure can be efficiently implemented with a kd-tree (Bentley, 1975). Once a policy is selected, it is either used as is, for exploitation and to update LP estimates on $\mathcal{G}_k$, or it is mutated by adding a Gaussian noise to all parameters of the policy. In the latter case, the new policy $\pi_{\theta'}^{i'}$ and its associated outcome $o^{i'}$ are added to the agent's policy database.

AMB does not assume that all goals are achievable. The agent autonomously select its goals but does not know initially which goals are achievable or not, which are easy and which are difficult, nor if certain goals need to be explored so that other goals become achievable. The efficiency of such goal exploration processes relies on a form of hindsight learning that leverages the fact that the data collected when targeting a goal can be informative to find better solutions to other goals. For example, a learner trying to achieve the goal of pushing an object on the right but actually pushing it on the left fails to progress on this goal, but learns as a side effect how to push it on the left. Additional implementation details can be found in Forestier et al. (2017).

3.1.1 The Minecraft Mountain Cart environment

Minecraft Mountain Cart (MMC), shown in figure 3.2, is an episodic environment in which the agent starts on the left of a rectangular arena and is given ten seconds (40 steps) to act on the environment using 2 continuous commands: *move* and *strafe*, both using values in $[-1; 1]$. *move(1)* moves the agent forward at full speed, *move(-0.1)* moves the agent slowly backward, etc. Similarly, *strafe(1)* moves the agent left at full speed and *strafe(-0.1)* moves it slowly to the right. Additionally, a third binary action allows the agent to use the currently handled tool.

The first challenge of this environment is to learn how to navigate within the arena's boundaries without falling in water holes (from which the agent cannot get out). Proper

[3] AMB has also been used in experiments with varying initial sensory context in Forestier et al. (2017). In such cases, LP estimates can be computed by concatenating outcome vectors o_k and o'_k with their respective context vectors c and c', e.g. the full initial state or a subset encompassing inter-episode variations.

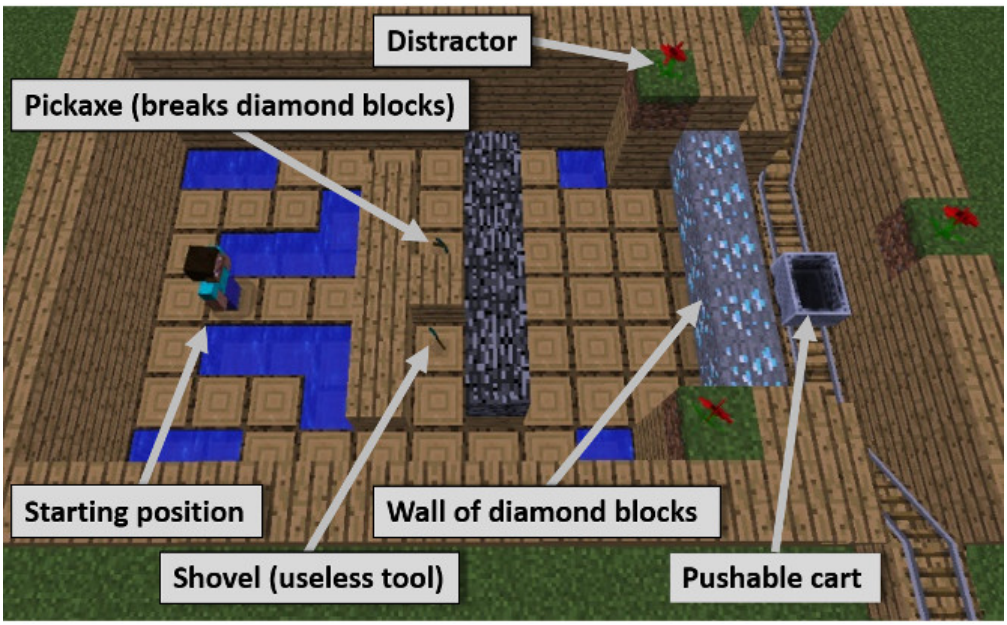


Figure 3.2: Minecraft Mountain Cart environment. If the agent manages to avoid falling into water holes it may retrieve and use a pickaxe to break diamond blocks and access the cart. A shovel (useless tool) serves as a controllable distractor, along with 3 flowers (uncontrollable distractors).

navigation might lead the agent to discover one of the two tools of the environment: a shovel and a pickaxe. The former is of no use, but the latter enables to break diamond blocks located further ahead in the arena. A last possible interaction is for the agent to get close enough to the cart to move it along its railroad. If given enough speed, the cart is able to climb the left or right slope.

The outcome of an episode is a vector composed of the end position of the agent (2D), shovel (2D), pickaxe (2D), cart (1D) and 3 distractors (2D each) positions along with a binary vector (5D) encoding the 5 diamond blocks' states. This environment is interesting to study modular IMGEP approaches since it is composed of a set of linked tasks of increasing complexity. Exploring how to navigate will help to discover the tools and, eventually, will allow the agent to break blocks and move the cart.

Contrary to previous IMGEP works, including parallel experiments from Forestier et al. (2017), In MMC we propose to study whether IMGEPs can be used with neural networks policies, instead of more classical robotics controllers such as Dynamic Movement Primitives (Schaal, 2006), which outputs multi-step continuous trajectories regardless of sensory inputs (i.e. open-loop controllers). Our considered neural network policies are closed-loop systems: In each step, they receive an observation vector, similar to the outcome vector: it provides the current positions of all objects normalized in $[-1; 1]$ (18D). We use neural nets composed of 1 hidden layer of 64 Relu units and 3D action output with tanh activation functions.

3.2 Experimental Results

In this section, to better understand how AMB and more generally IMGEPs can shape the development of autonomous agents, we analyse the performance of such approaches on MMC. We study AMB along with the following conditions:

Random (Rdm) Lower baseline, uses a randomly sampled policy for each new episode (sanity check).

Single Goal Space (SGS) Does not use any form of curriculum learning, i.e. it always chooses goals for the same complex target object, which is the cart in MMC.

Flat Random Goal Babbling (F-RGB) This IMGEP condition uses the entire goal space $\mathcal{G}$, containing all the variables of all objects. We use this condition to compare modular and non-modular (a.k.a. flat) IMGEPs.

Random Motor Babbling (RMB) An ablation of AMB, which samples objects/goal spaces to explore for each new episode at random (not using LP estimates).

Fixed Curriculum (FC) AMB variant using a curriculum designed by hand: instead of sampling goal spaces using LP estimates, it follows an expert training schedule: FC agents spend an equal amount of training episodes on each controllable object/goal-space in the following sequence: agent position, shovel, pickaxe, blocks, cart.

Table 3.1 shows a summary of the exploration results after 40k training episodes, in all goal spaces for all conditions. The remainder of this experimental section proposes multiple in-depth analysis of these results.

Space \ Condition	Rdm	SGS	F-RGB	RMB	AMB	FC
Agent Pos.	28,29,30	29,29,30	34,36,40	48,50,54	55,58,61	59,**63**,67
Shovel	5,5,6	5,6,7	8,11,13	25,27,30	32,34,37	34,**37**,42
Pickaxe	6,6,7	6,7,8	11,15,19	33,35,39	41,45,48	43,**51**,61
Blocks	3,3,3	3,3,3	3,11,19	69,77,84	73,**84**,93	100,**100**,100
Cart	0,0,0	0,0,0	0,0,1	5,162,409	56,360,886	386,**787**,1207

Table 3.1: Exploration results in all environments and conditions. We give the 25, 50 and 75 percentiles of the exploration results of all seeds (20 runs for Rdm, SGS, F-RGB, FC and 42 for AMB and RMB). Exploration measures the percentage of reached cells in a discretization of each goal space: we use 450 bins for each of the agent, pickaxe and shovel spaces (15 on the x-axis, 30 on the y-axis). The same measure is used for the block space, which is discrete with 32 possible combinations. For the cart space we measure exploration as the number of different outcomes reached.

3.2.1 Intrinsically Motivated Goal Exploration

Figure 3.3 shows an exploration map of a typical AMB run in MMC after 40k iterations. The agent successfully managed to learn all possible interactions within the environment, i.e. it learned to (1) navigate within the arena boundaries, (2) move the pickaxe and shovel, (3) use the pickaxe to break blocks and (4) move the cart located behind these blocks.

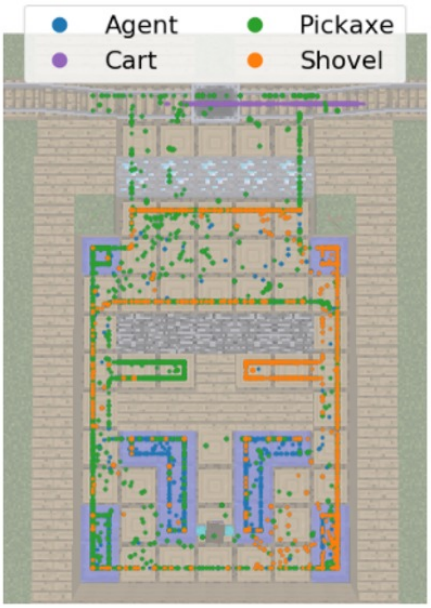


Figure 3.3: Example of overall exploration map of one AMB agent. We plot the end position of the agent, the agent with pickaxe, the agent with shovel, and the cart that were reached throughout training.

Discoveries – In order to understand the nested interaction structure of the exploration problem in MMC, we can look in more details how agents succeeded to move objects while exploring other objects. Indeed, in a nested interaction scenario, it is through the exploration of accessible objects, e.g. by acting towards the completion of a pickaxe goal, that interactions with new objects can be discovered, e.g. discovering that the pickaxe can break blocks.

To quantify exploration dependencies between objects in MMC, fig. 3.4 shows the proportion of episodes where an object of interest has been moved depending on the currently explored object. For instance, a *blue* curve from Fig. 3.4 (a) corresponds to the evolving percentage of episodes in which, while following an agent-end-position goal, the agent interacts (i.e. moves) the pickaxe. Random exploration with neural networks in the MMC environment is extremely challenging. An agent following random policies has 0.04% chances to discover the pickaxe, 0.00025% chances to break a single block and it never managed to move the cart (over 800k episodes). IMGEP agents reach better performances by leveraging the sequential nature of the environment: when exploring the agent space there is around 10% chances to discover the pickaxe, and exploring the pickaxe space has around 1% chances to break blocks. Finally, exploring the block space has about 8% chances to lead an agent to discover the cart.

3.2.2 Learned Skills

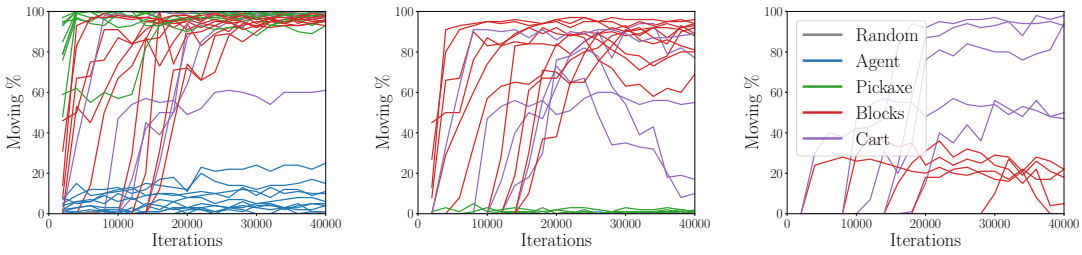
In addition to exploration measures, we also perform post-training tests of competence. Using modular approaches allows to easily test competence on specific objects of the environment. Fig. 3.5b shows an example in the cart space for an AMB agent. This agent successfully learned to move the cart close to the 5 queried locations.

For each of the RMB, AMB and FC runs, we performed a statistical analysis of competence in the cart and pickaxe spaces using 1000 and 800 uniformly generated goals, respectively. We were also able to test SGS for cart competence, as this condition was specifically designed to focus on this space. A goal is considered reached if the Euclidean distance between the outcome and the goal is lower than 0.05 in the normalized space (in range $[-1, 1]$) for each object. Results are shown in table 3.5c. Note that, since the pickaxe goal space is loosely defined as a rectangular area around the environment's arena, many goals are not reachable. SGS agents never managed to move the cart for any of the given goals. AMB appears to be significantly better than RMB on the pickaxe space ($p < 0.01$ on Welch's t-tests). However it is not in the cart space ($p = 0.09$), which might be due to the stochasticity of the environment. FC is not significantly better than AMB on the cart and pickaxe spaces.

3.2.3 Curriculum Learning

A modular sensory representation based on objects allows AMB agents to self-monitor their learning progress to control each object, and to accordingly explore objects with high learning progress. Here, we focus on the qualitative analysis of AMB's intrinsic reward generation along with a comparative analysis of aggregated exploration performances w.r.t other IMGEP variants and ablations.

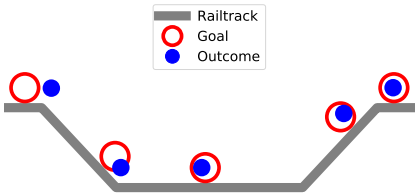
Intrinsic rewards based on LP – Figure 3.6 shows the evolution of intrinsic rewards, i.e. LP estimates, of two AMB agents in MMC. Both agents first explore the simpler agent position space and then quickly improve on the shovel and pickaxe spaces. Exploring the



(a) Pickaxe (b) Blocks (c) Cart

Figure 3.4: Stepping-stone structure of the MMC environment. We show the proportion of iterations that allowed to (a) move the pickaxe, (b) mine diamond blocks, and (c) move the cart, depending on the currently explored goal space (or random movements), for 10 AMB agents with different seeds. Exploring the agent space helps discover the pickaxe, exploring the pickaxe helps discover the blocks, and exploring the blocks helps discover the cart.

(a) Agent moving the cart



(b) 5 cart goals

	Pickaxe goals	Cart goals
FC	39,49,55	12,17,25
AMB	41,45,49	8,11,18
RMB	37,40,43	6,9,15
SGS	N/A	0,0,0

(c) Competence results

Figure 3.5: Learned skills in the Minecraft Mountain Cart. **(a):** Example of one AMB agent's trajectory for a cart goal. **(b):** Example of five final cart positions reached by an AMB agent when tasked to reach five different targets. This agent successfully learned to push the cart along the track. **(c):** Overall competence results in Minecraft Mountain Cart. We give the 25, 50 and 75 percentiles of the competence results of all seeds.

pickaxe space leads to discover how to progress in the block space. Finally, after some progress in the block space, the cart is discovered after 14k episodes for the first agent (left) and 26k episodes for the other (right). The 3 distracting flowers have an interest strictly equal to zero in both runs, which allows the agent to focus on controllable aspects of its environment. Evaluating the learning progress to move objects allows AMB agents to self-organize a learning curriculum focusing on the objects currently yielding the most progress and to discover stepping stones one after the other.

Comparative results – Fig. 3.7 shows the evolution of exploration of all conditions in MMC. Agents focusing their goal sampling on the cart space (SGS) have low performances across all goal spaces, especially for the cart and block spaces which are never discovered. Agents using learning progress sampling (AMB) explore significantly more than random sampling agents (RMB) across all goal spaces (Welch's t-tests at 40k iterations, $p < 0.04$). Contrary to RMB agents, which randomly choose objects to focus on, the strength of AMB is to focus only on objects that are learnable (distractor objects are ignored), and to reduce the relative interest of objects already explored for some time. Agents following a hard-coded curriculum (FC) reach higher median performances than AMB agents on every goal spaces.

F-RGB do not manage to reach more than 15% exploration when AMB and RMB reach 45% and 35%, respectively. Modular approaches significantly outperform F-RGB across all goal spaces (Welch's t-tests at 40k iterations, $p < 0.001$). Random agents do not manage to explore the block and cart spaces. The modular representation of the sensory space thus

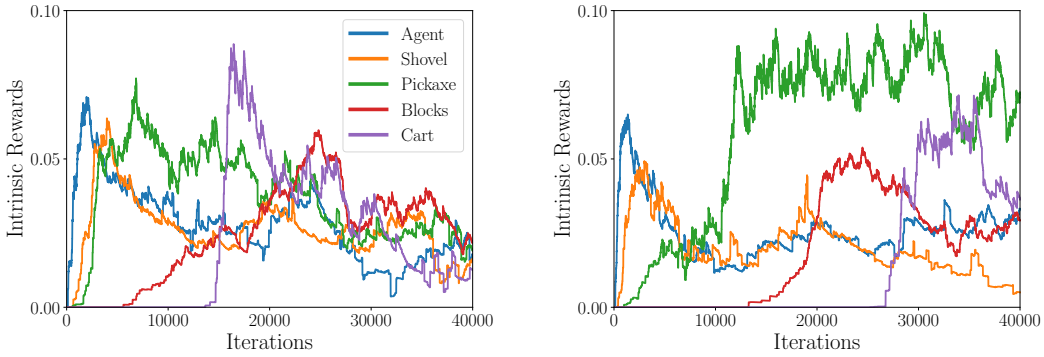
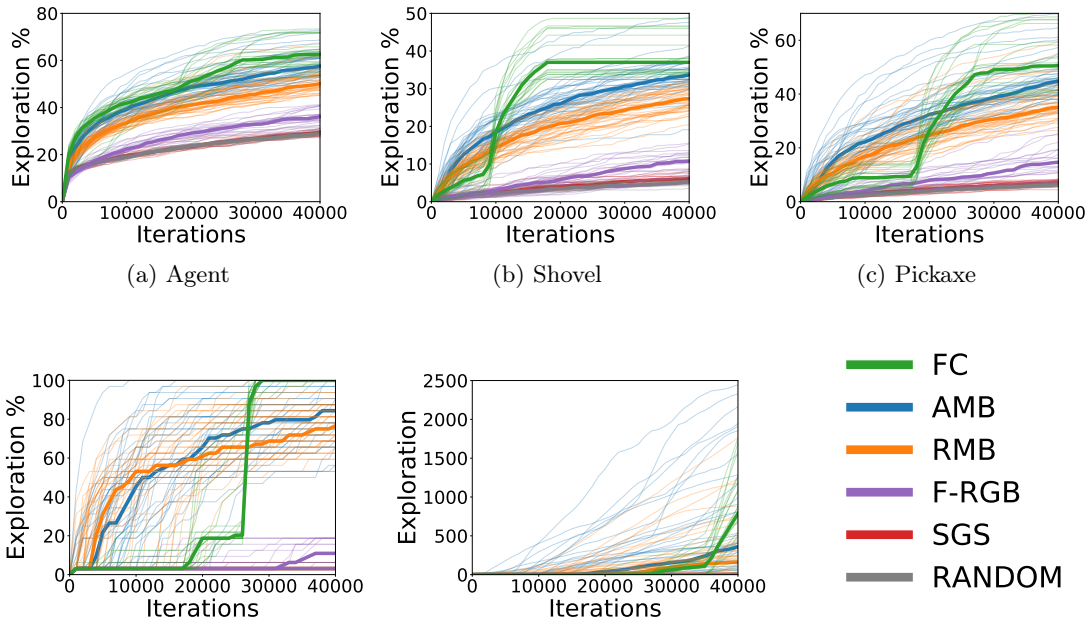


Figure 3.6: Examples of intrinsic rewards (i.e. LP-estimates) for two AMB agents in MMC. Agents first focus on exploring the space of their position until they discover the shovel or the pickaxe and start making progress to move them. When they discover how to mine blocks with the pickaxe and to push the cart, they make progress in those goal spaces, get intrinsic rewards and thus focus more on these.

greatly improves exploration efficiency compared to a flat intricate representation of the whole sensory feedback, as it allows agents to consider the different objects independently to monitor their behavior and select disentangled goals.



(a) Agent (b) Shovel (c) Pickaxe

(d) Blocks (e) Cart

Figure 3.7: Exploration results in MMC. Modular approaches (AMB and RMB) performs significantly better than the flat (F-RGB) approach. Agents actively generating their curriculum (AMB) perform better overall than agents choosing goal spaces randomly (RMB). Focusing on the cart space (SGS) is equivalent to performing random policies (Random). For the agent, pickaxe and shovel spaces, exploration is measured as the cumulative number of reached cells in a discretization of the 2D space. For the block and cart spaces we measure the number of unique outcomes reached.

3.3 Discussion

This first experimental chapter presented the AMB algorithm, an existing population-based modular IMGEP using LP estimates to organize exploration. We designed Minecraft Mountain Cart, a Minecraft tool-use environment with nested interactions, and showcased how AMB (and variants) are able to efficiently create learning curricula for agents growing populations of neural network controllers.

Limits of population-based agents – Although population-based agents can efficiently explore environments through the diversity of policies they can contain, they are also limited by memory requirements. This has to be compared with classical single-policy RL and DRL agent, which have fixed memory requirements, as they only maintain a single policy (ignoring policy backups and eventual Q-networks), trained through back-propagation (LeCun et al., 1989). Besides, compared to the collection of task-expert policies built by population-based approaches, the training of single-policy agents pushes towards finding robust policies, able to handle multiple interaction scenarios. Population-based approaches also bring additional computation complexity at inference time (i.e. action-selection time), as they need to run a meta-procedure to find the optimal policy for a given situation, while a DRL agent with a single policy can be directly launched in the environment. As such, in the remainder of this manuscript, we will focus on designing intrinsically motivated systems well suited for DRL agents (but our work could be used for any type of iterative policy learner, including population-based agents).

Limits of modular agents – The AMB approach can only be efficiently applied to environments for which an expert-defined division of the goal space into separate objects is provided. While such a division is easy to provide for tool-use scenarios with a fixed number of objects, other types of environments do not easily afford it, e.g. locomotion environments, or tool-use scenarios with a dynamically evolving set of objects. In chapter 4, we present ALP-GMM, an ACL approach suited to deal with task spaces without being provided expert knowledge.

From intrinsically motivated autotelic agents to Teacher-Student ACL – The learning system we presented in this chapter, composed of AMB as an intrinsic motivation system and a population-based approach to grow a repertoire of skill-specific policies, can be conceptualized as an *autotelic* agent (Colas et al., 2020b), from the Greek *auto* (self) and *telos* (end, goal). Autotelic agents autonomously select and pursue their own goals in an open-ended fashion. This work, among others (Colas et al., 2019, 2020a,b), showed that LP-based intrinsic signals are particularly efficient to shape self-generated training trajectories. In the next chapter, inspired by this literature, we will present the ALP-GMM algorithm and show how it can be efficiently used as an external (LP-based) *teacher* algorithm to provide a training curriculum that maximize extrinsic reward collection by a (Black-Box) DRL *student*.

Chapter 4

Teacher Algorithms for Black-Box DRL Agents in Continuously Parameterized Environments

Contents

In this chapter, we consider the problem of how an ACL algorithm can enable an *unknown* DRL agent to become good at a skill over a wide range of diverse environments. More precisely, DRL agents are considered as *black-boxes*, i.e. ACL algorithms exclusively rely on observing their students' behaviors, and cannot access to internal representations and algorithms. To do so, we study how such a *teacher* algorithm can learn to generate a learning curriculum, whereby it sequentially samples parameters controlling a stochastic procedural generation of environments, a.k.a. tasks (as explained in section 2.1.1, sampling environments is a specific form of task sampling). Because it does not initially know the capacities of its DRL *student*, a key challenge for the teacher is to discover which tasks are easy, difficult or unlearnable, and in what order to propose them to maximize the efficiency of learning over the learnable ones. To achieve this, this problem is transformed into a surrogate *continuous* bandit problem where the teacher samples tasks in order to maximize absolute learning progress of its student. Unlike in chapter 3, we no longer assume access to a meaningful partitioning of the task space.

We present ALP-GMM, a new ACL algorithm, modeling absolute learning progress with Gaussian mixture models. We also adapt existing ACL algorithms and provide a complete study in the context of DRL. Using parameterized variants of the BipedalWalker environment (a classical locomotion environment in DRL), we study their efficiency to personalize a learning curriculum for different learners (embodiments), their robustness to the ratio of learnable/unlearnable tasks, and their scalability to non-linear and high-dimensional task spaces. Videos and code are available at `https://github.com/flowersteam/teachDeepRL`.

4.1 Introduction

We address the *strategic student problem*. This problem is well known in the developmental robotics community (Lopes & Oudeyer, 2012b), and formalizes a setting where an agent has to sequentially select tasks to train on to maximize its average competence over the whole set of tasks after a given number of interactions. To address this problem, several works (Oudeyer et al., 2007b; Baranes & Oudeyer, 2013; Moulin-Frier et al., 2014), presented in section 2.2.2, proposed to use automatic curriculum learning strategies based on learning progress, and showed that population-based algorithms can benefit from such techniques. Inspired by these initial results, similar approaches were then successfully applied to DRL agents in continuous control scenarios with discrete sets of goals (Colas et al., 2019), i.e. tasks varying only by their reward functions. Promising results were also observed when learning to navigate in discrete sets of environments (Matiisen et al., 2017; Mysore et al., 2018), i.e. tasks varying by their state spaces (see section 2.1.1 for a definition of "task", "goal", and "environment").

Inspired by both this literature and our own experiments with AMB (chapter 3), this second experimental study proposes to assess, for the first time, whether LP-based curriculum learning methods are able to scaffold generalist DRL agents in continuously parameterized environments. We compare the reuse of RIAC (Baranes & Oudeyer, 2009) in this new context to Absolute Learning Progress - Gaussian Mixture Model, a.k.a. ALP-GMM, a new GMM-based approach inspired by earlier work on developmental robotics (Moulin-Frier et al., 2014), that is well suited for DRL agents. Both these methods rely on Absolute Learning Progress (ALP) as a surrogate objective to optimize with the aim to maximize average competence over a given task space. Importantly, we consider *stochastic* tasks, i.e. a task-encoding parameter vector does not map to a single deterministic task but rather to a distribution of tasks with similar properties. Studying such non-deterministic training regimes is closer to real-world scenarios where stochasticity is an issue.

Recent work (Wang et al., 2019b) already showed impressive results in continuously parameterized environments. The POET approach proved itself to be capable of generating and mastering a large set of diverse BipedalWalker environments. However, their work differs from ours as they evolve a population of agents where each individual agent is specialized for a single specific deterministic environment, whereas we seek to scaffold the learning of a single generalist agent in a training regime where it never sees the same exact environment twice.

As our approaches make few assumptions, they can deal with ill-defined parametric task spaces, that include unfeasible subspaces and irrelevant dimensions. This makes them particularly well suited to complex continuous task spaces in which expert-knowledge is

difficult to acquire. We formulate the Continuous Teacher-Student (CTS) framework to cover this scope of challenges, opening the range of potential applications.

Main contributions:

- A Continuous Teacher-Student setup enabling to frame teacher-student interactions for ill-defined continuous parameter spaces encoding stochastic tasks. See Sec. 4.3.
- Design of two BipedalWalker environments featuring parametric procedural content generation, well-suited to benchmark ACL approaches on continuous task spaces. See Sec. 4.4.3.
- ALP-GMM, an ACL approach based on Gaussian Model Mixture and absolute LP that is well suited to for DRL agents learning to master continuous task spaces. See Sec. 4.4.1.
- First study of ALP-based teacher algorithms leveraged to scaffold the learning of generalist DRL agents in continuously parameterized environments. See Sec. 4.5.

4.2 Related Work

As discussed in section 2.2.2, learning progress has often been used as an intrinsically motivated objective to automate curriculum generation in developmental robotics. For instance, this led to successful applications in population-based robotic control in simulated (Moulin-Frier et al. (2014); Forestier & Oudeyer (2016b); chapter 3) and real-world environments (Oudeyer et al., 2007b; Baranes & Oudeyer, 2013). LP was also used to accelerate the training of LSTMs and neural turing machines (Graves et al., 2017), and to personalize sequences of exercises for children in educational technologies (Clément et al., 2015).

A similar Teacher-Student framework was proposed in Matiisen et al. (2017), which compared teacher approaches on a set of deterministic navigation tasks in Minecraft (Johnson et al., 2016). While their work focuses on discrete sets of tasks, we tackle the broader challenge of dealing with *continuous* task spaces (i.e. infinite task sets), in which large parts of the task space may be unlearnable.

At the time of the present research project, another form of ACL has already been studied for continuous sets of tasks (Florensa et al., 2018). However, in their work, they studied how to learn a multiplicity of locomotion *goals* (varying by their reward function), where we tackle the more complex setting of learning to behave in a continuous set of *environments* (varying by their state space). Their GOAL-GAN algorithm also requires researchers to set a reward range of "intermediate difficulty" to be able to label each goal in order to train a GAN, which is highly dependent on both the learner's skills and the considered continuous set of goals. Besides, as the notion of intermediate difficulty provides no guarantee of progress, this approach is susceptible to focusing on unlearnable goals for which the learner's competence stagnates in the intermediate difficulty range. A comparative study of LP-based ACL methods w.r.t. GOAL-GAN is included in chapter 5.

4.3 The Continuous Teacher-Student Framework

In this section, we formalize our Continuous Teacher-Student framework. It is inspired from earlier works in developmental robotics (Baranes & Oudeyer, 2009; Lopes & Oudeyer, 2012b) and intelligent tutoring systems (Clément et al., 2015). The CTS framework is also close to earlier work on Teacher-Student approaches for discrete sets of tasks (Matiisen et al., 2017). In CTS however, teachers sample parameters mapping to stochastic tasks from a continuous task-encoding parameter space. In the remainder of this chapter, we will refer to task parameters and tasks interchangeably, as one task parameter directly maps to a stochastic task. Likewise, for simplicity, tasks will be assumed stochastic.

Student – In CTS, learning agents, called students, are confronted with episodic tasks, procedurally generated from n-dimensional task parameters $\tau \in \mathcal{T} \subset \mathbb{R}^n$, which can be formalized as Partially Observable MDPs (POMDP). For each interaction step, a student s collects an observation $o \in \mathcal{O}_\tau$, performs an action $a \in \mathcal{A}_\tau$, and receives a corresponding reward $r \in \mathbb{R}_\tau$. Upon task termination, an episodic reward $r_e = \sum_{t=0}^{T} r^{(t)}$ is computed, with T the length of the episode.

Teacher – The teacher interacts with its student s through the sequential sampling of task parameters in the aforementioned task-encoding parameter space $\mathcal{T}$, thereafter simply referred to as the "task space". For each interaction step, the teacher selects a task parameter τ used to generate a (stochastic) task that is presented m times to its student, and observes r_τ, the average of the m episodic rewards r_e. The new task-reward tuple is then added to a history database of interactions $\mathcal{H}^{int}$ that the teacher leverages to influence task selection in order to maximize the student's final competence return $c_\tau = f(r_\tau)$ across the task space. As students are considered as *black-box* learners, the teacher solely relies on its database history $\mathcal{H}^{int}$ for task sampling and does not have access to information about its student's internal state, algorithm, or perceptual and motor capacities. Given this, one can define a teacher algorithm as an ACL function $\mathcal{D}(\mathcal{H}^{int}) \rightarrow \tau$. This general teacher function formulation can be implemented in many different ways, e.g. a simple feed-forward DNN taking only the last task-reward pair of $\mathcal{H}^{int}$ as input. In practice, rather than directly using $\mathcal{H}^{int}$, ACL methods often condition their task sampling on an internal student-knowledge state iteratively inferred from the growing $\mathcal{H}^{int}$, e.g. a list of task subspaces with high learning potential (Baranes & Oudeyer, 2009; Matiisen et al., 2017; Colas et al., 2019). Formally, one can express the objective of the teacher function $\mathcal{D}(\mathcal{H}^{int}) \rightarrow \tau$ as

$$\begin{aligned} &\max_{\mathcal{D}} \int_{\tau \sim \mathcal{T}} w_\tau \cdot c_\tau \mid \mathcal{T}^s_{train} \, \mathrm{d}\tau, \\ &\text{with } \mathcal{T}^s_{train} = \left[\tau_1, \tau_2, ..., \tau_E\right]_{\tau_i \sim \mathcal{D}} \end{aligned} \tag{4.1}$$

with $c_\tau \mid \mathcal{T}^s_{train}$ a term which quantifies the student's competence on task τ *given* a previous training on a set of pre-defined budget of E tasks ($\mathcal{T}^s_{train}$) selected by the ACL function $\mathcal{D}$. w_τ is a factor weighting the relative importance of each task in the optimization process, enabling to specify whether to focus on specific subregions of the task space (i.e. harder target tasks). In practice, directly optimizing such an objective is often intractable for non-trivial scenarios, as each parameter update of the teacher must

be conditioned on post-training performance of its student. Instead, such ACL methods will need to optimize a surrogate objective, e.g. maximizing empirical (absolute) learning progress.

Task space assumptions – The teacher does not know the evolution of difficulty across the task space and therefore assumes a non-linear, piece-wise smooth function. The task space may also be ill-defined. For example, there might be subregions $\mathcal{U} \subset \mathcal{T}$ of the task space in which competence improvements on tasks $\tau \in \mathcal{U}$ is not possible given their state transition functions $\mathcal{P}_\tau$ (i.e tasks are either trivial or unfeasible). Additionally, given a task space $\mathcal{T} \in \mathbb{R}^n$, there might exist an equivalent task space $\mathcal{T}' \in \mathbb{R}^d$ with $d < n$, constructed with a subset of the n dimensions of $\mathcal{T}$, meaning that there might be irrelevant or redundant dimensions in $\mathcal{T}$.

In the following sections, we will restrict our study to CTS setups in which sampled tasks are presented only once to the student (i.e $m = 1$ and $r_\tau = r_e$) and do not prioritize the learning of specific subspaces (i.e $w_\tau = 1, \forall \tau \in \mathcal{T}$). As in the general ACL formalization proposed in section 2.3.1, we assume an ACL setup focused on a single non-resetable learner with a fixed teacher-student interaction budget. However, compared to the general ACL formalization, the CTS framework can be seen as being A) more specific (continuous task space, extrinsic reward maximization) and B) more restrictive (black-box students, i.e. task selection based on $\mathcal{H}^{int}$ rather than $\mathcal{H}$).

4.4 Methods

In this section, we will describe our absolute LP-based teacher algorithms, our reference teachers, and present the continuously parameterized BipedalWalker environments used to evaluate them.

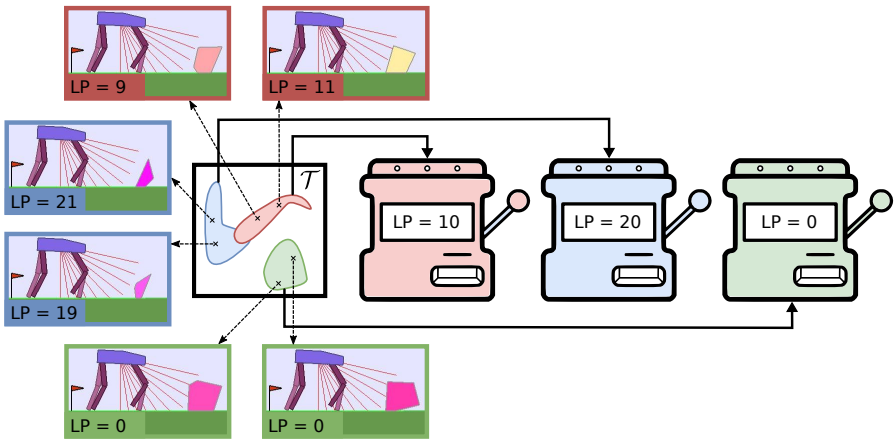


Figure 4.1: The common recipe between RIAC, COVAR-GMM and ALP-GMM (our proposed approach) is to dynamically detect subspaces having different LP value within the task space. Then, they consider each subspace as an arm of a Multi Armed Bandit setup, and compute each of their utility using a local aggregated LP value. More precisely, they consider *absolute* LP, as learning regress equates to forgetting, which is a valuable information indicating that re-training on the subspace must be done.

4.4.1 Absolute Learning-Progress-Based Teacher Algorithms

Inspired by previous works which proposed LP-based IMS (section 2.2.2), we frame our teacher approaches as a Multi-Armed Bandit setup in which arms are dynamically mapped to subspaces of the task space, and whose values are defined by an absolute average LP utility function (see figure 4.1). The objective of such teacher algorithms is to select subspaces on which to sample tasks in order to maximize ALP. ALP gives a richer signal than (positive) LP as it enables the teacher to detect when a student is losing competence on a previously mastered task subspace (thus preventing catastrophic forgetting).

Robust Intelligent Adaptive Curiosity (RIAC) – RIAC (Baranes & Oudeyer, 2009) is a task sampling approach whose core idea is to split a given task space in hyperboxes (called regions) according to their absolute LP, defined as the (absolute) difference of cumulative episodic reward between the newest and oldest tasks sampled in the region. Tasks are then sampled within regions selected proportionally to their ALP score. To avoid a known tendency of RIAC to oversplit the space (Florensa et al., 2018), we added a few minor modifications to the original architecture to constrain the splitting process, e.g. we enforce a minimal region size. Details can be found in appendix A.2.

Absolute Learning Progress Gaussian Mixture Model (ALP-GMM) – Another more principled way of sampling tasks according to LP measures is to rely on the well known Gaussian Mixture Model (Rasmussen, 2000) and Expectation-Maximization (Dempster et al., 1977) algorithms. This concept has already been successfully applied in the cognitive science field as a way to model intrinsic motivation in early vocal developments of infants (Moulin-Frier et al., 2014). In addition of testing for the first time their approach (referred to as COVAR-GMM) on DRL students, we propose a variant based on an ALP measure capturing long-term progress variations that is well-suited for RL setups. See appendix A.2 for a description of their method.

The key concept of ALP-GMM is to fit a GMM on a dataset of previously sampled tasks concatenated to their respective ALP measure. Then, the Gaussian from which to sample a new task is chosen using an EXP4 bandit scheme (Auer et al., 2002), where each Gaussian is viewed as an arm, and ALP is its utility. This enables the teacher to bias the task sampling towards high-ALP subspaces. To get this per-task ALP value, we take inspiration from our previous study of the AMB algorithms in chapter 3: for each newly sampled task τ_{new} and associated episodic reward r_{new}, the closest (Euclidean distance) previously sampled task τ_{old} (with associated episodic reward r_{old}) is retrieved using a nearest neighbor algorithm (implemented with a KD-Tree (Bentley, 1975)). We then have

$$alp_{new} = |r_{new} - r_{old}| \tag{4.2}$$

The GMM is fit periodically on a window $\mathcal{W}$ containing only the most recent task-ALP pairs (the last 250 in our experiments) to bound its time complexity and make it more sensitive to recent high-ALP subspaces. The number of Gaussians is adapted online by fitting multiple GMMs (here having from 2 to $k_{max} = 10$ Gaussians) and keeping the best one based on Akaike's Information Criterion (Bozdogan, 1987). Note that the

nearest neighbor computation of per-task ALP uses a database that contains all previously sampled tasks and associated episodic rewards, which prevents any forgetting of long-term progress. In addition to its main task sampling strategy, ALP-GMM also samples random tasks to enable exploration (here $p_{rnd} = 20\%$). See algorithm 1 for pseudo-code and figure 4.2 for a schematic view of ALP-GMM.

Algorithm 1 Absolute Learning Progress Gaussian Mixture Model (ALP-GMM)

Require: Student s, bounded task space $\mathcal{T}$, probability of random sampling p_{rnd}, fitting rate N, max number of Gaussians k_{max}

Initialize task-ALP First-in-First-Out window $\mathcal{W}$, set max size to N
Initialize task-reward history database $\mathcal{H}^{int}$
loop N times ▷ Bootstrap phase
    Sample random task-encoding parameters $\tau \in \mathcal{T}$
    Generate environment with τ, send it to s, observe episodic reward r_τ
    Compute ALP of τ based on r_τ and $\mathcal{H}^{int}$ (see equation 4.2)
    Store (τ, r_τ) pair in $\mathcal{H}^{int}$, store (τ, ALP_τ) pair in $\mathcal{W}$
loop ▷ Stop after sampling E tasks (including bootstrap)
    Fit a set of GMM having 2 to k_{max} kernels on $\mathcal{W}$
    Select the GMM with best Akaike Information Criterion
    loop N times
        $p_{rnd}\%$ of the time, sample a random task $\tau \in \mathcal{T}$
        Else, sample τ from a Gaussian chosen proportionally to its mean ALP value
        Generate environment with τ, send it to s, observe episodic reward r_τ
        Compute ALP of τ based on r_τ and $\mathcal{H}^{int}$
        Store (τ, r_τ) pair in $\mathcal{H}^{int}$, store (τ, ALP_τ) pair in $\mathcal{W}$
Return s

4.4.2 Teacher References

Random Task Curriculum (Random) – In Random, tasks are sampled randomly from the task space for each new episode. Although simplistic, similar approaches in previous work (Baranes & Oudeyer, 2013) proved to be competitive against more elaborate forms of CL.

Oracle – A hand-constructed approach, sampling random tasks in a fixed-size sliding window on the task space. This window is initially set to the easiest area of the task space and is then slowly moved towards complex ones, with difficulty increments only happening if a minimum average performance is reached. Expert knowledge is used to find the dimensions of the window, the amplitude and direction of increments, and the average performance threshold. Pseudo-code is available in appendix A.2.

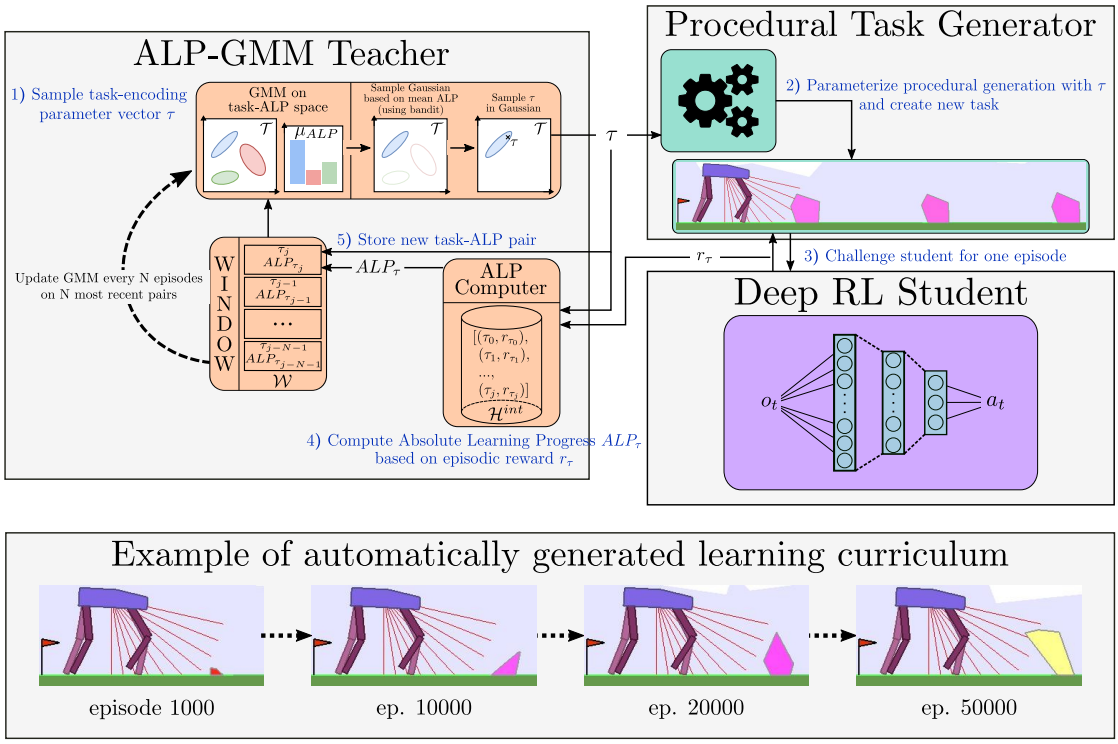


Figure 4.2: Schematic view of an ALP-GMM teacher's workflow

4.4.3 Parameterized BipedalWalker Environments with Procedural Generation

The BipedalWalker environment (Brockman et al., 2016) offers a convenient testbed for continuous control, allowing to easily build parametric variations of the original version (Ha, 2019; Wang et al., 2019b). The learning policy is embodied into a walker agent whose motors are controllable with torque (i.e. continuous action space). The observation space is composed of lidar sensors, head position and joint positions. Positive rewards are received for moving forward and penalties for torque usage and angular head movements. Agents are allowed 2000 steps to reach the other side of the map. Episodes are aborted with a -100 reward penalty if the walker's head touches an obstacle.

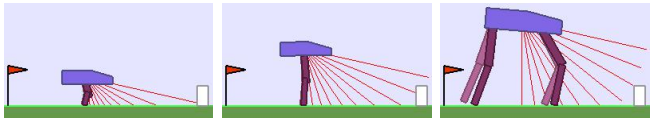
To study the ability of our teachers to guide DRL students, we design two continuously parameterized BipedalWalker environments enabling the procedural generation of walking tracks:

- *Stump Tracks* – A 2D parametric environment producing tracks paved with stumps varying by their height and spacing. Given a task-encoding parameter vector $[\mu_h, \Delta_s]$, a track is constructed by generating stumps spaced by Δ_s and whose heights are defined by independent samples in a normal distribution $\mathcal{N}(\mu_h, 0.1)$.

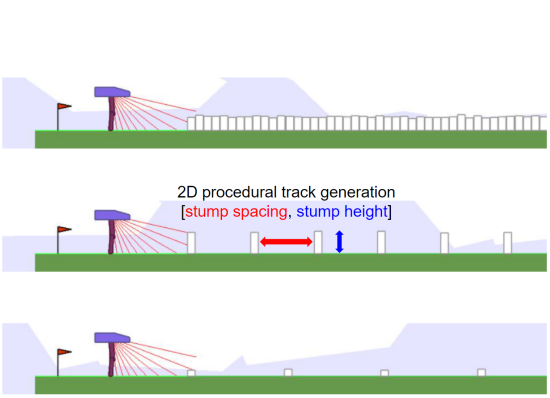
- *Hexagon Tracks* – A more challenging 12D parametric BipedalWalker environment. Given 10 offset values μ_o, each track is constructed by generating hexagons having their default vertices' positions perturbed by strictly positive independent samples in $\mathcal{N}(\mu_o, 0.1)$. The remaining 2 parameters are distractors defining the color of each

hexagon. This environment is challenging as there are no subspaces generating trivial tracks with 0-height obstacles (as offsets to the default hexagon shape are positive). This task space also has non-linear difficulty gradients as each vertices have different impacts on difficulty when modified.

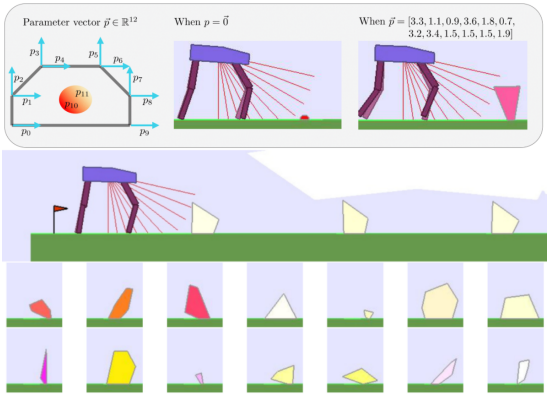
All of the experiments done in these environments were performed using OpenAI's implementation of Soft-Actor Critic (Haarnoja et al., 2018a) as the single student algorithm (see app. A.2 for details). To test our teachers' robustness to students with varying abilities, we use 3 different walker morphologies. Our considered set of walker morphologies along with examples of generated tracks are shown in figure 4.3.



(a) Walker morphologies



(b) *Stump Tracks* (all walker morphologies)



(c) *Hexagon Tracks* (quadrupedal morpho. only)

Figure 4.3: **Multiple students and environments to benchmark teachers. (a):** In addition to the default bipedal walker morphology (middle agent), we designed a bipedal walker with 50% shorter legs (left) and a bigger quadrupedal walker (right). **(b):** *Stump Tracks* environments, in which we test all morphologies. **(c):** *Hexagon Tracks*, a more difficult testbed, whose parameters control the generation of arbitrary hexagon obstacles. Here we show one track example and multiple potential hexagon generations. Note that we only present experimental results with quadrupedal walkers in Hexagon Tracks as smaller morphologies failed to learn.

4.5 Experimental Results

Performance metric – To assess the performance of all of our approaches on our BipedalWalker environments, we define a binary competence return measure stating whether a given track is mastered or not, depending on the student's episodic reward r_τ. We set the reward threshold to 230, which was used in Wang et al. (2019b) to ensure "reasonably efficient" walking gates for default bipedal walkers trained in environments similar to ours. Note that this reward threshold is only used for evaluation purposes

and in the Oracle condition. Performance is then evaluated periodically using a fixed evaluation set of 50 tracks generated by a uniform sampling in the task space. We then simply measure the percentage of mastered tracks. During evaluation, learning in DRL agents is turned off.

Through our experiments we answer three questions about ALP-GMM, COVAR-GMM and RIAC:

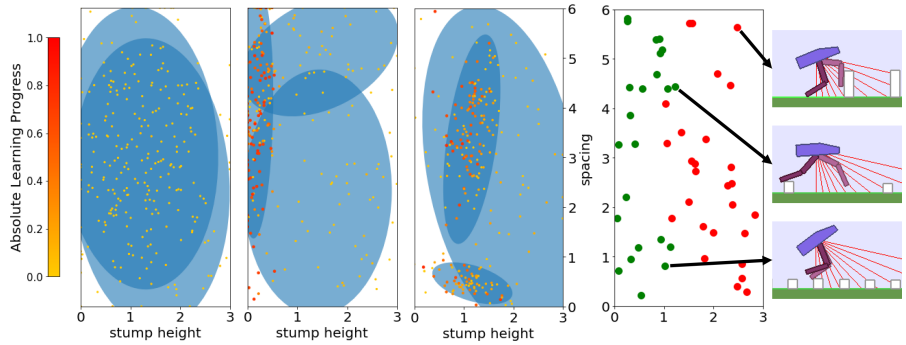
- Are ALP-GMM, COVAR-GMM and RIAC able to optimize their students' performance better than random approaches and teachers exploiting environment knowledge?
- How do their performance scale when the proportion of unfeasible tasks increases?
- Are they able to scale to high-dimensional sampling spaces with irrelevant dimensions?

4.5.1 How do ALP-GMM, COVAR-GMM and RIAC compare to reference teachers?

Emergence of curriculum – Figure 4.4 provides a visualization of the sampling trajectory observed in a representative ALP-GMM run for a default walker. Each plot shows the location of each Gaussian of the current mixture along with the 250 tasks subsequently sampled. At first (a), the walker does not manage to make any significant progress. After 1500 episodes (b) the student starts making progress on the leftmost part of the task space, especially for tracks with a stump-spacing task parameter higher than 1.5, which leads ALP-GMM to focus its sampling in that direction. After 15k episodes (c) ALP-GMM has shifted its sampling strategy to more complex regions. The analysis of a typical RIAC run is detailed in appendix A.3.

Performance comparison – Figure 4.5 shows learning curves for each condition paired with short, default and quadrupedal walkers. First, for short agents (a), one can see that Oracle is the best performing algorithm, mastering more than 20% of the test set after 20 million steps. This is an expected result as Oracle knows where to sample simple tracks, which is crucial when most of the task space is unfeasible, as is the case with short agents. ALP-GMM is the LP-based teacher with the highest final mean performance, reaching 14.9% against 10.6% for COVAR-GMM and 8.6% for RIAC. This performance advantage for ALP-GMM is statistically significant when compared to RIAC (Welch's t-test at 20M steps: $p < 0.04$), however there is no statistically significant difference with COVAR-GMM ($p = 0.16$). All LP-based teachers are significantly superior to Random ($p < 0.001$).

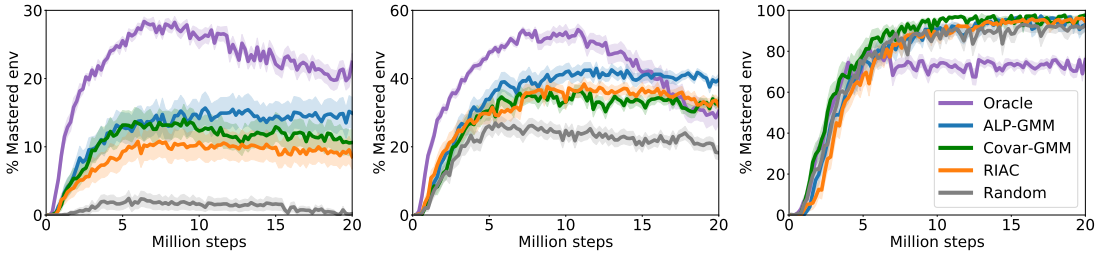
Regarding default bipedal walkers (b), our hand-made curriculum (Oracle) performs better than other approaches for the first 10 million steps and then rapidly decreases to end up with a performance comparable to RIAC and COVAR-GMM. All LP-based conditions end up with a final mean performance statistically superior to Random ($p < 10^{-4}$). ALP-GMM is the highest performing algorithm, significantly superior to Oracle ($p < 0.04$), RIAC ($p < 0.01$) and COVAR-GMM ($p < 0.01$).



(a) After 500 episodes (b) 1500 eps. (c) 15000 eps. (d) Mastered tracks

Figure 4.4: **Example of an ALP-GMM teacher paired with a Soft Actor-Critic student on Stump Tracks.** Figures (a)-(c) show the evolution of ALP-GMM task sampling in a representative run. Each dot represents a task and is colored according to its absolute learning progress value. After initial progress on the leftmost part of the space, as in (b), most ALP-GMM runs end up improving on tracks with 1 to 1.8 stump height, with the highest ones usually paired with spacing above 2.5 or below 1, indicating that tracks with large or very low spacing are easier than those in $[1, 2.5]$. Figure (d) shows for the same run which track of the test set are mastered (i.e $r_t > 230$, shown by green dots) after 17k episodes (amounting to 20 million steps).

For quadrupedal walkers (c), Random, ALP-GMM, COVAR-GMM and RIAC agents quickly learn to master nearly 100% of the test set, without significant differences apart from COVAR-GMM being superior to RIAC ($p < 0.01$). This indicates that, for this agent type, the task space of Stump Tracks is simple enough that training on random tracks is a sufficient curriculum learning strategy. Oracle teachers perform significantly worse than any other method ($p < 10^{-5}$).



(a) Short agents (b) Default agents (c) Quadrupedal agents

Figure 4.5: **Evolution of mastered tracks for Teacher-Student approaches in Stump Tracks.** The mean performance (32 seeded runs) is plotted with shaded areas representing the standard error of the mean.

Through this analysis we answered our first experimental question by showing how ALP-GMM, COVAR-GMM and RIAC, without strong assumptions on the environment, managed to scaffold the learning of multiple students better than Random. Interestingly, ALP-GMM outperformed Oracle with default agents, and RIAC, COVAR-GMM and ALP-GMM surpassed Oracle with the quadrupedal agent, despite its advantageous use of domain knowledge. This indicates that training only on tracks sampled from a sliding window

that end up on the most difficult task subspace leads to forgetting of simpler tasks. Our approaches avoid this issue through efficient tracking of their students' learning progress.

4.5.2 How do our approaches scale when the amount of unfeasible tasks increases?

A crucial requirement when designing all-purpose teacher algorithms is to ensure their ability to deal with task spaces that are ill-defined w.r.t to the considered student. To study this property, we performed additional experiments on Stump Tracks where we gradually increased the stump height dimension range, which increases the amount of unfeasible tasks.

Results are summarized in Table 4.1. To assess whether a condition is robust to increasing unfeasibility, one can look at the p-value of the Welch's t-test performed on the final performance measure between the condition run on the original task space and the same condition run on a wider space. High p-value indicates that there is not enough evidence to reject the null hypothesis of no difference, which can be interpreted as being robust to task spaces containing more unfeasible tasks. Using this metric, it is clear that ALP-GMM is the most robust condition among the presented LP-based teachers, with a p-value of 0.71 when increasing the stump height range from $[0, 3]$ to $[0, 4]$ compared to $p = 0.02$ for RIAC and $p = 0.05$ for COVAR-GMM. When going from $[0, 3]$ to $[0, 5]$, ALP-GMM is the only LP-based teacher able to maintain most of its performance ($p = 0.12$). Although Random also seems to show robustness to increasingly unfeasible task spaces ($p = 0.78$ when going from $[0, 3]$ to $[0, 4]$ and $p = 0.05$ from $[0, 3]$ to $[0, 5]$), it is most likely due to its stagnation in low performances. Compared to all other approaches, ALP-GMM remains the highest performing condition in both task space variations ($p < 0.02$).

Cond. \ Stump height	$[0, 3]$	$[0, 4]$	$[0, 5]$
ALP-GMM	$\mathbf{39.6} \pm 9.6$	$\mathbf{38.6} \pm 11.6$ $(p = 0.71)$	$\mathbf{34.3} \pm 15.8$ $(p = 0.12)$
COVAR-GMM	33.3 ± 7.1	27.6 ± 14.0 $(p = 0.05)$	24.1 ± 14.7 $(p = 0.01)$
RIAC	32.1 ± 12.2	$23.2 \pm 17.2 (p = 0.02)$	20.5 ± 15.4 $(p = 0.002)$
Random	18.2 ± 11.5	17.4 ± 11.8 $(p = 0.78)$	12.6 ± 11.0 $(p = 0.05)$

Table 4.1: **Impact of increasing the proportion of unfeasible tasks.** The average performance with standard deviation (after 20 million steps) on the original Stump Tracks' test-set is reported (32 seeds per condition).The additional p-values inform whether conditions run in the original Stump Tracks ($[0, 3]$) are significantly better than when run on variations with higher maximal stump height.

These additional experiments on Stump Tracks showed that our LP-based teachers are able to partially maintain the performance of their students in task spaces with higher proportions of unfeasible tasks, with a significant advantage for ALP-GMM.

4.5.3 Are our approaches able to scale to ill-defined high-dimensional task spaces?

To assess whether ALP-GMM, COVAR-GMM and RIAC are able to scale to task spaces of higher dimensionality containing irrelevant dimensions, and whose difficulty gradients are non-linear, we performed experiments with quadrupedal walkers on Hexagon Tracks, our 12-dimensional parametric BipedalWalker environment. Results are shown in Figure 4.6. In the first 20 million steps, one can see that Oracle has a large performance advantage compared to LP-based teachers, which is mainly due to its knowledge of initial progress niches. However, by the end of training, ALP-GMM significantly outperforms Oracle ($p < 0.02$), reaching an average final performance of 80% against 68% for Oracle. Compared to COVAR-GMM and RIAC, the final performance of ALP-GMM is also significantly superior ($p < 0.01$ and $p < 0.005$, respectively) while being more robust and having less variance (see appendix A.4). All LP-based approaches are significantly better than Random ($p < 0.01$).

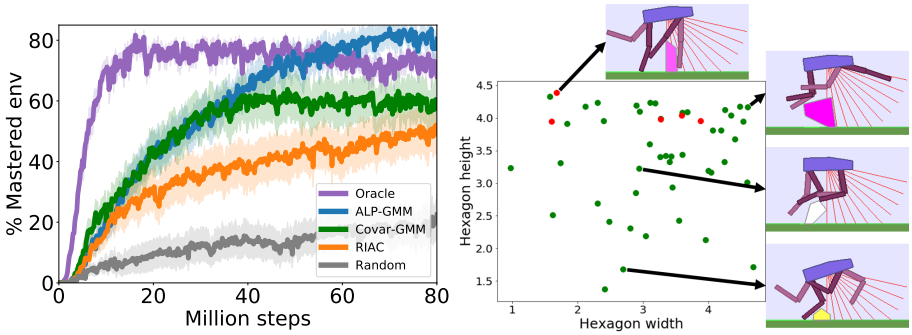


Figure 4.6: **Teacher-Student approaches in Hexagon Tracks. Left:** Evolution of mastered tracks for Teacher-Student approaches in Hexagon Tracks. 32 seeded runs (25 for Random) of 80 million steps where performed for each condition. The mean performance is plotted with shaded areas representing the standard error of the mean. **Right:** A visualization of which tracks of the test-set are mastered (i.e $r_t > 230$, shown by green dots) by an ALP-GMM run after 80 million steps.

Experiments on the Hexagon Tracks showed that ALP-GMM is the most suitable condition for complex high-dimensional environments containing irrelevant dimensions, non-linear task spaces and large proportions of initially unfeasible tasks.

Complementary experiments – To better grasp the general properties of our teacher algorithms, additional abstract experiments without DRL students were also performed for task spaces with increasing number of dimensions (relevant and irrelevant) and increasing ratio of initially unfeasible subspaces, showing that GMM-based approaches performed best (see appendix A.1).

4.6 Conclusion

This work demonstrated that LP-based teacher algorithms could successfully guide DRL agents to learn in difficult continuously parameterized environments with irrelevant dimensions and large proportions of unfeasible tasks. With no prior knowledge of its student's abilities and only loose boundaries on the task space, ALP-GMM, our proposed teacher, consistently outperformed random heuristics and occasionally even expert-designed curricula. ALP-GMM, which is conceptually simple and has very few crucial hyperparameters, opens-up exciting perspectives inside and outside DRL for curriculum learning problems.

One limitation of this work is that our comparative analysis is restricted to LP-based teachers, e.g. we did not study how such methods compare to GOAL-GAN (Florensa et al., 2018), an existing ACL approach (featured in our survey, see section 2.3.2), able to autonomously sample tasks using a surrogate objective based on *intermediate difficulty*. Among other motivations, this led to the work presented in the following chapter 5, which will provide a thorough benchmarking of teacher algorithms, including LP-based methods, GOAL-GAN, and other recent ACL algorithms.

Chapter 5

TeachMyAgent: a Benchmark for Automatic Curriculum Learning in DRL

Contents

To one day efficiently cope with the diversity of real-world situations, machine learners must be able to generalize their behaviors to a diversity of tasks. As discussed in chapter 2, this desideratum is at the core of the DRL field (section 2.1.3). In parallel to improving DRL algorithms themselves, one mean towards this end is to leverage ACL algorithms (section 2.3.2), i.e. to study how teacher algorithms can train DRL agents more efficiently across spaces of tasks, by adapting task selection to their evolving abilities. In the last few years, multiple ACL algorithms have been proposed (see section 2.3.2 for a survey). In chapter 4, we presented ALP-GMM, one such ACL algorithm, and demonstrated its performance advantages over other LP-based ACL methods. However, as in all other works proposing new ACL methods, computational limitations and lack of easy-to-use open-source code prevented us to perform an exhaustive comparison of our approach w.r.t. all relevant ACL methods. In short: *the ACL field is lacking a standard benchmark.*

While multiple standard benchmarks exist to compare DRL agents, there is currently no such thing for ACL algorithms. Thus, comparing existing approaches is difficult, as too many experimental parameters differ from paper to paper. In this chapter, to formalize comparative analysis in ACL, we identify several key challenges faced by ACL

algorithms. Based on these, we present *TeachMyAgent*, a benchmark of current ACL algorithms leveraging procedural task generation. It includes 1) challenge-specific unit-tests using variants of Stump Tracks, our procedural Box2D bipedal walker environment (from chapter 4), and 2) a new procedural Parkour environment combining most ACL challenges, making it ideal for global performance assessment. We then use *TeachMyAgent* to conduct a comparative study of representative existing approaches, showcasing the competitiveness of some ACL algorithms that do not use expert knowledge (such as ALP-GMM). We also show that the Parkour environment remains an open problem. We open-source our environments, all studied ACL algorithms (collected from open-source code or re-implemented), and DRL students in a Python package available at `https://github.com/flowersteam/TeachMyAgent`.

5.1 Introduction

DRL researchers have been increasingly interested in finding methods to train generalist agents (Rajeswaran et al., 2017; Zhang et al., 2018a; Vanschoren, 2018; Cobbe et al., 2019) to go beyond initial successes on solving single problems, e.g individual Atari games (Mnih et al., 2015) or navigation in fixed scenarios (Lillicrap et al., 2016; Haarnoja et al., 2018a). Many works proposed novel DRL learning architectures able to successfully infer multi-purpose action policies when given an experience stream composed of randomly sampled *tasks* (Schaul et al., 2015; Hessel et al., 2018; Cobbe et al., 2019; Hessel et al., 2019). Other works focused on designing appropriate benchmarks to study the generalization capacities of such agents. For instance, Cobbe et al. (2020) proposed a suite of 16 Atari-like environments, all relying on Procedural Content Generation (PCG) to generate a wide diversity of learning situations. The high-diversity induced by PCG has been identified as particularly beneficial to foster generalization abilities to DRL agents (section 2.1.3).

An important aspect not covered by these prior works is that they all rely on proposing randomly selected tasks to their agent, i.e. they do not consider using curriculum in learning. One can argue that random task selection is inefficient, especially when considering complex continuous task sets, a.k.a. task spaces, which can feature subspaces of varying difficulties ranging from trivial to unfeasible. Following this observation, many works attempted to train given generalist agents by pairing them with ACL algorithms (see section 2.3.2). The advantages of ACL over random task sampling for DRL agents have been demonstrated in diverse experimental setups, such as domain randomization for sim2real robotics (OpenAI et al., 2019; Mehta et al., 2019), video games (Salimans & Chen, 2018; Mysore et al., 2018), or navigation in procedurally generated environments, as in chapter 4 and other works (Florensa et al., 2018; Racaniere et al., 2020).

While this diversity of potential application domains and implementations of ACL hints a promising future for this field, it also makes comparative analysis complicated, which limits large-scale adoption of ACL. For instance, depending on the ACL approach, the amount of required expert knowledge on the task space can range from close to none – as in ALP-GMM– to a high amount of prior knowledge, e.g. initial task sampling subspace and predefined reward range triggering task sampling distribution shifts, as in OpenAI et al. (2019). Additionally, some ACL approaches were tested based on their ability to master an expert-chosen target subspace (Klink et al., 2020), while others were tasked to

optimize their performance over the entire task space, e.g. Baranes & Oudeyer (2009); Florensa et al. (2018) and ALP-GMM. Besides, because of the large computational cost and implementation efforts necessary for exhaustive comparisons, newly proposed ACL algorithms are often compared to only a subset of previous ACL approaches (see figure 5.1 for examples). This computation bottleneck is also what prevents most works from testing their ACL teachers on a diversity of DRL students, i.e. given a set of tasks, they do not vary the student's learning mechanism nor its embodiment. Designing a unified benchmark platform, where baselines would be shared and allow one to only run its approach and compare it to established results, could drive progress in this space.

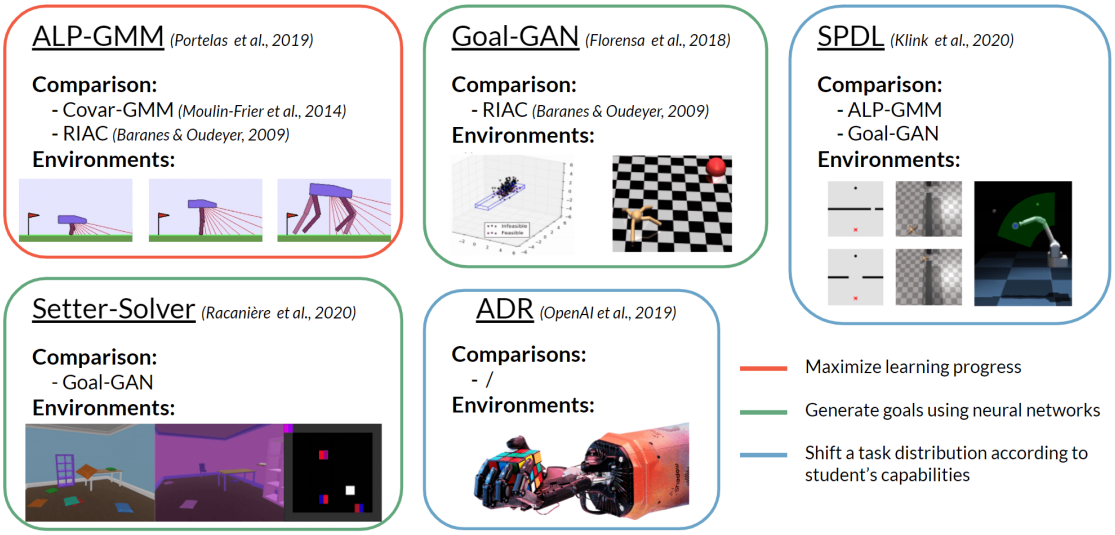


Figure 5.1: Existing ACL approaches lack a holistic comparative analysis.

Inspired by how the MNIST dataset (Lecun et al., 1998) or the ALE Atari games suite (Bellemare et al., 2013) respectively catalyzed supervised learning and single-task reinforcement learning research, we propose to perform this much-needed in-depth ACL benchmarking study. As such, we introduce *TeachMyAgent 1.0*[1], a teacher testbed featuring a) two procedural Box2D environments with challenging task spaces, b) a collection of pre-defined agent embodiments, and c) multiple DRL student models. The combination of these three components constitutes a large panel of diverse teaching problems. We leverage this benchmark to characterize the efficiency of an ACL algorithm on the following key teaching challenges:

1. *Mostly unfeasible task spaces* – While using PCG systems to generate tasks allows us to propose rich task spaces to DRL agents, which is good for generalization, such large spaces might contain a predominant amount of unfeasible (or initially unfeasible) tasks. A teacher algorithm must then have the ability to quickly detect and exploit promising task subspaces for its learner.

[1] http://developmentalsystems.org/TeachMyAgent/

2. *Mostly trivial task spaces* – On the contrary, the task space might be mostly trivial and contain only few challenging subspaces, which is a typical scenario when dealing with a skilled student (e.g. that is already trained, or that has an advantageous embodiment). In that case, the teacher has to efficiently detect and exploit the small portion of subspaces of relevant difficulty.

3. *Forgetting students* – DRL learners are prone to catastrophic forgetting Kirkpatrick et al. (2017), i.e. to overwrite important skills while training new ones. This has to be detected and dealt with by the teacher for optimal curriculum generation.

4. *Robustness to diverse students* – Being able to adapt curriculum generation to diverse students is an important desideratum to ensure a given ACL mechanism has good chances to transfer to novel scenarios.

5. *Rugged difficulty landscapes* – Another important property for ACL algorithms is to be able to deal with task spaces for which the optimal curriculum is not a smooth task distribution sampling drift across the space but rather a series of distribution jumps, e.g. as in complex PCG-task spaces.

6. *Working with no or little expert knowledge* – Prior knowledge over a task space w.r.t. a given student is a costly information gathering process that needs to be repeated for each new problem/student. Relying on as little expert knowledge as possible is therefore a desirable property for ACL algorithms (especially if aiming for out-of-the-lab or open-ended applications).

To precisely assess the proficiency of an ACL algorithm on each of these challenges independently, we extend our walker environment from chapter 4 into multiple unit-test variants, one per challenge, inspired by the structure of *bsuite* (Osband et al., 2020), a recent benchmark for RL agents. The second environment of our benchmark is the *Parkour* environment, inspired by Wang et al. (2020b). It features a complex task space whose parameters seed a neural network-based procedural generation of a wide diversity of environments, in which there exists drastically different learning curricula depending on the agent's embodiment (see figure 5.2). To assess the ability of existing ACL methods to robustly adapt to diverse students, we consider a random black-box student scenario in the Parkour environment, i.e. the morphology (e.g. walker or climber) of the learner is randomly selected for each new training run.

Scope – More precisely, we conduct an in-depth comparative study of ACL approaches suited for generalist DRL agents such as SAC (Haarnoja et al., 2018a) or PPO (Schulman et al., 2017) in single agent scenarios. We do not include works on self-play/multi-agent setups (Hernandez-Leal et al., 2018; Hernandez et al., 2019) nor single-agent population-based approaches (Forestier et al., 2017; Wang et al., 2020b). Also, we are interested in the problem of task selection from a continuous parameter space encoding the procedural generation of tasks. We leave the analysis of ACL methods for discrete task sets (Matiisen et al., 2017; Mysore et al., 2018), sets of task spaces (Forestier et al., 2017; Colas et al., 2019), or intrinsic reward learning (Pathak et al., 2017; Burda et al., 2019b; Raileanu & Rocktäschel, 2020) for future work. We assume this continuous space is given and relatively low-dimensional as it already poses strong teaching challenges: we therefore leave

the analysis of approaches that autonomously learn task representations for subsequent work (Jabri et al., 2019; Pong et al., 2020; Kovač et al., 2020).

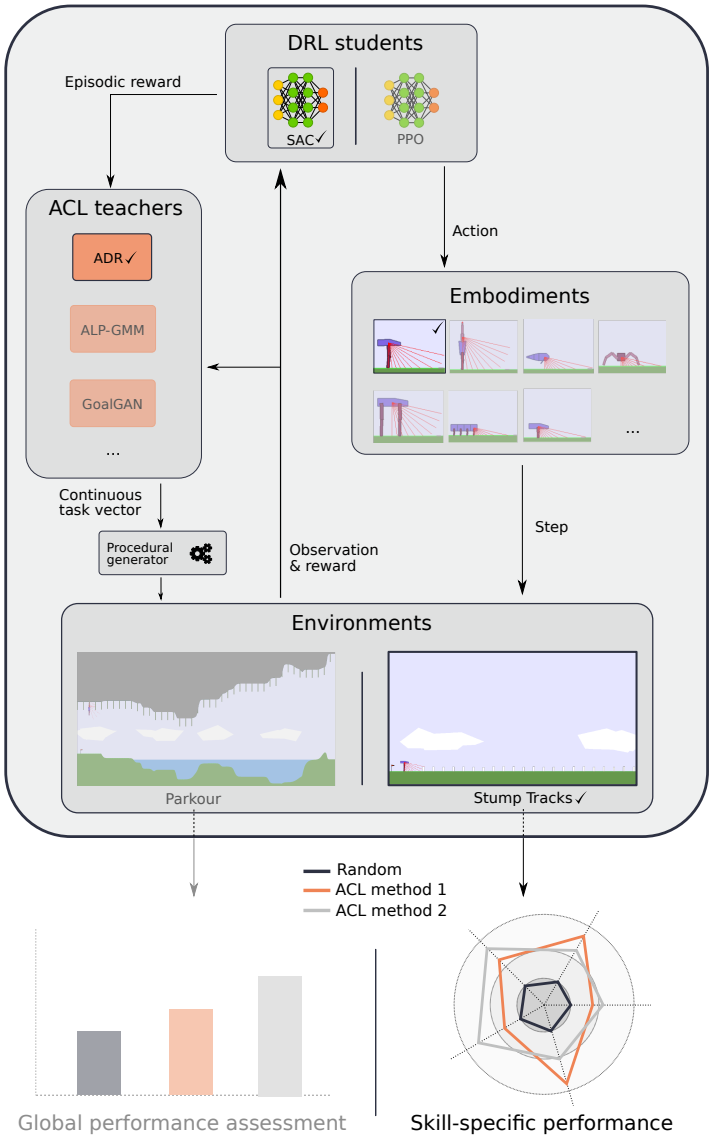


Figure 5.2: **TeachMyAgent**: A benchmark to study and compare teacher algorithms in continuous procedural environments.

Main contributions:

- Identification of multiple challenges to be tackled by ACL methods, enabling multi-dimensional comparisons of these algorithms.
- *TeachMyAgent 1.0*, a set of teaching problems (based on PCG environments) to study and compare ACL algorithms when paired with DRL students.
- Comparative study of representative existing ACL approaches, including both skill-specific unit-tests and global performance assessments, which highlights the competitiveness of methods not using expert knowledge and shows that our Parkour environment largely remains an open problem for current state-of-the-art ACL.

- Release of an open-source Python package, featuring 1) all environments, embodiments and DRL students from *TeachMyAgent*, 2) all studied ACL algorithms, that we either adapt to our API when code is available or re-implement from scratch if not open-sourced, 3) our experimental results as baselines for future works, and 4) tutorials & reproducibility scripts.

5.2 Related Work

Many environment suites already exist to benchmark DRL algorithms: some of them leverage video games, which provide challenging discrete action spaces, e.g. Atari 2600 games as in Bellemare et al. (2013) or Sonic The Hedgehog levels in Nichol et al. (2018). To study and develop DRL agents suited for complex continuous control scenarios, the community predominantly used the MuJoCo physics engine (Todorov et al., 2012). The Deep Mind Lab (Beattie et al., 2016) provides customizable puzzle-solving environments, particularly well suited to study goal-conditioned policies learning from pixels in rich 3D environments. At the intersection of DRL and Natural Language Processing, benchmark environments such as TextWorld (Côté et al., 2018) or BabyAI (Chevalier-Boisvert et al., 2019) were also designed to provide a testbed to develop autonomous agents receiving linguistic goals and/or interacting using language. The *bsuite* benchmark (Osband et al., 2020) leverages unit-tests to assess the core capabilities of DRL methods (e.g. generalization, memory). In all these previous works, the DRL agent is learning in one or few environments presented randomly and/or intrinsically chooses goals within those predefined environments, and the long-term community objective is to find more efficient learning architectures. On the contrary, the objective of *TeachMyAgent* is to foster the development of new teacher algorithms whose objective is, given a task space and a DRL student, to most efficiently organize the learning curriculum of their DRL student such that its performance is maximized over the task set. In other words, it is not about finding efficient learning architectures but about finding efficient curriculum generators.

Perhaps closest to our work is the Procgen benchmark (Cobbe et al., 2020), which features several atari-like environments, all having unique procedural generation systems allowing to generate a wide diversity of learning situations, particularly well suited to assess the generalization abilities of DRL agents. While they rely on an uncontrollable, random procedural generation, we assume control over it, which enables the use of ACL methods to select parameters encoding task generation. An interesting future work, parallel to ours, would be to modify the Procgen benchmark to allow direct control over the procedural generation.

Because of the current lack of any ACL benchmark, most recently proposed ACL algorithms relied on designing their own set of test environments. Florensa et al. (2018) used a custom MuJoCo Ant maze in which the ACL approach is in control of which end-position to target. Klink et al. (2020) used another MuJoCo Ant maze and ball-catching environment featuring a simulated Barrett WAM robot. While these previous works studied how to control goal selection in a given fixed environment, we are interested in the arguably more challenging problem of controlling a rich parametric procedural generation. In chapter 4, we already studied ACL in Stump Tracks, a procedural Box2D environment that we include and extend in *TeachMyAgent*, however we did not perform an extensive

comparative study as what we propose in this chapter. Racaniere et al. (2020) also used procedural generation to test their ACL approach, however they only compared their ACL algorithm to GOAL-GAN (Florensa et al., 2018), and did not open-source their environments. Additionally, in contrast with all previously cited ACL works, in *TeachMyAgent* we propose an in-depth analysis of each approaches through multiple unit-test experiments to fully characterize each teacher.

5.3 ACL Baselines

In the following paragraphs, we succinctly frame and present all the ACL algorithms that we compare using *TeachMyAgent*. More detailed explanations are left to appendix B.1.

Framework – Because we consider continuous task spaces and black-box students, we reuse the CTS framework proposed in chapter 4. Given a DRL student s and a n-dimensional task-encoding parameter space $\mathcal{T} \in \mathbb{R}^n$ (i.e. a task space), the process of ACL aims to learn a function $\mathcal{D}(\mathcal{H}^{int}) \rightarrow \tau$. $\mathcal{D}$ proposes new tasks to student s given a history $\mathcal{H}^{int}$ of episodic rewards obtained for each task. One can define the optimization objective of such an ACL policy given an experimental budget of E episodic tasks as:

$$\begin{aligned} &\max_{\mathcal{D}} \int_{\tau \sim \mathcal{T}} c_\tau \mid \mathcal{T}^s_{train} \, \mathrm{d}\tau, \\ &\text{with } \mathcal{T}^s_{train} = \left[\tau_1, \tau_2, ..., \tau_E\right]_{\tau_i \sim \mathcal{D}} \end{aligned} \tag{5.1}$$

with $c_\tau \mid \mathcal{T}_{train}$ the post-training competence of the student on task τ, which we simply define as the episodic reward. For simplicity, we drop the weighting parameter w_τ (see section 4.3), i.e. the objective is to maximize competence uniformly over the task space. Since it is usually difficult to directly optimize for this objective, various surrogate objectives have been proposed in the literature. See section 2.3.2 for a survey and classification of recent ACL works.

Expert-knowledge – To ease the curriculum generation process, multiple forms of expert knowledge have been provided in current ACL approaches. We propose to gather them in three categories: 1) use of an initial task distribution $\mathcal{T}_{init}$ to bootstrap the ACL process, 2) use of a target task distribution $\mathcal{T}_{target}$ to guide learning, and 3) use of a function interpreting the scalar episodic reward sent by the environment to identify mastered tasks (*Reward mastery range*). For each implemented ACL method, we highlight its required prior knowledge over the task space w.r.t a given DRL agent in table 5.1. We hope that this classification will ease the process of selecting an ACL method for researchers and engineers, as available expert knowledge is (arguably) often what conditions algorithmic choices in machine learning scenarios.

Implemented baselines – We compare seven ACL methods, chosen to be representative of the diversity of existing approaches, that can be separated in three broad categories. First, we include three methods relying on the idea of maximizing the learning progress of the student: RIAC (Baranes & Oudeyer, 2009), COVAR-GMM (Moulin-Frier

Table 5.1: Expert knowledge used by the different ACL methods. We separate knowledge required (req.) by algorithms, optional ones (opt.), and knowledge not needed (empty cell).

Algorithm	$\mathcal{T}_{init}$	$\mathcal{T}_{target}$	Reward mastery range
ADR	req.		req.
ALP-GMM	opt.		
COVAR-GMM	opt.		
GOAL-GAN	opt.		req.
RIAC			
SPDL	req.	req.	
SETTER-SOLVER		opt.	req.

et al., 2014), and our proposed ALP-GMM algorithm (chapter 4). We then add in our benchmark GOAL-GAN (Florensa et al., 2018) and SETTER-SOLVER (Racaniere et al., 2020), both generating tasks using deep neural networks and requiring a binary reward for mastered/not mastered tasks, pre-defined using expert knowledge. We append to our comparison two other ACL algorithms based on the idea of starting from an initial distribution of tasks and progressively shifting it depending on the student's capabilities: ADR (OpenAI et al., 2019) and SPDL (Klink et al., 2020). ADR relies on inflating a task distribution from a single initial task based on student mastery at each task distribution's border. SPDL shifts its initial distribution towards a target distribution. Finally, we also add a baseline teacher selecting tasks uniformly random over the task space (called Random).

5.4 The *TeachMyAgent* Benchmark

In the following section, we describe available environments and learners in *TeachMyAgent*. We propose two Box2D environments with procedural generation, allowing to generate a wide variety of terrains. Both our environments are episodic, use continuous action/observation spaces, and return scalar rewards. In addition, we provide two DRL algorithms as well as multiple agent morphologies. An experiment is thus constituted of an ACL method, an environment, and a learner (i.e. an embodied DRL algorithm).

5.4.1 Environments

Stump Tracks variants – As a first testbed, *TeachMyAgent* includes *Stump Tracks*, our walker environment presented in chapter 4 (section 4.4.3), which uses a 2D parametric PCG to generate new episodes. These parameters control the height and spacing of stumps laid out along a walking track. We chose to feature this environment as its low-dimensional task space is convenient for visualizations and modifications. More precisely, *TeachMyAgent* features multiple variants of the original Stump Tracks (e.g.

by extending the task space boundaries or shuffling it) to design our unit-tests of ACL challenges (see sec. 5.1 and sec. 5.5).

Parkour environment – Inspired by both Stump Tracks and another Box2D environment from Wang et al. (2020b), we present the parametric Parkour environment: a challenging task space with rugged difficulty landscape, few prior knowledge definable, and requiring drastically different learning curricula depending on the agent's embodiment.

It features an uneven terrain (see figure 5.3) composed of a ground and ceiling encoded through a Compositional Pattern-Producing Network (CPPN) (Stanley, 2007). This CPPN (whose weights and architecture are kept fixed) takes an additional input vector of bounded real numbers, which acts as the parameters controlling terrain generation. This neural network based generation enables to create a task space with a rugged difficulty landscape (see figure 5.3(a) and appendix B.2), requiring time-consuming exploration from an expert to seek trivial subspaces. The Parkour environment also features graspable objects, called "creepers", creating a niche for climbing morphologies. Similarly to the stumps in Stump Tracks, the creepers' generation is controlled by their height and the space between them. The Parkour's task space also contains a dimension controlling the "water" level of the track, ranging from 0 (no water) to 1 (entire parkour underwater). Water adds new physic rules aiming to imitate (in a simplified way) physics of water.

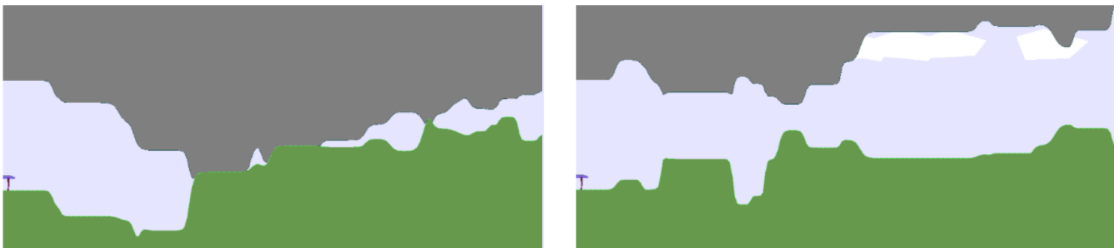
The resulting 6D task space (3 for the CPPN's input, 2 for creepers and 1 for water) creates a rich environment in which the optimal curriculum will largely depend on the agent's embodiment (e.g. swimming agents need high levels of water, while climbers and walkers need low levels). Note that, as in Stump Tracks, each episode lasts 2000 steps, agents are rewarded for moving forward (and penalized for using torque) and have access to lidars, head position, joint positions, and other environment-specific information (see appendix B.2).

5.4.2 Learners

Embodiments – As aforementioned, we introduce new morphologies using swimming and climbing locomotion (e.g. fish, chimpanzee, see figure 5.2). *TeachMyAgent* also features the short, default, and quadrupedal walker from chapter 4, as well as new walking morphologies such as the spider and the millipede (see figure 5.2).

DRL algorithms – To benchmark ACL algorithms, we rely on two different SOTA DRL algorithms: 1) Soft-Actor-Critic (Haarnoja et al., 2018a) (SAC), a classical off-policy actor-critic algorithm based on the dual optimization of reward and action entropy, and 2) Proximal Policy Optimization (PPO) (Schulman et al., 2017), a well-known on-policy DRL algorithm based on approximate trust-region gradient updates (see section 2.1). We use OpenAI *Spinning Up*'s implementation[2] for SAC and OpenAI Baselines' implementation[3] for PPO. See appendix B.3 for implementation details.

[2] https://spinningup.openai.com
[3] https://github.com/openai/baselines



(a) Example of random CPPN-based terrain generation (creepers and water not shown)

(b) Feasible parkour environments depending on the student's morphology, taken from our test set

Figure 5.3: *Parkour* environment examples. **(a):** The CPPN-based generation allows generating complex ground and ceiling landscape. The top left figure is a good example of an unfeasible task. **(b):** The full procedural generation system in the *Parkour* environment allows us to encode a diversity of tasks. These three bottom figures show task samples well suited for specific morphologies. For instance, the fish morphology (middle figure) requires submerged environments. An ACL method must explore this complex task space to figure out relevant subspaces given its student's abilities.

5.5 Experiments

We now leverage *TeachMyAgent* to conduct an in-depth comparative study of the ACL algorithms presented in section 5.3. After discussing experimental details, we undergo two separate experiments, aiming to answer the following questions:

- How do current ACL methods compare on each teaching challenges proposed in section 5.1 ?
- How do current ACL methods scale to a complex task space with limited expert knowledge ?

5.5.1 Experimental Details

For both our environments, we train our DRL students for 20 million steps. For each new episode, the teacher samples a new task vector used for the procedural generation of the environment. The teacher then receives the cumulative episodic reward that can be potentially turned into a binary reward signal using expert knowledge (as in GOAL-GAN and SETTER-SOLVER). Additionally, SPDL receives the initial state of the episode as well as the reward obtained at each step, as it is designed for non-episodic RL setup. Every 500000 steps, we test our student on a test set composed of 100 pre-defined tasks and monitor the percentage of test tasks on which the agent obtained an episodic reward greater than 230 (i.e. "mastered" tasks), which corresponds to agents that were able to

reach the last portion of the map (in both Stump Tracks and Parkour). We compare performance results using Welch's t-test as proposed in Colas et al. (2018), allowing us to track statistically significant differences between two methods. We perform a hyperparameter search for all ACL conditions through grid-search (see appendix B.1), while controlling that an equivalent number of configurations are tested for each algorithm. See appendix B.3 for additional experimental details.

5.5.2 Challenge-Specific Comparison with Stump Tracks Variants

First, we aim to compare the different ACL methods on each of the six challenges we identified and listed in section 5.1. For this, we propose to leverage the Stump Tracks environment to create five experiments, each of them designed to highlight the ability of a teacher in one of the first five ACL challenges (see appendix B.3 for details):

1. *Mostly unfeasible task space*: growing the possible maximum height of stumps, leading to almost 80% of unfeasible tasks.
2. *Mostly trivial task space*: allowing to sample stumps with negative height, introducing 50% of new trivial tasks.
3. *Forgetting student*: resetting the DRL model twice throughout learning (i.e. every 7 million steps).
4. *Diverse students*: using multiple embodiments (short bipedal and spider) and DRL students (SAC and PPO).
5. *Rugged difficulty landscape*: applying a random transformation to the task space such that feasible tasks are scattered across the space (i.e. among unfeasible ones).

Additionally, in order to compare methods on our sixth challenge (i.e. the need of prior knowledge), we propose to perform each of our five experiments in three conditions:

- *No expert knowledge*: none of the prior knowledge listed in table 5.1 is given. Hence only methods not requiring it can run in this setup.
- *Low expert knowledge*: only reward mastery range information is accessible. We consider this as low prior knowledge as, while it requires some global knowledge about the task space, it does not require assumptions on the difficulty of specific subspaces of the task space.
- *High expert knowledge*: all the expert knowledge listed in table 5.1 is given.

Note that in both the *No expert knowledge* and *Low expert knowledge* setups, SPDL (and ADR in *Low expert knowledge*) uses an initial task distribution randomly chosen as a subset of the task space. Moreover, in order to make a fair comparison in the *High expert knowledge* condition, we modified the vanilla version of COVAR-GMM and ALP-GMM such that they can use an expert-given initial task distribution.

Using these 15 experiments (5 challenges in 3 expert knowledge setups), we here introduce what is, to our knowledge, the first unit-test like experiment of ACL methods, allowing one to compare teachers in each of the challenges we previously introduced. Moreover, performing each of the five experiments in three expert knowledge setups allows us to show how the (un)availability of expert knowledge impacts performance for each method, which is hard to infer from each approach's original paper as they tend to focus only on the most ideal scenario. See appendix B.3 for a detailed explanation of each experimental setup.

To conduct our analysis, each ACL method is used in 15 experiments with 32 seeds, except ADR, GOAL-GAN and SETTER-SOLVER which cannot run in the *No expert knowledge* setup (i.e. only 10 experiments). We then calculate the aforementioned percentage of mastered test tasks on our test set (identical for all experiments), and average it over seeds.

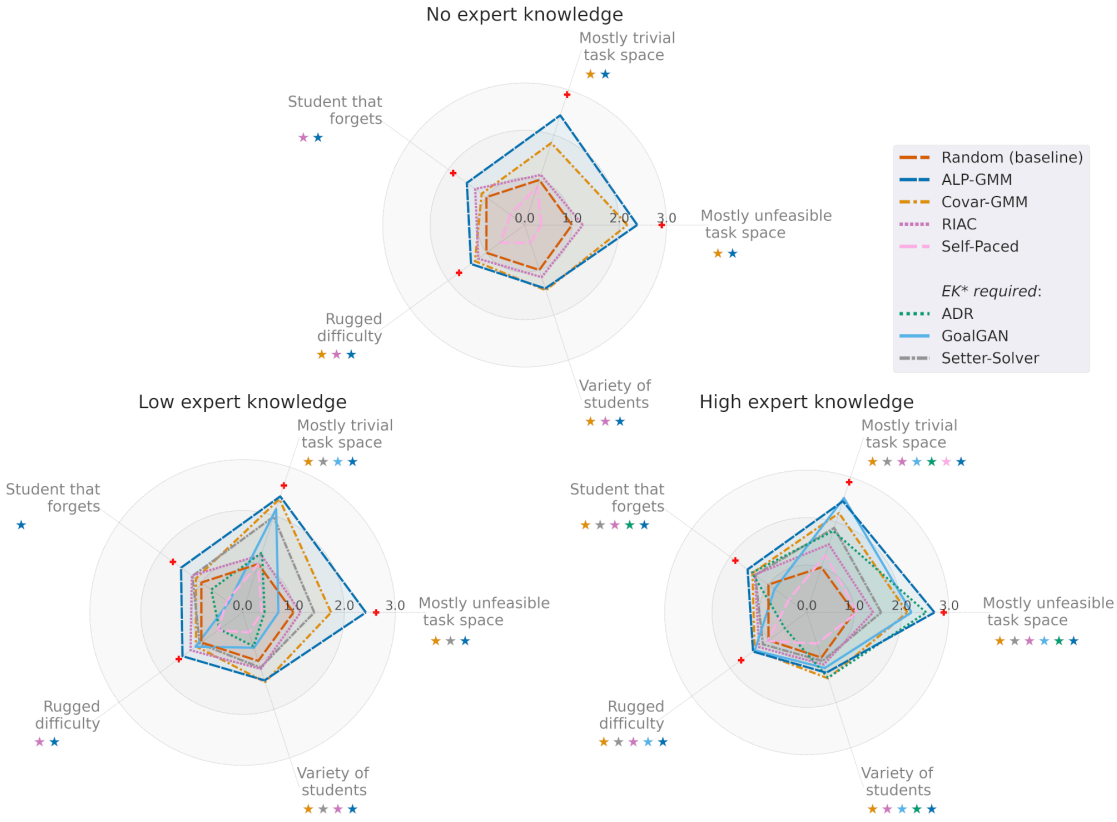


Figure 5.4: *EK: Expert Knowledge.* Post-training performance of each ACL method as a ratio of Random's results on multiple teaching challenges, done with 3 different expert knowledge levels. We use ✚ to show estimations of upper-bound performances in each challenge, except for *Variety of students* (see appendix B.4.1). On each axis, we indicate which method performed significantly better than Random ($p < 0.05$) using colored stars matching each method's color (e.g. ★ for COVAR-GMM, ★ for ADR). See appendix B.4.2 for details.

Experimental Results

Performance results of all conditions can be visualized in figure 5.4 as a ratio of the Random teachers' performance, our lower-baseline (see appendix B.4.2 for additional results).

Expert-knowledge-free methods – One can see that methods not requiring any expert knowledge (e.g. ALP-GMM or COVAR-GMM) obtain very similar performances in *No expert knowledge* and in *High expert knowledge* setups (although expert knowledge does benefit them in terms of sample efficiency (see app. B.4.2 for details). Comparing their performance without prior knowledge to the results obtained by other teachers when they have access to high expert knowledge shows how competitive expert-knowledge-free methods can be.

Expert knowledge dependency – The *Low expert knowledge* setup highlights the dependence of methods relying on an initial distribution of easy tasks (e.g. ADR and GOAL-GAN), as it is not given in this scenario. As a result, in this setup, ADR obtains end performances not significantly different from Random in all challenges, and GOAL-GAN only outperforms Random in the *Mostly trivial task space* challenge ($p < 0.05$). This has to be compared with their performance on the *High expert knowledge* setup, in which both approaches reach the top 3 results on 3/5 challenges.

ADR & GOAL-GAN – Both ADR and GOAL-GAN have one strong weakness in a challenge (*Rugged difficulty* for ADR and *Forgetting student* for GOAL-GAN) that lead them to a performance worse than Random (significantly for ADR with $p < 0.05$) in all expert knowledge setups. For ADR, it can be explained by the fact that its expansion can get stuck by subspaces of very hard (or unfeasible) difficulty, and for GOAL-GAN, by its inability to adapt quickly enough to the student's regressing capabilities because of its inertia to update its sampling distribution (updating the buffer and training the GAN). We provide a more in-depth analysis of these two cases in appendix B.4.2.

SPDL – One can see that SPDL's performance seem very poor in our experimental setup: its end performance is significantly inferior to Random in 11/15 experiments ($p < 0.05$). This can be explained by the fact that SPDL, by design, optimizes performance over a Gaussian target distribution, while our test set is uniformly sampled over the task space. See appendix B.1 for details and potential fixes.

5.5.3 Global Performance Analysis using the Parkour

The second experiment we propose aims to more broadly benchmark ACL methods' performance in the Parkour environment, which features most of the previously discussed ACL challenges: 1) most tasks are unfeasible, 2) before each run, unknown to the teacher, the student's embodiment is uniformly sampled among three morphologies (bipedal walker, fish and chimpanzee), requiring the teacher to adapt curriculum generation to a diversity of student profiles, and 3) tasks are generated through a CCPN-based PCG, creating a rich task space with rugged difficulty landscape and hardly-definable prior knowledge.

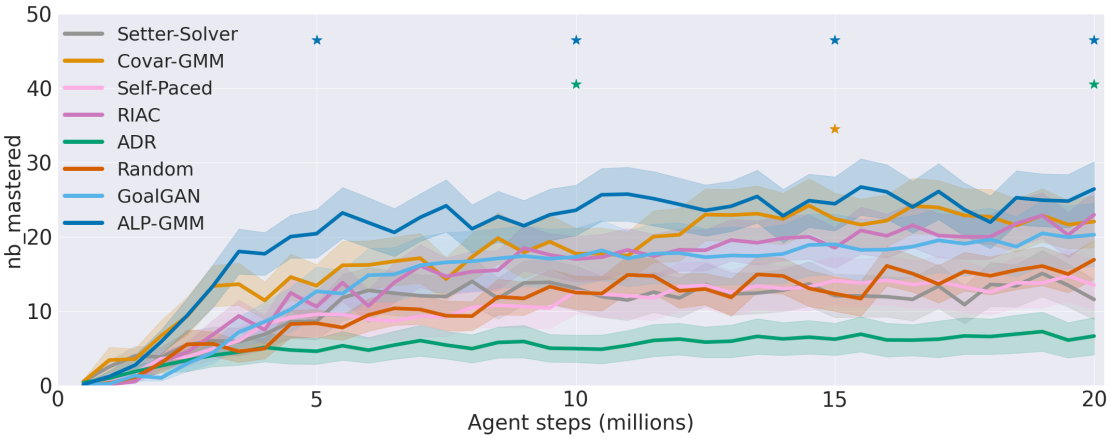


Figure 5.5: Averaged performance (48 seeds, with standard error of the mean) for each ACL method on Parkour. We calculate every 5 million steps which method obtained statistically different ($p < 0.05$) results from Random and indicate it with a star.

We perform 48 seeded experiments (i.e. 16 seeds per morphology). To evaluate performance, three test sets were hand-designed (one per embodiment) such that each contains an even distribution between easy, medium and hard tasks. In terms of expert knowledge for teachers, we only give reward mastery range. Without a straightforward initial easy task distribution to give to teachers requiring such knowledge (ADR and SPDL), we set it randomly over the space for each new run. See appendix B.3 for details.

Experimental Results

We present the evolution of performance of each teacher averaged over all seeds (and thus all embodiments) in figure 5.5 and gather the detailed results in appendix B.4.3. Interestingly, one can observe that best-performing methods do not use expert knowledge. This is explained by the fact that few prior knowledge is provided to the teachers in these experiments and, as shown in the challenge-specific experiments, most methods using expert knowledge heavily rely on them to reach high-performance. However, one can see that, while SPDL and SETTER-SOLVER remain at the performance level of Random, GOAL-GAN's performance along training is (mostly) not significantly different from those of COVAR-GMM and RIAC, two methods not relying on expert knowledge, as opposed to GOAL-GAN. ADR seems to plateau very fast and finally reach an average performance significantly worse than Random ($p < 0.05$). Indeed, as the difficulty landscape of the Parkour environment is rugged, and the initial "easy" task distribution randomly set, ADR is unable to progressively grow its sampling distribution towards feasible subspaces. Finally, when looking specifically to each embodiment type, results show the incapacity of all teachers to make the DRL student learn an efficient policy with the climbing morphology (i.e. at most 1% of mastered tasks by the end of training across all teachers), although we are able to show that high-performing policies can be learned when considering a subspace of the task space (see our case study in appendix B.4.3). This might be due to the complexity of learning the climbing gait w.r.t walking or swimming, as it requires for

instance good coordination skills between the arms and the grasping actions. For the two other morphologies (bipedal walker and fish), results obtained are also low (respectively less than 60% and 50%) and have a high variance (especially for the fish) considering that our test sets contain feasible tasks. This makes the Parkour environment an open challenge for future work on designing ACL algorithms.

5.6 Open-Source Release of *TeachMyAgent*

With the open-source release of *TeachMyAgent* (version *1.0*), we hope to provide a tool that can be used as a step towards thorough comparison and better understanding of current and future ACL methods. *TeachMyAgent*'s documented repository features the code of our environments, embodiments, DRL students, as well as implementations of all ACL methods compared in this paper. All of these parts use APIs we provide such that one can easily add its ACL method, learning algorithm, and new embodiment or environment. We hope this will foster community-driven contributions to extend *TeachMyAgent* in order to broaden its impact and adapt it to the future of ACL. We also provide the code we used to reproduce our experiments, as well as Jupyter notebooks allowing to generate all the figures showed in this paper. Finally, we release the results of our benchmark, allowing one to load them and compare its ACL method against baselines without having to reproduce our large-scale experiments.

5.7 Discussion and Conclusion

In this chapter, we presented *TeachMyAgent 1.0*, a first extensive testbed to design and compare ACL algorithms. It features unit-tests environments to assess the efficiency of a given teacher algorithm on multiple core skills, and the Parkour environment, which provides a challenging teaching scenario that has yet to be solved. We used *TeachMyAgent* to conduct a comparative study of existing ACL algorithms. Throughout our experiments, we identified that 1) current ACL approaches not using expert knowledge matched and even outperformed (e.g. ALP-GMM) other approaches using high amounts of expert knowledge, and 2) the Parkour environment is far from solved, which makes it a good candidate as a testbed when designing new ACL approaches.

Limitations & future work. – An obvious extension of this work is the addition of recent ACL approaches proposed during or after our experimental campaign (Zhang et al., 2020; Jiang et al., 2021a,b). So far, all studied ACL algorithms struggled to detect feasible task subspaces in Parkour, hinting that more research is needed to improve the "progress niche detection" ability of current teacher algorithms.

TeachMyAgent currently only features environments with low-dimensional PCG systems. Designing new environments with higher-dimensional PCG, that might require to learn low dimensional representations on which to apply ACL algorithms, is an interesting avenue. Besides, our current list of environments only studies 2D locomotion tasks inspired by our initial work on ALP-GMM as well as other works on DRL and 2D locomotion (Song et al., 2018; Ha, 2019; Gaier & Ha, 2019; Wang et al., 2019b, 2020b). While we put maximal effort in building a thorough and fair analysis of ACL methods, we

believe extending *TeachMyAgent* with other environments, such as ProcGen (Cobbe et al., 2020) or robotic manipulation tasks, would make the benchmark even more informative. Similarly, extending the benchmark to consider environment-conditioned goal selection (Racaniere et al., 2020; Campero et al., 2021) – i.e. where teachers have to observe the initial episode state to infer admissible goals – is also worth investigating.

TeachMyAgent provides comparisons in terms of expert knowledge requirements and teaching performances (asymptotic performance, sample efficiency) on a diversity of teaching challenges. In future work, another relevant metric to record and study would be the wall-clock time and compute efficiency of each ACL approach. Featuring such metrics would be especially important for ML researchers and engineers looking for an ACL method to plug into their existing learning pipeline while controlling computation/time overhead.

Outreach Project: an Interactive Web-Demo on DRL

Accessible at https://developmentalsystems.org/Interactive_DeepRL_Demo/

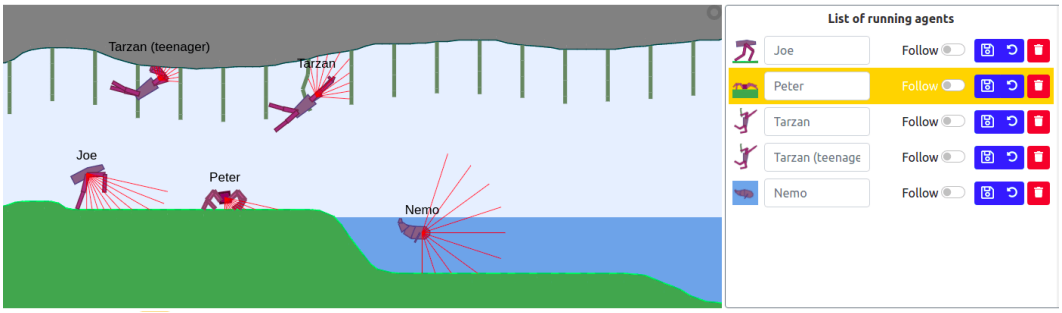
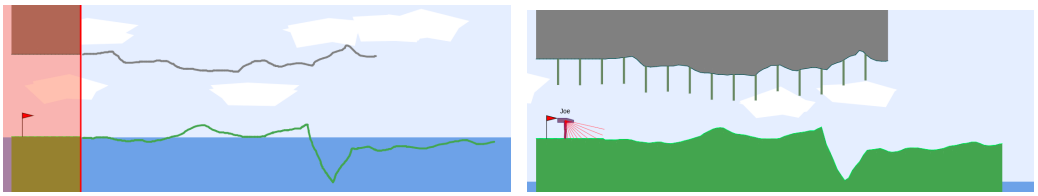


Figure 5.6: An interactive (javascript) demo, allowing to simultaneously visualize multiple trained DRL agents in custom *Parkour* environments

As part of Paul Germon's internship, which I co-supervised with Clément Romac and Pierre-Yves Oudeyer, we leveraged the *Parkour* environment from *TeachMyAgent* to build an interactive DRL demonstration in Javascript (link above, see figure 5.6). The main objective of this demonstration is to showcase how DRL (with ACL) can enable the training of embodied agents with robust locomotion policies, able to generalize to never seen before scenarios. The demonstrations allows to draw custom parkours through a user-friendly drawing tool (see figure 5.7), affording fun experimentations on the limits of the generalization capabilities of said learners.



(a) Users can draw floor and ceiling curves (b) Drawings are turned into parkours

Figure 5.7: Our demo features a drawing tool, allowing to create unique parkours to challenge the abilities of our agents.

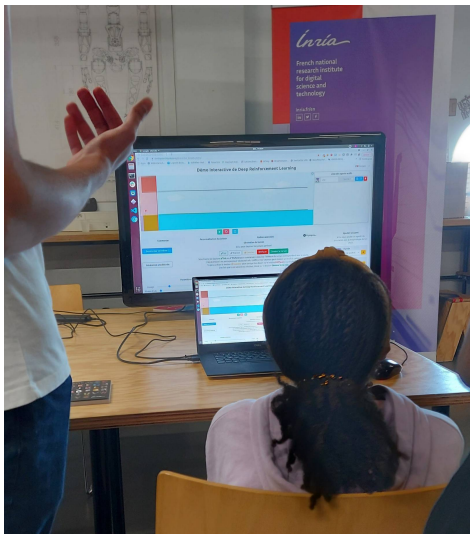


Figure 5.8: A child trying out our demonstration at *CapSciences Bordeaux*.

In October 2021, we used our demo at the science fair *CapSciences Bordeaux* (figure 5.8) as a support to seed discussions with participants (often young children and their parents) around autonomous machine learners, DRL and generalization.

Beyond educational demonstrations, our interactive javascript environment could also be leveraged in future work as a visualization tool for the *TeachMyAgent* benchmark. For instance, it could be used by researchers as an interactive leaderboard, i.e. to study and compare the learned locomotion policies of top ACL-DRL learning systems.

Chapter 6

Meta-ACL for Black-Box Students

Contents

Previous chapters focused on the notion of *automatic curriculum learning*, i.e. teaching algorithms whose objective is to adapt task selection to the evolving abilities of their student throughout training. Section 2.3 formalized then surveyed this field. Chapter 3 presented experiments using ACL for population-based agents. Chapter 4 and 5 presented experimental contributions in applying ACL methods to DRL agents. More precisely, these works, as in all other ACL studies, focused on *independent* teaching scenarios: given a task space, a single DRL student is paired to an ACL algorithm whose objective is to infer an appropriate curriculum *on the fly*, i.e. throughout a single non-resetable interaction rollout with its student. In short, this chapter proposes to consider a broader problem: how to efficiently teach *multiple* students? Given N such students, and assuming they might have similarities in their learning capabilities, how to escape from independently performing N tabula rasa curriculum generations in a vanilla ACL fashion ? How to leverage *meta* knowledge from teaching student A to bootstrap the teaching of student B ?

In this chapter, we introduce the concept of Meta-ACL, i.e. algorithms seeking to generalize curriculum generation to multiple students, and formalize it in the context of black-box DRL learners. We present AGAIN, a first instantiation of Meta-ACL, and showcase its benefits for curriculum generation over classical ACL in multiple simulated environments including procedurally generated locomotion environments with learners of varying morphologies. Interestingly, we also showcase that AGAIN can outperform ACL in

a classical single student scenario through retraining instead of training longer. Videos and code are available at https://sites.google.com/view/meta-acl.

6.1 Introduction

As discussed in section 2.1.3, multiple authors demonstrated the benefits of Procedural Content Generation (PCG) as a tool to create rich task spaces to train generalist agents (Justesen et al., 2018; Risi & Togelius, 2019; Cobbe et al., 2019). Chapter 4 and 5 showcased that ACL methods could be successfully applied on these rich task spaces to scaffold DRL students. However, the current limit of ACL is that, when applied to such large continuous task spaces, that often have few learnable subspaces, it either relies on human expert knowledge that is hard/costly to provide (and which undermines how automatic the ACL approach is), or it loses a lot of time finding tasks of appropriate difficulty through *task exploration*. This random search for progress niches over the task space is a costly *tabula rasa* process. While it seems acceptable to perform given a single agent to train, it becomes suboptimal whenever considering the training of multiple agents that might have similarities in their capabilities profiles.

Beyond training single DRL learners with ACL to generalize over task spaces, we propose to go further and work on training (unknown) distributions of students on continuous task spaces. We propose to use the term *Classroom Teaching* (CT) to refer to such problems. CT defines a family of problems in which a teacher algorithm is tasked to sequentially generate multiple curricula tailored for each of its students, all having potentially varying abilities. CT differs from the problems studied in population-based developmental robotics (e.g. chapter 3) and evolutionary algorithms (section 2.1.3, e.g. Wang et al. (2019b)) as in CT there is no direct control over the characteristics of learners, and the objective is to foster maximal learning progress over all learners rather than iteratively populating a pool of high-performing task-expert policies. Studying CT scenarios brings DRL closer to assisted education research problems and might stimulate the design of methods that alleviate the expensive use of expert knowledge in current SOTA methods (Koedinger et al., 2013; Clément et al., 2015).

Given multiple students to train, no expert knowledge, and assuming at least partial similarities between each students' optimal curriculum, current *tabula rasa* exploratory-ACL approaches that do not reuse knowledge between different students do not seem like the optimal choice. This motivates the research of what we propose to call *meta automatic curriculum learning* mechanisms, i.e. algorithms learning to generalize ACL over multiple students. In this work we formalize this novel setup and propose AGAIN, a first Meta-ACL baseline algorithm inspired from ALP-GMM (chapter 4). Given a new student to train, our approach is centered on the extraction of adapted curriculum priors from a history of previously trained students. The prior selection is performed by matching competence vectors that are built for each student through pre-testing. We show that this simple method can bring significant performance improvements over classical ACL in both a toy environment without DRL students and in Box2D locomotion environments with DRL learners.

Motivations – We argue that finding ways to efficiently train multiple agents is

a valid research endeavor, which could find potential application areas. For example, in robotics, when considering a specific set of problems to solve, researchers often rely on iteratively modifying the morphology of their robot (e.g. they change motors, add DoF, change end-effector) based on the performance obtained by a given morphology on the considered task set. In such cases, the resulting sequence of agents to train bear strong similarities with each others. If the complexity of the task set requires to use curriculum generation procedures for each agent, the ability to leverage previous teaching data to bootstrap training seems more appropriate than brute-force task space exploration. Similar issues arise when considering the iterative enhancement of DRL architectures (regardless of embodiment). On a more speculative note, it appears not too far-fetched to envision that, one day, industrial automation might significantly rely on robotic systems that are iteratively trained (e.g. using DRL in simulation) to master a significant range of tasks. The sensorimotor apparatus of such systems might have to be modified to the specific environmental condition of each industrial customer. In such cases, providing sample efficient ways to train these set of robotic systems appears as an important component for economic viability.

Main contributions:

- Introduction to the concept of Meta-ACL, i.e. algorithms that generalize curriculum generation in Classroom Teaching scenarios, and an approach to study these algorithms. Formalization of the interaction flows between Meta-ACL algorithms and (unknown) DRL student distributions.
- Introduction of AGAIN, a first Meta-ACL baseline algorithm which learns curriculum priors to fasten the identification of learning progress niches for new DRL students.
- Design of a toy-environment and of a parametric Box2D locomotion environment featuring a multi-modal distribution of possible agent embodiments well suited to study Meta-ACL.
- Analysis of AGAIN on these environments, demonstrating the performance advantages of this approach over classical ACL, including (and surprisingly) when applied to a single student.

6.2 Related Work

As aforementioned, this work is tightly related to the ACL literature. While such works focus on training students through independent runs, we propose to investigate how one can share information across multiple trainings. Within DRL, *policy distillation* (Teh et al., 2017; Czarnecki et al., 2019) consists in leveraging one or several previously trained policies to perform *behavior cloning* on a new policy (e.g. to speed up training and/or to leverage task-experts to train a multi-task policy). Our work can be seen as proposing a complementary toolbox, aiming to perform *curriculum distillation* on a continuous space of tasks.

Similar ideas were developed for supervised learning (Yim et al., 2017; Furlanello et al., 2018; Hacohen & Weinshall, 2019). Hacohen & Weinshall (2019) propose an approach to

infer a curriculum from past training for an image classification task: they train their network once without curriculum and use its predictive confidence for each image as a difficulty measure exploited to derive an appropriate curriculum to re-train the network. Although we are mainly interested in training a classroom of diverse students, section 6.5.3 presents similar experiments in a DRL scenario, showing that our Meta-ACL procedure can be beneficial for a single learner that we train once and re-train using curriculum priors inferred from the first run.

Our work bears some similarities with Turchetta et al. (2020), which used a data-driven approach to autonomously infer curricula for multiple DRL agents. However, in their work, each agent is trained to perform a single navigation task: the "curriculum" consists in the selection of safety constraints to apply during training, chosen among a pre-defined discrete set (e.g. reset agent to previous state if it reaches a dangerous location). Agents only differ by their network initializations. By contrast, we consider the problem of training *generalist* students with varying morphologies (and network initializations), with a teacher algorithm choosing tasks from a *continuous* task space.

In Narvekar & Stone (2020), authors also propose to study how to generalize curriculum generation for policy learners. Given a space of navigation goals as independent target objectives in a grid-world environment, they show that they are able to efficiently generate training curricula for each new target goal presented to a SARSA (Rummery & Niranjan, 1994) policy learner. Curriculum generation is performed by a high-level DQN agent, conditioned on the student's weights (i.e. its knowledge state) and the target goal, whose actions are to select on which source goal (9 predefined possibilities) to train the student. Compared to this work, we study how to generalize curriculum generation to *diverse* learners given a single space of task to master. We consider complex DRL students and locomotion environments with continuous actions.

6.3 Meta Automatic Curriculum Learning Framework

In the following paragraphs, we formalize the concept of Meta-ACL. Because we focus on teaching black-box DRL students in continuous task spaces, we re-use our Continuous Teacher Student formalization from chapter 4 to specify teacher-student interactions.

Black-box students – The Meta-ACL framework assumes the existence of policy learners, a.k.a students s, capable of interacting in episodic control tasks (which can be defined as POMDPs). These students are assumed non-resetable, as in classical ACL scenarios. Their objective is to maximize reward collection. Such students are confronted with tasks drawn from a *continuous* task space. We do not assume expert knowledge over this task space w.r.t students, e.g. task subspaces could be trivial for some students and unfeasible for others. The objective of Meta-ACL is precisely to autonomously infer such prior knowledge from experience in scenarios where human expert knowledge is either hard or impossible to use (e.g. environments featuring complex PCG). Similarly, we consider *black-box* students, i.e. the internal states of students are not accessible to teachers, and we do not assume which learning mechanisms are used (e.g. SAC (Haarnoja et al., 2018a), PPO (Schulman et al., 2017), evolutionary algorithms).

Automatic Curriculum Learning – Given a black-box student s to train on a continuous *task space* $\mathcal{T}$, an ACL algorithm can be formalized as a function $\mathcal{D}(\mathcal{H}_s^{int}) \to \tau$ which sequentially samples (parameterized) tasks for s given its training history $\mathcal{H}_s^{int}$, i.e. the list of episodic rewards obtained for each task s was trained on. The objective of the experimenter is to find parameters of $\mathcal{D}$ which maximize the following theoretical objective:

$$\begin{aligned} &\max_{\mathcal{D}} \int_{\tau \sim \mathcal{T}} c_\tau \mid \mathcal{T}_{train}^s \, \mathrm{d}\tau, \\ &\text{with } \mathcal{T}_{train}^s = \left[\tau_1, \tau_2, ..., \tau_E\right]_{\tau_i \sim \mathcal{D}} \end{aligned} \tag{6.1}$$

with E the episode budget and $c_\tau \mid \mathcal{T}_{train}$ the post-training competence of the student on task τ, which we simply define as the episodic reward. Since direct optimization of such a post-training performance is difficult, ACL is often approached using proxy objectives (e.g. learning progress, intermediate difficulty).

Meta-ACL for Classroom Teaching. – We now present the concept of Meta-ACL applied to a Classroom Teaching scenario, i.e. there is no longer a single student s to be trained, but a set of students with varying abilities (e.g. due to morphology and/or learning mechanisms), sequentially drawn from an unknown student distribution $\mathcal{S}$. The notion of meta-learning refers to *any type of learning guided by prior experience with other tasks* (Vanschoren, 2018). In classical meta-RL settings, these "tasks" are defined as distinct MDPs. The novelty of our setup is to consider meta-learning at the level of teacher algorithms, for which *tasks* correspond to distinct *students* to train. In other words, Meta-ACL approaches must leverage knowledge from curricula built for previous students to improve the curriculum generation for new ones. More precisely, a Meta-ACL algorithm can be formulated as a function:

$$\begin{aligned} &\hat{\mathcal{D}}(\mathcal{H}_{s^K}^{int}, X) \to \tau \;\; s.t. \;\; X = f(\mathcal{H}^{\mathcal{S}}) \\ &\mathcal{H}^{\mathcal{S}} = [\mathcal{H}_{s^0}^{int}, \mathcal{H}_{s^1}^{int}, ..., \mathcal{H}_{s^{K-1}}^{int}] \end{aligned} \tag{6.2}$$

s^K is the student being trained by $\hat{\mathcal{D}}$. f is a function extracting curriculum priors X over a history $\mathcal{H}^{\mathcal{S}}$ of past K student trainings, resulting from the scaffolding of K previous students with an ACL or Meta-ACL policy. Given our formalization of ACL (see eq. 6.1), the experimenter's evaluation objective for Meta-ACL can be expressed as follows:

$$\max_{\hat{\mathcal{D}}} \int_{s \sim \mathcal{S}} \int_{\tau \sim \mathcal{T}} c_{s,\tau} \mid \mathcal{T}_{train}^s \, \mathrm{d}\tau \mathrm{d}s. \tag{6.3}$$

As in the case of the ACL evaluation objective expressed in eq. 6.1, direct optimization of eq. 6.3 is difficult as it implies the joint maximization of multiple students' performance. In our experiments, we reduce the Meta-ACL problem to the sequential independent training of a set of new students by leveraging priors from previous student trainings (with the hope to maximize performance over the entire set). Figure 6.1 provides a visual transcription of the workflow of a Meta-ACL algorithm. While in our experiments we use a fixed-size history $\mathcal{H}^{\mathcal{S}}$ of ACL-trained students (to make our experiments computationally tractable), $\mathcal{H}^{\mathcal{S}}$ could be grown incrementally by collecting training trajectories online from Meta-ACL trainings.

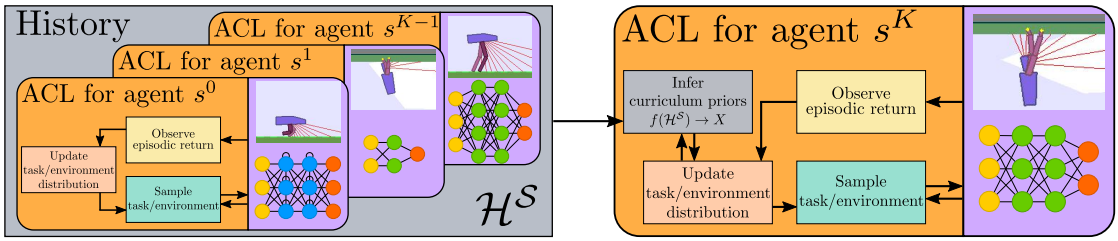


Figure 6.1: In *Meta Automatic Curriculum Learning* (Meta-ACL), the objective is to leverage previous teaching experience to improve the curriculum generation of a new agent, whose embodiment and learning mechanisms have potentially never been seen before: the teacher has to *generalize* over students.

6.4 AGAIN: a First Meta-ACL Baseline

In this section, we present AGAIN (Alp-Gmm And Inferred Progress Niches), our proposed Meta-ACL algorithm, and connect it to the formalism described in section 6.3. In essence, AGAIN is based on characterizing the performance of a new student, finding the most similar (performance-wise) previously trained student, and extracting promising curriculum data to resume teaching the new student. We first give a broad overview of the approach and then provide detailed explanations of key components.

Overview – Figure 6.2 provides a schematic pipeline of our Meta-ACL approach. Given a history $\mathcal{H}^{\mathcal{S}}$ of previously trained students and a new student $s^K \sim \mathcal{S}$ to train, AGAIN starts by (1) pre-training s^K using ALP-GMM, our proposed ACL algorithm from chapter 4 (well suited for teaching scenarios without expert knowledge). After this pre-training, AGAIN (2) challenges the student with a set of test tasks to construct a multidimensional *Competence Profile* CP^{pre} of the student. This vector is then (3) used to select a previously trained student s^i similar to s^K from which the training history

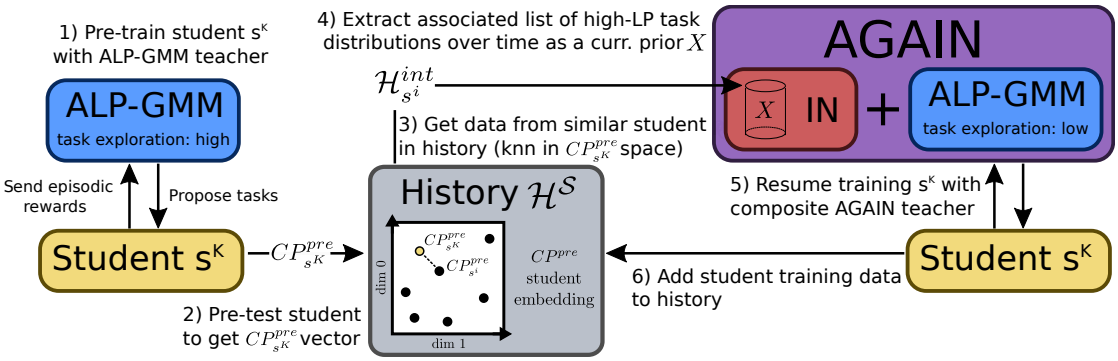


Figure 6.2: Schematic pipeline of AGAIN, our proposed Meta-ACL approach. Given a new student, AGAIN first performs a preliminary run with a high-exploration ALP-GMM curriculum generator. It then pre-test its new student and compare it to previous ones to infer an expert curriculum (IN). The training of the new student is then resumed with a combination of IN and a low-exploration ALP-GMM.

$\mathcal{H}^{int}_{s^i}$ is recovered. Based on $\mathcal{H}^{int}_{s^i}$, (4) AGAIN infers a set of curriculum priors X (a list of high-LP task subspaces). Finally, (5) the training of s^K can resume using a composite curriculum generator using both an expert curriculum derived from X (for exploitation), and ALP-GMM (for exploration).

ALP-GMM (1)

ALP-GMM is a LP-based ACL technique for continuous task spaces that does not assume prior knowledge. In short, ALP-GMM frames the task sampling problem into a non-stationary Multi-Armed bandit setup (Auer et al., 2002) in which arms are Gaussians spanning over the task space. The utility of each Gaussian is defined with a local LP measure derived from episodic reward comparisons, i.e. from the training history $\mathcal{H}^{int}$. The essence of ALP-GMM is to periodically fit a Gaussian Mixture Model (GMM) on recently sampled task parameters *concatenated with their respective LP*. The Gaussian from which to sample a new task is chosen proportionally to its mean LP dimension. Task exploration happens initially through a bootstrapping period of random task sampling and during training through residual random task sampling. See chapter 4 for a detailed description with visualizations and pseudo-code.

Curriculum priors: selection (2,3)

How to extract curriculum priors X from previous teaching data $\mathcal{H}^{\mathcal{S}}$, i.e. to implement function f from equation 6.2. As a first step towards more end-to-end approaches, we rely on a multi-step procedure inspired from knowledge assessment in Intelligent Tutoring Systems setups studied in the educational data mining literature (Vie, 2016; Vie et al., 2018). Because our procedure relies on extracting a set of high learning-progress subspaces, it assumes that all students in $\mathcal{H}^{\mathcal{S}}$ were either trained with AGAIN or ALP-GMM.

For a new student s^K, given its capabilities on the considered task space, how to select the most relevant previously trained student in $\mathcal{H}^{\mathcal{S}}$, from which to extract curriculum priors ? We propose to use *pre-tests* to derive a competence profile vector $CP^{pre} \in \mathbb{R}^m$ for all trained students. Each dimensions of CP^{pre} contains the episodic return of the student on the corresponding pre-test task. Given that we do not assume access to expert knowledge, we build this pre-test task set by selecting m tasks uniformly over the task space (and leave the automatic construction of adaptive task sets for future work). We use the same task set to build a post-training CP vector $CP^{post} \in \mathbb{R}^m$ whose dimensions are summed up to get a score $j_s \in \mathbb{R}$, used to evaluate the end performance of students in $\mathcal{H}^{\mathcal{S}}$. After the initial pre-training of s^K with ALP-GMM, curriculum priors can be obtained in 3 steps:

1. pre-test s^K to get its CP vector $CP^{pre}_{s^K}$,
2. infer the k most similar previously trained students in CP space (using a k-nearest neighbor algorithm), and
3. use the training history $\mathcal{H}^{int}$ of the student with maximal post-training score j_s among those k.

In essence, this method is about re-using curriculum data from a similarly-skilled and successfully-trained student.

Curriculum priors: curation (4)

Let s^i be the student from $\mathcal{H}^S$ resulting from the aforementioned selection procedure for the new student s^K. Assuming ALP-GMM (or AGAIN) as the underlying teacher used for s^i, we can extract curriculum priors X by considering the ordered sequence of GMMs X_{raw} that were periodically fitted throughout training (i.e. throughout $\mathcal{H}^{int}_{s^i}$):

$$\begin{aligned} X_{raw} &= \{p(1), ..., p(T)\} \\ s.t. \quad p(t) &= \sum_{i=1} LP_{ti}\mathcal{N}(\boldsymbol{\mu_{ti}}, \boldsymbol{\Sigma_{ti}}), \end{aligned} \tag{6.4}$$

with T the total number of GMMs in the list and LP_{ti} the Learning Progress of the i^{th} Gaussian from the t^{th} GMM. Since the LP value of each Gaussian can be considered as its utility, we propose a simple method to leverage X_{raw}: X can be obtained from X_{raw} by keeping only Gaussians with LP_{ti} above a predefined threshold δ_{LP}, which creates a curated list X containing only Gaussians located on task subspaces on which s^i experienced learning progress.

AGAIN (5)

Given that a curriculum prior X, in the form of a list of high-LP Gaussian over time has been constructed, how to use it to generate an improved curriculum for student s^K. We propose to leverage X by deriving an "expert" curriculum from it, named Inferred progress Niches (IN).

IN – Given a GMM of X, a task can be selected by 1) sampling a Gaussian proportionally to its LP_{ti} value, and 2) sampling the Gaussian to obtain the task parameters. But how to decide which GMMs to use along the training of the new student s^K ?

While the simplest way to obtain such a curriculum would be to start sampling tasks from the first GMM and step to the next GMM at the same rate as in the initial ALP-GMM run, we propose a more flexible *reward-based* method. This method requires us to record the mean episodic reward obtained by the previously trained student s^i for each GMM of X (which can be done without additional assumptions or computational overhead). Given this, to select which GMM from X is used to sample tasks over time along the training of s^K, we start with the first GMM and only iterate over X once the mean episodic reward over tasks recently sampled from the current GMM matches or surpasses the mean episodic reward recorded during the initial ALP-GMM run. In app. C.3, we show experimentally that this *reward-based* variant outperforms other potential methods. See app. C.1 for algorithmic details.

AGAIN – Simply using IN directly for s^K lacks adaptive mechanisms towards the characteristics of the new student (e.g. new embodiment, different initial parameters),

which could lead to failure cases where the expert curriculum misses important aspects of training (e.g. detecting task subspaces that are being forgotten). Additionally, the meta-learned ACL algorithm must have the capacity to emancipate from the expert curriculum once the trajectory is completed (i.e. go beyond X). This motivates why our approach combines IN with an ALP-GMM teacher after the initial pre-training. The resulting Alp-Gmm And Inferred progress Niches approach (AGAIN) samples tasks from a GMM that is composed of the current mixture of both ALP-GMM and IN. See appendix C.1 for details and pseudo-code of AGAIN, and section 4.4.1 for more details on ALP-GMM.

6.5 Experiments and Results

We organize the analysis of our proposed Meta-ACL algorithm around 3 experimental questions:

- What are the properties and important components of AGAIN (sec. 6.5.1)? In this section, we will leverage a toy environment without DRL students to conduct systematic experiments.
- Does AGAIN scale well to Meta-ACL scenarios with DRL students (sec. 6.5.2)? Here, we will present a new Box2D locomotion environment that will be used to conduct our experiments.
- Can AGAIN be used for single learners (sec. 6.5.3)? Here we will show that it can be useful to derive curriculum priors even for a single student (i.e. without any teaching history $\mathcal{H}^{\mathcal{S}}$).

Considered baselines and AGAIN variants. – In the aforementioned experiments, we compare AGAIN to the following conditions:

- IN, an ablation, which directly use the expert curriculum instead of combining it with ALP-GMM
- AGAIN_RND, an ablation which do not perform pre-tests and instead directly extract curriculum priors from a randomly selected student in the teaching history $\mathcal{H}^{\mathcal{S}}$.
- AGAIN_GT, an AGAIN variant with privileged information: instead of performing pre-tests, this condition is given access to the ground truth student distribution to condition from which previously trained student to extract curriculum priors.
- *Random*, an ACL condition, randomly sampling tasks for its student.
- ADR, an existing ACL condition from (OpenAI et al., 2019), used in section 6.5.1 and 6.5.2.
- *Oracle*, an expert-made ACL approach, used in 6.5.3.

See appendix C.2 for details on these conditions.

6.5.1 Analyzing Meta-ACL in a Toy Environment

To provide in-depth experiments on AGAIN, we first emancipate from DRL students through the use of a toy testbed (adapted from the toy environment used in chapter 4, see app. A.1). The objective of this environment is to simulate the learning of a student within a 2D task-encoding parameter space $\mathcal{T} = [0, 1]^2$, a.k.a task space. This fake task space is uniformly divided in 400 square cells $C \subset \mathcal{T}$, and each task $\tau \in \mathcal{T}$ sampled by the teacher is directly mapped to an episodic reward r_τ based on sampling history and whether C is considered "locked" or "unlocked". Three rules enforce reward collection in $\mathcal{T}$:

1. Every cell C starts "locked", except a randomly chosen one that is "unlocked".
2. If C is "unlocked" and $\tau \in C$, then $r_\tau = min(|C|, 100)$, with $|C|$ the cumulative number of tasks sampled within C while being "unlocked" (if C is "locked", then $r_\tau = 0$).
3. If $|C| >= 75$, adjacent cells become "unlocked".

Given these rules, one can model students with different curriculum needs by assigning them different initially unlocked cells, which itself models what is "easy to learn" initially for a given student, and from where it can expand. See figure 6.3 for illustrations of this toy environment.

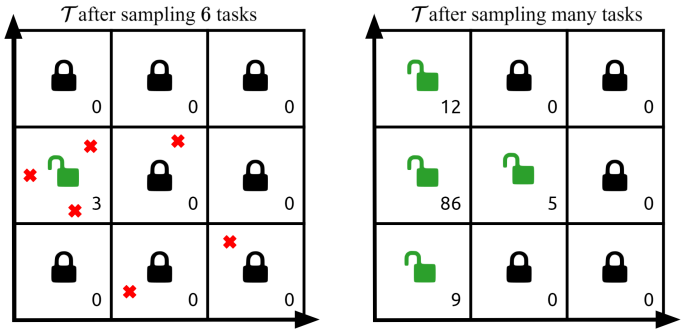


Figure 6.3: A toy environment to study teacher algorithms. **Left:** Teachers must focus their sampling strategy in unlocked areas ("learnable" cells) of the 2D task space. Red crosses illustrate this sampling procedure. Successful task sampling increments the competence counter of the cell. **Right:** If the teacher manages to focus its sampling strategy in unlocked cells, new cells will be unlocked, which simulates competence improvement.

Results

Instead of performing a pre-test to construct the CP^{pre} vector of a student, we directly compute it by concatenating $|C|$ for all cells, giving a 400-dimensional CP^{pre} vector. This vector is computed after 20k training episodes out of 200k. To study AGAIN, we first populate our history of trained students $\mathcal{H}^{\mathcal{S}}$ by training with ALP-GMM an initial

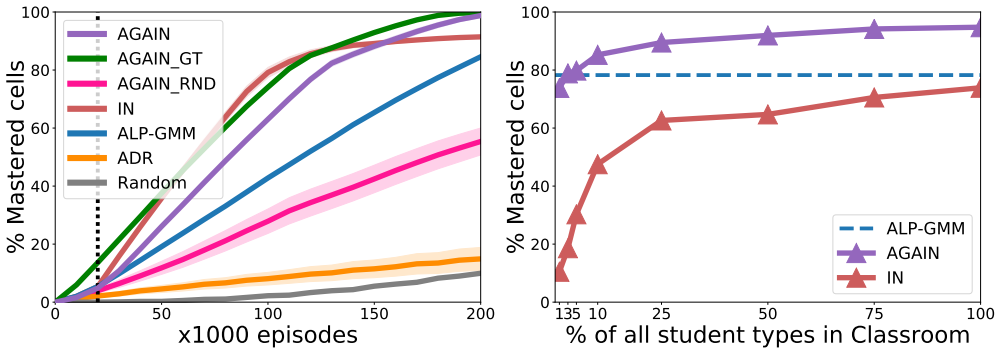


Figure 6.4: **Left:** By leveraging meta-learned curriculum priors w.r.t to its students, AGAIN outperforms regular ACL approaches. Avg. perfs. with *sem* (standard error of the mean) plotted, 48 seeds. The vertical dashed black line indicates when pre-training ends for Meta-ACL conditions. **Right:** Impact of classroom size and sparsity on Meta-ACL performances. Post-training (200k ep.) avg perfs. plotted, 96 seeds.

classroom of 128 students drawn randomly from 4 fixed possible student types (i.e. 4 possible initially unlocked cell positions), and then test it on a new fixed set of 48 random students.

Comparative analysis – Figure 6.4 (left) showcases performance across training for our considered Meta-ACL conditions and ACL baselines. Both AGAIN and IN significantly outperform ALP-GMM ($p < .001$ for both, using Welch's t-test at 200k episodes). The initial performance advantage of IN w.r.t AGAIN is due to the greedy nature of IN, which only exploits the expert curriculum while AGAIN complements it with ALP-GMM for exploration. By the end of training, AGAIN outperforms IN ($p < .001$) thanks to its ability to emancipate from the curriculum priors it initially leverages. The regular CP-based curriculum priors selection used in AGAIN outperformed the random selection used in AGAIN_RND ($p < .001$ at 200k episodes), while being not significantly inferior to the Ground Truth variant AGAIN_GT ($p = 0.16$). Because we assume no expert knowledge over the set of students to train, i.e. their respective initial learning subspace is unknown, ADR– which relies on being given an initial easy task – fails to train most students when given randomly selected starting subspace (among the 4 possible ones). By contrast, this showcases the ability of AGAIN to autonomously and efficiently infer such expert knowledge.

Varying classroom size experiment – An important property that must be met by a meta-learning procedure is to have a monotonic increase of performance as the database of information being leveraged increases. Another important expected aspect of Meta-ACL is whether the approach is able to generalize curriculum generation to students that were never seen before. To assess whether these properties hold on AGAIN, we consider the full student distribution of the toy environment, i.e. 400 possible student types. We populate a new history $\mathcal{H}^{\mathcal{S}}$ by training (with ALP-GMM) a 400-students classroom (one per student type). We then analyze the end performance of AGAIN and IN on a fixed test set of 96 random students when given increasingly smaller subsets of $\mathcal{H}^{\mathcal{S}}$. The smaller the subset, the harder it becomes to generalize over new students. Results, shown in fig. 6.4 (right),

demonstrate that both AGAIN and IN do have monotonic performance increasments as the classroom grows. With as little as 10% of possible students in the classroom, AGAIN statistically significantly ($p < .001$) outperforms ALP-GMM on the new student set, i.e. it generalizes to never seen before students.

6.5.2 Meta-ACL for DRL students in the Walker-Climber Environment

To study Meta-ACL with DRL students, we present *Walker-Climber*, a new Box2D locomotion environment[1] with a 2D parametric PCG that encodes a large space of tasks (see fig. 6.5). The first task parameter controls the spacing between walls that are positioned along the track, while the second task parameter sets the y-position of a gate that is added to each wall. Positive rewards are collected by going forward. To simulate a multi-modal distribution of students well suited to study Meta-ACL, we randomize the student's morphology for each new training (i.e. each seed): It can be embodied in either a bipedal walker, which will be prone to learn tasks with near-ground gate positions, or a two-armed climber, for which tasks with near-roof gate positions are easiest. We also randomize the student's limb sizes, which can vary from the length visible in fig. 6.5 to 50% shorter.

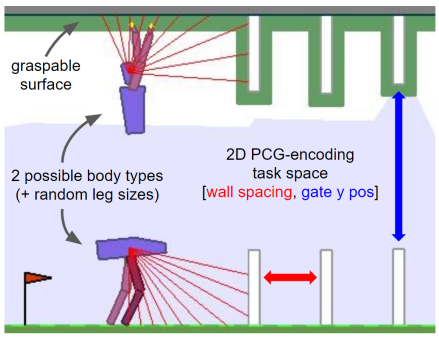


Figure 6.5: Our proposed *Walker-Climber* parametric env. to study Meta-ACL with DRL students.

Results

In the following experiments, our Meta-ACL variants leverage a teaching history $\mathcal{H}^{\mathcal{S}}$ built from a classroom of 128 randomly drawn Soft-Actor-Critic (Haarnoja et al., 2018a) students (with varying embodiments and initial policy weights) trained with ALP-GMM. We then compare ACL and Meta-ACL variants on a fixed set of 64 new students and report the mean percentage of mastered environments (i.e. $r > 230$) from 2 fixed expert test sets (one per embodiment type) across training. The CP^{pre} vector is built using a uniform pre-test set of $m = 225$ tasks, performed after 2 million agent steps out of 10. See appendix C.4 for additional experimental details.

Qualitative view – Figure 6.6 (left) showcases the evolution of task sampling when using AGAIN to train a new student. Three distinct phases emerge along training: 1) A pre-training exploratory phase used to gather information about the student's capabilities, 2) After building the CP^{pre} vector and inferring the most appropriate curriculum priors from $\mathcal{H}^{\mathcal{S}}$, AGAIN paces through the resulting IN curriculum while mixing it to ALP-GMM,

[1] This environment bears some similarities (e.g. it features climbers) with the *Parkour* environment presented in chapter 5. In practice, this work was done in parallel to the *TeachMyAgent* benchmark, hence the overlap.

and 3) AGAIN emancipates from IN after completing it.

Comparative analysis – As shown in figure 6.6 (right), through its use of curriculum priors, AGAIN outperforms the teaching performances of ALP-GMM on our environment. AGAIN's students master an average of 41% of the test set at 10M steps, compared to 31% for ALP-GMM's ($p < .001$) after 10.5M steps (0.5M training steps added to account for AGAIN additional pre-test time). AGAIN performs better than its AGAIN_RND random prior selection variant, and is not statistically different ($p = 0.8$) from ground truth sampling (AGAIN_GT), although only by the end of training. While AGAIN and IN initially have comparable performances, after 7 Millions training steps – a point at which most students trained with IN or AGAIN reached the last Gaussian mixture from IN– AGAIN starts to outperform IN, with a significant advantage by the end of training ($p < 0.02$). This showcases the advantage of emancipating from the expert curriculum once completed. As in the toy environment experiments, when given randomly selected starting subspaces (since we assume no expert knowledge), ADR fails to train most students. Figure 6.6 (top) showcases which tasks of the post-training test set were mastered for each SAC student trained with Random, ALP-GMM or AGAIN. AGAIN better exploits the learning capacities of its students, leading to a superior overall mastery of the task space.

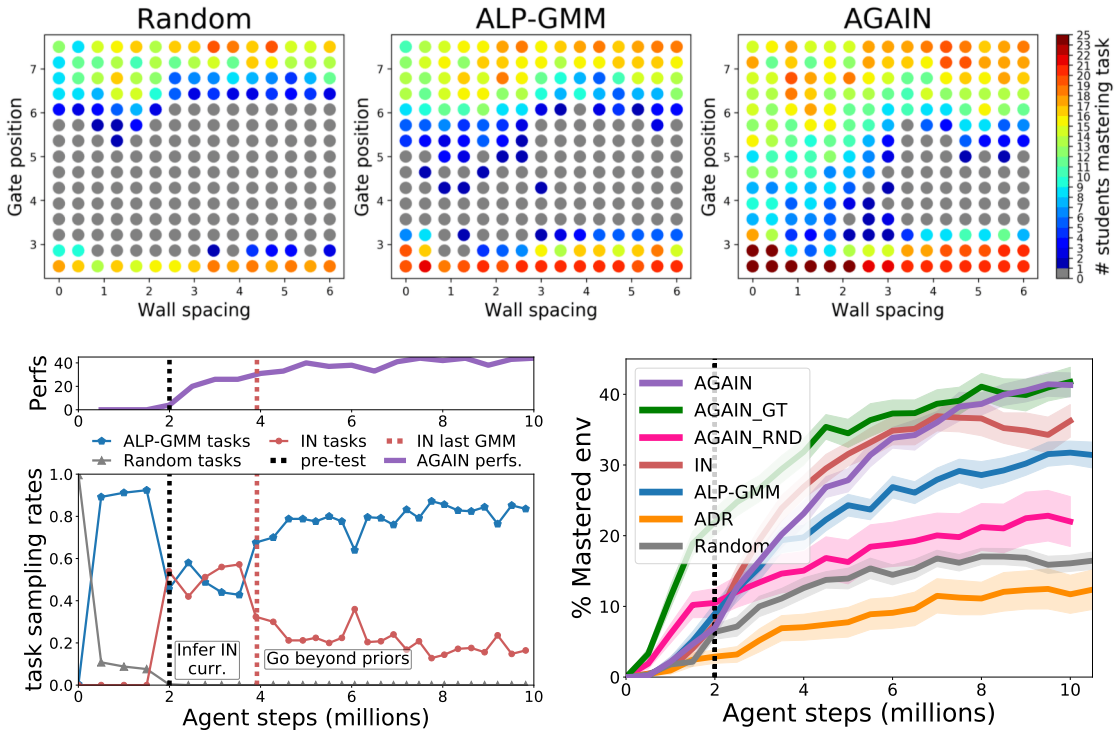


Figure 6.6: **Bottom Left:** Example of evolution of task sampling when using AGAIN in *Walker-Climber*. (1 seed). **Bottom Right:** Average performances of AGAIN with variants and baselines in the same env.. 64 seeds, sem plotted. The vertical dashed black line indicates when pre-training ends for Meta-ACL conditions. **Top:** Overall post-training performances for each student training with Random, ALP-GMM and AGAIN. Each test task (dot) is colored according to how many students (out of 64) mastered it (i.e. obtained $r > 230$).

6.5.3 Applying Meta-ACL to a Single Student: Trying AGAIN instead of Trying Longer

Given a single DRL student to train (i.e. no history $\mathcal{H}^S$) and an expert knowledge-free setup, current ACL approaches leverage task-exploration (as in ALP-GMM). We hypothesize that these additional tasks presented to the DRL learner could have a cluttering effect on the gathered training data, i.e. it adds noise in its already brittle gradient-based optimization and leads to suboptimal performances. We propose to address this problem by modifying AGAIN to fit this no-history setup. To do so, we assume the ability to restart the student once along training. More precisely, instead of pre-testing the student to find appropriate curriculum priors from $\mathcal{H}^S$, we split the training of the target student into a two-stage approach, where 1) the DRL student is first trained with ALP-GMM (with high-exploration), and then 2) we extract curriculum priors from the training history of the first run and use them to re-train the same agent *from scratch*.

Results

We test our modified AGAIN along with variants and baselines on the *Stump Tracks* environment proposed in chapter 4, which generates walking tracks paved with stumps whose height and spacing are defined by a PCG-encoding 2D vector. As in our experiments in chapter 4, we test our approaches with both the default walker and a modified short-legged walker, which constitutes an even more challenging scenario (as the task space is unchanged). Performance is measured by tracking the percentage of mastered tasks from a fixed test set. See app. C.5 for additional results.

Figure 6.7 showcases our proposed approach on the short (top) and default (bottom) walker setups, with a SAC student (Haarnoja et al., 2018a). In both cases, AGAIN statistically significantly outperform ALP-GMM ($p < 0.05$). This performance gap is most striking in the short walker setup. This result is expected: this hard training scenario is more likely to benefit from adaptive curriculum generation since there are less feasible task subspaces w.r.t. the default walker setup.

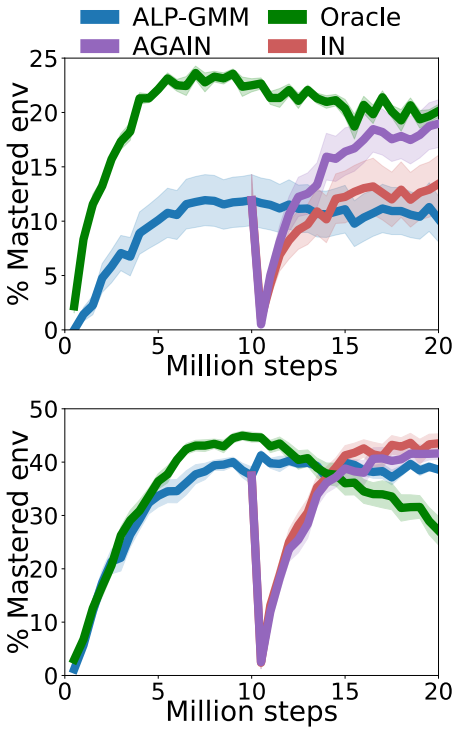


Figure 6.7: Given a single DRL student to train, AGAIN outperforms ALP-GMM in a parametric BipedalWalker environment. sem plotted, 32 seeds. **Top:** Experiments with *short* bipedal walkers. **Bottom:** Experiments with *default* bipedal walkers.

AGAIN vs IN – In the default walker experiments, AGAIN and IN reach similar end-performances ($p > 0.05$). This is unsurprising: in this simple setting preliminary trainings on 10 million environment steps with ALP-GMM always (in our 32 seeds) find

feasible task subspaces to focus on. This means appropriate curriculum priors can be consistently extracted for re-training, i.e. IN curricula are sufficient, and complementing it with exploration as in AGAIN is unnecessary. However, on the short walker scenario, mixing ALP-GMM with IN is essential: while IN end performances are not statistically significantly superior to ALP-GMM, AGAIN clearly outperforms ALP-GMM ($p < 0.01$), reaching a mean end performance of 19.0. This is due to the difficulty of the short walker scenario: after the preliminary 10 million training steps, 16/32 SAC students did not manage to learn any locomotion policy. All these run failures led to many GMMs lists $\mathcal{C}$ used in IN to be of very low-quality, i.e. low-quality curriculum priors, which illustrates the advantage of AGAIN that is able to complement them with further exploration.

6.6 Conclusion and Discussion

In this work we attempted to motivate and formalize the study of Classroom Teaching problems, in which a set of diverse students have to be trained optimally, and we proposed to attain this goal through the use of Meta-ACL algorithms. We then presented AGAIN, a first Meta-ACL baseline, and demonstrated its advantages over classical ACL and variants for CT problems in both a toy environment and in *Walker-Climber*, a new parametric locomotion environment with DRL learners. We also showed how AGAIN can bring performance gains over ACL in classical single student ACL scenarios.

Limitations & future work. – AGAIN is a first Meta-ACL *baseline*, i.e. a first step aiming to seed further research. Many parts of its learning pipeline could be improved. For instance, in future work, instead of building large pre-test sets spanning over the task space, AGAIN could use adaptive approaches to build compact pre-test sets, e.g. using decision tree-based test pruning methods. AGAIN relies on pre-defining the length of the initial pre-training period. This hyperparameter is crucial and must be carefully selected by the experimenter: if pre-training is too short, the pre-tests and resulting competence profiles of students will not be easily separable. If too long, pre-training will drain training time for the main training session, thus hindering performances. An interesting avenue for future work would be to study how to avoid relying on pre-tests to select curriculum priors, e.g. to extract similarity measures between students based on their training history $\mathcal{H}^{int}$. Additionally, although for simplicity we focused on extracting useful curriculum priors from a single student in the history of previously trained students, combining curriculum priors from multiple previously trained learners, or even adaptively switching from which student to extract curriculum priors along training, appears like interesting research directions.

While AGAIN is built on top of an existing ACL algorithm, developing an end-to-end Meta-ACL algorithm that generates curricula using a DRL teacher-policy trained across multiple students is also a promising line of work to follow. In practice, approaching this task-level control problem with classical DRL algorithms is challenging because of sample efficiency: an ACL policy has to be learned and exploited along interaction windows typically around a few tens of thousands of steps. This has to be compared to the tens of millions or sometimes billions of interaction steps necessary to train a DRL policy for robotic control tasks. For this reason, most recent ACL research has focused on reducing

the teaching problem into a Multi Armed Bandit setup, which ignores the sequential dependency over student states implied in POMDP settings (Matiisen et al., 2017; Mysore et al., 2018; Colas et al., 2019), including chapter 4. One potential research direction towards this end-to-end Meta-ACL goal would be to study how to modify the DQN curriculum generator proposed in Narvekar & Stone (2020) to fit a Classroom Teaching scenario.

Chapter 7

SocialAI: Environments for the Development of Socio-Cognitive Skills in DRL

Contents

The long-term objective of the present research and of the AI/ML field is to contribute towards building proficient machine learners. But what does it mean to be *proficient*? Ideally, a proficient artificial agent should be able to efficiently interact among humans. To do so, such an agent should possess cognitive abilities akin to those of humans. A core challenge towards this (never-ending?) quest for *human-level AI* (McCarthy, 2007) is to know on which aspects of human intelligence to focus research efforts. So far, based on evidence from developmental sciences stressing the importance of self-experimentation and incremental learning in child development (section 1.1), the present manuscript focused on studying ACL algorithms, i.e. how to best organize the presentation of new challenges to learn from such that complex behaviors can be acquired efficiently. While navigation (chapter 3) and locomotion (chapter 4,5 and 6) environments provide relevant testbeds to study ACL, this last experimental chapter proposes a complementary research direction focused on perhaps the most important aspect of human cognition: *social intelligence*.

We argue that studying social intelligence – more precisely the acquisition of *social skills* – and studying ACL methods are related research fields. Both are based on a

fundamental perspective on cognitive development: intelligent agents are sculpted in large part by the properties of their (evolving) environments (Nisioti & Moulin-Frier, 2020). Thus, to sculpt intelligent agents, one needs to sculpt environments. One way to achieve this is to do ACL, i.e. to control which environments should be presented – and in which order – to a learning agent. Another approach, if one is interested in human-like intelligence, is to develop environments that include some of the fundamental learning challenges humans have: social skills (Eppe & Oudeyer, 2021).

In the following chapter, we will see that, within the DRL field, the study of social skills motivated multiple works, especially on embodied language use. However, in light of the diversity of social scenarios humans experience on a daily basis, current works focus on rather simple and rigid social interactions. We will then explain how concepts from cognitive sciences could help AI to draw a roadmap towards human-like intelligence, with a focus on its social dimensions. As a first step, we propose to expand current research to a broader set of core social skills. To do this, we present *SocialAI*, a set of environments to study the acquisition of social skills by DRL agents using multiple grid-world environments featuring other (scripted) social agents. We then study the limits of a recent SOTA DRL approach when tested on *SocialAI* and discuss important next steps towards proficient social agents. Videos and code are available at `https://sites.google.com/view/socialai`.

7.1 Introduction

How do human children manage to reach the social and cognitive complexity of human adults? As discussed in chapter 1, an influential perspective on this question are Jean Piaget's foundational theories of cognitive development (Piaget, 1952). For Piaget, the child is a solitary thinker, a "little scientist" deciding which experiments to perform to challenge its assumptions and improve its representation of the world. While he acknowledged that social context can assist development, for him cognitive maturation happens mainly through the child's solitary exploration of their world. Social proficiency appears as a by-product of the child's internal maturation.

While Piaget's theories are centered on internal mechanisms, for Vygotsky, a soviet scholar from the 1920s, a main driver towards "higher-level" cognition are socio-cultural interactions with other human beings (Vygotsky & Cole, 1978). For him, many high-level cognitive functions a child develops first appear at the social level and then develop at the individual level. This leap from interpersonal processes to intrapersonal processes is referred to as *internalization*. A typical example of this process is learning to count. Children first learn to count at loud, i.e. with language and social guidance, which is an interpersonal process. As the child improves, it will learn to count in his head, no longer requiring any external guidance: counting became internalized, and will be a first step towards more complex forms of abstract thinking. Vygotsky's theories influenced multiple works within cognitive science (Clark, 1996; Hutchins, 1996), primatology (Tomasello, 1999) and the developmental robotics branch of AI (Billard & Dautenhahn, 1998; Brooks et al., 2002; Cangelosi et al., 2010; Mirolli & Parisi, 2011).

Out of these two perspectives, the Piagetian view on development is most aligned with mainstream DRL research, which mainly focuses on sensorimotor development through navigation and object manipulation problems rather than language based social

interactions (Mnih et al., 2015; Lillicrap et al., 2016; Andrychowicz et al., 2017). The study of language has been mostly separated from DRL, into the field of Natural Language Processing (NLP), which is mainly focused on learning (disembodied) language models for text comprehension and/or generation (e.g. using large text corpora as in Brown et al. (2020)).

In the last few years however, recent advances in both DRL and NLP made the machine learning community reconsider experiments with language based interactions (Luketina et al., 2019; Bender & Koller, 2020). Text-based exploratory games have been leveraged to study the capacities of autonomous agents to properly navigate through language in abstract worlds (Côté et al., 2018; Prabhumoye et al., 2020; Ammanabrolu et al., 2021). While these environments allow meaningful abstractions, they neglect the importance of embodiment for language learning, which has long been identified as an essential component for proper language understanding and grounding (Cangelosi et al., 2010; Bisk et al., 2020). Following this view, many works attempted to use DRL to train embodied agents to leverage language, often in the form of language-guided RL agents (Chevalier-Boisvert et al., 2019; Colas et al., 2020a; Hill et al., 2020b; Akakzia et al., 2021) and Embodied visual Question Answering (EQA) (Das et al., 2018; Gordon et al., 2018), and more recently on interactive question production and answering (Abramson et al., 2020). Language use has also been studied in multi-agent emergent communication settings, both in embodied and disembodied scenarios (Mordatch & Abbeel, 2018; Jaques et al., 2019; Lowe et al., 2020; Woodward et al., 2020).

One criticism that could be made over aforementioned works in light of Vygotsky's theory is the simplicity of the "social interactions" and language-use situations that are considered: in language-conditioned works, the interaction is merely just the agent receiving its goal as natural language within a simple and rigid interaction protocol (Luketina et al., 2019). In EQA, language-conditioned agents only need to first navigate and then produce simple one or two words answers. And because of the complexity of multi-agent training, studies on emergent communication mostly consider simplistic language (e.g. communication bits) and tasks.

To catalyze research on building proficient social agents, we propose to identify a richer set of socio-cognitive skills than those currently considered in most of the DRL and NLP literature. We do not claim to provide an exhaustive list, but rather a preliminary set of important social abilities, aiming to seed further investigations. We organize this set along 3 dimensions. Proficient social agents must be able to master *intertwined multimodality*, i.e. coordinating multimodal actions based on multimodal observations. They should also be able to build an (explicit or implicit) *theory of mind*, i.e. inferring other's mental state, e.g. beliefs, desires, emotions, etc. Lastly, they should be able to learn diverse and complex *pragmatic frames*, i.e. social interaction protocols described as "verbal or non-verbal patterns of goal-oriented behaviors that evolve over repeated interactions between learners and teachers" Bruner (1985).

Based on these target socio-cognitive skills, we present *SocialAI* 1.0, a set of grid-world environments as a first step to foster research in this direction (see fig. 7.1). To study complex social scenarios in reasonable computational time, we consider single-agent learning among scripted agents (a.k.a. Non-Player-Characters or NPCs) and use low-dimensional observation and action spaces. We also use templated language, enabling to

emphasize the under-studied challenges of dealing with more complex and diverse social and pragmatic situations. To showcase the relevance of *SocialAI*, we study the failure case of a current SOTA DRL approach on this suite of environments through detailed case studies.

Social agents are not objects. – Although social peers could be seen as merely complex interactive objects, we argue they are in essence quite different. Social agents (e.g. humans) can have very complex and changing internal states, including intents, moods, knowledge states, preferences, emotions, etc. The resulting set of possible interactions with peers (social affordances) is essentially different than those with objects (classical affordances). In cognitive science, an affordance refers to what things or events in the environment afford to an organism (de Carvalho, 2020). A flat surface can afford "walking-on" to an agent, while a peer can afford "obtaining directions from". The latter is a social affordance, which may require a social system and conventions (e.g. politeness), implying that social peers have complex internal states and the ability to reciprocate. Successful interaction might also be conditioned on the peer's mood, requiring communication adjustments. Training an agent for such social interactions most likely requires drastically different methods – e.g. different architectural biases – than classical object-manipulation training. In *SocialAI* we simulate such social peers using scripted NPCs. We argue that studying isolated social scenarios featuring NPCs in tractable environments is a promising first step towards designing proficient social agents able to engage with humans. We argue NPCs are not mere objects. Indeed, across our environments, our NPCs can deliver complex social interactions such as multi-step verbal interactions, they can require a basic form of politeness to condition their compliance, or they can provide multimodal (embodied) demonstrations of a task.

Grounding language in social interactions. – In AI, *natural language* often refers to the ability of an agent to use a large vocabulary and complex grammar. We argue that this is but one dimension of the *naturalness* of language. Another, often overlooked, dimension of this *naturalness* refers to language grounding, i.e. the ability to anchor the meaning of language in physical, pragmatic and social situations (Steels, 2007). The large literature on language grounding has so far mostly focused on grounding language into physical action (Cangelosi et al., 2010; Chevalier-Boisvert et al., 2019; Colas et al., 2020a): here the meanings of sentences refer to actions to be made in interaction with objects (e.g. "Grasp the blue box"). However, natural language as used by humans is also strongly grounded in social contexts: not only the interpretation of language requires understanding the social context (e.g. taking into account intents or beliefs of others), but meanings can refer to social actions (e.g. "Help your friend to learn his dance lesson"). Here, an important aspect of language naturalness refers to the diversity of kinds of pragmatic social situations in which it is grounded: the work presented here aims at making steps in this direction.

Main contributions:

- An outline of a set of core socio-cognitive skills necessary to enable artificial agents to efficiently act and learn in a social world.

- *SocialAI*, a set of grid-world environments including complex social situations with scripted NPCs to study the capacity of DRL agents to learn socio-cognitive skills organized across several dimensions.
- Performance assessment of a SOTA DRL approach on *SocialAI* and analysis of its failure using multiple case studies.

7.2 Background

7.2.1 Earlier calls for socially proficient agents

This work aims to connect the recent DRL & NLP literature to the older developmental robotics field (section 2.2.1), which studies how to leverage knowledge from the cognitive development of human babies into embodied robots. Within this field, multiple calls for developing the social intelligence of autonomous agents have already been formulated (Billard & Dautenhahn, 1999; Lindblom & Ziemke, 2003; Mirolli & Parisi, 2011). This emphasis on the importance of social interactions for learning is probably what led Bruner to conceptualize the notion of pragmatic frames (Bruner, 1985), which has later been reused for example as a conceptual tool to theorize language development (Rohlfing et al., 2016). We intend to further motivate the relevance of this notion to enable further progress in DRL and AI.

7.2.2 Human-Robot Interaction

Interactions with knowledgeable human teachers is a well-studied form of social interaction. Many works within the Human-Robot Interaction (HRI) and the Interactive Learning field studied how to provide interactive teaching signals to their agents, e.g. providing instructions (Grizou et al., 2014), demonstrations (Argall et al., 2009; Grollman & Billard, 2011), or corrective advice (Celemin & Ruiz-del Solar, 2015). In Vollmer et al. (2016), authors review this field, showing that many of the considered interaction protocols can be reduced to a restricted set of pragmatic frames. They note that most of these works consider single rigid pragmatic frames. Echoing this observation, this work invites to study a broader set of social situations, e.g. requiring agents to both move and speak, and even to learn to interact in a *diversity* of pragmatic frames. Catalyzing research on DRL and social skills seems even more relevant now that many application-oriented works are beginning to leverage RL and DRL into real-world humanoid social robots (Akalin & Loutfi, 2021).

7.2.3 Recent Works on Language Grounded DRL

Building on NLP, developmental robotics, and previous works on classical goal-conditioned DRL (Colas et al., 2020b), a renewed interest emerged towards the development of embodied autonomous agents able to process language (Luketina et al., 2019). Most approaches were proposed to design language conditioned agents in instruction-following scenarios (e.g. “go to the red bed”). In Hermann et al. (2017), authors train a

DRL model from pixels in a 3D world by augmenting instruction-following with auxiliary tasks (language prediction and temporal autoencoding), enabling their agent to follow object-relative instructions ("pick the red object next to the green object"). Multiple other approaches were studied to ease learning. Some works proposed to leverage pre-trained language models (Hill et al., 2020b) or demonstrations (Fu et al., 2019; Lynch & Sermanet, 2020), from which rewards can be learned (Bahdanau et al., 2019). Descriptive feedbacks have also been used (Nguyen et al., 2021). In Colas et al. (2020b), authors propose to combine descriptive feedbacks with imagining new goals. Hill et al. (2020a) try to assess to which extent vanilla language conditioned agents are able to perform systematic generalization (combining known concepts/skills into new ways). In embodied visual question answering works, agents are conditioned on questions, requiring them to navigate within environments and then produce an answer ("what color is the bed ?") (Das et al., 2018; Gordon et al., 2018). Compared to these previous works, *SocialAI* aims to enlarge the set of considered scenarios by studying how language conditioned agents are able to ground and produce language within diverse forms of social interactions among embodied social peers.

Closely related to the present work is Abramson et al. (2020), which is an invitation to focus research on building embodied multimodal agents suited for human-robot interactions in the real-world. Towards this goal, authors propose a series of experiments on a simulated 3D playroom environment designed for multi-agent interactive scenarios featuring both autonomous agents and/or human players. The main focus of the paper is in proposing novel ways to leverage human demonstrations to bootstrap agents performance and allow meaningful interactive sessions with humans (as scaffolding a randomly acting agent is a tedious journey). Because of the complexity of their considered experiments (imitation learning, 3D environments, pixel-based, human in the loop, etc.), their work only considers the two now-common social interaction scenarios: visual question answering and instruction-following. The novelty of their setup is that these questions/instructions are alternatively produced or tackled by learning agents in an interactive fashion. In *SocialAI*, we focus on a lighter experimental pipeline (2D grid-world, low dimensional symbolic pixels, no humans) such that we are able to study a broader range of social scenarios, requiring multi-steps conversations and interactions with multiple (scripted) agents within a single episode.

7.2.4 Testbeds on Embodied Agents and Language

Multiple benchmarks featuring language and embodied agents already exists. The BabyAI (Chevalier-Boisvert et al., 2019) and gSCAN benchmarks (Ruis et al., 2020) test language conditioned agents on grid-world environments. BabyAI focuses on assessing the sample efficiency of tested agents while gSCAN targets systematic generalization. Misra et al. (2018) extends this type of benchmark to 3D environments. In contrary to *SocialAI*, these benchmarks do not consider multimodal action spaces, i.e. agents do not produce language. Besides, they only consider a single rigid social interaction protocol: instruction-following.

Related to instruction-following benchmarks are testbeds for embodied visual question answering (Gordon et al., 2018; Das et al., 2018). Here agents are conditioned on questions:

they must navigate within an environment to collect information and produce an answer (i.e. a one or two word output).

Puig et al. (2021) propose a new benchmark to test social perception in machine learning models. Learning agents must infer the intent of a scripted agent in a 3D world (using a single demonstration) to better collaborate towards a similar goal in a new environment. Here again, despite being novel and relevant, only a single social interaction is considered: cooperation towards a common goal, which in that case does not require language use nor understanding.

In between classical disembodied NLP testbeds (Johnson et al., 2017; Wang et al., 2018; Zadeh et al., 2019) and previously discussed embodied language benchmarks is the LIGHT environment (Urbanek et al., 2019), A multiplayer text adventure game allowing to study social settings requiring complex dialogue production (Ammanabrolu et al., 2021; Prabhumoye et al., 2020). While they consider a text world, i.e. a virtual embodiment, the *SocialAI* environments we propose tackle the arguably harder and richer setting of egocentric embodiment among embodied social peers. Text worlds have also been used in combination with an embodied environment to demonstrate how language-based planning (in text worlds) can benefit instruction-following (Shridhar et al., 2021).

Within the multi-agent RL field, Mordatch & Abbeel (2018) propose embodied environments to study the emergence of grounded compositional language. Here language is merely a discrete set of abstract symbols that can only be used one at a time (per step) and whose meanings must be negotiated by agents. While symbol negotiation is an interesting social situation to study, we leave it to future work and consider scenarios in which agents must enter an already existing social world (using non-trivial language). In Jaques et al. (2019), authors present multi-agent social dilemma environments requiring the emergence of cooperative behaviors through communication. In their work, communication is strictly non-verbal, while we consider both non-verbal communication (e.g. gaze following) and language based communication.

7.3 Social Skills for Socially Competent Agents

Social skills have been extensively studied in cognitive science (Riggio, 1986; Beauchamp & Anderson, 2010) and developmental robotics (Cangelosi et al., 2010). Based on a literature survey, this section identifies *a* set of core social skills when aiming to train socially competent artificial agents.

Intertwinded Multimodality

Intertwinded multimodality refers to the ability to interact and use multiple modalities (verbal and non-verbal) in a coordinated manner. A proficient agent should be able to act using both primitive actions (moving) and language actions (speaking), and to process both visual and language observations of social peers. Importantly, socially competent agents must be able to interact using multiple modalities in an *intertwined* fashion. By intertwined multimodality we refer to an agent's ability to adapt its multimodal interaction sequence, rather than following a pre-established progression of modalities. For example,

in EQA(Das et al., 2018), the progression is always as follows: 1) a question is given to the agent at the beginning of the episode, 2) the agent moves through the environment to gather information, and 3) upon finding an answer it responds (in language) and the episode ends. By the term *intertwined multimodality* we aim to emphasize that the modalities often interchange and the question of "when to use which modality" is non-trivial, e.g. sometimes the relevant information can be obtained by *asking* for it and sometimes by *looking* for it.

Theory of Mind

Theory of Mind (ToM) refers to the ability of an agent to attribute to others and itself mental states, including beliefs, intents, desires, emotions and knowledge (Wellman, 1992; Flavell, 1999). An agent that has ToM perceives other participants as *minds* like itself. This enables the agent to theorize about other's intents, knowledge, lack of knowledge etc. Here we outline some, of many, different perspectives of ToM to better demonstrate how ToM is essential for human social interactions:

Inferring intents: the agent is able to infer, based on verbal or non-verbal cues, what others will do or want to do, e.g. that some social peers are liars/trustworthy.

False belief: the agent understands that someone's belief (including its own) can be faulty (Baillargeon et al., 2010).

Imitating or emulating social peer's behavior: agent can imitate a behavior or a goal seen in a social peer, e.g. upon observing a peer cut onions the agent is able to cut the onions himself, either with the same movement or with its own strategy.

Pragmatic Frames

Pragmatic frames refer to the regular patterns characterizing the unfolding of possible social interactions (equivalent to an interaction protocol or a grammar of social interactions). Pragmatic frames simplify learning by providing a stable structure to social interactions. An example of a pragmatic frame are turn taking games. By playing those games, a child extracts the rule of each participant having his "turn". It can then generalize this rule to a conversation, where it understands that it shouldn't speak while someone else is speaking. We propose to outline several facets of pragmatic frames that proficient social agents should be able to master:

Learning a pragmatic frame: The agent is able to learn a frame through social interactions, without it being manually hand-coded (e.g. as in instruction-following scenarios). Through rich social interactions (e.g. dialogues) with one or several peers, the agent should be able to infer the structure of the interaction pattern (frame), extract potential instructions, and leverage them appropriately.

Teaching frames: A specific type of pragmatic frames involves a teacher explicitly teaching a certain content via a *slot*. A slot refers to the place in the interaction

sequence holding the variable learning content. A parent teaching a child words with the help of a picture book is one such teaching frame. Upon seeing a picture with a dog, a parent might point to the dog, say "Look, it's a dog", and establish eye contact to verify that the child understood the message. Upon, however, seeing a picture of a cat he might say "Look, it's a cat". Here "dog" and "cat" are learning contents and the slot is the location of those words in the sequence ("Look, it's a <*slot*>"). A socially competent agent should be able to learn such a frame and extract the *learning content* from it.

Roles: The agent is able to not only understand the relevance of various participants for achieving a shared goal but also learn about the others' role just from playing its own. For example, in a setting where one agent opens the door to enable another agent to exit the room, the *exiting* agent should be able to learn what the role of opening the door consists of. This exiting agent should then be able to open the door for another agent with little or no additional training. The social interaction described above consists of one frame viewed from two different perspectives corresponding to two different roles. Socially proficient agents should be able to easily learn this whole frame by experiencing it just from their own perspective.

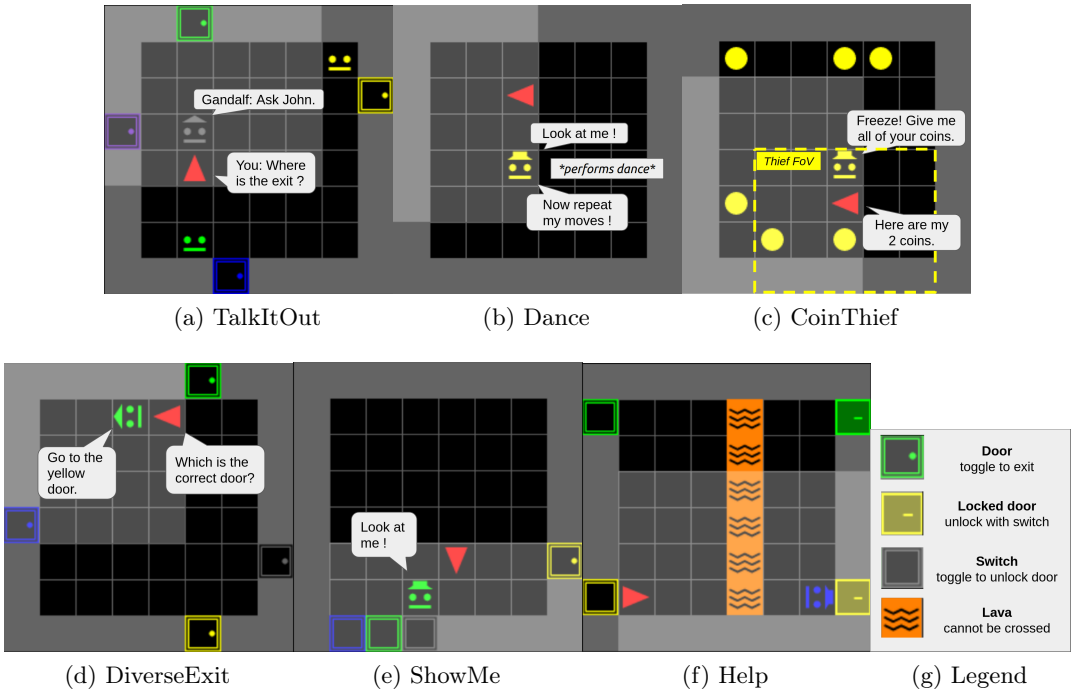
Diversity: The agent can learn many different frames and differentiate between them. Furthermore, the agent can reuse those frames in new situations and even negotiate and construct new ones.

Frame changes: An agent is able to detect and adjust to a change of the current pragmatic frame. For example, while playing football, we are able to participate in *small talk* with another player.

7.4 The SocialAI 1.0 Benchmark

As a first step to catalyze research on developing socially proficient autonomous agents, we present *SocialAI* 1.0 (see fig. 7.1), a set of grid-world environments designed to challenge learners on important aspects of the core social skills mentioned in sec. 7.3. In this section, we briefly present each environment (see app. D.1 for details) and highlight how they require various subsets of the aforementioned core social skills.

Common components – The key design principle of *SocialAI* environments is to allow the study of complex social situations in reasonable computational time. As such, we consider single-room grid-world environments (8×8 grid), based on minigrid (Chevalier-Boisvert et al., 2018). The learning agent can both navigate using discrete actions (e.g. turn left/right, go forward, toggle) and use template-based language generation (environment-dependent). As observations, the agent receives a partial 7×7 agent-centric symbolic pixel grid (see highlighted cells in fig. 7.1), with 4 dimensions per cell (type, color, status, orientation). Additionally, the agent receives the history of observed language outputs from NPCs preceded by the NPC's name (e.g. "John: go to the red door"). A positive reward is given only upon successful completion of the social scenario (discounted by time taken).



(a) TalkItOut (b) Dance (c) CoinThief

(d) DiverseExit (e) ShowMe (f) Help (g) Legend

Figure 7.1: *SocialAI 1.0* is composed of multiple grid-world environments featuring scripted NPCs, well-suited to study how to design socially proficient DRL agents.

In the following description of environments, unless stated otherwise, the agent, all objects and all NPCs are spawned randomly for each new episode. Each description highlights the socio-cognitive skills required to solve the environment (see table 7.1 for a recapitulating overview).

TalkItOut – The agent has to exit the room using one of the four doors (by uttering "Open Sesame" in front of it). The environment features a wizard and two guides (one lying, one trustworthy). To find out which door is the correct one the agent has to ask the trustworthy guide for directions, and to find out which guide is trustworthy it has to query the wizard (which requires a preliminary politeness formula: "Hello, how are you?"). Solving TalkItOut requires mastering *intertwined multimodality*, basic *Theory of Mind* (inferring ill-intentions), and a basic *pragmatic frame* (the agent must stand near NPCs to interact with them).

Dance – A NPC demonstrates a 3-steps dance pattern (randomly generated for each episode) and then asks the agent to reproduce this dance. Each dance step is composed of a movement action and, half of the time, of an utterance. To solve *Dance*, agent must reproduce the full dance sequence. Multiple trials are authorized. Only trials performed after the NPC completed his dance are recorded. This requires the agent to be able to infer that the NPC is setting up a *teaching pragmatic frame* ("Look at me" + `do_dance_performance` + "Now repeat my moves"), requiring the agent to *imitate* a social peer, process *multimodal observations* and produce *multimodal actions*.

CoinThief – In a room containing 6 coins, a thief NPC spawns near the agent, and utters that the agent must give "all of its coins". To obtain a positive reward, the agent must give (using language) exactly the number of coins *that the thief can see* (the thief's field of view is a 5×5 square, i.e. a smaller version than the agent's). This requires *Theory of Mind* as the agent must understand that the thief holds *false belief* over the agent's total number of coins and must infer how many coins he actually sees.

ShowMe – The agent has to exit the room through the locked door. To unlock the door it has to press the correct button, and to find out which button is the correct one it has to look at the NPC. The NPC waits for the agent to establish eye contact, then presses the correct button and exits the room. Solving ShowMe requires that the agent infers the *teaching pragmatic frame* and imitates the NPC's goals (pressing a button, and exiting the room).

DiverseExit – The agent has to exit the room using the correct door (one out of four). To find out which door is the correct one it has to ask the NPC. There are twelve different NPCs which can be present in the environment (each episode a random one is chosen). Each NPC prefers to be asked (using language) for directions differently (e.g. by standing close, by poking him, etc), i.e. via a different pragmatic frame. To solve DiverseExit the agent has to learn the diversity of frames and, most importantly, which one to use with which NPC.

Help – the environment consists of two roles (the Exiter and the Helper), one played by the agent and another by the NPC. The Exiter is placed on the right side of the room and has to exit the room using one of the two doors. The doors are locked and each has a corresponding unlocking switch on the left wall. The Helper, placed on the left side of the room, has to press the switch unlocking the door by which the agent wants to exit. Episodes are ended without reward if both switches are pressed. The agent is trained in the Exiter role, but tested in the Helper role. To solve Help the agent needs to learn about both roles just from training as the Exiter. i.e. learn the full pragmatic frame just from seeing its own perspective of it.

SocialEnv – In this meta-environment, which contains all previous ones, we consider a multi-task setup, in which the agent is facing a randomly drawn environment, i.e. it has to infer what is the current social scenario he is spawned in (using pragmatic information collected through interaction). Mastering this environment requires to be proficient in *all of the core social skills* we proposed.

7.5 Experiments and Results

To showcase the relevance of *SocialAI* as a testbed for research towards building machine learners able to acquire rich socio-cognitive abilities, and to provide initial target baselines to outperform, we test a recent DRL architecture on our environments. Through global performance assessment and multiple case-studies, we demonstrate that this agent essentially fails to learn due to the *social complexity* of *SocialAI*'s environments.

Table 7.1: List of core socio-cognitive skills required in each environment.

Social Skills \ Envs	TalkItOut	Dance	CoinThief	ShowMe	DiverseExit	Help	SocialEnv
Intertwined m.m.	++	++	+	+	++		++
ToM - inferring intent	++	+	+	+	+	++	++
ToM - false belief			++				++
ToM - imitating peers		++	+	++		+	++
ToM-joint attention				++	+	+	++
P. Frames - Diversity	+	+	+	+	++	+	++
P. Frames -Teaching		+			+		++
P. Frames - Roles						++	++

Baselines – Our main baseline is a PPO-trained (Schulman et al., 2017) DRL architecture proposed in Hui et al. (2020). We chose this model as it was designed for language-conditioned navigation in grid worlds, which is similar to our setup (although in our case language input is not fixed but varies along interactions). We modify the original architecture to be Multi-Headed, since our agent has to both navigate and talk, and thereafter name the resulting condition *PPO*. We also consider a variation of this baseline trained with additional intrinsic exploration bonuses (PPO +Explo). We consider two different exploration bonuses to reward the discovery of either new utterances or new visual objects. In each environment, we determined empirically the optimal set of exploration bonus (visual only, utterance only, or both), and only report results for the best configuration. Finally, as a lower-baseline, we consider an ablated, non-social version of our PPO agent, from which observation inputs emanating from NPCs are removed (*Unsocial PPO*). See appendix D.2.1 for details.

7.5.1 Overall Results

For each condition, 16 seeded runs of 30 million environment steps are performed on each environment. Performance is defined as the percentage of episodes that were solved (success rate). Post-training performance results are gathered in Table 7.2. All considered agents essentially fail to learn, on all environments. In *ShowMe*, *DiverseExit* and *Help*, both PPO and PPO with exploration bonus (PPO +Explo) performance are not statistically significantly different from Unsocial PPO, our lower-baseline agent that doesn't observe the NPC ($p > 0.5$ in all cases, using a post-training Welch's t-test). This implies that our agents are not able to leverage NPC-related inputs, i.e. they are not socially proficient. On both *TalkItOut* and *DiverseExit*, PPO agents converge to a local optimum of 25% success rate, which corresponds to ignoring the NPC and going to any door.

Table 7.2: Success rates (mean ± std. dev.) of considered baselines on *SocialAI* after 30 Millions environment steps (on a fixed test set of 500 environments). Our DRL agents fail to learn.

Env \Cond	PPO	PPO + Explo	Unsocial PPO
TalkItOut	0.25 ± 0.01	0.12 ± 0.03	0.25 ± 0.01
Dance	0.03 ± 0.01	0.03 ± 0.01	0.01 ± 0.0
CoinThief	0.45 ± 0.08	0.47 ± 0.04	0.38 ± 0.02
DiverseExit	0.25 ± 0.02	0.25 ± 0.01	0.24 ± 0.01
ShowMe	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Help	0.12 ± 0.05	0.11 ± 0.04	0.15 ± 0.06
SocialEnv	0.06 ± 0.02	0.08 ± 0.02	N/A

7.5.2 Case-studies

To better understand why our agents are failing to learn (and as a sanity check for our implementations), we present additional performance analysis of three environment-specific case studies highlighting different social skills categories of sec. 7.3: *TalkItOut* (Intertwined Multimodality), *CoinThief* (Theory of Mind), and *Help* (Pragmatic Frames).

Case study - TalkItOut

TalkItOut is challenging because the agent has to master a non-trivial progression of modalities. To talk with an NPC, apart from the language modality, both vision, and primitive actions have to be used to move close to the NPC (which is mandatory for communication). Furthermore, the agent has to learn to infer, from verbal-cues of the dialogue with the wizard, which guide is the ill-intended one (i.e. a facet of ToM). For this experiment we construct an ablation environment where the ill-intended NPC is removed, which greatly reduces the social complexity as 1) all NPCs are now well-intended, and 2) dialogue with only the trustworthy guide is sufficient to solve the task.

Results – Figure 7.2a shows the training success rates of all our baselines. We can see that, in both environments, the PPO condition gets stuck at 25% success rate, i.e. the local optimum of ignoring the NPC and going to a random door. Adding exploration bonus (PPO +Explo condition) enables the agent to overcome this local optimum, however only in the ablation environment does this result in solving the task. This shows that the social complexity introduced by the lying guide is too challenging for our conditions. These experiments suggest that our agents lack sufficient biases for both mastering intertwined multimodal interactions and inferring different intents of social peers.

Case-study - CoinThief

To assess whether it is the social complexity of the CoinThief environment that prevents our agents to learn high-performing policies, we consider a simplified version of the environment in which coins visible to the NPC have a different visual encoding from

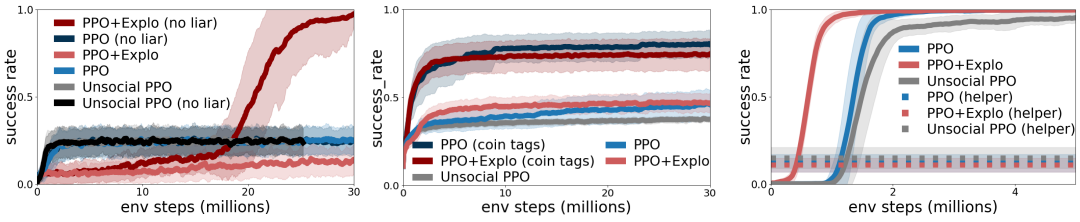
other coins. This modification removes the need to infer the NPC's field of view, i.e. the correct number of coins can be given to the NPC without any form of social awareness.

Results – Performance curves for our PPO variants on CoinThief and on the simplified CoinThief (with *coin tags* for NPC-visible coins) are shown in figure 7.2b. For both PPO and PPO +Explo, statistically significant improvements are obtained on the simplified environment w.r.t. vanilla CoinThief ($p < 0.001$): both approaches respectively reach a final performance of 0.81 and 0.75 (not statistically significantly different, $p = 0.07$).

Case study - Help

The Help environment aims to test the ability of the agent to learn about the other's role from training only on its own i.e. to learn the whole pragmatic frame just from seeing its own perspective on it. We train the agent to achieve a shared goal on one role and then evaluate in a zero-shot manner on the other.

Results – Figure 7.2c shows the training success rates for the agent in the Exiter role. The horizontal dotted lines depict the performance of the same final agents on the Helper role (depicted by the same colors). We can see that in training the Exiter role is easily solved, reaching almost perfect success rate in less than two million environment steps. Furthermore, we can see that the agent with the exploration bonus (PPO +Explo) is able to solve the task faster. When the same agents are evaluated in the Helper role, their performance drastically drops ($\leq 15\%$ success rate). Qualitative analysis shows that this non-zero success rate on Helper role is due to agents acting as if in the Exiter role, which sometimes make them press the switch due to the stochastic nature of the PPO action sampling. The agent doesn't show any implication of understanding that the roles have been reversed. These unsurprising results outline the inability of standard RL techniques to transfer the knowledge about the task to the opposite role. The agent only learns its perspective of the pragmatic frame and not the frame itself. It doesn't understand that its goal is shared with the NPC.



(a) TalkItOut (b) CoinThief (c) Help

Figure 7.2: Evolution of success rates along training in three environment specific case-studies. Mean and std. deviation plotted, 16 seeds per condition.

7.6 Discussion

In this chapter, we classified and described a first set of core socio-cognitive skills needed to obtain socially proficient autonomous agents. We presented *SocialAI*, an open-source testbed to study the acquisition of social skills for DRL learners, leveraging the computational simplicity of grid-world environments to enable the study of complex social situations. We then studied how a current SOTA DRL approach was unable to acquire the range of social capabilities required to efficiently interact in *SocialAI*. By analyzing the failure cases of this approach through multiple case studies, we were able to highlight the relevance of *SocialAI* as a tool to catalyze future research on socially proficient DRL agents.

Limitations & Future Work

This chapter presents ongoing work, a *SocialAI version 1.0*, for which we envision many updates. In the following paragraphs, we propose to discuss such promising directions for extensions and future research.

Richer social interactions – An obvious iteration would be to work on proposing more elaborated social scenarios (although this is a questionable endeavor, given the apparent failure of SOTA agents in our currently featured environments). In future work, we could study how to design NPCs with more elaborated internal states, e.g. by making them more adaptive to the learner's behavior. While we currently only consider environments with fixed sets of pragmatic frames, another interesting avenue is to design environments with *emergent* pragmatic frames, i.e. pragmatic frames that are negotiated between participants (a crucial component lacking from human-robot interaction methods (Vollmer et al., 2016)).

Towards a single parametric environment – Currently, we feature multiple environments, on which DRL agents are trained independently. Ideally, we would like to be able to train a DRL agent such that it is able to master all environments jointly, since the objective is not to obtain task-experts but rather an agent with general social skills. The *SocialEnv* environment is a first attempt towards proposing such a holistic environment. In practice however, because of the complexity of each of the 6 environments featured in *SocialEnv*, and the fact that they are randomly presented to the agent at each new episode, we doubt that such an abrupt and unstructured training regime could afford agents to learn the required social skills to solve all our environments jointly. A more promising alternative would be to design a parametric social environment generator, rich enough to encode a space of social tasks including all our current environments along with a diversity of simpler and/or different scenarios. We believe that, given such a procedural environment, a machine learner featuring an ACL method to select on which scenario to train on might be able to learn non-trivial social skills. Studying how to adapt existing ACL methods to control the selection of social tasks is an interesting area for future work.

Testing generalization – To avoid overfitting issues and study generalization, we implemented multiple forms of stochasticity in all our environments (e.g. randomized

initial positions and colors of agents, objects, and NPCs, multiple NPC policies), such that there is little chance to experience the same episode twice, and therefore little chance to be tested on environments seen in training. The failure case we presented showcased that this in-distribution generalization challenge remains to be solved. Assuming progress on such initial challenges (perhaps through the use of a holistic procedural environment as aforementioned), future work should focus on testing more complex forms of generalization. One promising avenue towards this would be to study combinatorial generalization, i.e. the ability to construct new behaviors based on combinations of known building blocks (Battaglia et al., 2018).

A need for architectural biases – The present chapter also suggests that architectural improvements are needed for DRL agents to learn to behave appropriately in multimodal social environments. One avenue towards this is to endow agents with mechanisms enabling to learn models of others' minds, which has been identified in cognitive neuroscience works as a key ingredient of human social proficiency (Vélez & Gweon, 2021). Some ideas have already been formulated regarding how to enable agents to master theory of mind, such as by using a meta-learning approach (through the observation and modeling of populations of agents) (Rabinowitz et al., 2018), or by leveraging inverse RL (Jara-Ettinger, 2019). This also points to the general open-question of what parts of biases need to be "innate", and what others could be learned through practicing diverse social interaction games in the lifetime of an agent.

Chapter 8

Discussion

Contents

In this last chapter, we will summarize the contributions proposed throughout this thesis. We will then discuss both existing and future new avenues towards building efficient machine learners by leveraging ACL and other systems shaping or biasing artificial development.

8.1 Summary of Contributions

As explained in our introductory chapter, the present research aims to contribute to the design of proficient developmental machine learners. Towards this, we presented a series of works approaching this challenge through an *environmental* perspective, i.e. through the *design of* – or *evolution of* – environments as a way to sculpt the intelligence of machine learners.

The main axis of our research focused on the study and design of automatic curriculum learning algorithms. In chapter 2, we situated ACL within the deep reinforcement learning and developmental robotics fields, which both seek to design robust, generally capable machine learners. We then formalized and surveyed ACL methods. ACL algorithms catalyze the training of machine learners by autonomously adapting the selection of their learning experiences, leading to sample efficient training and higher asymptotic performances. Many ACL methods have been proposed, using a diversity of surrogate objectives to organize curriculum generation. Learning progress is one such objective, that has been theoretically and empirically studied within the DevRob community as a powerful form of intrinsic motivation system.

In chapter 3, as a first experimental step, we studied an existing LP-based ACL approach, named Active Model Babbling, which is able to autonomously select on which objects to focus exploration and learning in multi-object scenes. We designed *Malmo Mountain Cart*, a Minecraft environment featuring tool-use and nested interactions. We used this environment to showcase that AMB could successfully train population-based

agents, enabling them to grow a set of small neural-network controllers able to cover a wide range of behaviors.

In chapter 4, we presented the ALP-GMM algorithm, a new LP-based approach particularly well-suited to train DRL agents in environments with parametric PCG encoding a continuous space of tasks. Studying ACL in such setups is important since the training diversity induced by PCG-environments has been identified as beneficial towards learning generalist policies. Using multiple parametric 2D walker environments, we showcased that agents trained with ALP-GMM were better able to handle the diversity of locomotion challenges of the task space than when using random curriculum or previous LP-based teachers.

In chapter 5, we identified the lack of a standardized testbed for ACL methods. When released, new ACL algorithms (including ALP-GMM) are only compared to a subset of existing approaches. ACL researchers test their methods using different environments, different assumptions on available expert-knowledge, and different experimental protocols. As an attempt to simplify comparative studies, we released *TeachMyAgent*, an open-source benchmark for ACL methods focused on controlling the parametric procedural generation of tasks. *TeachMyAgent* features multiple SOTA ACL methods collected from open-source code or implemented from scratch. To compare them, we designed both 1) unit-test environments centered on different ACL challenges (e.g. forgetting students, mostly unfeasible task spaces), and 2) a procedural Parkour environment combining most ACL challenges, ideal for global performance assessment. We used *TeachMyAgent* to conduct a comparative study of representative existing approaches, showcasing the competitiveness of ACL algorithms that do not use expert knowledge, such as ALP-GMM, while demonstrating that the Parkour environment remains an open problem.

An important objective of the present research is to show that, given a single DRL student to train on a challenging task space, an ACL algorithm is able to explore this space such that it can find relevant tasks for its student throughout the training session. However, in chapter 6, we argue that if multiple DRL students were to be trained on this task space, performing a sequence of independent ACL training runs appears suboptimal. Instead of repeating multiple times a costly task space exploration from tabula rasa, how could we leverage previous curriculum generations to bootstrap subsequent student training runs? Towards answering this question, we formalized and studied the concept of Meta-ACL, i.e. teacher algorithms learning to optimize curriculum generation over multiple students. We presented AGAIN, a first Meta-ACL baseline, and showcased its performance advantages compared to classical ACL when training multiple DRL students with varying morphologies on challenging task spaces.

Our previous experimental chapters can be seen as computational descendants of the Piagetian theories of cognitive development (section 1.1). By exploiting ACL ideas initially developed to model intrinsically motivated learning, we designed and studied teachers algorithms able to shape the training trajectories of learning agents. As in most DRL works, to concentrate research efforts on the study of core algorithmic blocks, we used simple experimental setups featuring navigation and/or locomotion, and discarded any form of social interactions, seen as important downstream applications rather than development testbeds. This approach is questionable from a Vygotskian perspective. For Vygotsky, socio-cultural interactions shape cognitive maturation, and not the opposite.

In chapter 7, as in a growing strand of works in DRL, we proposed to focus on the acquisition of social skills. We argued that, so far, DRL works on social intelligence focused on simple social interactions (e.g. rigid question-answering, text-based worlds). As a first step towards a longer-term research endeavor, we presented *SocialAI* (1.0), a set of environments to study – but more importantly to *foster* (Nisioti & Moulin-Frier, 2020) – the acquisition of a broad set of social skills for DRL agents, using multiple grid-world environments featuring other (scripted) social agents. We then studied the limits of a current SOTA DRL agent when tested on *SocialAI* and discussed important next steps towards proficient social agents.

8.2 Perspectives for Future Work

Through the analysis of a (non-exhaustive) set of interesting recent works related to the present research, the objective of this section is to identify promising directions for follow-up works.

Unsupervised Environment Design

Instead of sampling environments from a procedural generation system, Dennis et al. (2020) propose to directly train a teacher policy to construct environments in multiple steps. They showcase this approach using discrete 2D maze environments: their teacher policy (i.e. a DNN) generates mazes by selecting where to place the agent, the goal, and a fixed number of obstacles. They frame this problem as *Unsupervised Environment Design* (UED), and propose the PAIRED algorithm as a method to organize environment generation in such settings. In PAIRED, a teacher algorithm, called the *adversary*, generates environments for the main DRL student, called the *protagonist*, and another DRL student, called the *antagonist*. The objective of the adversary is to maximize regret, defined as the performance difference between the protagonist and the antagonist. This leads to the adaptive generation of environments that are both solvable and challenging for the protagonist.

The PAIRED algorithm can be seen as an interesting way to bring the strength of adversarial self-play training into single-agent environments. PAIRED could also be applicable to the continuous task space settings we focused on in our work: the adversary would have its action space set to the task space, and will "construct" the environment in a single step. Compared to LP-based approaches implemented as continuous bandit algorithms such as ALP-GMM, PAIRED implies a significant compute overhead, as three DRL policies must be trained, instead of only one. Its gradient-based teacher update procedure can also be a drawback, as it makes it slow to adapt to its student (Jiang et al., 2021a).

Regardless of their proposed ACL approach, the multi-step environment generation procedure presented in their work is interesting in itself. It allows researchers to consider ambitious procedural environments that could not be easily leveraged using one-step PCG-encoding parameter vectors. For instance, one could imagine a bounded Minecraft scene, initially empty, in which the teacher algorithm is a policy which iteratively construct a task by placing various block types and tools (or even NPCs) within the bounded

space. In this example, considering directly the underlying one-step task space – with one parameter dimension for each block position, bounded by the number of possible objects – would be intractable for non-trivial scenes. However, a potentially interesting method to recover a lower-dimensional task space in such settings would be to consider the vector of policy parameters as the task space itself (provided that the policy is not too complex).

Prioritized Level Replay

In Jiang et al. (2021b), authors present Prioritized Level Replay (PLR), an ACL algorithm controlling task selection without assuming access to a parametric task space (as in our work) nor an iterative task-construction pipeline (as in Dennis et al. (2020)). PLR only requires the ability to replay a previously proposed task. PLR is based on scoring the learning potential of each encountered task (called levels) based on the TD-error of its student. More precisely, PLR records for each task the average of the absolute TD-errors obtained throughout the episode, which is a measure of the discrepancy between estimated and obtained returns. For each new episode, it either queries the PCG system for a new random task or uses its task replay buffer to sample a previously seen task to train its student on. This algorithm can be understood as a forward and online version of the Prioritized Experience Replay (Schaul et al., 2016) algorithm, which also leverages TD-errors as learning potential estimates but uses it to bias offline replay buffer sampling. In other words, given the terminology we presented in our survey of ACL (section 2.3.2), PLR leverages a TD-error based surrogate objective for data collection ACL, while PER uses it for data exploitation ACL. In Jiang et al. (2021a), authors contextualize their method as a form of unsupervised environment design, allowing them to provide theoretical convergence guaranties, and inspiring them to propose a revised version of PLR, named PLR$^{\perp}$. They then present experiments showing that PLR$^{\perp}$ significantly outperforms PLR (and PAIRED). Compared to PLR, PLR$^{\perp}$ uses a different scoring function and, interestingly, it no longer uses episodes in randomly proposed tasks to update the DRL policy: it trains less but performs better.

Compared to ALP-GMM, PLR and PLR$^{\perp}$ makes a more restrictive assumption on the underlying student: it must be composed of a value-function estimator, and this function must be accessible to the teacher. ALP-GMM is student-agnostic, it does not require to access nor observe its internal state: students are black-boxes. An interesting avenue would be to compare and to combine ALP-GMM and PLR$^{\perp}$. Additionally, the performance advantages of PLR$^{\perp}$ over PLR suggests an algorithmic update to ALP-GMM. More generally, it also applies to any ACL method using residual random task sampling, i.e. methods combining their task curriculum with a small amount of random task sampling for exploration purposes. Maybe such approaches should not use training data on random tasks to update the policy of their student. In addition to theoretical motivations from Jiang et al. (2021a), another point of view is to consider that these gradient updates from random tasks might have a cluttering effect in the optimization process of the student. This perspective is aligned with the hypothesis and observations made while applying AGAIN to a single agent trained twice (section 6.5.3). In rich task spaces, random tasks are often trivial or unfeasible for a given student, leading to gradient updates which might essentially be noise.

Architectural biases promoting generalization

ACL methods controlling the selection of learning experiences – be it goals, environments or offline transitions – can be seen as mechanisms helping artificial agents to learn policies able to handle a diversity of situations. Arguably, this is an important step towards learning policies able to generalize to never-seen-before problems (section 2.1.3). Another approach to promote generalization is to focus on the student's learning system, i.e. to study and design architectural biases promoting generalization. Recent examples include the design of auxiliary losses, e.g. contrastive losses to promote time consistency in state representations (Mazoure et al., 2020; Stooke et al., 2021). Contrastive loss approaches have also been used as a way to train agents able to build object-centered representation of their world (Kipf et al., 2020). In Tang & Ha (2021), authors present a DRL architecture with permutation-invariant properties, enabling to be robust to strong input perturbations.

In Dohare et al. (2021), authors thoroughly expose a well-known yet poorly understood issue among machine learning practitioners: neural network architectures trained with backpropagation (LeCun et al., 1989) lose their learning plasticity over training time. Authors observe that this issue is particularly problematic when learning to approximate target functions when confronted with non-stationary inputs, as in continual learning and reinforcement learning scenarios (a shared concern in the literature, e.g. Igl et al. (2021)). They note that the backpropagation algorithm relies heavily on the network's weight initialization to perform well. Multiple works have studied how to properly initialize neural networks (Glorot & Bengio, 2010; Sutskever et al., 2013; He et al., 2015). Careful weight initialization allows to start learning in an ideal state where networks weights are diverse and unsaturated. The problem is that, by definition, network *initialization* is done only once, leading to a saturation and loss of weight diversity throughout training, hindering further learning. To alleviate this, authors propose the Continual BackPropagation algorithm (CBP), which periodically resets neural network units based on a measure of their contribution to the network's outputs (useless units are removed). Authors show that their approach allows to maintain an agent's ability to continually learn to adjust its policy when faced with changing input distributions in both supervised learning and RL settings. This mechanism can be seen as an interesting and generally-applicable form of architectural bias promoting the learning capacities of open-ended agents.

The interplay of ACL mechanisms and architectural biases promoting learning and generalization is an interesting future avenue for research. For instance, it would be interesting to study whether the aforementioned CBP algorithm could be used as a way to strengthen the learning capacities of ACL-trained DRL students when learning to master large spaces of tasks, i.e. learning to master diverse and evolving input distributions. CBP could also be used as a way to mitigate the detrimental impact of training on random tasks, by preventing noisy updates to gradually saturate the network optimization process (see section 6.7).

ACL for large language models

Although we argued that the training of *embodied* and *interactive* agents is of uttermost importance towards the long-term goal of building artificial peers able to assist us in the

real world, this perspective is only one facet of the wider machine learning field. For instance, within the NLP community, a large group of researchers is interested in building disembodied systems able to understand and generate language. Application areas for such agents include a diversity of language processing tasks, such as text translation or summarization, question answering, sentiment analysis, or chatbots. Recently, impressive performance leaps on such tasks have been obtained through the use of large autoregressive and attention-based DNN architectures, e.g. transformer networks (Vaswani et al., 2017). Multiple works showed that such *Language Models* (LMs) could be pre-trained for text prediction on massive unlabeled text corpora crawled from the internet and then re-used for zero or few-shot learning on various downstream NLP tasks (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; Qiu et al., 2020).

Within this domain, a typical trend has been to scale up the size – i.e. the number of parameters – of such language models, as it was shown to steadily correlate with performance improvements on downstream tasks, e.g. as in Brown et al. (2020). To avoid such large LMs to overfit to their training data, ever bigger pre-training datasets have been assembled. This increase in model capacity and dataset size is staggering: from 340 millions parameters LM and 16 GB dataset size in Devlin et al. (2019) to 1570 *billions* parameters LM and 745 GB dataset sizes in Fedus et al. (2021). Training such gigantic models requires so much compute that it raises ethical questions concerning the environmental costs entailed by experimental campains (Bender et al., 2021).

Interestingly, to train large LMs in a robust and sample efficient (i.e. energy-efficient) manner, a few works started to use curriculum learning approaches in their experimental pipelines. In Press et al. (2021), authors propose a two-staged curriculum, where their LM is first trained to do word-prediction on short input sequences, before training on larger training sequences. Reducing input sequence length simplifies the prediction task as it reduces the size of the textual context on which to perform attention-based computations. In Li et al. (2021) authors also propose to control input sequence length, this time by gradually increasing the input sequence up to a maximal value reached after a fifth of the training time. A similar curriculum schedule is also used in Brown et al. (2020) to control the batch size.

So far, curriculum learning methods tested on LMs are fixed pacing heuristics, which have only been tried to control batch sizes or input sequence length. In an ongoing follow-up project, we are starting to investigate whether more elaborated forms of curriculum generation could be beneficial in such setups. More precisely, we are currently testing whether ALP-GMM could be used to control the selection of learning batches for language models. Each "task" sampled by ALP-GMM corresponds to a group of similar text blocks from the dataset. Similarity between text blocks is measured by mapping each block into a task space (or rather a block space). A coherent task space can be created by pre-processing a vector of characteristics for each block of a given text dataset, e.g. average word frequency/length, sentence length. To compute a performance measure for a given task presented to the LM, which is necessary to estimate learning progress, one can use the inverse of the LM's loss on the task (i.e. on the batch of similar blocks). Just as in RL problems, using ACL mechanisms in such NLP settings might allow practitioners to avoid wasting training time on irrelevant data points (e.g. corrupted text blocks with random letters) and properly allocate compute resources to focus on promising task subspaces.

On the importance of imitation

The present research focused exclusively on reinforcement learning, i.e. learning problems in which an artificial agent improves its policy through interaction within an environment, guided by (intrinsic or extrinsic) reward collection. We showed that when dealing with complex learning situations (e.g. learning a continuous space of related tasks), ACL can facilitate policy learning through the adaptive shaping of learning experiences. Ultimately, we hope that such learning mechanisms will get us closer to human-like artificial agents. However, as we discussed in chapter 7, humans are not merely solitary reinforcement learning agents: a large part – often regarded as central for many psychologists (Vygotsky & Cole, 1978; Tomasello, 2019) – of our knowledge and cognitive abilities is built through social interactions. And a core social affordance is *imitation*.

Imitation Learning (IL) has long been used in the robotics and RL literature (Schaal, 1999; Argall et al., 2009), and usually consist in exploiting a set of expert demonstrations, i.e. recordings of expert interaction sequences containing both the sequence of states s visited by the expert and the actions a that were taken in each of them. Given such demonstration, typical families of methods to learn a policy are *behavioral cloning* (Bain & Sammut, 1995) (learning to match the expert policy through supervised learning) and *inverse reinforcement learning* (Russell, 1998; Ng et al., 2000) (approximating the expert's reward function and using it to do RL).

Interestingly, a growing number of works study how to combine both RL-based learning with imitation learning. In Hester et al. (2018), authors present a modified DQN agent able to alternate between reinforcement learning and learning from demonstrations (they add persistent expert transitions in DQN's replay buffer). They show that the combination of both outperforms pure imitation and is vastly more sample efficient than DQN without demonstrations when tested on Atari games. In robotic settings, Nair et al. (2018) also propose to combine learning from demonstrations (collected from teleoperation) for complex manipulation tasks using DDPG agents. Behavior cloning is also often used in population based works as a way to bootstrap the learning of new generations of agents, e.g. by learning to imitate previous elite policies (Vinyals et al., 2019; Team et al., 2021).

ACL for imitation – We believe this research direction gathered around coupling RL and IL is promising, and could also be combined with ACL mechanisms. Similar ideas have already been explored in the developmental robotics community. For instance, Nguyen & Oudeyer (2012) proposed a population-based system able to autonomously switch between intrinsically motivated goal exploration and training on demonstration data in a 2D robotic arm simulation featuring a ball. Authors were able to show that combining both autonomous learning with IL led to performance improvements in downstream test goals (end-effector-positioning and ball-throwing) w.r.t. to only using IL or autonomous learning. More precisely, they used a learning progress-based intrinsic motivation signal to 1) select on which goal to train on, 2) select which data collection strategy to use among autonomous exploration or expert demonstrations, and 3) if using demonstrations, LP is also used to select from which expert it is most efficient to query demonstrations given the considered goal. While in their work they considered simple controllers, an interesting avenue would be to study how it can transfer to DRL settings. A few works already

proposed ACL methods leveraging expert demonstrations in DRL settings (Florensa et al., 2017; Salimans & Chen, 2018). However, they only focused on a specific scenario: given a single demonstration and a resetable environment, ACL can control on which state of the demonstration should an agent start interacting (which allows to bootstrap exploration). Beyond this setting, ACL could control the adaptive selection of expert demonstrations (e.g. using PLR to learn from specific demonstrations), or even decide when to query a compliant expert for specific demonstrations.

Towards seamless social imitation – Classical IL often requires accessing to the actions a of the expert demonstrations. From a human perspective, this is unrealistic: it requires to ask for "muscle commands". As such, a growing number of works focuses on approaches able to perform imitation from expert *observation* sequences (Torabi et al., 2019). Similarly, most IL works consider *first-person* demonstrations, i.e. the agent is re-experiencing the expert's interaction, which assumes an identical environment and embodiment for both parties. Performing IL from third-person observations appears like a more realistic setup, and is being under investigation in the literature (Stadie et al., 2017; Sharma et al., 2019). We believe more work is needed in this direction, especially on studying how architectural biases (e.g. intrinsic motivation mechanisms) could enable agents to *extract* demonstration data from their own interaction in social settings. A first milestone in this direction was recently presented in Ndousse et al. (2021). Authors showed that, through the addition of an auxiliary next-state prediction task, DRL agents learning to perform navigation tasks among expert policies were able to learn to imitate social peers to overcome hard-exploration scenarios. One of the objectives of the *SocialAI* project, whose preliminary version is presented in chapter 7, is to provide rich social scenarios in which to study such learning systems.

Towards open-ended artificial agents

To some extent, the computational studies we present in this manuscript could be understood as being *open-ended*, in the sense that we consider continuous task spaces built through procedural generation, i.e. our DRL students must learn an infinity of tasks. However, these infinite task spaces are *bounded*: from this perspective, their open-ended nature is questionable. This is even more striking when considering human's development, which consist in a never-ending accumulation of countless behaviors and knowledge of various natures. How to approach this amazing learning ability ?

To escape from pre-defined training boundaries, an interesting avenue is to autonomously learn goal/task/skill representations from vision or state-based observations, and leverage them to organize open-ended exploration and learning (Laversanne-Finot et al., 2018; Eysenbach et al., 2019; Pong et al., 2020; Kovač et al., 2020; Choi et al., 2021). For instance, in Laversanne-Finot et al. (2018), authors presented an intrinsically motivated agent able to learn a disentangled world representation, i.e. mapping raw pixel images to a low-dimensional feature space with each dimension focused on specific scene entities. They performed experiments using a 2D robotic simulation of an arm surrounded by 2 balls (one being graspable). Authors showed that, by using such a feature space as a task space, their agent (a LP-based IMGEP, see section 2.2.2) could learn to position its arm and the ball at various locations, at a similar learning speed than

if using the ground truth task space. Could we scale up these pioneering results to more complex environments, e.g. 3D scenes with variable number of visible objects, requiring to learn disentangled representations online ? Although promising, so far, progress towards making such autonomous agents is slow, as current techniques to learn disentangled world representations provide mitigated results (Locatello et al., 2020).

Another interesting direction is to leverage *language* to frame open-ended learning. As we discussed in chapter 7, studying how to train embodied agents able to use and understand language efficiently is a desirable and ongoing research direction (Luketina et al., 2019). In Colas et al. (2020a), authors showed that language can be used as an internal tool to *imagine* and pursue out-of-distribution goals. In essence, their work aims to model *exploratory play*, a typical child behavior which consist in using egocentric speech to self-generate unique goals through the generative properties of language (Piaget, 1952; Vygotsky & Cole, 1978). To model this phenomenon, their approach relies on the compositional nature of language, and assumes a (disembodied) social partner providing descriptive feedback. They performed experiments on a 2D multi-object scene with a DRL agent equipped with such an intrinsically motivated system. Throughout training, their agent was able to 1) explore some of the feasible interactions in its environment (moving and combining objects), 2) learn to map textual descriptive feedbacks to corresponding physical goals, and 3) generate and try to achieve never seen before goals by imagining them through construction grammar rules. In other words, language was used as a *task space*, and social guidance was leveraged to bootstrap its exploration. This work is promising from an open-ended learning perspective: scaling such learners to complex language spaces with proper social guidance (e.g. scheduled with ACL) is an interesting avenue towards open-ended development.

Importantly, the concept of "open-endedness" does not exclusively refer to the never-ending learning abilities of humans. The study and modeling of open-ended systems is a field of research in itself (Stanley et al., 2017; Stanley, 2019). Arguably, the most impressive biological open-ended process is not human learning, it is *evolution*: it created nature itself, including us, in a single run. Understanding and modeling this phenomenon is what motivates the *artificial life* field (Langton, 1995). Natural evolution also motivated a large strand of works building efficient artificial agents using evolutionary algorithms (see section 2.1.3). Recently, multiple works showcased that impressive behaviors could be learned through the combination of evolutionary processes with powerful DRL learning systems, e.g. training multiple generations of learners, with each one bootstrapping its behavior on elite elders (Vinyals et al., 2019; Wang et al., 2019b, 2020b; Team et al., 2021).

8.3 Conclusion

Throughout this manuscript, we mentioned and discussed a large panel of existing works focused on building embodied artificial agents able to efficiently learn robust policies. More precisely, inspired by developmental sciences and developmental robotics, we focused on the design – or evolution of – environments as a lever to catalyze artificial learning. This led to the study of *automatic curriculum learning* algorithms to guide learners in complex task spaces, but also to the design of *social environments* to provide diverse learning scenarios fostering the acquisition of social skills.

Why are we still waiting for our mechanical alter ego? – We started our introductory chapter by wondering which obstacles prevented humanity from building its long-desired artificial social peer. We hope that after reading this manuscript, this question is clearer. Human cognitive development is a complex multidimensional process. Embodied. Self-motivated. Rooted in social interactions. Open-ended. Although not fully understood, the tremendous technological and scientific explosion of the last century allowed ML researchers to study and model various aspects of our cognition. The long term picture remains blurry, but these first steps suggest many interesting future avenues. The present research attempted to further motivate that a core block to train tomorrow's autonomous artificial agents is to adopt a developmental perspective. There is no shortcut: learning must be done *step by step*.

Appendices

Appendix A

ALP-GMM and LP-based Teacher Algorithms

Contents

A.1 Experiments on an n-dimensional Toy Environment

To emancipate our study of teacher algorithms from DRL students, we propose to use a toy testbed. The objective of this environment is to simulate the learning of a student within a n-dimensional task-encoding parameter space $\mathcal{T} = [0, 1]^n$, a.k.a task space. This fake task space is uniformly divided in hypercubes. Each task $\tau \in \mathcal{T}$ sampled by the teacher is directly mapped to an episodic reward r_τ based on a) sampling history and b) status of the hypercube from which the task originates (is it "locked" or "unlocked"). Three rules enforce reward collection in $\mathcal{T}$:

- Sampling a task in an "unlocked" hypercube results in a positive reward ranging from 1 to 100 depending on the amount of already sampled tasks in the hypercube: if 10 tasks were sampled in it, the next one will yield a reward of 11. Sampling a task located in a "locked" hypercube does not yield any reward.
- At first, all hypercubes are "locked" except for one, located in a corner.
- Sampling 75 tasks in an unlocked hypercube unlocks its neighboring hypercubes.

These rules model a policy learning student, for which a teacher algorithm must detect an initially feasible task subspace (the initially unlocked hypercube), and then focus and expand task sampling from here (to unlock adjacent hypercubes).

Experimental Results

Results are displayed in Figure A.1. We use the median percentage of unlocked hypercubes as a performance metric. A first experiment was performed on a 2D toy space with 10 hypercubes per dimensions. In this experiment, one can see that all LP-based approaches outperform Random by a significant margin. COVAR-GMM is the highest performing algorithm. This first toy-space will be used as a point of reference for our following analysis, for which all conditions were tested on a panel of toy spaces with varying number of meaningful dimensions (first row of Figure A.1), irrelevant dimensions (second row) and number of hypercubes (third row).

By looking at the first row of Figure A.1, one can see that increasing the dimension size – i.e. expanding the task space – seems to be most detrimental for RIAC than for GMM-based approaches. RIAC, which was between ALP-GMM and COVAR-GMM in terms of median performance in our reference experiment is now clearly under-performing them on all 3 toy spaces. In the 3D and 4D cases RIAC is even outperformed by the Random condition after 70k episodes and 290k episodes, respectively. For the 6D toy space, RIAC consistently outperforms Random, reaching a median final performance of 80% after 1M episodes. In this 6D toy space ALP-GMM and COVAR-GMM both reach 100% of median performance after 1M episodes. COVAR-GMM is the highest performing condition in each toy-space, closely followed by ALP-GMM.

The second row of Figure A.1 shows how performances of our approaches vary when adding irrelevant dimensions to the 2D toy-space. To better grasp the properties of these additional dimensions, one can see that Random is not affected by them. With 10,20 and 50 additional useless dimensions, RIAC is consistently inferior to GMM-based conditions in terms of median performance. RIAC median performance is only above Random during the first 55k episodes. ALP-GMM is the highest performing algorithm throughout training for toy spaces with 20 and 50 irrelevant dimensions, closely followed by COVAR-GMM. In the toy space with 10 irrelevant dimensions, ALP-GMM outperforms COVAR-GMM in the first 40k episodes but end up reaching a 100% median performance after 52k episodes against only 44k episodes for COVAR-GMM.

The last row shows how performance changes according to the number of hypercubes. Given our toy space rules, increasing the number of hypercubes reduces the initial area where reward is obtainable in the task space, and therefore allows us to study the sensitivity of our approaches to detect learnable subspaces. Random struggles in all 3 toy-spaces compared to other conditions and to its performances on the reference experiment with 10 hypercubes per dimensions. COVAR-GMM and RIAC are the best performing conditions for toy-spaces with 20 hypercubes per dimensions. However, when increasing to 50 and 100 hypercubes per dimensions, COVAR-GMM remains the best performing condition but RIAC is now under-performing compared to ALP-GMM.

Overall, these experiments showed that GMM-based approaches scaled better than RIAC on task spaces with large number of relevant or irrelevant dimensions, and large number of (initially) unfeasible task spaces. Among these GMM-based approaches, contrary to experiments with DRL students on BipedalWalker environments, COVAR-GMM proved to be better than ALP-GMM for these toy spaces.

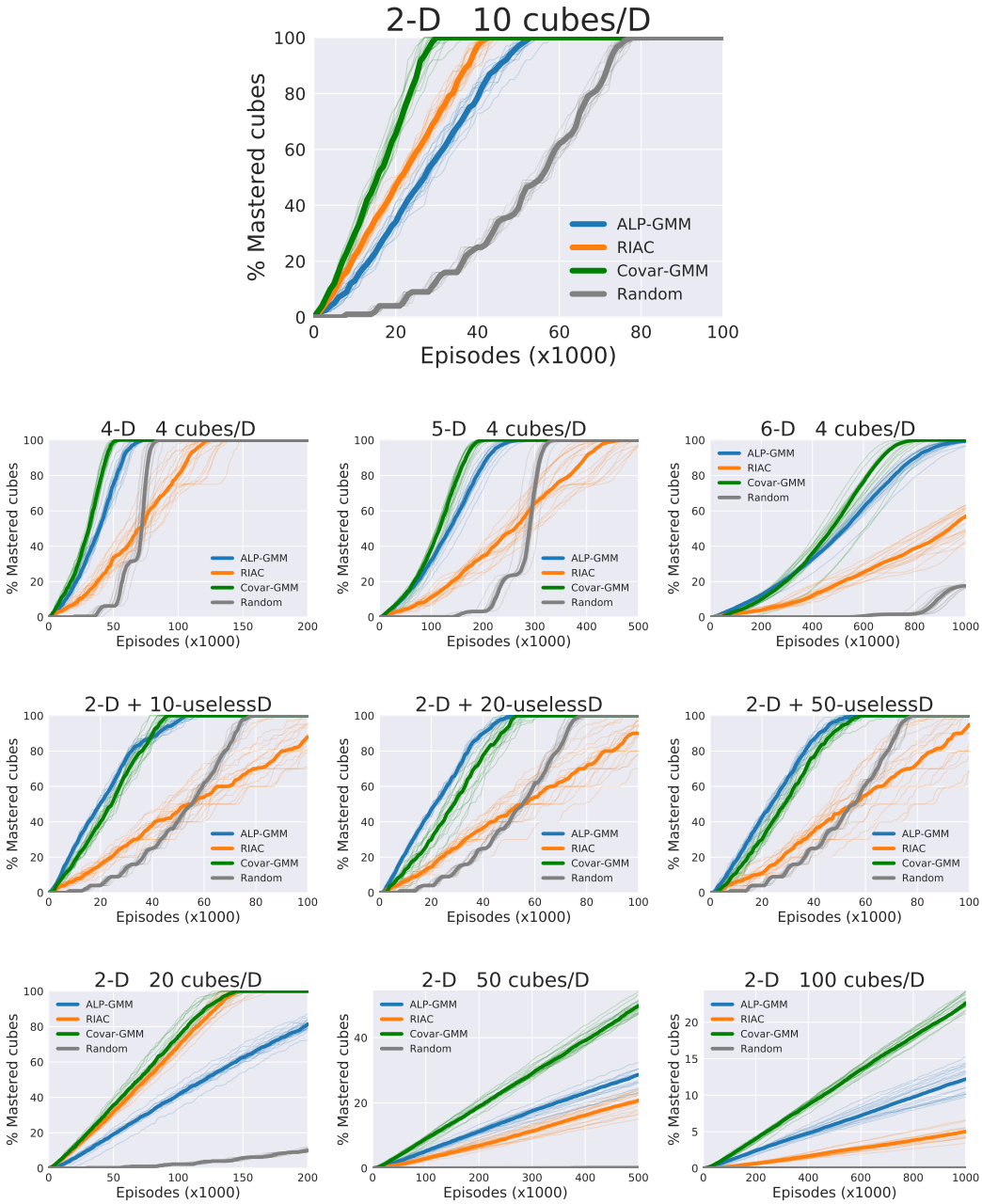


Figure A.1: **Evolution of performance on n-dimensional toy-spaces.** The impact of 3 aspects of the task space are tested: growing number of meaningful dimensions (top row), growing number of irrelevant dimensions (middle row) and increasing number of hypercubes (bottom row). The median performance (percentage of unlocked hypercubes) is plotted with shaded curves representing the performance of each run. 20 repeats were performed for each condition (for each toy-space).

A.2 Implementation details

A.2.1 Soft-Actor Critic

All of our experiments were performed with OpenAI's implementation[1] of SAC as our DRL student. We used the same 2-layered (400, 300) network architecture with ReLU for the Q, V and policy networks. The policy network's output uses tanh activations. The entropy coefficient and learning progress were respectively set to 0.005 and 0.001. Gradient steps are performed every 10 environment steps by selecting 1000 samples from a replay buffer with a fixed sized of 2 millions.

A.2.2 LP-based Teachers

RIAC – Proposed in Baranes & Oudeyer (2009), Robust Intelligent Adaptive Curiosity is based on the recursive splitting of the task space in hyperboxes, called regions. One region is split in two whenever a pre-defined number max_s of sampled tasks originate from the region. The split value is chosen such that there is maximal LP difference between the two regions. Beyond being provided a bounded task space, RIAC does not require expert knowledge. To avoid a known tendency of RIAC to oversplit the task space (Florensa et al., 2018), we added a few modifications to the original architecture. The workflow of our modified RIAC could be summarized as follows (hyperparameters settings are given in parentheses):

1. When collecting a new task-reward pair, it is added to its respective region. If this region reaches its maximal capacity max_s (= 200), a split attempt is performed.

2. When trying to split a parent region p into two children regions c_1 and c_2, n (= 50) candidate splits on random dimensions and thresholds are generated. If c_1 or c_2 have less than min_s (= 20) individuals, the split is rejected. Likewise, to avoid having extremely small regions, a minimum size min_d is enforced for each region's dimensions (set to 1/6 of the initial range of each dimensions of the task space). The split with the highest score, defined as $card(c_1) \cdot card(c_2) \cdot |alp(c_1) - alp(c_2)|$, is kept. If no valid split was found, the region flushes its oldest points (the oldest quarter of pairs sampled in the region are removed).

3. At sampling time, several strategies are combined:

   - 20%: a random task is chosen in the entire space.
   - 70%: a region is selected proportionally to its ALP and a random task is sampled within the region.
   - 10%: a region is selected proportionally to its ALP and the worst task with lowest associated episodic reward is slightly mutated (by adding a Gaussian noise $\mathcal{N}(p, 0.1)$).

[1] https://github.com/openai/spinningup

We send the reader back to the original papers of RIAC (Baranes & Oudeyer, 2009, 2013) for detailed motivations and pseudocode descriptions.

COVAR-GMM – Originating from the developmental robotics field (Moulin-Frier et al., 2014), this approach inspired the design of ALP-GMM. In COVAR-GMM, instead of fitting a GMM on the task space concatenated with ALP as in ALP-GMM, they concatenate each tasks with its associated episodic return and time (relative to the current window of considered tasks). New tasks are then chosen by sampling on a Gaussian selected proportionally to its positive covariance between time and episodic reward, which emulates positive LP. Contrary to ALP-GMM, they ignore negative learning progress and do not have a way to detect long term LP (i.e LP is only measured for the currently fitted datapoints). Although not initially present in COVAR-GMM, we compute the number of Gaussians online as in ALP-GMM to compare the two approaches solely on their LP measure. Likewise, COVAR-GMM is given the same hyperparameters as ALP-GMM (see section 4.4).

Oracle – Oracle has been manually and iteratively crafted based on knowledge acquired over multiple runs of the algorithm. It uses a step size $\sigma_W = \frac{1}{30}R$, with R a vector containing the maximal distance for each dimension of the task space. Before each new episode, the window ($W_{size} = \frac{1}{6}R$) is slid toward more complex task distributions by σ_W only if the average episodic reward of the last 50 proposed tasks is above $r_{thr} = 230$. See Algorithm 2 for pseudocode.

Figure A.2 provides a visualization of the evolution of Oracle's task sampling for a typical run in Stump Tracks. One can see that the final position of the task sampling window (corresponding to a subspace that cannot be mastered by the student) is reached after 10500 episodes (c) and remains the same up to the end of the run, totaling 15000 episodes. This end-of-training focus on a specific part of the task space is the cause of the forgetting issues of Oracle (see section 4.5.1).

Algorithm 2 Oracle

Require: Student s, bounded task space $\mathcal{T}$, initial sampling window position W_{pos}, window step size σ_W, memory size m_{size}, reward threshold r_{thr}, window-size W_{size}.

Set sampling window $W \subset \mathcal{T}$ to W_{pos}
loop
    Sample random task-encoding parameters $\tau \in W$
    Generate environment with τ send it to s, observe episodic reward r_p
    If the mean competence over the last m_{size} episodes exceeds r_{thr} **then**
        $w_{pos} = w_{pos} + \sigma_w$
Return s

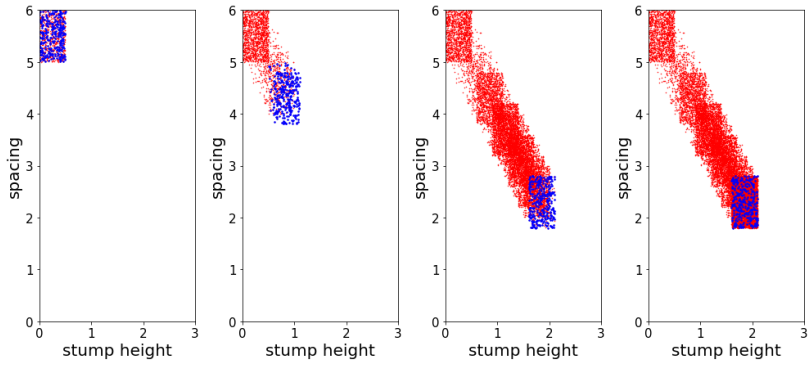
A.2.3 Parameterized BipedalWalker Environments

In BipedalWalker environments, observations vectors provided to walkers are composed of 10 lidar sensors (providing distance measures), the hull angle and velocities (linear and angular), the angle and speed of each hip and knee joints along with a binary vector which

informs whether each leg is touching the ground or not. This sums up to 24-dimensions for our two bipedal walkers and 34 for the quadrupedal version. To account for its increased weight and additional legs, we increased the maximal torque usage and reduced the torque penalty for quadrupedal agents.

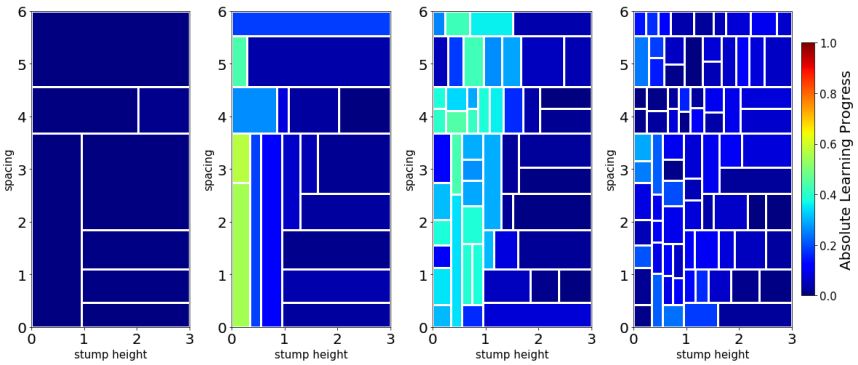
Task space bounds – In Stump Tracks, the range of the mean stump height μ_h is set to $[0, 3]$, while the spacing Δ_s range lies in $[0, 6]$. In Hexagon Tracks, the range of the 12 dimensions of the space are set to $[0, 4]$.

A.3 Additional Visualizations for Stump Tracks Experiments



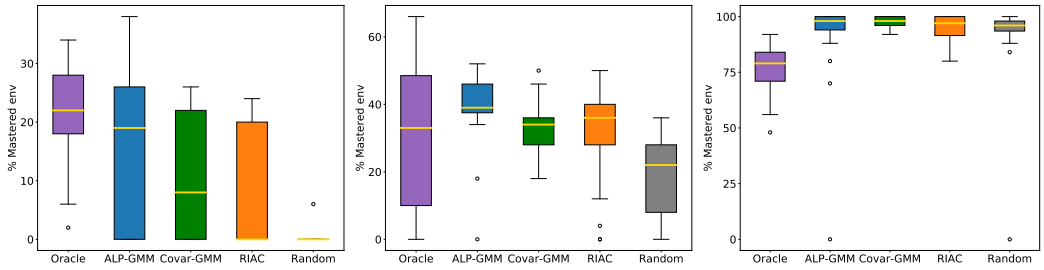
(a) 1000 episodes (b) 2000 episodes (c) 10500 episodes (d) 15000 episodes

Figure A.2: **Evolution of Oracle task sampling for a default bipedal walker on Stump Tracks.** Blue dots represent the last 300 sampled tasks, red dots represent all other previously sampled tasks. At first (a), Oracle starts by sampling tasks in the easiest subspace (i.e large stump spacing and low stump height). After 2000 episodes (b), Oracle slid its sampling window towards stump tracks whose stump height lies between 0.6 and 1.1 and a spacing between 3.7 and 4.7. After 10500 episodes (c) this Oracle run reached a challenging subspace that his student will not be able to master. By 15000 episodes, The sampling window did not move as the mean reward threshold was never crossed.



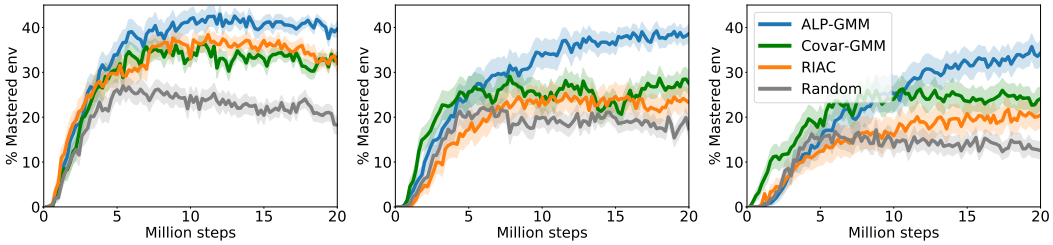
(a) 500 episodes (b) 1500 episodes (c) 15000 episodes (d) 20000 episodes

Figure A.3: **Evolution of RIAC task sampling for a default bipedal walker on Stump Tracks.** At first (a), RIAC do not find any learning progress signal in the space, resulting in random splits. After 1500 episodes, RIAC focuses its sampling on the leftmost part of the space, corresponding to low stump heights, for which the SAC student manages to progress. After 15k episodes (c), RIAC spreaded its sampling to task subspaces corresponding to track distributions with stump heights up to 1.5, with the highest stumps paired with high spacing. By the end of the training (d) the student converged to a final skill level, and thus LP is no longer detected by RIAC, except for simple track distributions in the leftmost part of the space in which occasional forgetting of walking gates leads to ALP signal.



(a) Short agents (b) Default agents (c) Quadrupedal agents

Figure A.4: **Box plot of the final performance of each condition on Stump Tracks after 20M steps.** Gold lines are medians, surrounded by a box showing the first and third quartile, which are then followed by whiskers extending to the last datapoint or 1.5 times the inter-quartile range. Beyond the whiskers are outlier datapoints. **(a)**: For short agents, Random always end-up mastering 0% of the track distributions of the test set, except for a single run that is able to master 3 track distributions (6%). LP-based teachers obtained superior performances than Random while still failing to reach non-zero performances by the end of training in 13/32 runs for ALP-GMM, 15/32 for COVAR-GMM and 19/32 for RIAC.
(b): For default walkers, LP-based approaches have less variance than Oracle (visible by the difference in inter-quartile range) whose window-sliding strategy led to catastrophic forgetting occurring in a majority of runs. Random remains the least performing algorithm.
(c): For quadrupedal walkers, Oracle performs significantly worse than any other condition ($p < 10^{-5}$). Additional investigations on the data revealed that, by sliding its sampling window towards track distributions with higher stump heights and lower stump spacing, Oracle's runs mostly failed to master track distributions that were both hard and distant from its sampling window within the task space: that is, tracks with both high stump heights (> 2.5) and high spacing (> 3.0).



(a) max stump height of 3 (b) max stump height of 4 (c) max stump height of 5

Figure A.5: **Evolution of mean performance of Teacher-Student approaches when increasing the amount of unfeasible tracks in Stump Tracks with default bipedal walkers.** 32 seeded runs where performed for each condition. The mean performance is plotted with shaded areas representing the standard error of the mean. ALP-GMM is the most robust LP-based teacher and maintains a statistically significant performance advantage over all other conditions in all 3 settings. Random performances are most impacted when increasing the number of unfeasible tracks. ALP-GMM is more robust than RIAC when going from a maximal stump height of 3 to 4 and 3 to 5. Note that for all 3 experiments, for comparison purposes, the same test set was used and contained only track distributions with a maximal stump height of 3.

A.4 Additional Visualization for Hexagon Tracks Experiments

To better understand the properties of all of the tested conditions in Hexagon Tracks, we analyzed the distributions of the percentage of mastered environments of the test set after training for 80 Millions (environment) steps. Using Figure A.6, one can see that ALP-GMM both has the highest median performance and narrowest distribution. Out of the 32 repeats, only Oracle and ALP-GMM always end-up with positive final performance scores whereas COVAR-GMM, RIAC and Random end-up with 0% performance in 8/32, 5/32 and 16/25 runs, respectively. Interestingly, in all repeats of any condition, the student manages to master part of the test set at some point (i.e non-zero performance), meaning that runs that end-up with 0% final test performance actually experienced catastrophic forgetting. This showcase the ability of ALP-GMM to avoid this forgetting issue through efficient tracking of its student's absolute learning progress.

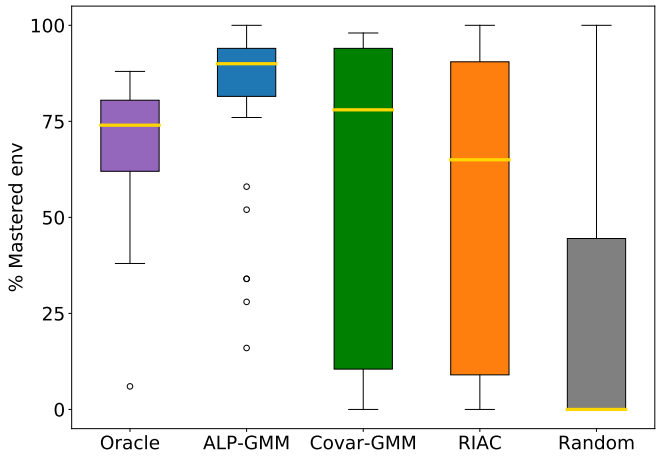


Figure A.6: **Box plot of the final performance of each condition run on Hexagon Tracks after 80M steps.** Gold lines are medians, surrounded by a box showing the first and third quartile, which are then followed by whiskers extending to the last datapoint or 1.5 times the interquartile range. Beyond the whiskers are outlier datapoints.

Appendix B

TeachMyAgent

Contents

B.1 Details on ACL Baselines

In this section, we give details about our implementations of ACL methods, as well as their hyperparameters tuning.

B.1.1 Implementation Details

Random – We use as baseline a random teacher, which samples tasks using a uniform distribution over the task space.

ADR – OpenAI et al. (2019) introduced *Automatic Domain Randomization* (ADR), an ACL method relying on the idea of *domain randomization* (Tobin et al., 2017; Peng et al., 2018b). Instead of sampling tasks over the whole task space, ADR starts from a distribution centered on a single easy task for a given student and progressively grows the distribution according to the student's performance. Using this mechanism, it increases the difficulty of the tasks proposed to the student while still sampling in previously seen regions (to prevent potential forgetting).

This sampling distribution $\mathcal{T}_\phi$ is parameterized by $\phi \in \mathbb{R}^{2d}$ (with d the number of dimensions of the task space). For each dimension, a lower and upper boundary are set $\phi = \{\phi_i^L, \phi_i^H\}_{i=1}^d$ allowing to sample uniformly on each dimension using these boundaries and obtain a task τ:

$$\mathcal{T}_\phi(\tau) = \prod_{i=1}^{d} U(\phi_i^L, \phi_i^H)$$

At the beginning, ϕ is centered on a single example (i.e. $\phi_i^L = \phi_i^H \; \forall i$). Then, at each episode, 1) ADR starts by sampling a new task $\tau \sim \mathcal{T}_\phi$. Following this, 2) ADR chooses with a probability p_b whether to modify τ in order to explore the task space or not. It thus samples a value ϵ uniformly in $[0; 1]$ and checks whether $\epsilon < p_b$. If this is not the case, ADR simply sends τ to the environment.

Otherwise, 3) ADR selects uniformly a dimension of the task space. Let j refer to this dimension. Following this, 4) one of the two boundaries ϕ_j^L or ϕ_j^H is selected (50% chances for each boundary). Finally, 5) ADR replaces the j-th value of τ by the selected boundary and sends τ to the environment.

Moreover, ADR keeps a buffer D_i^L and D_i^H for each dimension i in the task space. Every time ϵ is greater than p_b and a value of τ is replaced by one of the selected boundary, ADR stores the episodic reward obtained at the end of the episode in the buffer associated to the selected boundary (e.g. the episodic reward is stored in D_k^L if the k-th value of lambda was replaced by ϕ_k^L).

Every time one of the buffers' size reaches m, the average $\overline{p}$ of episodic reward stored is calculated. Then, $\overline{p}$ is compared to two thresholds t_L and t_H (being hyperparameters of ADR) in order to know whether the boundary associated to the buffer must be reduced or increased.

As an example, let's say that D_k^L's size reached m, meaning that ϕ_k^L is the associated dimension (i.e. a τ sampled got its k-th value replaced by ϕ_k^L m times). Its average episodic reward $\overline{p}$ is calculated. It is first compared to t_L and, if $\overline{p} < t_l$, ϕ_k^L is increased by Δ (as ϕ_k^L is a lower boundary, this means that the task space is reduced). Similarly, if $\overline{p} > t_l$, ϕ_k^L is decreased by Δ (expanding the task space).

If instead of D_k^L we take D_k^H, our task space has to be expanded or reduced in the same way: if $\overline{p} < t_L$ then ϕ_k^H is reduced by Δ (as it is now an upper boundary of the task space) and if $\overline{p} > t_H$ then ϕ_k^H is increased by Δ. Finally, note that whenever one buffer's size reaches m, it is then emptied.

As no implementation was provided by the authors, we propose here an implementation being as close as possible to the algorithms given in OpenAI et al. (2019).

RIAC – Proposed in Baranes & Oudeyer (2009), Robust Intelligent Adaptive Curiosity is based on the recursive splitting of the task space in hyperboxes, called regions. Regions are assigned LP-values and are used for LP-proportional task sampling. We reuse the implementation and hyperparameters from chapter 4, see appendix A.2 for details.

COVAR-GMM – COVAR-GMM was proposed in Moulin-Frier et al. (2014) and adapted for DRL settings in chapter 4. As for RIAC, it does not require expert knowledge and is based on learning progress. The core idea of COVAR-GMM is to fit a Gaussian Mixture Model (of maximum size max_k) every n episodes on recently sampled tasks *concatenated with both a time dimension and a competence dimension*. The Gaussian from which to sample a new task is then chosen proportionally to its respective learning progress, defined as the positive correlation between time and competence. Additionally, in order to preserve exploration, COVAR-GMM has a probability p_{rnd} of sampling a task uniformly random instead of using one of its Gaussians. We use the implementation and hyperparameters from chapter 4 which uses Absolute Learning Progress (ALP) instead of LP.

Moreover, as aforementioned in section 5.5, we modified the implementation to make it use expert knowledge (i.e. an initial distribution) when provided. Hence, instead of uniformly sampling tasks over the whole task space during the bootstrap phase at the beginning of training, COVAR-GMM samples tasks from an initial Gaussian distribution of tasks provided by the expert.

ALP-GMM – ALP-GMM is an ACL algorithm inspired from COVAR-GMM, proposed in chapter 4. Instead of relying on time competence correlation, which only allows to compute ALP over a single GMM fit, it computes a per-task ALP from the entire history of sampled tasks using a knn-based approach. Recent tasks are periodically used to fit a GMM on recently sampled tasks *concatenated with their respective ALP value*. The Gaussian from which to sample is then selected based on its mean ALP dimension. ALP-GMM does not require expert knowledge and has the same hyperparameters as COVAR-GMM. We reused the implementation and hyperparameters (except max_k, n and p_{rnd}) from chapter 4, which also features detailed explanations on ALP-GMM.

Additionally, as for COVAR-GMM, we added the possibility to ALP-GMM to bootstrap curriculum generation with an initial Gaussian distribution if the latter is provided, instead of starting ACL by randomly proposing tasks.

GOAL-GAN – Another teacher algorithm we included in this benchmark is called GOAL-GAN, and relies on the idea of sampling goals (i.e. states to reach in the environment) where the agent performs neither too well nor to badly, called *Goals Of Intermediate Difficulty* (GOID). As discussed in section 2.1.1, goals are a particular form of task. We will thus call them tasks instead of goals in the following description. For sampling, Florensa et al. (2018) proposed to use a modified version of a *Generative Adversarial Network* (GAN) (Goodfellow et al., 2014) where the generator network is used to generate tasks for the student (conditioned on a random noise vector), and the discriminator is trained to classify whether these tasks are of "intermediate difficulty". To define such an "intermediate difficulty", GOAL-GAN uses a binary reward signal defining whether the student succeeded in the proposed task. As our environments return scalar rewards, this implies a function interpreter hand-designed by an expert (in our case we set a threshold

on the scalar reward, as explained in appendix B.3). For each task sampled, the teacher proposes it multiple times ($n_{rollouts}$) to the student and then calculates the average of successes obtained (i.e. a value between 0 and 1). Using a lower threshold R_{min} and an upper threshold R_{max}, GOAL-GAN calculates if the average lies in this interval of tasks neither too easy (with an average of successes very high) nor too hard (with an average of successes very low). If this is the case, this task is labelled as 1 for the discriminator (0 otherwise). This new task is then stored in a buffer (except if it already exists in the buffer a task at an euclidean distance smaller than ϵ from our new task). Every time a task has to be sampled, in order to prevent the GAN from forgetting previously seen GOIDs, the algorithm has the probability p_{old} of uniformly sampling from the buffer instead of using the GAN. Finally, the GAN is trained using the tasks previously sampled every n episodes.

Note that, in order to help the GAN to generate tasks in a feasible subspace of the task space at the beginning of training, GOAL-GAN can also pretrain its GAN using trivial tasks. In the original paper, as tasks are states, authors proposed to use the student to interact with the environment for a few steps, and use collected states as achievable tasks. However, in our case, this is not possible. We thus chose to reuse the same trick as the one in Klink et al. (2020), that uses an initial Gaussian distribution to sample tasks and label them as positives (i.e. tasks of intermediate difficulty) in order to pretrain the GAN with them. See appendix B.3 for the way we designed this initial distribution.

We reused and wrapped the version[1] of GOAL-GAN implemented by Klink et al. (2020), which is a slightly modified implementation of the original one made by Florensa et al. (2018). Our generator network takes an input that has the same number of dimensions as our task space, and uses two layers of 256 neurons with ReLU activation (and TanH activation for the last layer). Our discriminator uses two layers of 128 neurons. For ϵ, we used a distance of 10% on each dimension of the task space. As per Florensa et al. (2018), we set R_{min} to 0.25 and R_{max} to 0.75. Finally, as in the implementation made by Klink et al. (2020), we set the amount of noise δ added to each goal sampled by the generator network as a proportion of the size of the task space.

Self-Paced – Proposed by Klink et al. (2020), *Self-Paced Deep Reinforcement Learning* (SPDL) samples tasks from a distribution that progressively moves towards a target distribution. The intuition behind it is similar to the one behind ADR: the idea is to start from an initial task space and progressively shift it towards a target space, while adapting the pace to the agent's performance. However here, all task distributions (initial, current and target) are Gaussian distributions. SPDL thus maintains a current task distribution from which it samples tasks and changes it over training. This distribution shift is seen as an optimization problem using a dual objective maximizing the agent's performance over the current task space, while minimizing the Kullback-Leibler (KL) divergence between the current task distribution and the target task distribution. This forces the task selection function to propose tasks where the agent performs well while progressively going towards the target task space.

Initially designed for non-episodic RL setups, SPDL, unlike all our other teachers,

[1] https://github.com/psclklnk/spdl

receives information at every step of the student in the environment. After an offset of n_{OFFSET} first steps, and then every n_{STEP} steps, the algorithm estimates the expected return for the task sampled $\mathrm{E}_{p(c)}[J(\pi, c)]$ using the value estimator function of the current student (with $p(c)$ the current task distribution, π the current policy of the student, and $J(\pi, c)$ the expected return for the task c with policy π).

With this, SPDL updates its current sampling distribution in order to maximize the following objective w.r.t. the current task distribution $p(c)$:

$$\max_{p(c)} \mathrm{E}_{p(c)}[J(\pi, c)]$$

Additionally, a penalty term is added to this objective function, such that the KL divergence between $p(c)$ and the target distribution $\mu(c)$ is minimized. This penalty term is controlled by an α parameter automatically adjusted. This parameter is first set to 0 for K_α optimization steps and is then adjusted in order to maintain a constant proportion ζ between the KL divergence penalty and the expected reward term (see Klink et al. (2020) for more details on the way α is calculated). This optimization step is made such that the shift of distribution is not bigger than ϵ (i.e. $s.t. D_{\text{KL}}(p(c)||q(c)) \leq \epsilon$ with a shift from $p(c)$ to $q(c)$).

We reused the same implementation made by Klink et al. (2020) and wrapped it to our teacher architecture. However, as shown in section 5.5, using a Gaussian target distribution does not match with our Stump Tracks test set where tasks are uniformly sampled over the whole task space. In order to solve this issue, some adaptations to its architecture could be explored, e.g. we tried using a truncated Gaussian as target distribution to get closer to a uniform distribution (but, so far, we did not observe performance improvements).

For the value estimators, we used the value network of both our PPO and SAC implementations (with the value network sharing its weights with the policy network for PPO). For the calculation of α, we chose to use the average reward, as in the experiments of Klink et al. (2020). We did not use the lower bound restriction on the standard deviation of the task distribution σ_{LB} proposed in Klink et al. (2020) as our target distributions were very large (see appendix B.3).

SETTER-SOLVER – Finally, the last ACL algorithm we implemented here is SETTER-SOLVER (Racaniere et al., 2020). In a very similar way to GOAL-GAN, this method uses two neural networks: a *Judge* (replacing the discriminator) and a *Setter* (replacing the generator) outputting a task given a feasibility scalar in $[0; 1]$. During the training, the *Judge* is trained to output the right feasibility given a task sampled, and is used in the *Setter*'s losses to encourage the latter to sample tasks where the predicted feasibility was close to the real one. The *Setter* is also trained to sample tasks the student has succeeded (i.e. using a binary reward signal as GOAL-GAN) while maximizing an entropy criterion encouraging it to sample diverse tasks.

For the implementation, Racaniere et al. (2020) provided code to help reproducibility that implements both the *Setter* and *Judge*, but did not include neither losses nor optimization functions. Therefore, we provide here our own implementation of the full SETTER-SOLVER algorithm trying to be as close as possible to the paper's details. We

reused the code provided for the two neural networks and modified it to add losses, optimizers, and some modifications to better integrate it to our codebase. We kept the tricks added in the code provided by authors that uses a non-zero uniform function to sample the feasibility and a clipped sigmoid in the *Setter*'s output. Concerning the generator network, we kept the hyperparameters of the paper (i.e. a RNVP (Dinh et al., 2017) with three blocks of three layers) except the size of hidden layers n_{HIDDEN} that we optimized. We also reused the three layers of 64 neurons architecture for the *Judge* as per the paper. Note that we used an Adam optimizer with a learning rate of $3 \cdot 10^{-4}$ for both the *Setter* and the *Judge*, as it was not precised for the *Judge* in Racaniere et al. (2020). We optimized the upper bound δ of the uniformly sampled noise that is added to succeeded tasks in the validity *Setter*'s loss, as well as the update frequency n.

We did not use the conditioned version of the *Setter* or *Judge*. Indeed, first we generate the task before obtaining the first observation in our case as opposed to Racaniere et al. (2020), and also because the first observation of an embodiment is always the same as both our environments have a startpad (see appendix B.2). Finally, we did not use the additional target distribution (called *desired goal distribution* in the original paper) loss that uses a Wassertein discriminator (Arjovsky et al., 2017) to predict whether a task predicted belongs to the target distribution. Indeed, as shown in Racaniere et al. (2020), using the targeted version of SETTER-SOLVER offers more sample efficiency but leads to similar final results. Moreover, in our case, a target distribution is known only in the *High expert knowledge* setup of the challenge-specific experiments, in addition of having this part not implemented at all in the code provided by authors. We thus leave this upgrade to future work.

B.1.2 Hyperparameters Tuning

In order to tune the different ACL methods to our experiments, we chose to perform a grid-search using our Stump Tracks environment with its original task space. As the Parkour is partly extended from it, in addition of the challenge-specific experiments, this environment offered us an appropriate setup. Each point sampled in the grid-search was trained for 7 million steps (instead of the 20 millions used in our experiments) with 16 seeds in order to reduce the (already high) computational cost. At the end of training, we calculated the percentage of mastered tasks on test set for each seed. The combination of hyperparameters having the best average over its seeds was chosen as the configuration for the benchmark.

In order to make the grid-search as fair as possible between the different ACL methods, given that the number of hyperparameters differs from one method to another, we sampled the same number of points for each teacher: 70 ($\pm$10). The hyperparameters to tune for each teacher, as well as their values, were chosen following the recommendations given by their original paper.

Moreover, we chose to tune the teachers in what we call their "original" expert knowledge version (i.e. they have access to the same amount of prior knowledge as the one they used in their paper). Hence, teachers requiring expert knowledge use our high expert knowledge setup, and algorithms such as ALP-GMM use no expert knowledge.

Table B.1 shows the values we tested for each hyperparameter and the combinations that obtained the best result.

Table B.1: Hyperparameters tuning of the ACL methods.

ACL method	Hyperparameter	Possible values	Best value
ADR	t_L	$[0, 50]$	0
ADR	t_H	$[180, 230, 280]$	180
ADR	p_b	$[0.3, 0.5, 0.7]$	0.7
ADR	m	$[10, 20]$	10
ADR	Δ	$[0.05, 0.1]$	0.1
RIAC	max_s	$[50, 150, 250, 350]$	150
RIAC	n	$[25, 50, 75, 100]$	75
RIAC	min_d	$[0.0677, 0.1, 0.1677, 0.2]$	0.1
COVAR-GMM	n	$[50, 150, 250, 350]$	150
COVAR-GMM	max_k	$[5, 10, 15, 20]$	15
COVAR-GMM	p_{rnd}	$[0.05, 0.1, 0.2, 0.3]$	0.1
ALP-GMM	n	$[50, 150, 250, 350]$	150
ALP-GMM	max_k	$[5, 10, 15, 20]$	10
ALP-GMM	p_{rnd}	$[0.05, 0.1, 0.2, 0.3]$	0.05
GOAL-GAN	δ	$[0.01, 0.05, 0.1]$	0.01
GOAL-GAN	n	$[100, 200, 300]$	100
GOAL-GAN	p_{old}	$[0.1, 0.2, 03]$	0.2
GOAL-GAN	$n_{rollouts}$	$[2, 5, 10]$	2
SPDL	n_{OFFSET}	$[100000, 200000]$	200000
SPDL	n_{STEP}	$[50000, 100000]$	100000
SPDL	K_α	$[0, 5, 10]$	0
SPDL	ζ	$[0.05, 0.25, 0.5]$	0.05
SPDL	ϵ	$[0.1, 0.8]$	0.8
SETTER-SOLVER	n	$[50, 100, 200, 300]$	100
SETTER-SOLVER	δ	$[0.005, 0.01, 0.05, 0.1]$	0.05
SETTER-SOLVER	n_{HIDDEN}	$[64, 128, 256, 512]$	128

B.2 Environment Details

In this section, we give details about our two environments, their PCG algorithm, as well as some analysis about their task space. Note that our two environments follow the OpenAI Gym's interface and provide after each step, in addition of usual information (observation, reward, and whether the episode terminated), a binary value set to 1 if the cumulative reward of the episode reached 230. Additionally, we provide extra information and videos of our environments and embodiments, as well as policies learned at at http://developmentalsystems.org/TeachMyAgent/.

B.2.1 Stump Tracks

We used variants of the Stump Tracks environment from chapter 4, whose PCG is parameterized by 2 scalars: stumps' height μ_s and spacing Δ_ss. As in chapter 4, μ_s is used as the mean of a Gaussian distribution with standard deviation 0.1. Each stump has thus its height sampled from this Gaussian distribution and is placed at a distance Δ_s from the previous one. We bound differently this task space depending on the experiment we perform, as explained in appendix B.3.

We kept the same observation space with 10 values indicating distance of the next object detected by lidars, head angle and velocity (linear and angular), as well as information from the embodiment (angle and speed of joints and also whether the lower limbs have contact with the ground). For information concerning the embodiment, the size of observation depends on the embodiment, as the number of joints varies (see below in B.2.3). We also kept the action space controlling joints with a torque.

B.2.2 Parkour

We introduce the Parkour, a Box2D parkour track inspired from the Stump Tracks and the environment introduced in Wang et al. (2020b). It features different milieu in a complex task space.

Procedural Generation

CPPN-encoded terrain – First, similarly to the Stump Tracks, our Parkour features a ground (that has the same length as the one in Stump Tracks) where the agent starts at the leftmost side and has to reach the rightmost side. However, this ground is no longer flat and rather, as in Wang et al. (2020b), generated using a function outputted by a neural network called CPPN (Stanley, 2007). This network takes in input a x position and outputs the associated y position of the ground. Using this, one can slide the CPPN over the possible x positions of the track in order to obtain the terrain. This method has the advantage of being able to easily generate non-linear and very diverse terrains as shown in Wang et al. (2020b), while being light and fast to use as this only needs inference from the network. While CPPNs are usually used in an evolutionary setup where the architecture and weights are mutated, we chose here to rather initialize an arbitrary

architecture and random weights and keep them fixed. For this architecture, we chose to use a four layers feedforward neural network with 64 units per layer and an alternation of TanH and Softplus activations (except for the output head which uses a linear activation) inspired from Ha (2016). Weights were sampled from a Gaussian distribution with mean 0 and standard deviation of 1. In addition of its x input, we added to our network three inputs that are set before generating the terrain as parameters controlling the generation. This vector θ of size 3 acts in a similar way as noise vector does in GANs for instance. Its size was chosen such that it allows to analyze the generation space and maintain the overall task space's number of dimensions quite small. As for the parameters in Stump Tracks, we bounded the space of values an ACL method could sample in θ. For this, we provide three hand-designed setups (easy, medium and hard) differing by the size of the resulting task space and the amount of feasible tasks in it (see appendix B.3.4).

Moreover, in addition of the y output of the ground, we added another output head y_c in order to create a ceiling in our tracks. As in Stump Tracks, the terrain starts with a flat startpad region (with a fixed distance between the ground and the ceiling) where the agent appears. Once $Y = (y_i)_{i \in X}$ and $Y_c = (y_{c_i})_{i \in X}$ generated by the CPPN, with X all the possible x positions in the track, we align them to their respective startpad:

$$y_i = y_i + startpad_g - y_0 \; \forall i \in Y$$

$$y_{c_i} = y_{c_i} + startpad_c - y_{c_0} \; \forall i \in Y_c$$

with $startpad_g$, $startpad_c$ being respectively the y position of the ground startpad and ceiling startpad, and y_0, y_{c_0} respectively the first y position of the ground and the ceiling outputted by our CPPN.

Using this non-linear generator (i.e. CPPN) allows us to have an input space where the difficulty landscape of the task space is rugged. Indeed, in addition of generating two non-linear functions for our ground and ceiling, the two latter can cross each other, creating unfeasible tasks (see figure B.2). Additionally, our CPPN also makes the definition of prior knowledge over the input space more complex, as shown in figure B.1.

Finally, as shown in figure B.2, we smoothed the values of Y and Y_c by a parameter δ ($= 10$ in the training distribution) in order to make the roughness of the terrains adapted to our embodiments.

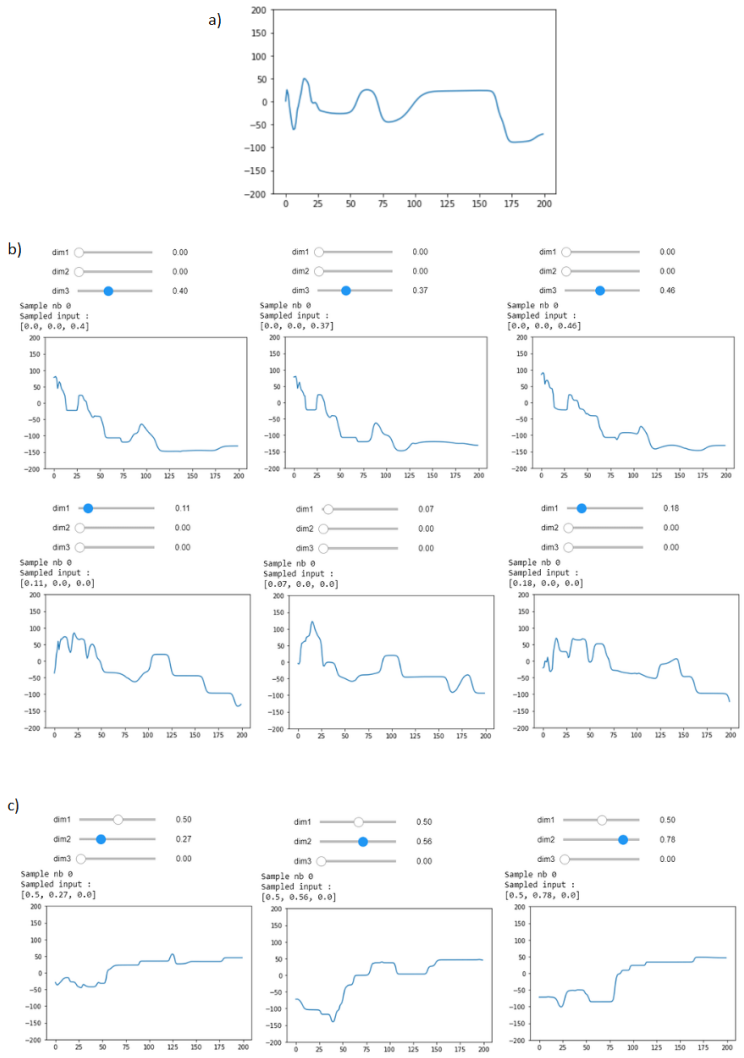


Figure B.1: Overview of the input space of θ. First, in a) one can see the function generated when all the values of the input vector are set to zero. Secondly, in b) we can see that small changes over the space lead to similar functions and that big changes lead to very different results, showing that local similarity is maintained over the task space. Finally, c) shows how the difficulty landscape of θ can be rugged, as moving along the second dimension leads to terrains having a very different difficulty level.

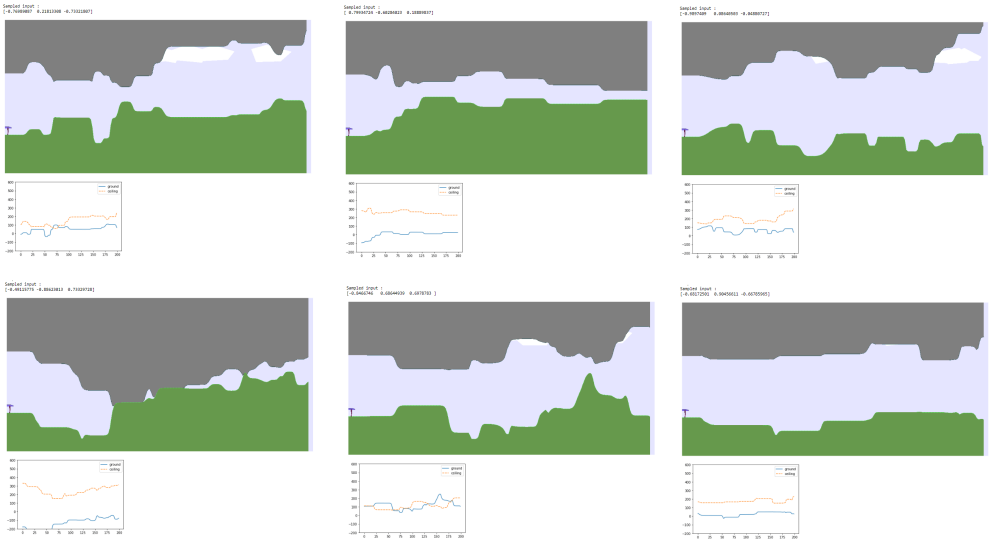


Figure B.2: Here are some examples of randomly generated tasks in the Parkour environment. Most of them seem too hard for a classic bipedal walker, justifying the need for ACL. The bottom left task is a good example of an unfeasible task, no matter which embodiment is used.

Creepers – Once the terrain generated, we add what we call "creepers". Similarly to the stumps, we create objects at distance Δ_c from one another and of height sampled using a Gaussian distribution of mean μ_c and standard deviation 0.1 (the width can also be controlled but was fixed to 0.25 in our experiments). However, creepers are not obstacles for agents as stumps but rather graspable objects that embodiments can go through. Moreover, even though not used in our experiments, we provide the possibility to make creepers more realistic by dividing every creeper in multiple rectangles of height at most 1 linked with a rotating joint. As shown on our website , this creates creepers on which the climbers can swing.

Water – Finally, we added a last dimension to our task space controlling the "water" level. Water is simulated using a rectangle object that the agent can go through and in which physics change (see below). This rectangle's width equals the terrain's width and its height is controlled by a parameter $\tau \in [0; 1]$ with 0 being an arbitrary lower limit the ground can reach and 1 the highest point of the current ceiling (generated by the CPPN for the current task).

Physics

As previously mentioned, we introduced creepers and water along with new physics. First, in order to make our creepers graspable by the agents, we added sensors to the end of limb of certain embodiments (see section B.2.3 below). Every time one of these sensors enters in contact with a creeper, we look at the action in the action space of the agent that is associated to this sensor. If its value is greater than 0, we create a rotational joint between the sensor and the creeper at the contact point. As long as this action is greater than 0, the joint remains. As soon as the action goes negative or equals 0, we delete the

joint (releasing the agent's limb from the creeper) and start watching again for contact. Note that, in order to better see whether the agent grasps a creeper, we color its sensors in red when a joint exists and in yellow otherwise (see our website). Additionally, in order to help the learning agent, we also make the ceiling graspable.

Secondly, concerning the water, we simulated a buoyancy force (inspired from Campbell (2013)) when an object enters water given its density compared to the water's density (set to 1). In addition, we implemented a "drag" and a "lift" force that simulate the resistance applied when an object moves in water and slows down the movement. Finally, we added a "push" force applied to an object having an angular velocity. This force simulates the fact that applying a torque to a rotational joint makes the object attached to the joint "push" the water and move (i.e. have a linear force applied). With these forces, we were able to simulate in a simplified way some physics of water, which resulted in very natural policies learned from our agents (see our website).

Finally, we simulated the fact that each embodiment is suited for one (or several) milieu, creating types of agents. Indeed, we first consider swimming agents that die (i.e. the actions sent by the DRL student to the environment no longer have effects on the motors of the embodiment) after spending more than 600 consecutive steps outside water. On the contrary, the two other types named climbers and walkers cannot survive underwater more than 600 consecutive steps. Both walkers and swimmers are allowed to have collisions with their body (including their head in the Parkour), whereas climbers are not allowed to touch the ground with any part of their body. Note that, while walkers appear with their legs touching the ground, swimmers appear a bit above the ground and climbers appear with all of their sensors attached to the ceiling (see figure 5.2).

All of these physics introduce the fact that an ACL teacher has to propose tasks in the right milieu for the current embodiment (i.e. mostly underwater for swimmers so that they do not die, with creepers and a ceiling high enough for climbers so that they do not touch the ground or die in water and with no water for walker so that they do not drown) in order to make the student learn.

Observation and Action Space

As in Stump Tracks, the agent is rewarded for moving forward and penalized for torque usage. An episode lasts 2000 steps unless the agent reaches the end of the track before or if a part of its body touches the ground if the embodiment is a climber. We also reused the 10 lidars per agent that were used in the Stump Tracks with all the lidars starting from the center of the head of the morphology. However, we modified them such that three configurations of covering exist (see the three tasks shown in figure 5.2):

- 90° from below the agent to ahead of it (used by walkers, as in Stump Tracks)
- 180° from below the agent to above it (used by swimmers)
- 90° from ahead of the agent to above it (used by climbers)

Moreover, in addition of the distance to the next object detected by each lidar, we added an information concerning the type of object detected by the lidar (-1 if water,

1 if creeper, 0 otherwise) such that the agent knows whether the object detected is an obstacle or can be passed through. Note also that once their origin point overlaps an object (e.g. water), lidars no longer detect it. Hence the lidars of an agent underwater no longer detect water (which would have made lidars useless as they would have only detected water). Therefore, in order to inform the DRL student whether the embodiment is underwater, we added an observation that is set to 1 if the agent's head is under the water level and 0 otherwise. Similarly, we added a binary observation telling whether the agent is dead or not (i.e. the actions we send to its motors no longer have impact). In addition, we kept the same information concerning the agent's head as in Stump Tracks (angle, linear velocity and angular velocity) as well as observations for each motor (angle and speed of joint as well as contact information for some of the attached limb). Finally, we added two binary observations per sensor (if the agent has sensors) telling whether the sensor has contact with a graspable surface and whether it is already attached with a joint. Without considering the information about motors and sensors which depend on the morphology, all of the information listed above create an observation vector of size 26. Note that, additionally, we provide the information to the teacher at each step whether the cumulative reward of the episode has reached 230 for the users using a binary reward.

Finally, for the action space, we kept the same behaviour as the one used in Stump Tracks (i.e. each agent has motors which are controlled through a torque value in $[-1; 1]$). Moreover, we added an action in $[-1; 1]$ per sensor for climbers to say whether this sensor must grasp (if it has contact with a graspable surface) or release.

B.2.3 Morphologies

We included in our benchmark the classic bipedal walker as well as its two modified versions introduced in chapter 4: the short bipedal and the quadrupedal. For these three agents, we kept in their implementation the additional penalty for having an angle different than zero on their head, which was already in chapter 4. Additionally, we created new walkers such as the spider or the millipede shown in figure 5.2. See our repository and website for the exhaustive list of embodiments we provide.

We introduce another type of morphologies: climbers. We propose two agents: a chimpanzee-like embodiment, as well as its simplified version without legs (reducing the action space to simplify the learning task). These two agents have two arms with two sensors at their extremity, allowing them to grasp creepers or the ceiling.

Both walkers and climbers have a density of 1 on their legs and arms, and a density of 5 on their body and head, making them simply "sink" in water.

Finally, we created swimming morphologies with each of their body part having the same density as the water, making them in a zero-gravity setting when fully underwater. We propose a fish-like embodiment (see figure 5.2) with a fin and a tale that can wave its body to move (as well as moving its fin).

Note that we also included an amphibious bipedal walker allowed to survive both underwater and outside water. This gave interesting swimming policies as shown on our website (`http://developmentalsystems.org/TeachMyAgent/`).

B.3 Experimental Details

In this section, we give details about the setups of our experiments.

B.3.1 DRL Students

We used the 0.1.1 version of OpenAI Spinningup's implementation of SAC that uses Tensorflow, as in chapter 4. We modified it such that a teacher could set a task at each reset of the environment. We also kept the same hyperparameters as the ones used in chapter 4:

- A two layers feedforward network with 400/300 units per hidden layer (ReLU activation) for both the value and policy network (using TanH activation on the output layer for the latter)
- An entropy coefficient of 0.005
- A learning rate of 0.001
- A mini-batch update every 10 steps using 1000 randomly sampled experiences from a buffer of size 2 millions

For PPO, we used OpenAI Baselines' (Tensorflow) implementation. We used the same two layers neural network as in SAC for the policy and value networks (which share weights). We modified the runner sampling trajectories from the environment in order to use a single synchronous runner instead of multiple asynchronous ones. We used the environment's wrappers proposed in the OpenAI Baselines' implementation to clip actions and normalize observations and rewards. We added a test environment (as well as a teacher that sets tasks) to test the agent's performance every 500000 steps (as done with SAC). We also normalize the observations and rewards in this test environment using the same running average as the one used in the training environment, so that an agent does not receive different information from both environments. We send to the teacher and monitor the original values of reward and observation sent by the environment instead of normalized ones. We set the RIAC factor of the Generalized Advantage Estimator to 0.95, the clipping parameter ϵ to 0.2 and the gradient clipping parameter to 0.5. Finally, we tuned the following hyperparameters using a grid-search on Stump Tracks for 10 millions steps with stumps' height and spacing respectively in $[0; 3]$ and $[0; 6]$:

- Size of experiences sampled between two updates: 2000
- Number of epochs per update: 5
- Learning rate: 0.0003
- Batch size: 1000
- Value function coefficient in loss: 0.5
- Entropy coefficient in loss: 0.0

Note that for both our DRL students, we used $\gamma = 0.99$.

B.3.2 General Experimental Setup

We call an experiment the repetition, using different seeds, of the training of a DRL student for 20 millions steps using tasks chosen at every reset of the environment by a given ACL teacher. The seed is used to initialize the state of random generators used in the teacher, DRL student and environment. We provide to the teacher the bounds (i.e. a *min* and *max* value for each dimension) of the task space before starting the experiment. The DRL student then interacts with the environment and asks the ACL teacher to set the task (i.e. a vector controlling the procedural generation) at every reset of the environment. Once the episode ends, the teacher receives either the cumulative reward or a binary reward (set to 1 if the episodic reward is grater than 230) for GOAL-GAN and SETTER-SOLVER. Teachers like SPDL can additionally access to the information sent by the environment at every step, allowing non-episodic ACL methods to run in our testbed.

Every 500000 steps of the DRL student in the environment, we test its performance on 100 predefined tasks (that we call test set). We monitor the episodic reward obtained on each of these tasks. We also monitor the average episodic reward obtained on the tasks seen by the student during the last 500000 steps. We ask the teacher to sample 100 tasks every 250000 steps of the DRL student and store these tasks to monitor the evolution of the generated curriculum (see at `http://developmentalsystems.org/TeachMyAgent/`). For this sampling, we use the non-exploratory part of our teachers (e.g. ALP-GMM always samples from its GMM or ADR never sets one value to one of its bounds) and do not append these monitoring tasks to the buffers used by some teachers to avoid perturbing the teacher's process.

In our experiments we were able to run 8 seeds in parallel on a single Nvidia Tesla V100 GPU. In this setup, evaluating one ACL method requires approximately (based on ALP-GMM 's wall-clocktime):

- 4608 gpu hours for all skill-specific experiments with 32 seeds.
- 168 gpu hours for the 48 seeded Parkour experiment.

Running both experiments would require 4776 gpu hours, or 48 hours on 100 Nvidia Tesla V100 GPUs. Users with smaller compute budgets could reduce the number of seeds (e.g. divide by 3) without strong statistical repercussions.

B.3.3 Stump Tracks Variants

We used the Stump Tracks environment to create our challenge-specific comparison of the different ACL methods. We leveraged its two dimensional task space (stumps' height and spacing) to create experiments highlighting each of the 6 challenges listed in section 5.1. Each experiment used 32 seeds.

Test Sets

We used the same test set in all our experiments on Stump Tracks to easy comparative analysis. This test set is the same as the one used in chapter 4 with 100 tasks evenly

distributed over a task space with $\mu_s \in [0;3]$ and $\Delta_s \in [0;6]$.

Experiments

In the following paragraphs, we detail the setup of each of our experiments used in the challenge-specific comparison.

Expert knowledge setups – We allow three different amounts of prior knowledge about the task to our ACL teachers:

- *No expert knowledge*
- *Low expert knowledge*
- *High expert knowledge*

First, in the *No expert knowledge* setup, no prior knowledge concerning the task is accessible. Hence, no reward mastery range (ADR, GOAL-GAN and SETTER-SOLVER) is given. Additionally, no prior knowledge concerning the task space like regions containing trivial tasks for the agent (e.g. for ADR or SPDL's initial distribution) or subspace containing the test tasks (e.g. for SPDL's target distribution) are known. However, we still provide these two distribution using the following method:

- *Initial distribution*: we sample the mean $\mu_{INITIAL}$ of a Gaussian distribution uniformly random over the task space. We choose the variance of each dimension such that the standard deviation over this dimension equals 10% of the range of the dimension (as done when expert knowledge is accessible).
- *Target distribution*: we provide a Gaussian distribution whose mean is set to the center of each dimension and standard deviation to one fourth of the range of each dimension (leading to more than 95% of the samples that lie between the min and max of each dimension). This choice of target distribution was made to get closer to our true test distribution (uniform over the whole task space), while maintaining most of the sampled tasks inside our bounds. However, it is clear that this target distribution is not close enough to our test distribution to make SPDL proposing a good curriculum and lead to an agent learning an efficient policy to perform well in our test set. As mentioned in section 5.5 and appendix B.1, using a Gaussian target distribution is not suited to our setup and would require modifications to make the target distribution match our true test distribution.

Hence in this setup, only ALP-GMM, RIAC, COVAR-GMM and SPDL (even though its initial and target distribution do not give insightful prior knowledge) can run.

In the *Low expert knowledge* setup, we give access to reward mastery range. Therefore, GOAL-GAN, SETTER-SOLVER and ADR can now enter in the comparison. The initial distribution is still randomly sampled as explained above. It is used by GOAL-GAN to pretrain its GAN at the beginning of the training process, but also by ADR which starts with a single example being $\mu_{INITIAL}$.

Finally, for the *High expert knowledge* setup, we give access to the information about regions of the task space. While the standard deviation of the initial distribution is still calculated in the same way (i.e. 10% of the range of each dimension), we set $\mu_{INITIAL}$ to $[0; 6]$, with the values being respectively μ_s and Δ_s. Hence, ADR now uses the task $[0; 6]$ as its initial task and GOAL-GAN pretrains its GAN with this distribution containing trivial tasks for the walking agent (as stumps are very small with a large spacing between them). SPDL also uses this new initial distribution, but keeps the same target distribution as we could not provide any distribution matching our real test distribution (i.e. uniform). Note that, as mentioned in appendix B.1, ALP-GMM and COVAR-GMM use this initial distribution in their bootstrap phase in this setup.

Mostly unfeasible task space – In this experiment, we use SAC with a classic bipedal walker. We consider stumps with height greater than 3 impossible to pass for a classic bipedal walker. Hence, in order to make most of the tasks in the task space unfeasible, we use in this experiment $\mu_s \in [0; 9]$ (and do not change $\Delta_s \in [0; 6]$) such that almost 80% of the tasks are unfeasible.

Mostly trivial task space – Similarly, we use in this experiment $\mu_s \in [-3; 3]$ (the Stump Tracks environments clips the negative values with $\mu_s = \max(0, \mu_s)$). Hence 50% of the tasks in the task space will result in a Gaussian distribution used to generate stumps' height with mean 0. We also use SAC with a classic bipedal walker.

Forgetting students – We simulate the catastrophic forgetting behaviour by resetting all the variables of the computational graph of our DRL student (SAC here) as well as its buffers every 7 million steps (hence twice in a training of 20 million steps). All variables (e.g. weights, optimizer's variables, etc.) are reinitialized the same way they were before starting the training and the experience buffer used by SAC is emptied. Note that we also use the classic bipedal walker as embodiment and did not modify the initial task space ($\mu_s \in [0; 3]$ and $\Delta_s \in [0; 6]$).

Rugged difficulty landscape – In order to create a rugged difficulty landscape over our task space, we cut it into 4 regions of same size and shuffle them (see algorithm 3). The teacher then samples tasks in the new task space using interpolation (see algorithm 4) which is now a discontinuous task space introducing peaks and cliffs in difficulty landscape. While the cut of regions is always the same, the shuffling process is seeded at each experiments.

Algorithm 3 Cutting and shuffling of the task space.

Input: Number of dimensions $\mathcal{D}$, bounds $(min_i)_{i\in[\mathcal{D}]}$ and $(max_i)_{i\in[\mathcal{D}]}$, number of cuts k
for $d \in [\mathcal{D}]$ **do**
  Initialise arrays $\mathcal{O}_d, \mathcal{S}_d$
  $size \leftarrow |max_d - min_d|/k$
  for $j \in [k]$ **do**
    Store pair $(min_d + j * size, min_d + (j+1) * size)$ in $\mathcal{O}_d$ and $\mathcal{S}_d$
  Shuffle order of pairs in $\mathcal{S}_d$

Algorithm 4 Interpolate sampled task in the shuffled task space.

Input: Number of dimensions $\mathcal{D}$, task vector $\mathcal{T}$, number of cuts k
Initialise the vector $\mathcal{I}$ of size $\mathcal{D}$
for $d \in [\mathcal{D}]$ **do**
  for $j \in [k]$ **do**
    Get pair o_j in $\mathcal{O}_d$
    Initialize l with the first element of o_j
    Initialize h with the second element of o_j
    if $l \leq \mathcal{T}_d \leq h$ **then**
      Get pair s_j in $\mathcal{S}_d$
      Get β as the interpolation of $\mathcal{T}_d$ from the interval o_j to the interval s_j
      Set $\mathcal{I}_d = \beta$
      End the loop
return $\mathcal{I}$

Robustness to diverse students – To highlight the robustness of an ACL teacher to diverse students, we perform 4 experiments (each with 32 seeds) and then aggregate results. We use the initial task space of Stump Tracks but use both PPO and SAC and two different embodiments: the short bipedal walker and the spider. Each embodiment is used both with PPO and SAC (hence 2 experiments per embodiment and thus a total of 4 experiments). We then aggregate the 128 seeds into a single experiment result.

B.3.4 Parkour Experiments

We perform a single experiment in the Parkour environment using 48 seeds. Among these seeds, 16 use a classic bipedal walker, 16 a chimpanzee and 16 a fish embodiment. We set the bounds of the task space to the following:

- CPPN's input vector $\theta \in [-0.35, 0.05] \times [0.6, 1.0] \times [-0.1, 0.3]$
- Creepers' height $\mu_c \in [0; 4]$
- Creepers' spacing $\Delta_c \in [0; 5]$
- Water level $\tau \in [0; 1]$

Note that the above CPPN's input space is considered as our *medium* one. We also provide an *easy* ($\theta \in [-0.25, -0.05] \times [0.8, 1.0] \times [0.0, 0.2]$) and *hard* ($\theta \in [-1, 1] \times [-1, 1] \times [-1, 1]$) alternative. The easy and medium spaces were designed from our hard task space. Their boundaries were searched such that the task space contains feasible tasks while maintaining diverse terrains.

Test Sets

Unlike in the Stump Track experiments, we needed in the Parkour environment different test sets as our three embodiments (i.e. bipedal walker, chimpanzee, fish) are not meant to act and live in the same milieu (e.g. swimmers do not survive in tasks not containing water). Creating a test set composed of uniformly sampled tasks would not allow us to assess the performance of the current embodiment. Hence, we made for the Parkour three different test sets, each constituted of 100 tasks. As the task space previously defined is composed of mostly unfeasible tasks for any embodiment, we hand-designed each of the three test sets to showcase the abilities of each morphology type, as well as showing the ability of the learned policy to generalize. Each test set has 60 tasks that belong to the training task space and 40 out-of-distribution tasks (using tasks outside the medium CPPN's input space as well as smoothing values different than 10 for the δ parameter). They also share the same distribution between easy (1/3), medium (1/3) and hard (1/3) tasks. We chose each task such that it seems possible given the physical capacities of our embodiments. See figure B.3 for some examples of the test

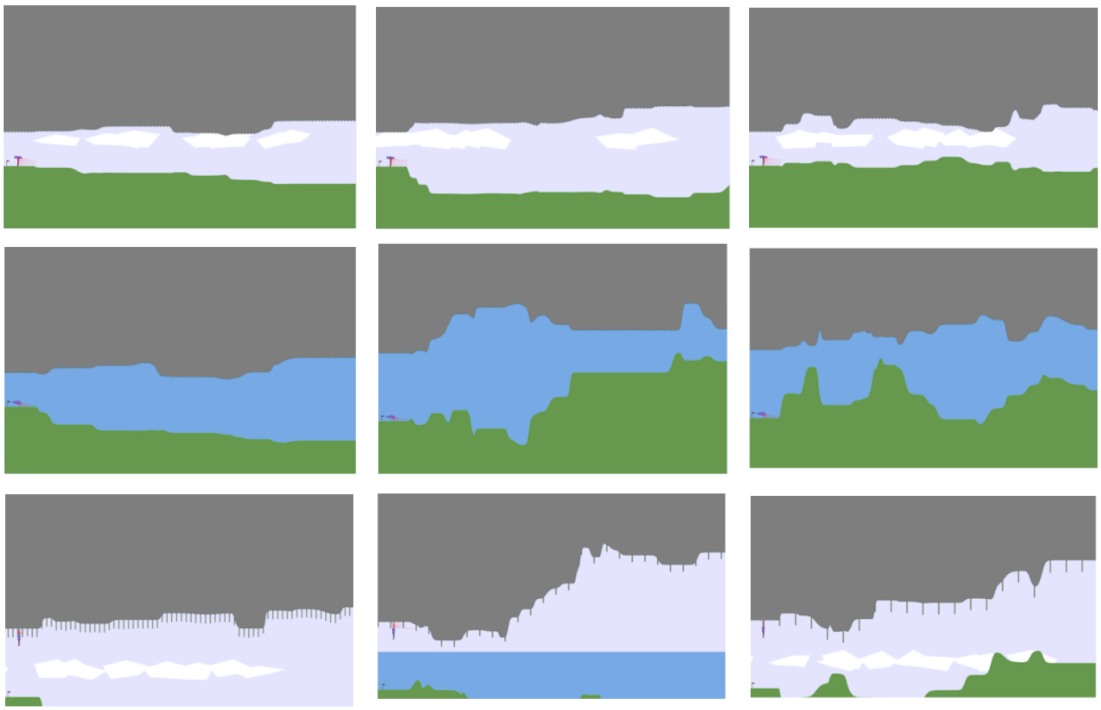


Figure B.3: We show some examples of the tasks belonging to our Parkour's test sets. First line shows tasks from the walkers' test set, second line from the swimmers' one and finally last line for climbers.

tasks.

B.4 Additional Results

In this section, we provide additional results on experiments presented in section 5.5 as well as case studies. As mentioned in appendix B.3, we monitor both the episodic reward on each of the test tasks and the average episodic reward on training tasks every 500000 steps for each seed. We use the episodic reward on test tasks to calculate our percentage of "mastered" tasks metric, which calculates the percentage of tasks on which the agent obtained an episodic reward greater than 230. Additionally, we compare two algorithms in an experiment using Welch's t-test between their population of seeds.

B.4.1 Original Stump Tracks

We trained our SAC student for 20 millions steps on the original Stump Track task space (i.e. $\mu_s \in [0;3]$ and $\Delta_s \in [0;6]$) with each teacher and each expert knowledge setup. We used the best performance of each prior knowledge configuration as a baseline indication in figures 5.2 and B.5. As in our challenge-specific experiments, we used 32 seeds as well as the same test set of 100 evenly distributed tasks. Results can be found in table B.2.

Table B.2: Percentage of mastered tasks after 20 millions steps on the original Stump Tracks challenge (i.e. $\mu_s \in [0;3]$ and $\Delta_s \in [0;6]$). Results shown are averages over 32 seeds along with the standard deviation. We highlight the best results in bold, which then acted as an upper baseline indication in the challenge-specific comparisons.

Algorithm	No EK	Low EK	High EK
ADR	-	24.1 (± 20.8)	43.4 (± 7.2)
ALP-GMM	**52.1** (± 5.9)	**47.1** (± 13.9)	49.3 (± 5.9)
COVAR-GMM	43.0 (± 9.1)	40.25 (± 16.5)	45.2(± 10.1)
GOAL-GAN	-	29.9 (± 26.2)	**51.9** (± 7.3)
RIAC	40.5 (± 8.4)	39.6 ((± 11.2)	42.2 (± 5.4)
SPDL	20.8 (± 19.4)	18.5 (± 20.8)	34.0 (± 10.6)
Setter-Solver	25.3 (± 10.7)	36.6(± 10.2)	37.4 (± 9.8)

B.4.2 Challenge-Specific Comparison

Overall Results

We here show the performance after 20 millions steps of each ACL teacher on each challenge. Results are gathered in tables B.3, B.4 and B.5, as well as in figure B.4 where we show the results of Welch's t-test between all methods on every challenge.

Table B.3: Percentage of mastered tasks after 20 millions steps with **no** prior knowledge in each challenge. Results shown are averages over all seeds along with the standard deviation. We highlight the best results in bold.

Algorithm	Mostly unf.	Mostly triv.	Forgetting stud.	Rugged dif.	Diverse stud.
Random	18.0 (± 10.5)	22.2 (± 15.2)	27.8 (± 14.6)	30.3 (± 7.7)	22.3 (± 11.5)
alp-gmm	**42.8** (± 6.6)	**43.7** (± 6.0)	**42.1** (± 6.9)	**42.5** (± 4.8)	31.5 (± 9.2)
covar-gmm	39.0 (± 9.9)	32.7 (± 16.0)	31.3 (± 16.2)	39.4 (± 7.4)	**32.3** (± 10.6)
riac	22.1 (± 14.5)	20.0 (± 10.9)	36.8 (± 6.9)	36.4 (± 7.9)	25.9 (± 11.3)
spdl	6.4 (± 10.2)	15.3 (± 9.9)	10.4 (± 12.9)	19.3 (± 16.2)	8.9 (± 14.4)

Table B.4: Percentage of mastered tasks after 20 millions steps with **low** prior knowledge in each challenge. Results shown are averages over all seeds along with the standard deviation. We highlight the best results in bold.

Algorithm	Mostly unf.	Mostly triv.	Forgetting stud.	Rugged dif.	Diverse stud.
Random	18.0 (± 10.1)	18.0 (± 7.1)	27.8 (± 14.6)	30.3 (± 7.7)	22.3 (± 11.5)
adr	7.8 (± 17.9)	22.2 (± 15.2)	21.2 (± 21.2)	17.0 (± 19.6)	15.6 (± 19.1)
alp-gmm	**43.5** (± 13.0)	**43.0** (± 9.0)	**41.6** (± 12.5)	**44.2** (± 7.1)	31.3 (± 9.4)
covar-gmm	31.2 (± 16.8)	42.0 (± 8.4)	31.5 (± 18.4)	34.3 (± 10.7)	**32.1** (± 9.6)
goal-gan	12.7 (± 16.2)	38.4 (± 16.1)	9.3 (± 15.8)	34.7 (± 19.1)	16.2 (± 17.5)
riac	20.5 (± 14.0)	21.3 (± 8.8)	34.3 (± 12.5)	38.3 (± 11.3)	26.0 (± 11.7)
spdl	6.7 (± 10.2)	17.9 (± 12.2)	10.6 (± 12.2)	18.1 (± 15.8)	9.2 (± 14.2)
Setter-Solver	25.3 (± 10.7)	35.5 (± 8.9)	33.9 (± 12.5)	31.6 (± 11.3)	25.4 (± 9.0)

Table B.5: Percentage of mastered tasks after 20 millions steps with **high** prior knowledge in each challenge. Results shown are averages over all seeds along with the standard deviation. We highlight the best results in bold.

Algorithm	Mostly unf.	Mostly triv.	Forgetting stud.	Rugged dif.	Diverse stud.
Random	18.0 (± 10.1)	18.0 (± 7.1)	27.8 (± 14.6)	30.3 (± 7.7)	22.3 (± 11.5)
adr	45.3 (± 6.7)	32.5 (± 6.2)	39.8 (± 10.8)	17 (± 20.9)	32.3 (± 9.7)
alp-gmm	**48.4** (± 11.2)	44.3 (± 14.2)	**43.0** (± 9.0)	**42.5** (± 7.3)	29.8 (± 8.8)
covar-gmm	38.2 (± 11.9)	39.6 (± 10.3)	39.5 (± 12.5)	41.3 (± 7.0)	**32.6** (± 10.2)
goal-gan	39.7 (± 10.1)	**45.6** (± 13.5)	23.4 (± 19.7)	41.2 (± 12.6)	27.5 (± 9.4)
riac	25.2 (± 12.3)	22.1 (± 11.1)	37.7 (± 12.5)	37.7 (± 8.8)	25.8 (± 11.7)
spdl	19.1 (± 12.5)	22.9 (± 6.9)	12.9 (± 11.2)	31.0 (± 11.2)	15.4 (± 15.1)
Setter-Solver	28.2 (± 9.7)	33.7 (± 10.8)	37.4 (± 8.7)	34.7 (± 8.1)	24.0 (± 9.8)

Figure B.4: Performance of the different teachers at the end of training in every experiment of our challenge-specific comparison. We plot as bars the average percentage of mastered tasks for each ACL method. Additionally, we compare in every experiment all possible couples of teacher methods using Welch's t-test and annotate the significantly different ($p < 0.05$) ones.

Case Study: Sample Efficiency

In this section, we take a look at the sample efficiency of the different ACL methods using their performance after only 5 millions steps. We reuse the same radar chart as in section 5.5 in figure B.5.

Looking at results, one can see the impact of ACL in the mostly unfeasible challenge, as some methods (e.g. ALP-GMM or ADR with high expert knowledge) already reach twice the performance of random after only 5 million steps. This highlights how leveraging a curriculum adapted to the student's capabilities is key when most tasks are unfeasible. On the opposite, when the task space is easier (as in the mostly trivial challenge), Random samples more tasks suited for the current student's abilities and the impact of Curriculum Learning is diminished.

Having the difficulty landscape rugged makes the search for learnable and adapted subspaces harder. Figure B.5 shows that only 5 millions steps is not enough, even for teachers like ALP-GMM or COVAR-GMM theoretically more suited for rugged difficulty landscapes, to explore and leverage regions with high learning progress.

Finally, one can see the strong impact of a well set initial distribution of tasks in the beginning of learning. Indeed, both ADR and GOAL-GAN already almost reach their final performance (i.e. the one they reached after 20 millions steps shown in figure 5.2) after 5 millions steps in the *High expert knowledge* setup, as they know where to focus and do not need exploration to find feasible subspaces. Similarly, adding expert knowledge to ALP-GMM increases its performance compared to the no and low expert knowledge setups, helping it focus the bootstrapping process on a feasible region. Leveraging this initial task distribution, GOAL-GAN obtains the best results in 3/5 challenges after 5 millions steps with high expert knowledge. This shows, in addition of the results from section 5.5, that GOAL-GAN is a very competitive method, especially when it has access to high expert knowledge.

Figure B.5: **Performance of ACL methods measured after 5 millions steps.** Results are presented as an order of magnitude of the performance of Random. Performance is defined as the average percentage of mastered test tasks over all 32 seeds. We also provide the same indications (✚) of the best performance (measured after 5 millions steps) on the original Stump Tracks experiment as in figure 5.2. Finally, we indicate on each axis which method performed significantly better than Random ($p < 0.05$) using colored stars matching each method's color (e.g. ★ for COVAR-GMM, ★ for ADR). *EK: Expert Knowledge.*

Case Study: On the Difficulty of GOAL-GAN and SPDL to Adapt the Curriculum to Forgetting Students

As mentioned in section 5.5, both GOAL-GAN and SPDL struggled on the forgetting student challenge, no matter the amount of expert knowledge. In order to better understand their behaviour in this challenge, we plot in figure B.6 both the evolution of their percentage of mastered tasks and their average training return. We also add ALP-GMM and ADR (two students that performed well in this challenge) as baselines for comparison. While ADR and ALP-GMM make the student quickly recover from a reset (i.e. the percentage of mastered tasks quickly reaches the performance it had before the reset) and then carry on improving, both GOAL-GAN and SPDL suffer from resets and do not manage to recover, leading to a poor final performance.

This phenomenon could be explained by multiple factors. First, in the case of SPDL, even though the algorithm tries to shift its sampling distribution such that it maximizes

the student's performance, the optimization methods also has to minimize the distance to the target distribution, which is a Gaussian spanning over the entire task space. However, resetting the student's policy requires the ACL method to revert back to the initial simple task distribution that it proposed at the beginning of training. Such a reverse process might not easily be achievable by SPDL, which optimization procedure progressively shifts its sampling distribution towards the target one.

Concerning GOAL-GAN, the performance impact of student resets is most likely due to its use of a buffer of "Goals of Intermediate Difficulty", used to train the goal generator. Upon student reset, this buffer becomes partially obsolete, as the student is reinitialized, making it lose all learned walking gaits, i.e. everything must be learned again. This means the goal generator will propose tasks that are too complex for a student that is just starting to learn. Because GOAL-GAN cannot reset its buffer of "Goals of Intermediate Difficulty" (which would require knowledge over the student's internal state), it impairs its ability to quickly shift to simpler task subspaces.

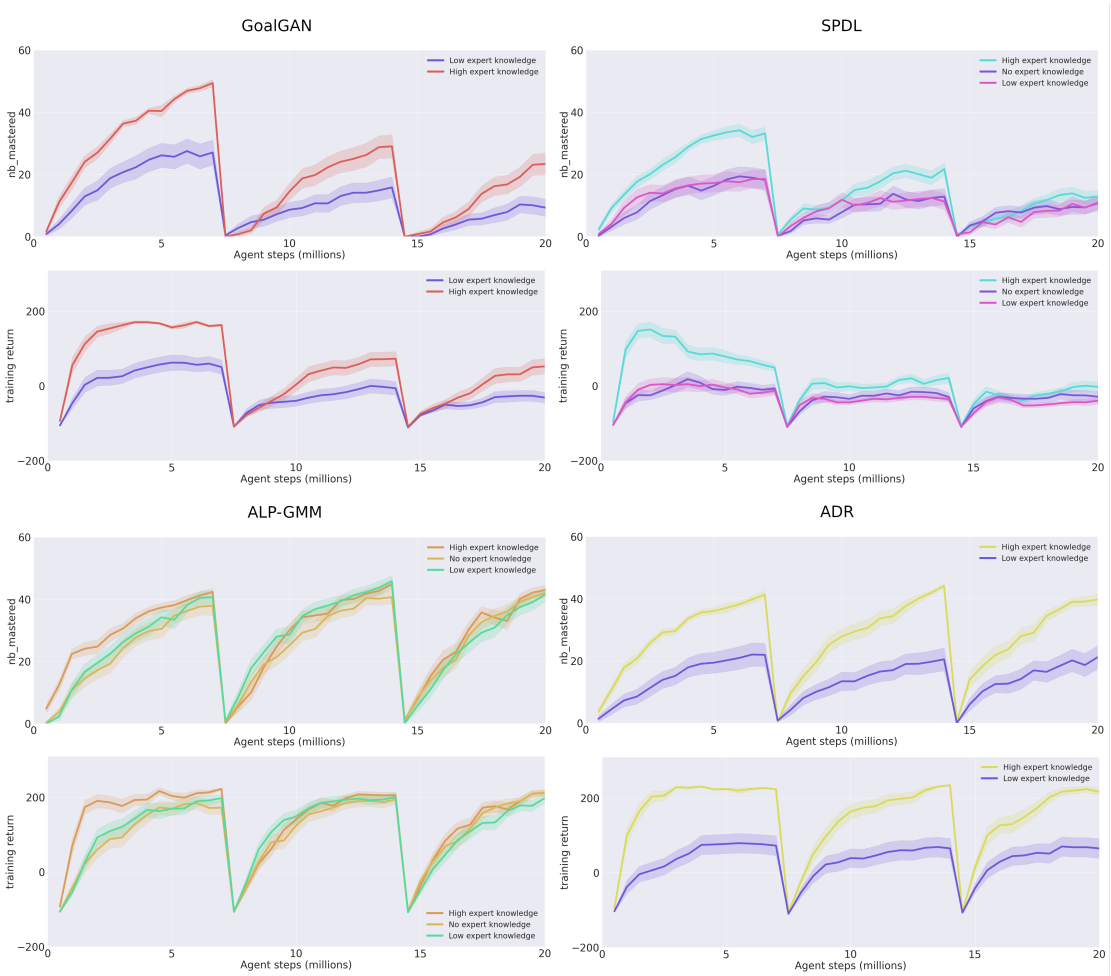


Figure B.6: Percentage of mastered test tasks and average training return of GOAL-GAN, SPDL, ALP-GMM and ADR on the forgetting student challenge. Curves are averages over 32 seeds along with the standard error of the mean.

Case Study: Impact of Expert Knowledge on ALP-GMM

As aforementioned, ALP-GMM is a method initially not requiring any expert knowledge. Moreover, it relies on an exploration (bootstrap) phase to fill its buffer, usually using uniform sampling over the task space. In *TeachMyAgent*, we provide an extended version of it where we added the possibility to use expert knowledge by bootstrapping from an initial distribution instead of a uniform distribution. In this case study, we take a look at the impact such a theoretical improvement had on their performance. We focus on the mostly unfeasible and forgetting student challenges, as the first highlighted the most how prior knowledge can help an ACL method (helping it start in a feasible region) and the latter showed interesting results for this case study, in addition of being easy to analyse (as it only uses a bipedal walker on the original task space of the Stump Tracks). We gather these results in figure B.7. Note that both the no and low expert knowledge setups are the same for ALP-GMM, meaning that any difference between their results is only due to variance in both the student's learning and ACL process.

When looking at these results, one can see that the high expert knowledge setup is significantly better than the two other setups at the beginning of the training in both challenges. These results can also be completed by our sample efficiency case study (see figure B.5), showing that adding expert knowledge to ALP-GMM makes it more sample efficient. Then, as training advances, the difference becomes statistically insignificant ($p > 0.05$). Finally, while the final results given in tables B.3, B.4, and B.5 show an improved percentage of mastered tasks in almost all challenges, with a notable difference (at least +5) on the mostly unfeasible challenge, results on the original Stump Tracks experiments (table B.2) show better results with no expert knowledge. It is thus not clear whether adding this prior knowledge to ALP-GMM benefits the whole training instead of just the beginning. Note that similar behaviours were also obtained with COVAR-GMM, even though they were not as significant as the ones of ALP-GMM.

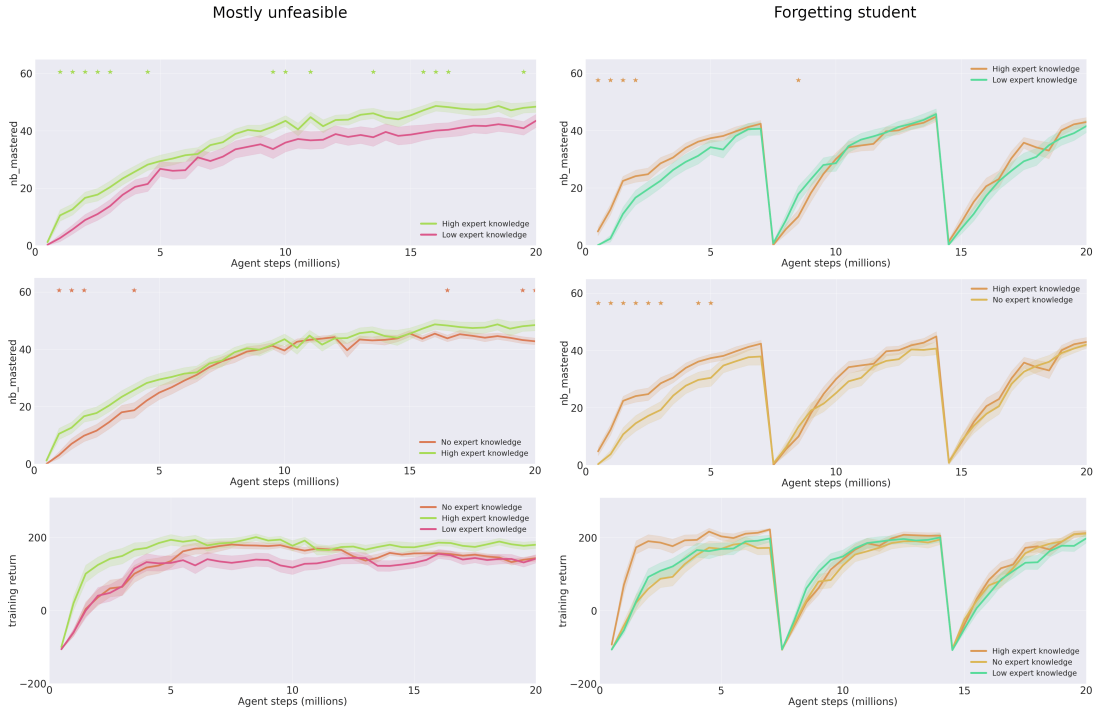


Figure B.7: Percentage of mastered test tasks and average training return of ALP-GMM on both the mostly unfeasible and the forgetting student challenges. Curves are averages over 32 seeds along with the standard error of the mean. We compare the impact of high expert knowledge compared to the two other setups using Welch's t-test and highlight significant ($p < 0.05$) differences with stars.)

Case study: What ADR Needs

ADR is a very efficient and light method, that, when all its expert knowledge requirements are satisfied, competes with the best teachers. However, in order to obtain such an efficient behaviour, ADR needs certain conditions that are implied by its construction. First, as explained in appendix B.1, ADR starts its process using a single task, and makes the assumption that this latter is easy enough for the freshly initialized student. It then progressively grows its sampling distribution around this task if the student manages to "master" the proposed tasks. While this behaviour seems close to SPDL's, ADR does not have any target distribution to help it shift the distribution even if the student's performance are not good enough. Hence, ADR can get completely stuck if it is initialized on a task lying in a very hard region, whereas SPDL would still try to converge to the target distribution (even though the performance would not be as good as if its initial distribution was set in an easy subspace). Similarly, GOAL-GAN also uses an initial distribution at the beginning of the training which, as shown in the results, has a strong impact on the final performance. However, even without it, GOAL-GAN is still able to reach a decent performance in certain challenges (e.g. mostly trivial) unlike ADR. This observation can also be seen in the Parkour's experiments, where GOAL-GAN reaches the top 4 while ADR obtains the worst performance. In order to highlight this explanation,

we provide the results of ADR in the mostly unfeasible and mostly trivial challenges in figure B.8, in addition of the clear difference between expert knowledge setups showed by figure 5.2 and tables B.3, B.4, and B.5. Using figure B.8, one can see the clear and significant ($p < 0.05$) difference between the two expert knowledge setups.

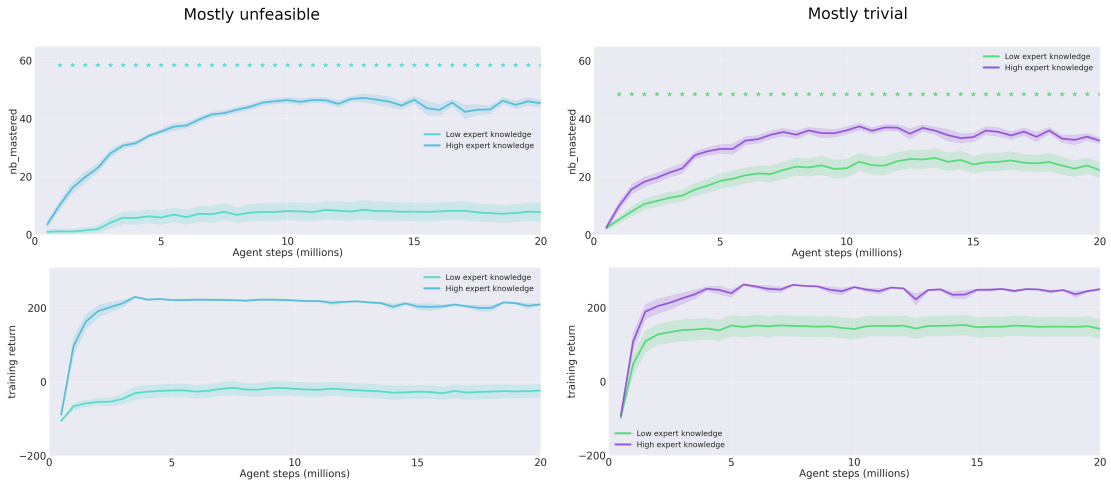


Figure B.8: Percentage of mastered test tasks and average training return of ADR on both the mostly unfeasible and mostly trivial challenges. Curves are averages over 32 seeds along with the standard error of the mean. We compare the impact of high to low expert knowledge using Welch's t-test ($p < 0.05$) and highlight significant differences with stars.)

In addition of an initial task well set using prior knowledge about the task space, ADR needs a difficulty landscape not too rugged to be able to expand and reach regions with high learning progress for the student. Indeed, when looking at its algorithm, one can see that the sampling distribution grows in a certain direction only if the student is able to master the tasks proposed at the edge of the distribution on this direction. If it is not the case (i.e. if this region of the task space is too hard for the current student's capabilities), the sampling distribution will shrink. This simple mechanism makes the strong assumption that if the difficulty is too hard at one edge of the distribution, there is no need to go further, implicitly saying that the difficulty further is at least as hard as the one at the edge. While this works well in the vanilla task space of our Stump Tracks environment (our difficulty is clearly smooth and even monotonically increasing), any task space with a rugged difficulty landscape would make the problem harder for ADR. Indeed, as it reaches a valley in the difficulty landscape surrounded by hills of unfeasible (or too hard for the current student's abilities) tasks, ADR can get stuck. In order to highlight this behaviour, we use our rugged difficulty landscape challenge, where we created a discontinuous difficulty landscape where unfeasible regions can lie in the middle of the task space. Figure B.9 shows how ADR is unable to solve this challenge, no matter the amount of expert knowledge it uses, leading to the worst performance of our benchmark (significantly worse than Random at $p < 0.05$). Note that this issue also happens in our Parkour experiments, as the difficulty of the task space is very rugged (see section 5.5).

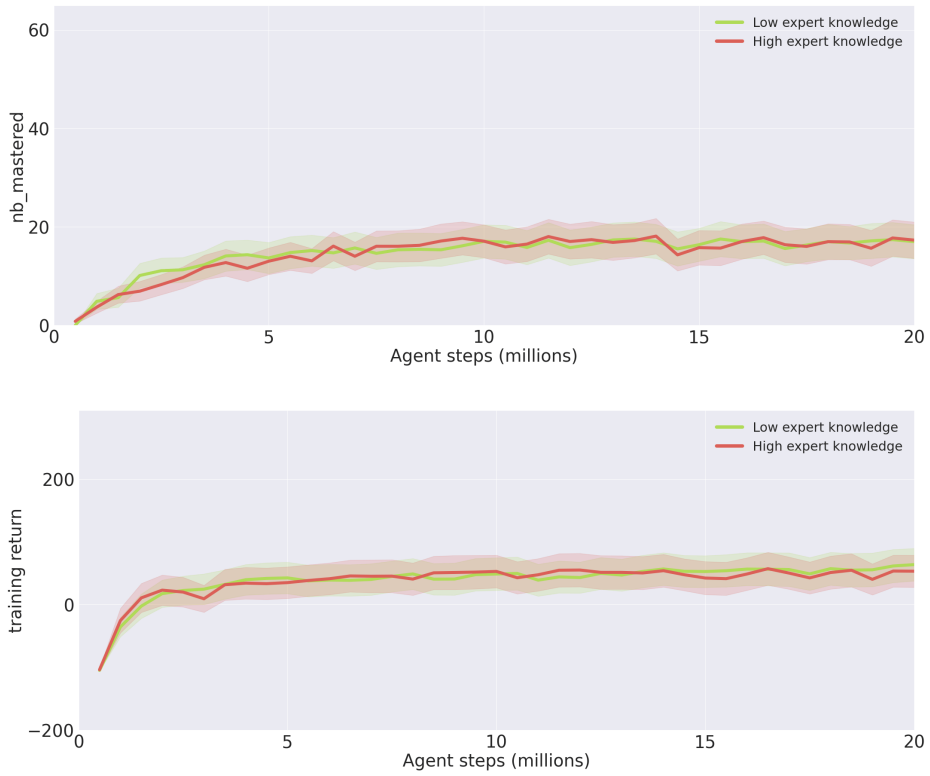


Figure B.9: Percentage of mastered test tasks and average training return of ADR on rugged difficulty landscape challenge. Curves are averages over 32 seeds along with the standard error of the mean.

B.4.3 Parkour

Overall Results

In this section, we present the performance of our teacher algorithms on the Parkour track experiments. We present the final results in table B.6 as well as a comparison in figure B.10 using Welch's t-test. We also provide insights concerning the obtained policies at `http://developmentalsystems.org/TeachMyAgent/`. When looking at the overall results, one can see that ALP-GMM is the only teacher performing significantly better than Random throughout training. COVAR-GMM's performance are very close to ALP-GMM, as well as RIAC, which obtains very similar results to GOAL-GAN. While being very close to Random, SETTER-SOLVER's results are not significantly different from ALP-GMM's results by the end of the training. Finally, while SPDL's behavior is very similar to Random, ADR reaches a plateau very soon and eventually obtains significantly worse results than the random teacher.

Table B.6: Percentage of mastered tasks after 20 millions steps on the Parkour track. Results shown are averages over 16 seeds along with the standard deviation for each morphology as well as the aggregation of the 48 seeds in the overall column. We highlight the best results in bold.

Algorithm	BipedalWalker	Fish	Climber	Overall
Random	27.25 (± 10.7)	23.6 (± 21.3)	0.0 (± 0.0)	16.9 (± 18.3)
ADR	14.7 (± 19.4)	5.3 (± 20.6)	0.0 (± 0.0)	6.7 (± 17.4)
ALP-GMM	**42.7** (± 11.2)	36.1 (± 28.5)	0.4 (± 1.2)	**26.4** (± 25.7)
Covar-GMM	35.7 (± 15.9)	29.9 (± 27.9)	0.5 (± 1.9)	22.1 (± 24.2)
GoalGAN	25.4 (± 24.7)	34.7 (± 37.0)	0.8 (± 2.7)	20.3 (± 29.5)
RIAC	31.2 (± 8.2)	**37.4** (± 25.4)	0.4 (± 1.4)	23.0 (± 22.4)
SPDL	30.6 (± 22.8)	9.0 (± 24.2)	**1.0** (± 3.4)	13.5 (± 23.0)
Setter-Solver	28.75 (± 20.7)	5.1 (± 7.6)	0.0 (± 0.0)	11.3 (± 17.9)

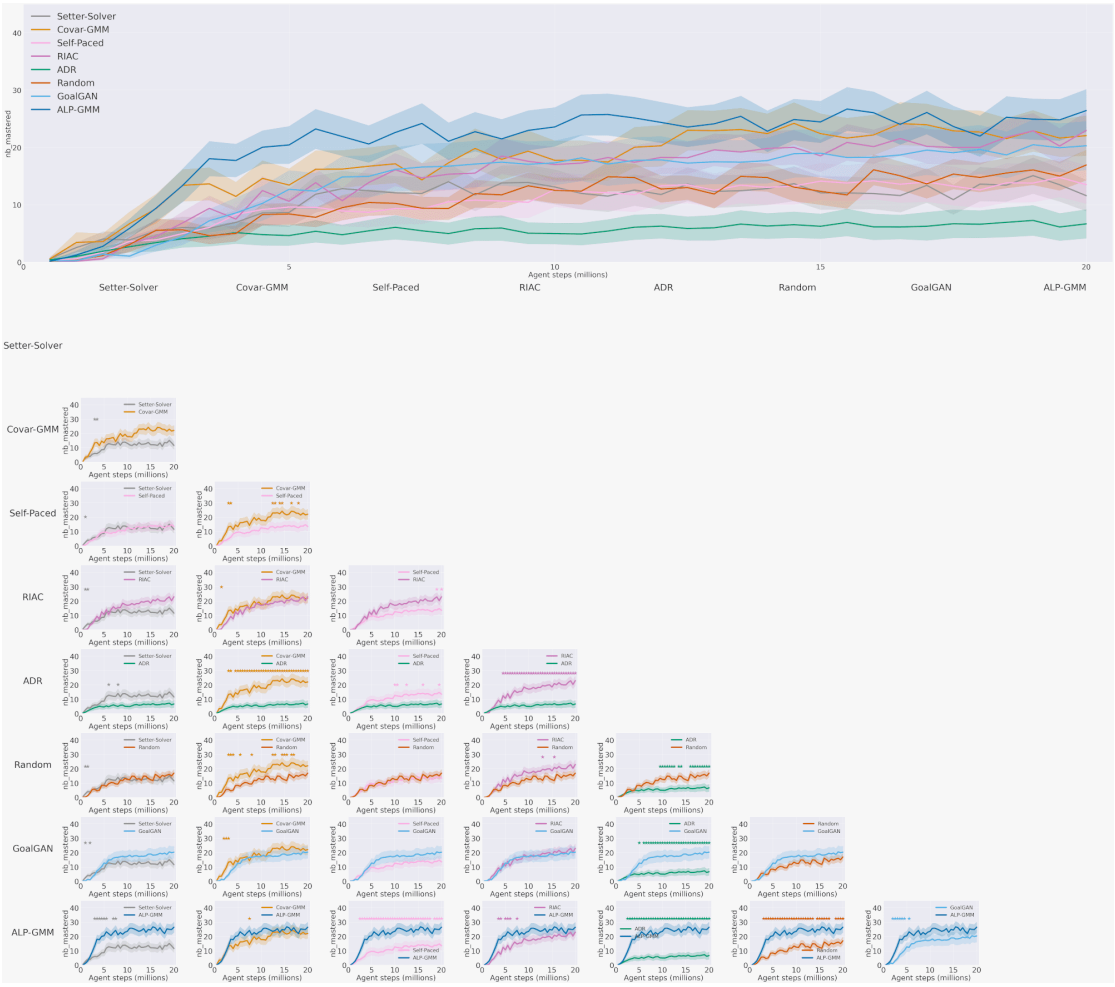
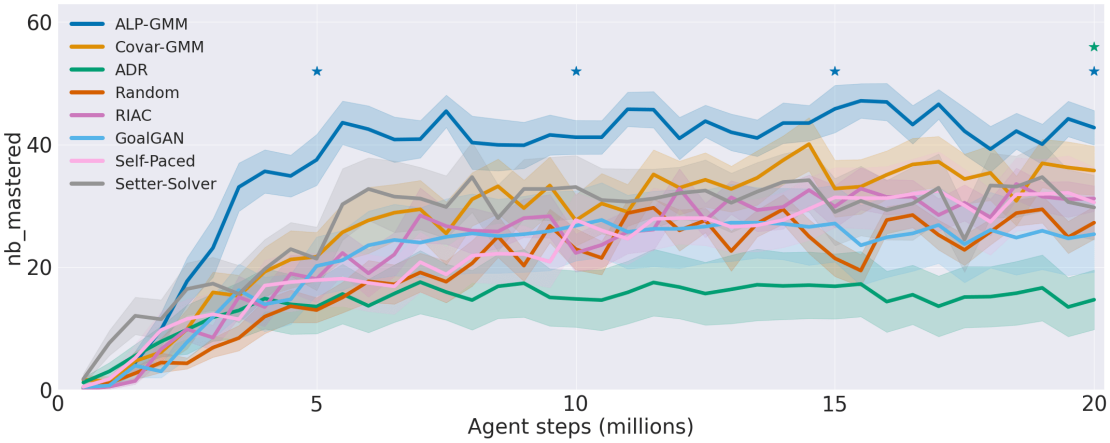
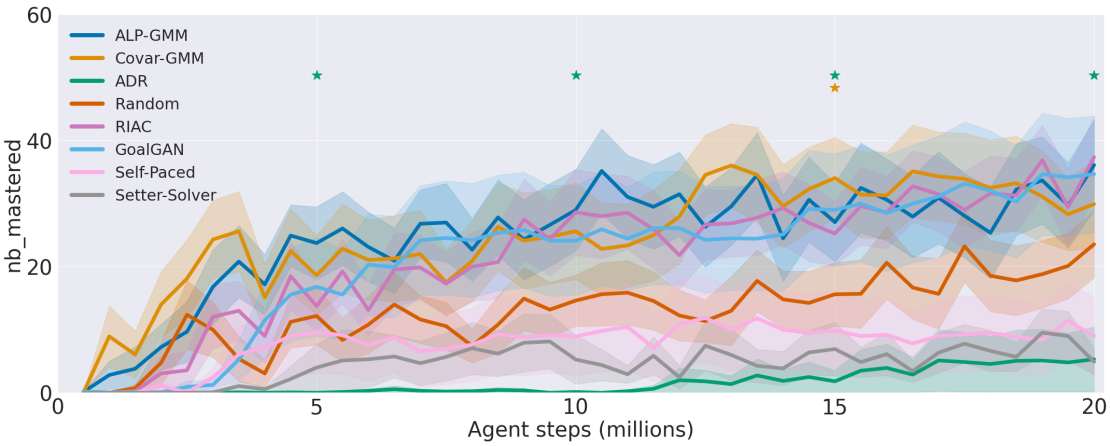


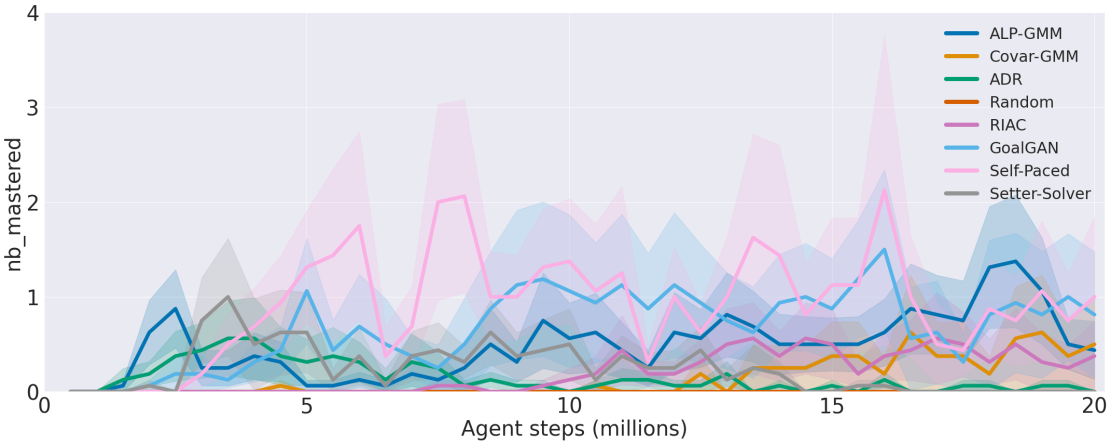
Figure B.10: **Comparison of the ACL methods on the Parkour experiments**. Upper figure shows the average percentage of mastered tasks over the 48 seeds with the standard error of the mean. We then extract all the possible couples of methods and compare their two curves. At each time step (i.e. every 500000 steps), we use Welch's t-test to compare the two distributions of seeds. If a significant difference exists ($p < 0.05$), we add a star above the curves at this time step.



(a) **Bipedal walker** morphology



(b) **Fish** morphology



(c) **Climber** morphology

Figure B.11: Average percentage of mastered tasks over 16 seeds for each tested morphology, along with the standard error of the mean. We calculate every 5 millions steps which method obtained statistically different ($p < 0.05$) results from Random and indicate it with a star.

Case Study: Learning Climbing Locomotion

As shown in figures 5.5 and B.11c, none of the ACL methods implemented in *TeachMyAgent* managed to find a curriculum helping the student to learn an efficient climbing policy and master more than 1% of our test set. While learning climbing locomotion can arguably appear as a harder challenge compared to the swimming and walking locomotion, we present in this case study the results of an experiment using our easy CPPN's input space (see appendix B.3.4), as well as no water (i.e. the maximum level is set to 0.2, leading to no tasks with water). Using this, we show that simplifying the task space allows our Random teacher to master more than 6% our test set with its best seed reaching 30% at the end of learning in only 10 millions steps. In comparison, our results in the benchmark show a best performance of 1% of mastered tasks (SPDL) with its best seed reaching only 14% by the end of learning. As this simpler task space contains more feasible tasks, these results show that the poor performance obtained with the chimpanzee embodiment are due to the inability of the implemented ACL algorithms to find feasible subspaces for their student. This also hints possible better performance by future methods in this totally open challenge of *TeachMyAgent*. See figure B.12 for the evolution of percentage of mastered tasks by the Random teacher in this simpler experiment.

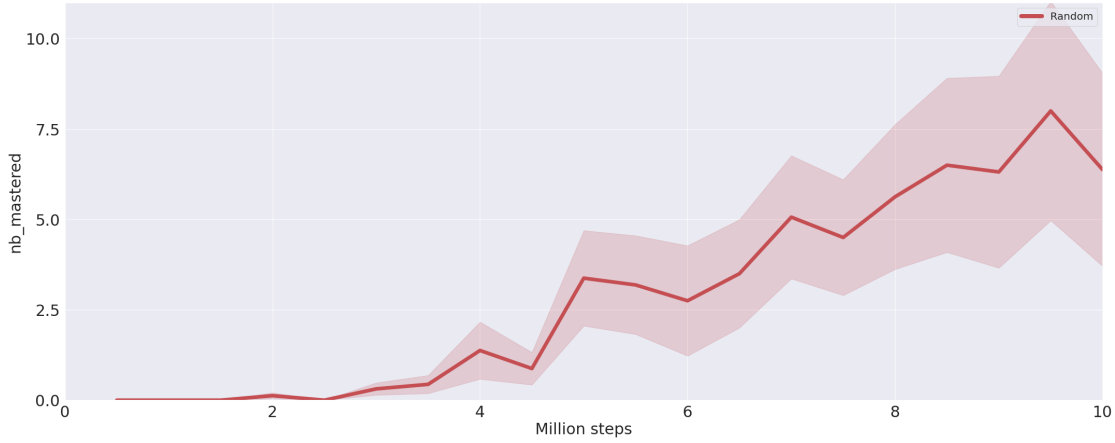


Figure B.12: **Random teacher in the easy CPPN's input space with no water**. Average percentage of mastered tasks over 16 seeds using our chimpanzee embodiment along with the standard error of the mean.

Appendix C

Meta Automatic Curriculum Learning

Contents

C.1 AGAIN

ALP-GMM – We refer the reader to section 4.4.1 for detailed descriptions of ALP-GMM. AGAIN uses ALP-GMM as an underlying ACL algorithm. In all of our experiments we use the same hyperparameters as in chapter 4 ($N = 250$, $k_{max} = 10$), except for the percentage of random task sampling ρ_{rnd} which we set to 10% (we found it to perform better than 20%) when running ALP-GMM. Note that in chapter 6, we refer to ALP as LP for simplicity (i.e. LP_{ti} in X from eq. 6.4 is equivalent to the mean ALP of Gaussians in ALP-GMM).

IN variants. – In order to filter the list X_{raw} (see eq. 6.4) of GMMs extracted from a previous student from $\mathcal{H}^{\mathcal{S}}$ trained with ALP-GMM into X and use it as an expert curriculum, we remove any Gaussian with a LP_{ti} below $\delta_{LP} = 0.2$ (the LP dimension is normalized between 0 and 1, which requires the experimenter to choose an approximate potential reward range, set to $[-150, 350]$ for all experiments on Box2D locomotion environments (sec. 6.5.2 and sec. 6.5.3). When all Gaussians of a GMM are discarded, the GMM is removed from X. In practice, it allows to 1) remove non-informative GMMs corresponding to the initial exploration phase of ALP-GMM, when the learner has not made any progress (hence no LP detected by the teacher), and 2) discard X_{raw} entirely and its associated student s from the teaching history if ALP-GMM never detected high-LP Gaussians, i.e. it failed to train student s. We propose 3 variants to iterate over X to generate a task-sampling curriculum:

- *Pool-based (IN-P), algo. 6:* A rather crude approach is to disregard the ordering of X and merge the trajectory of GMMs into a single large GMM. This GMM is used for every task sampling step, i.e. the curriculum is fixed.

- *Time-based (IN-T), algo. 7:* In this version X is stepped in periodically at a rate N, which we set to 250 (same as the fitting rate of ALP-GMM).

- *Reward-based (IN-R), algo. 8:* A more adaptive option is to iterate over X only once the mean episodic reward over tasks recently sampled from the current GMM matches or surpasses the mean episodic reward recorded during the initial ALP-GMM run (on the same GMM). The IN-R approach requires extracting additional data from the first run, in the form of a list $\mathcal{R}_{raw}$:

$$\mathcal{R}_{raw} = \{\mu_r^1, ..., \mu_r^t, \mu_r^T\} \quad s.t \quad |\mathcal{R}_{raw}| = |X_{raw}|, \tag{C.1}$$

  with T the total number of GMMs in the first run (same as in X_{raw}), and μ_r^t the mean episodic reward obtained by the first DRL agent during the last 50 tasks sampled from the t^{th} GMM. $\mathcal{R}$ is simply obtained by removing any μ_r^t that corresponds to a GMM discarded while extracting X from X_{raw}. The remaining rewards are then used as thresholds in IN-R to decide when to switch to the next GMM in X.

Regardless of the selection process, given a GMM, a new task is selected by sampling its parameters from a Gaussian selected proportionally to its LP_{ti} value.

AGAIN – In AGAIN (see algo. 9), the idea is to use both IN (R,T or P) and ALP-GMM (without the random bootstrapping period) for curriculum generation. Our main experiments use IN-R as it is the highest performing variant (see app. C.3). This means that in the main sections of chapter 6, AGAIN = AGAIN-R and IN = IN-R. We combine the changing GMM of IN and ALP-GMM over time, simply by building a GMM G containing Gaussians from the current GMM of IN and ALP-GMM. By selecting the Gaussian in G from which to sample a new task using their respective LP, this approach allows to adaptively modulate the task sampling between both, shifting the sampling towards IN when ALP-GMM does not detect high-LP subspaces and towards ALP-GMM when the current GMM of IN have lower-LP Gaussians. While combining ALP-GMM to IN, we reduce the residual random sampling of ALP-GMM from $\rho_{high} = 10\%$, used for the pretrain phase, to either $\rho_{low} = 2\%$ for experiments presented in sec. 6.5.1 and sec. 6.5.3, or $\rho_{low} = 0\%$ for experiments done in the *Walker-Climber* environment in sec. 6.5.2 (here we found $\rho_{low} = 0\%$ to be beneficial in terms of performances w.r.t. $\rho_{low} = 2\%$, which means that the task-exploration induced by the periodic GMM fit of ALP-GMM was sufficient for exploration). In AGAIN-R and AGAIN-T, when the last GMM $p(T)$ of the IN curriculum is reached, we switch the fixed LP_{Ti} values of all IN Gaussians to periodically updated LP estimates, i.e. we allow AGAIN to modulate the importance of $p(T)$ for task sampling depending on its current student's performance.

Algorithm 5 Pretrain phase (helper function)

Require: Student policy s_θ, teacher training history $\mathcal{H}^{\mathcal{S}}$, task-encoding parameter space $\mathcal{T}$, LP threshold δ_{LP}, experimental pre-train budget E_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}

Init s_θ, train it for E_{pre} env. steps with ALP-GMM $(\rho_{high}, \mathcal{T})$ # See algo. 1
Pre-test s_θ with m tasks selected uniformly over $\mathcal{T}$ and get CP_s^{pre} # Pre-test phase
Apply knn algorithm in CP space of $\mathcal{H}^{\mathcal{S}}$, get k students closest to CP_s^{pre}
Among those k, keep the one with highest summed post training CP^{post}, extract its X_{raw}
Get X from X_{raw} by removing any Gaussian with $LP_{ti} < \delta_{LP}$.
Return X

Algorithm 6 Inferred progress Niches - Pool-based (IN-P)

Require: Student policy s_θ, teacher training history $\mathcal{H}^{\mathcal{S}}$, task-encoding parameter space $\mathcal{T}$, LP threshold δ_{LP}, update rate N, experimental budget E, experimental pre-train budget E_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}

Launch Pretrain phase and get expert GMM list X # See algo. 5
Initialize pool GMM G_{IN}, containing all Gaussians from X
loop Stop after sampling E tasks (including pre-train)
    Sample τ from a Gaussian in G_{IN} chosen proportionally to its LP_{ti}
    Generate env. with τ, send it to student s_θ
Add student's training data to $\mathcal{H}^{\mathcal{S}}$
Return s_θ

Algorithm 7 Inferred progress Niches - Time-based (IN-T)

Require: Student policy s_θ, teacher training history $\mathcal{H}^{\mathcal{S}}$, task-encoding parameter space $\mathcal{T}$, LP threshold δ_{LP}, update rate N, experimental budget E, experimental pre-train budget E_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}

Launch Pretrain phase and get expert GMM list X # See algo. 5
Initialize expert curriculum index i_c to 0
loop Stop after sampling E tasks (including pre-train)
    Set i_c to $min(i_c + 1, len(X))$
    Set current GMM G_{IN} to i_c^{th} GMM in X
    loop N times
        Sample τ from a Gaussian in G_{IN} chosen proportionally to its LP_{ti}
        Send $E(a \sim \mathcal{A}(p))$ to student s_θ
Add student's training data to $\mathcal{H}^{\mathcal{S}}$
Return s_θ

Algorithm 8 Inferred progress Niches - Reward-based (IN-R)

Require: Student policy s_θ, teacher training history $\mathcal{H}^{\mathcal{S}}$, task-encoding parameter space $\mathcal{T}$, LP threshold δ_{LP}, update rate N, experimental budget E, experimental pre-train budget E_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{high}

Launch Pretrain phase and get expert GMM list X # See algo. 5
Initialize reward First-in-First-Out window $\mathcal{W}$, set max size to N
Initialize expert curriculum index i_c to 0
loop Stop after sampling E tasks (including pre-train)
    If $\mathcal{W}$ is full, compute mean reward μ_w from $\mathcal{W}$
        If μ_w superior to i_c^{th} reward threshold in $\mathcal{R}$, set i_c to $min(i_c + 1, len(X))$
    Set current GMM G_{IN} to i_c^{th} GMM in X
    Sample τ from a Gaussian in G_{IN} chosen proportionally to its LP_{ti}
    Generate env. with τ, send it to student s_θ and add episodic reward r_τ to $\mathcal{W}$
Add student's training data to $\mathcal{H}^{\mathcal{S}}$
Return s_θ

Algorithm 9 Alp-Gmm And Inferred progress Niches (AGAIN)

Require: Student policy s_θ, teacher training history $\mathcal{H}^{\mathcal{S}}$, task-encoding parameter space $\mathcal{T}$, LP threshold δ_{LP}, update rate N, experimental budget E, experimental pre-train budget E_{pre}, pre-test set size m, number of neighbors for student selection k, random sampling ratio ρ_{low} and ρ_{high}

Launch Pretrain phase and get expert GMM list X # See algo. 5
Setup new ALP-GMM ($\rho_{rnd} = 0, \mathcal{T}$) # See algo. 1
Setup either IN-T, IN-P or IN-R # See algo. 7, 6 and 8
loop Stop after sampling E tasks (including pre-train)
    Get composite GMM G from the current GMM of both ALP-GMM and IN
    ρ_{low}% of the time, sample a random parameter $\tau \in \mathcal{T}$
    Else, sample τ from a Gaussian chosen proportionally to its LP
    Generate env. with τ, send it to student s_θ and observe episodic reward r_τ
    Send (p, r_p) pair to both ALP-GMM and IN
Add student's training data to $\mathcal{H}^{\mathcal{S}}$
Return s_θ

C.2 Considered ACL and Meta-ACL Teachers

Meta-ACL variants – Our proposed approach, AGAIN, is based on the combination of an inferred expert curriculum with ALP-GMM, an exploratory ACL approach. In appendix C.1, we present 3 approaches to use such an expert curriculum, giving the AGAIN-R, AGAIN-P and AGAIN-T algorithms (only AGAIN-R is used in our main experiments). In our experiments, we also consider ablations where we only use the expert curriculum, giving the IN-R, IN-P and IN-T variants. We also consider two additional AGAIN variants that do not use our proposed CP-based student selection method:

- AGAIN with Random curriculum prior selection (AGAIN_RND), a lower-baseline which do not perform pre-tests and instead directly extract curriculum priors from a randomly selected student in the teaching history $\mathcal{H}^{\mathcal{S}}$.
- AGAIN with Ground Truth selection (AGAIN_GT), an upper-baseline using privileged information. Instead of performing the knn algorithm in the CP space (see section 6.4), this approach directly uses the true student distribution. For instance, in the *Walker-Climber* environment, given a new student s, AGAIN_GT selects the k previously trained students from $\mathcal{H}^{\mathcal{S}}$ that are morphologically closest to s (i.e. same embodiment type and closest limb sizes), and extracts curriculum priors X from the student with the highest score j_s (see sec. 6.4).

Note that both for AGAIN_RND and AGAIN_GT, there is no need to pre-test the student, which means we can use the IN expert curriculum directly at the beginning of training rather than after a pre-training phase.

ACL conditions – A first natural ACL approach to compare our AGAIN variants to is ALP-GMM, the underlying ACL algorithm in AGAIN. We also add as a lower-baseline a random curriculum teacher (Random), which samples tasks' parameters randomly over the task space.

In both the toy environment (sec. 6.5.1, toy env. for short) and the *Walker-Climber* environment (sec. 6.5.2), we additionally compare to Adaptive Domain Randomization (ADR), an ACL algorithm proposed in OpenAI et al. (2019), which is based on inflating a task distribution sampling from a predefined initially feasible task τ_{easy} (w.r.t a given student). Each lower and upper boundaries of each dimension of the sampling distribution are modified independently with step size Δ_{step} whenever a predefined mean reward threshold r_{thr} is surpassed over a window (of size q) of tasks occasionally sampled (with probability ρ_b) at the sampling dimension boundary. More details can be found in appendix B.1 or OpenAI et al. (2019). In our experiments, as we do not assume access to expert knowledge over students sampled within the student distribution, we randomize the setting of τ_{easy} uniformly over the task space in *Walker-Climber* experiments and uniformly over the 4 possible student starting subspaces in toy env. experiments. Based on the hyperparameters proposed in OpenAI et al. (2019) and on informal hyperparameter search, we use $[\rho_b = 0.5, r_{thr} = 1, \Delta_{step} = 0.05, q = 10]$ in toy env. experiments and $[\rho_b = 0.5, r_{thr} = 230, \Delta_{step} = 0.1, q = 20]$ in *Walker-Climber* experiments.

In experiments described in sec 6.5.3, we compare our approaches to an oracle condition (Oracle), which is a hand-made curriculum that is very similar to IN-R, except that the list X is built using expert knowledge before training starts (i.e. no pre-train and pre-test phases), and all reward thresholds μ_r^i in $\mathcal{R}$ (see eq. C.1) are set to 230, which is an episodic reward value often used in the literature as characterizing a default walker having a "reasonably efficient" walking gate in environments derived from the Box2D gym environment BipedalWalker, e.g. in Wang et al. (2019b) or chapter 4. In practice, Oracle starts proposing tasks from a Gaussian (with std of 0.05) located at the simplest subspace of the task space (i.e. low stump height and high stump spacing) and then gradually moves the Gaussian towards the hardest subspaces (high stump height and low stump spacing) by small increments (50 steps overall) happening whenever the mean episodic reward of the DRL agent over the last 50 proposed tasks is superior to 230.

C.3 Analyzing Meta-ACL in a Toy Environment

In this section, we report the full comparative experiments done in the toy environment, which includes comparisons with AGAIN-T and AGAIN-P to AGAIN-R, shown in table C.1. We also provide visualizations of the CP-based curriculum priors selection process (see fig. C.2) happening after the pretraining phase in AGAIN along with a visualization of the fixed set of 96 randomly drawn students used to perform the varying classroom experiments reported in sec. 6.5.1 (see fig. C.1).

Additional comparative analysis – Table C.1 summarizes the post-training performances obtained by our considered Meta-ACL conditions and ACL baselines on the toy environment on a fixed student test-set of 48 randomly drawn students (among 4 possible student types). Meta-ACL conditions are given a teaching history $\mathcal{H}^{\mathcal{S}}$ created by training an initial classroom of 128 students. Using a Reward-based iterating scheme over the inferred expert curriculum (AGAIN-R and IN-R) outperforms the Time-based and Pool-based variants ($p < .001$). This result was expected as both these last two variants do not have flexible mechanisms to adapt to the student being trained. The pool based variants (AGAIN-P and IN-P), which discard the temporal ordering of the expert curriculum are the worst performing variants, statistically significantly inferior to both Reward-based and Time-based conditions ($p < .001$).

Table C.1: **Experiments on the toy environment.** The average performance with standard deviation after 200k episodes is reported (48 seeds per conditions). For Meta-ACL variants we report results with column 1) the regular CP-based curriculum prior selection performed after 20k pre-training episodes, column 2) An ablation that performs the selection at random before training, and column 3) An oracle condition selecting before training the curriculum prior using student ground truth type. * Denotes stat. significant advantage w.r.t. ALP-GMM (Welch's t-test at $200k$ ep. with $p < 0.05$).

Condition	Regular	Random	Ground Truth
AGAIN-R	98.8 +- 4.8*	55.4 +- 32.2	99.8 +- 0.9*
IN-R	91.4 +- 3.4*	26.3 +- 41.1	92.5 +- 3.0*
AGAIN-T	84.3 +- 3.8	38.6 +- 34.1	89.0 +- 1.7*
IN-T	79.0 +- 12.0	30.3 +- 37.3	88.9 +- 1.7*
AGAIN-P	38.2 +- 7.5	9.3 +- 9.2	14.8 +- 1.2
IN-P	40.6 +- 6.4	9.2 +- 9.0	15.1 +- 1.2
ALP-GMM	84.6 +- 3.4		
ADR	14.9 +- 27.4		
Random	10.0 +- 0.8		

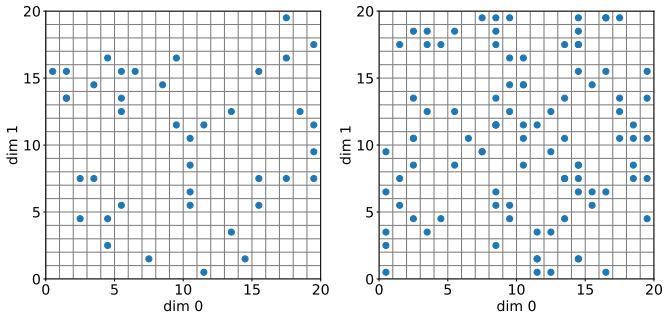
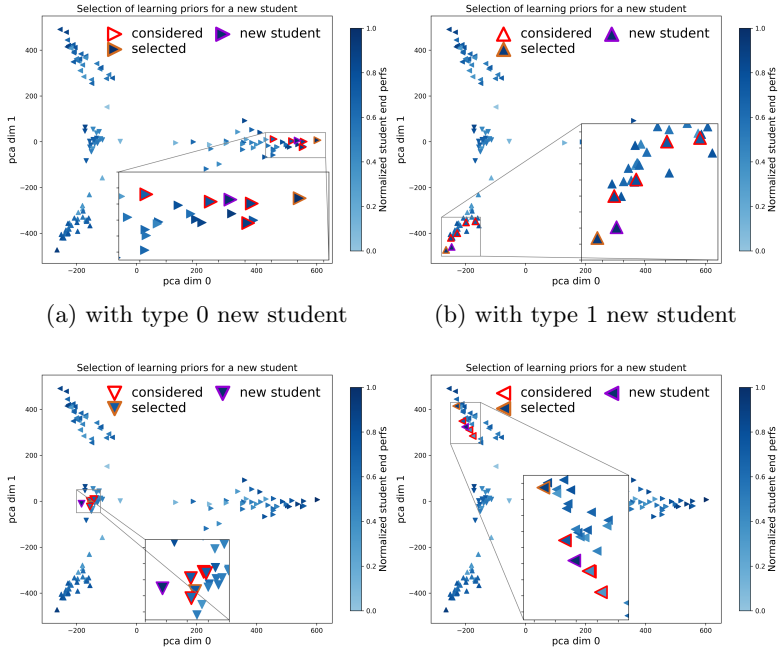


Figure C.1: Additional visualizations for the varying classroom size experiments (sec. 6.5.1). **Left:** Visualization of the starting cells of students from a 10% sample of a classroom of 400 students (one per student type) trained with ALP-GMM and used to populate the teaching history $\mathcal{H}^{\mathcal{S}}$. Each blue circle marks the starting cell of each student (i.e. its type) within the 2D parameter space $\mathcal{T}$, which is an initial learning subspace that needs to be detected by the teacher for successful training. **Right:**Visualization of the fixed set of 96 randomly drawn students that have to be trained by Meta-ACL variants given $\mathcal{H}^{\mathcal{S}}$. As not all student types are represented in $\mathcal{H}^{\mathcal{S}}$, Meta-ACL approaches have to generalize their curriculum generation to these new students.

C.4 Meta-ACL for DRL Students in the *Walker-Climber* environment

In this section we give additional details on the *Walker-Climber* environment presented in section 6.5.2, and we provide additional details and visualizations on the experiments that were performed on it.



(a) with type 0 new student (b) with type 1 new student

(c) with type 2 new student (d) with type 3 new student

Figure C.2: **Examples of student selection process in 4-student type toy environment.** In all figures, we plot the 2D PCA visualization of the CP^{pre} vectors (after pre-training) of the initial classroom (128 students) trained with ALP-GMM and used to populate the teaching history $\mathcal{H}^{\mathcal{S}}$ used by AGAIN variants in our 4-student type toy env experiments (see sec. 6.5.1). We then use these 4 figures to showcase the selection process happening in 4 different AGAIN-R runs (one per student type). Each triangle represents a student, whose ground truth type (i.e. its initial learning cell) is denoted by the orientation of the triangle. Given a new student to train, AGAIN pretrains the student, constructs its CP vector (purple border triangle), infers the k closest previously trained students from $\mathcal{H}^{\mathcal{S}}$ (red and golden border triangles), and use the one with highest end of training performance (i.e. highest score s, see sec. 6.4), denoted by a golden border triangle, to infer curriculum priors for the new student.

Details on the Walker-Climber environment. – In our experiments, we bound the wall spacing dimension of the task space to $\Delta_w = [0, 6]$, and the gate y position to $\mu_{gate} = [2.5, 7.5]$. In practice, a single task-encoding parameter tuple (μ_{gate}, Δ_w) encodes a stochastic task , since for each new wall along the track we add an independent Gaussian noise to each wall's gate y position μ_{gate}. Examples of *Walker-Climber* tasks randomly sampled within these bounds are available in figure C.3 (right). At the beginning of training a given DRL policy, the agent is embodied in either a bipedal walker morphology with two joints per legs or a two-armed climber morphology with 3-joints per arms ended by a grasping "hand". Both morphologies are controlled by torque. Climbers have an additional action dimension $g \in [-1, 1]$ used to grasp: if $g \in [-1, 0[$, the climber closes its gripper, and if $g \in]0, 1]$ it keeps it open. To avoid falling (which aborts the episode with a -100 penalty) while moving forward to collect rewards, climber agents must learn to swing themselves forward by successive grasp-and-release action sequences. To increase the diversity of the student distribution, we also randomize limb sizes. See figure C.3 (left) for examples of randomly sampled embodiments.

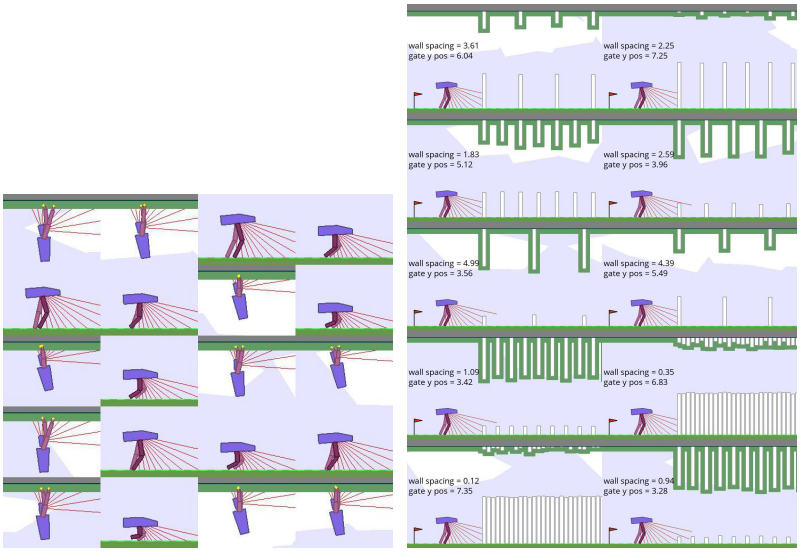


Figure C.3: Visualizations of the student space and the task space of the *Walker-Climber* environment.**Left:** Examples of possible agent embodiments (randomly set for a given DRL learner before training starts). **Right:** Examples of randomly sampled *Walker-Climber* tracks.

Soft Actor-Critic students – In our experiments, we use an implementation of Soft Actor-Critic provided by OpenAI[1] (MIT license). We use a 2 layered (400,300) network for V, Q1, Q2 and the policy. Gradient steps are performed each 10 environment steps, with a learning rate of 0.001 and a batch size of 1000. The entropy coefficient is set to 0.005.

Evaluation procedure – To report the performance of our students on the *Walker-Climber* environment, we use two separate test sets, one per embodiment type. For walkers we use a 100-tasks test set, uniformly sampled over a subspace of the task space with $\Delta_w \in [0, 6]$ and $\mu_{gate} \in [2.5, 3.6]$, which loosely corresponds to walking tracks from the test set of our Stump Tracks environment from chapter 4. For climbers, because there are no similar experiments in the literature (this work was parallel to the one presented in chapter 5, and since it is hard to infer beforehand what will be achievable by such a morphology, we simply use a uniform test set of 225 tasks sampled over the full task space. Importantly, the customized test set used for walkers is solely used for visualization purposes. In our AGAIN approaches, we pre-test all students with the expert-knowledge-free set of 225 tasks uniformly sampled over the task space.

Compute resources – Each of the 576 seeds required to reproduce our experiments (128 seeds for the classroom and 7×64 seeds for our 7 conditions) takes 36 hours on a single cpu. This amounts to around 21 000 cpu hours. Each run requires less than 1 GB of RAM.

[1]https://github.com/openai/spinningup

Visualizing student diversity. – To assess whether our proposed multi-modal distribution of possible students in the *Walker-Climber* environment do have diverse competence profiles (which is desirable as it creates a challenging Meta-ACL scenario), we plot the 2D PCA of the post training CP vector for each students of the initial classroom trained with ALP-GMM (used to populate $\mathcal{H}^S$). The result, visible in figure C.4 (top), shows that climber-students and walker-students are located in two independent clusters, i.e. they do have clearly different competence profiles. The spread of each of the clusters also demonstrates that variations in initial policy parameters and limb sizes also creates students with diverse learning potentials. The competence differences between walkers and climbers can also be seen in Figure C.4 (left and right), which shows the episodic reward obtained for each of the 225 tasks of the CP vector after training by a representative walker student (left) and climber student (right).

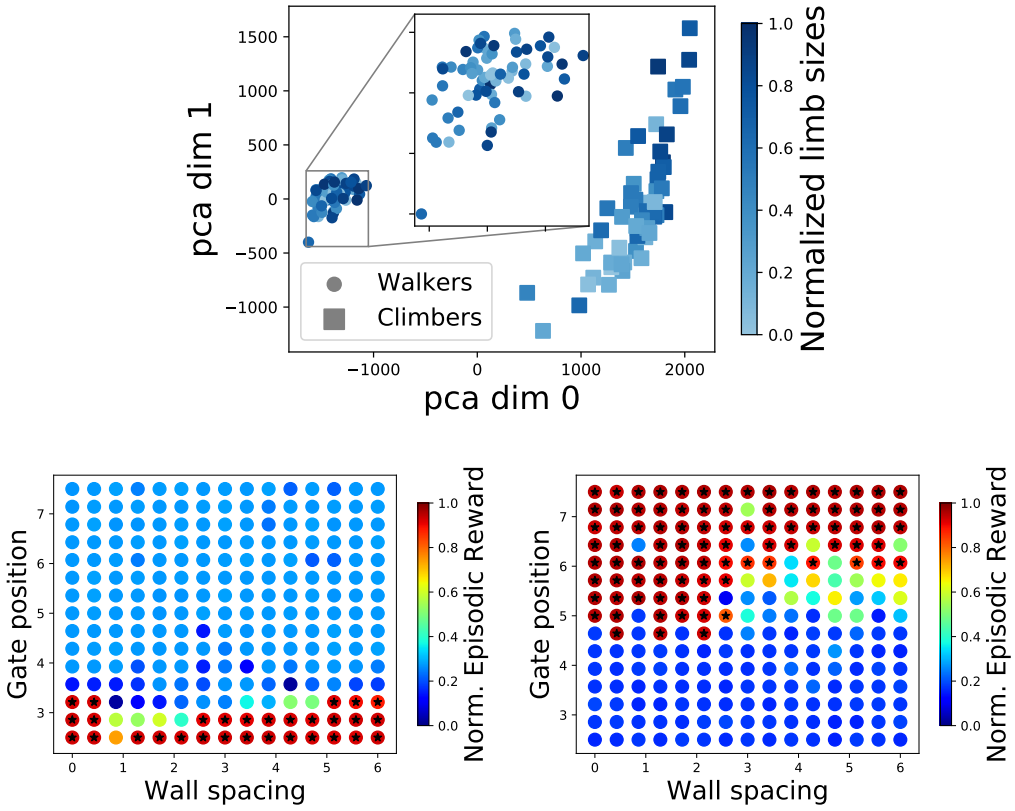


Figure C.4: **top:** PCA of classroom's CP vector (128 students) after being trained for 10M student steps with ALP-GMM. **left and right:** Episodic reward obtained for each task that compose the CP vector by a walker-student (left) and a climber-student (right) of this classroom. Stars are added for all tasks for which the agent obtained more than $r = 230$ (which corresponds to an efficient locomotion policy). Walkers only manage to learn tasks with very low gate positions while climbers learn only tasks with medium to high gate positions.

C.5 Applying Meta-ACL to a Single Student: Trying AGAIN instead of Trying Longer

In the following section we report all experiments on applying AGAIN variants to train a single DRL student (i.e. no history $\mathcal{H}^{\mathcal{S}}$), which is briefly presented in sec. 6.5.3.

Parametric BipedalWalker env. – We test our modified AGAIN variants along with baselines on teh *Stump Tracks* environment from chapter 4, which generates walking tracks paved with stumps whose height and spacing are defined by a 2D parameter vector used for the procedural generation of tasks. We keep the original bounds of this task space, i.e. we bound the stump-height dimension to $\mu_h \in [0, 3]$ and the stump-spacing dimension to $\delta_s \in [0, 6]$. As in chapter 4, we also test our teachers when the learning agent is embodied in a modified short-legged walker, which constitutes an even more challenging scenario (as the task space is unchanged, i.e. more unfeasible tasks). The agent is rewarded for keeping its head straight and going forward and is penalized for torque usage. The episode is terminated after 1) reaching the end of the track, 2) reaching a maximal number of 2000 steps, or 3) head collision (for which the agent receives a strong penalty). See figure C.5 for visualizations.

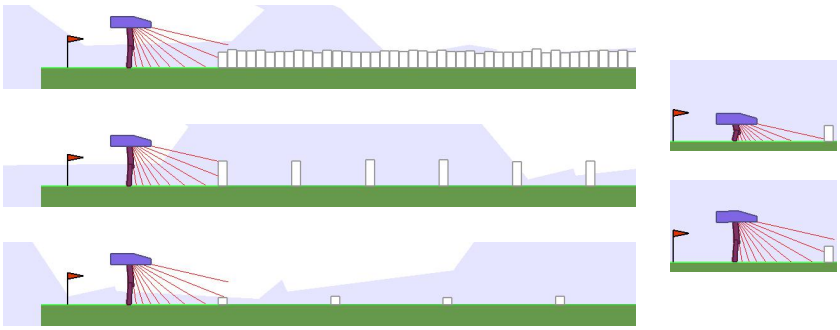


Figure C.5: Parameterized BipedalWalker environment. **Left:** Examples of generated tracks. **Right:** The two walker morphologies tested on the environment.

Results – To perform our experiments, we ran each condition for either 10Millions (IN and AGAIN variants) or 20Millions (others) environment steps (30 repeats). The preliminary ALP-GMM runs used in IN and AGAIN variants correspond to the first 10 Million steps of the ALP-GMM condition (whose end-performance after 20 Million steps is reported in table C.2. All teacher variants are tested when paired with a Soft-Actor Critic (Haarnoja et al., 2018a) student, with same hyperparameters as in the *Walker-Climber* experiments (see app. C.4). Performance is measured by tracking the percentage of mastered tasks (i.e. $r > 230$) from a fixed test set of 100 tasks sampled uniformly over the task space. We thereafter report results for 2 independent experiments done with either default walkers or short walkers.

Is re-training from scratch beneficial? - The end performances of all tested conditions are summarized in table C.2 (performance curves are available in figures C.6 and C.7). Interestingly, retraining the DRL agent from scratch in the second run gave superior end

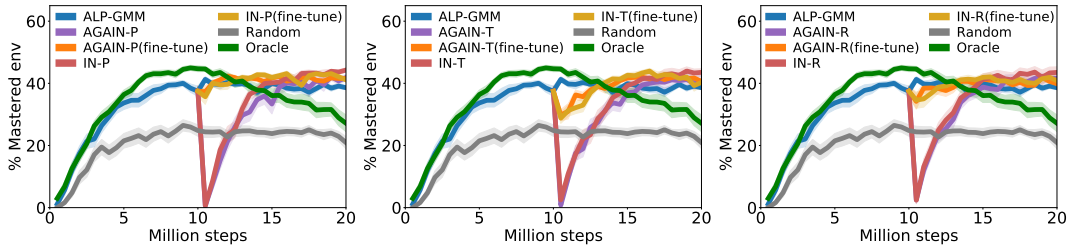
performances than fine-tuning using the weights of the first run *in all tested variants*. This showcases the brittleness of gradient-based training and the difficulty of transfer learning. Despite this, even fine-tuned variants reached superior end-performances than classical ALP-GMM, meaning that the change in curriculum strategy in itself is already beneficial.

Is it useful to re-use ALP-GMM in the second run? - In the default walker experiments, AGAIN-R, T and P conditions mixing ALP-GMM and IN in the second run reached lower mean performances than their respective IN variants. However, the exact opposite is observed for IN-R and IN-T variants in the short walker experiments. This can be explained by the difficulty of short walker experiments for ACL approaches, leading to 16/30 preliminary 10M steps long ALP-GMM runs to have a mean end-performance of 0, compared to 0/30 in the default walker experiments. All these run failures led to many GMMs lists X used in IN to be of very low-quality, which illustrates the advantage of AGAIN that is able to emancipate from IN using ALP-GMM.

Highest-performing variants. - Consistently with the precedent analysis, mixing ALP-GMM with IN in the second run is not essential in default walker experiments, as the best performing ACL approach is IN-P. This most likely suggests that the improved adaptability of the curriculum when using AGAIN is outbalanced by the added noise (due to the low task-exploration). However in the more complex short walker experiments, mixing ALP-GMM with IN is essential, especially for AGAIN-R, which substantially outperforms ALP-GMM and other AGAIN and IN variants (see fig. 6.7), reaching a mean end performance of 19.0. The difference in end-performance between AGAIN-R and Oracle, our hand-made expert using privileged information who obtained 20.1, is not statistically significant ($p = 0.6$).

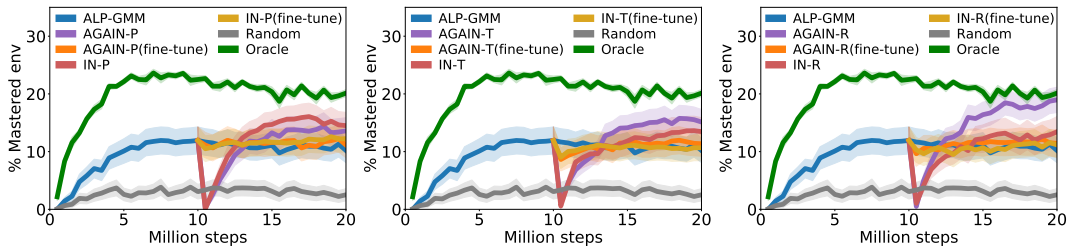
Condition	Short walker	Default walker
AGAIN-R	$19.0 \pm 12.0^{*}$	$41.6 \pm 6.3^{*}$
AGAIN-R (fine-tune)	11.4 ± 12.9	39.9 ± 4.6
IN-R	13.4 ± 14.4	$43.5 \pm 9.6^{*}$
IN-R (fine-tune)	11.2 ± 12.3	40.8 ± 5.6
AGAIN-T	15.1 ± 11.9	40.6 ± 11.5
AGAIN-T (fine-tune)	11.4 ± 11.8	$40.6 \pm 3.8^{*}$
IN-T	13.5 ± 13.3	$43.5 \pm 6.1^{*}$
IN-T (fine-tune)	10.7 ± 12.3	40.3 ± 7.6
AGAIN-P	13.6 ± 12.5	$41.9 \pm 5.1^{*}$
AGAIN-P (fine-tune)	11.1 ± 12.0	$41.5 \pm 3.9^{*}$
IN-P	14.5 ± 12.6	$\mathbf{44.3} \pm 3.5^{*}$
IN-P (fine-tune)	12.2 ± 12.5	$41.1 \pm 3.8^{*}$
ALP-GMM	10.2 ± 11.5	38.6 ± 3.5
Oracle	$\mathbf{20.1} \pm 3.4^{*}$	$27.2 \pm 15.2^{-}$
Random	$2.5 \pm 5.9^{-}$	$20.9 \pm 11.0^{-}$

Table C.2: **Experiments on Parametric BipedalWalker** The avg. perf. with std. deviation after 10 Millions steps (IN and AGAIN variants) or 20 Million steps (others) is reported (30 seeds). For IN and AGAIN we also test variants that do not retrain the weights of the policy used in the second run *from scratch* but rather *fine-tune* them from the preliminary run. $^{*/-}$ Indicates whether perf. difference with ALP-GMM is statistically significant ie. $p < 0.05$ in a post-training Welch's student t-test (* for performance advantage w.r.t ALP-GMM and $^{-}$ for perf. disadvantage).



(a) **Pool-based** IN (b) **Time-based** IN (c) **Reward-based** IN

Figure C.6: **Evolution of performance across 20M environment steps of each condition with default bipedal walker.** Each point in each curve corresponds to the mean performance (30 seeds), defined as the percentage of mastered tracks (ie. $r > 230$) on a fixed test set. Shaded areas represent the standard error of the mean. Consistently with chapter 4, which implements a similar approach, Oracle is prone to forgetting with default walkers due to the strong shift in task subspace (which is why it is not the best performing condition for default walker experiments.



(a) **Pool-based** IN (b) **Time-based** IN (c) **Reward-based** IN

Figure C.7: **Evolution of performance across 20M environment steps of each condition with short bipedal walker.** Each point in each curve corresponds to the mean performance (30 seeds), defined as the percentage of mastered tracks (ie. $r > 230$) on a fixed test set. Shaded areas represent the standard error of the mean.

Appendix D

SocialAI

Contents

D.1 Environment Details

D.1.1 Action Space

The action space of our environments consists of two modalities, (*primitive actions* and *language*), which results in a 3D discrete action vector.

The first dimension corresponds to the primitive action modality, which are identical to actions available in minigrid (Chevalier-Boisvert et al., 2018), from which our code is based on. It consists of 7 actions (turn left, turn right, move forward, pickup, drop, toggle, done). In all the environments *pickup* and *drop* actions do not do anything and *done* terminates the episode. We kept these actions as we intend to use them in future iterations of the benchmark. In TalkItOut, Dance, and CoinThief *toggle* terminates the episode with 0 reward. In DiverseExit, ShowMe and Help *toggle* opens doors and presses buttons. In DiverseExit, it can also be used to *poke* the NPC. In Dance and CoinThief,

only a subset of 3 primitive actions are available (to simplify these environments): *turn left*, *turn right* and *move forward*. In SocialEnv, all actions behave as in the original environment that is sampled for each new episode. The *second* and *third* dimensions regard the language modality. The second dimension selects a template and the third a noun. The full grammar for each environment is shown in table D.1. In SocialEnv all grammars are merged to a single one containing all templates and nouns used in all single environments. Both modalities can also be undefined, i.e. a no-op action is taken. Examples of action vectors are shown in table D.2.

D.1.2 Observation Space

The multimodal observation space consists of the *vision* modality and the *language* modality.

The *vision* modality is manifested as a 7×7 grid displaying the space in front of the agent (shown as highlighted grids in figure 7.1). Each location of this grid is encoded as 4 integers for the object type, color, status and orientation (used for NPCs). *status* is used to refer to object states (e.g. door is *open*) or, if the object is an NPC, it is used to inform about the NPC type (e.g. *wizard* NPC. For example, a blue wizard NPC facing down will be encoded as $(11, 2, 0, 1)$ and a blue guide NPC facing up will be encoded as $(11, 2, 1, 3)$.

The *language* modality is represented as a string containing the currently heard utterances, i.e. utterances uttered by NPCs next to the agent, and their names (ex. "John: go to the green door"). In case of silence, an "empty indicator" symbol is used. As it is often more convenient to concatenate all the utterances heard, to simplify the implementation of the agent, the implementation of the environment also supports giving the full history of heard utterances with the "empty indicator" symbols removed as additional information.

D.1.3 Reward

In all of our environments, the extrinsic reward is given upon completing the task. The reward is calculated by the following equation:

$$r_{extr} = 1.0 - 0.9 * \frac{t}{t_{max}} \tag{D.1}$$

, where t is the number of steps the agent made in the environment and t_{max} is the maximum allowed number of steps.

D.1.4 TalkItOut

This environment consists of three NPCs and four doors, and the goal of the agent is to exit the room using the correct (one out of four) door in $t_{max} = 100$ steps. The agent can find out which door is the correct one by asking the true guide. To find out which guide is the correct one, the agent has to ask the wizard. Before talking to any NPC, the

Table D.1: Template-based grammars for all the environments. An utterance is constructed by selecting a template (top table) and a noun (bottom table).

Action	Template			
	TalkItOut, DiverseExit	CoinThief	Dance, ShowMe, Help	SocialEnv
0	Where is <noun>	Here is <noun>	Move your <noun>	Where is <noun>
1	Open <noun>		Shake your <noun>	Open <noun>
2	Which is <noun>			Close <noun>
3	How are <noun>			How are <noun>
4				Move your <noun>
5				Shake your <noun>
6				Here is <noun>
7				Which is <noun>

Action	Noun				
	TalkItOut	DiverseExit	CoinThief	Dance, ShowMe, Help	SocialEnv
0	sesame	sesame	1	body	sesame
1	the exit	the exit	2	head	the exit
2	the wall	the correct door	3		the wall
3	you	you	4		the floor
4	the ceiling	the ceiling	5		the ceiling
5	the window	the window	6		the window
6	the entrance	the entrance			the entrance
7	the closet	the closet			the closet
8	the drawer	the drawer			the drawer
9	the fridge	the fridge			the fridge
10	oven	oven			oven
11	the lamp	the lamp			the lamp
12	the trash can	the trash can			the trash can
13	the chair	the chair			the chair
14	the bed	the bed			the bed
15	the sofa	the sofa			the sofa
16					the correct door
16					1
17					2
18					3
19					4
20					5
21					6
22					body
23					head

Table D.2: Examples of various actions in the environment. Second and third dimension must both either be defined or not.

Action	description
(1, -, -)	moves left without speaking
(1, 1, 5)	moves left and utters "Open the window"
(-, 1, 5)	doesn't move but utters "Open the window"
(-, -, -)	nothing happens

agent has to stand in front of it and introduce himself by saying "How are you?". Upon finding out which door is the correct one, the agent has to stand in front of it and utter "Open sesame". Then the episode ends, and the reward is given. If the agent executes *done*, *toggle* or utters "Open sesame" in front of the wrong door the episode ends with no reward. An example of a dialogue that might appear in a successful episode is shown in table D.3

Table D.3: An example of a successful episode in the TalkItOut environemnt

True guide: John
Correct door color: blue
agent goes to the wizard
Agent: How are you?
Wizard: I am fine.
Agent: Where is the exit?
Wizard: Ask John.
agent goes to one guide
Agent: How are you?
Jack: I am fine.
Agent: Where is the exit?
Jack: Go to the red door.
agent goes to the other guide
Agent: How are you?
John: I am fine.
Agent: Where is the exit?
John: Go to the blue door.
agent goes to the blue door
Agent: Open sesame

For each episode the colors of doors and NPCs are selected randomly from a set of six and the names of the two guides are selected randomly from a set of two (Jack, John), i.e. in one episode either Jack or John will be the truth speaking guide and the other

will be the lying guide. Furthermore, the grid width and height are randomized from the minimal size of 5 up to 8 and the NPCs and the agent are placed randomly inside (omitting locations in front of doors).

Required Social Skills

In the remainder of this section, we use the TalkItOut environment to provide an in-depth example of the detailed list of social skills required in one of our environments (we revert to more straightforward descriptions for all subsequent environment descriptions).

Intertwined multimodality – To solve TalkItOut the agent must use both modalities both in the action and in the observation space. Furthermore, this multimodality is intertwined because the progression in which the modalities are used is non-trivial. To discuss this notion further, let's imagine an example of instruction following. The progression of modalities here is trivial because the agent always *listens* for the command first and then *looks* and *moves/acts* to complete the task. Another good example is embodied question answering. Here the agent again always first *listens* to the question, then *looks* and *moves* in the environment to finally, at the end, *speak* the answer.

In our environment, however, the agent must always choose which modality to use based on the current state. Furthermore, it will often be required to switch between modalities many times. For example, to talk to an NPC the agent first *looks* to find the NPC, then it *moves* to the NPC, finally the agent *speaks* to it and *listens* to the response. This progression is then used, if needed, for other NPCs, and finally a similar one used to go to the correct door and open it. Furthermore, depending on the current configuration of environment, the progression can also be different. Usually, after finding out the correct door the agent needs to *look* for it and *move* to it to *speak* the password, but if the true guide is already next to the correct door only *looking* for the door and *speaking* the password is required.

Theory of Mind – Since the agent must be able to infer good or bad intentions of other NPCs, a basic form of ToM is needed. Primarily, the agent needs to infer that the wizard is well-intended, wants to help, and is therefore trustworthy. Using the inferred trust in the wizard, it is possible to infer the good intentions of the true guide, and likewise the bad intentions of the false guide. On the other hand, as the false guide chooses which false direction to give each time asked, it is also possible to infer its ill-intentions by asking him many times in the same episode and observing this inconsistency. If an NPC gives different answers for the same question in the same episode, then it is evident its intentions are bad.

Pragmatic frames – Pragmatic frames were not the focus of this environment, and are studied in more detail in other environments, they are present in this environment only in a simple form. To talk with an NPC the agent needs to stand next to it and introduce itself by saying "how are you", and to get an answer the agent needs to ask "where is the exit". These simple rules (a.k.a. social conventions) are pragmatic frames, i.e. grammars describing possible and impossible interactions. For example, it is impossible

to communicate if you are far and get directions if you ask "Where is the floor". The agent needs to be able to extract these rules and use them in relation to all NPCs.

D.1.5 Dance

A Dancer NPC demonstrates a 3-steps dance pattern (randomly generated for each episode) and then asks the agent to reproduce this dance. Each dance step is composed of a primitive action, randomly selected among rotating left, right, or moving forward. 50% of the time, a randomly selected utterance among 4 possible ones (see table D.1) is also performed simultaneously with the primitive action. In the first step of each episode, the NPC utters "Look at me!". It then performs the dance in the next 3 steps. Finally, at the fifth step, the NPC utters "Now repeat my moves", and starts to record the agents' actions. In contrary to TalkItOut, the agent does not need to be close to the NPC to interact with it (i.e. both are "shouting"). To solve *Dance*, the agent must reproduce the full dance sequence. Multiple trials are authorized within the $t_{max} = 20$ steps of an episode. Only trials performed after the NPC completed his dance are recorded.

The Dance environment requires the agent to be able to infer that the NPC is setting up a *teaching pragmatic frame* ("Look at me!" + `do_dance_performance` + "Now repeat my moves!"), requiring the agent to *imitate* a social peer, process *multi-modal observations* and produce *multi-modal actions*.

D.1.6 CoinThief

In a room containing 6 coins (randomly placed), a Thief NPC spawns near the agent, i.e. in one of the 4 cells adjacent to the agent (selected randomly for each new episode). At step 0, the Thief NPC utters "Freeze! Give me all your coins!". The agent can "give coins" by uttering "here is $< nb >$", with $< nb >$ ranging from 0 to 6 (see table D.1. Note that the agent does not need to collect coins by navigating within the environment, it only has to utter. To obtain a positive reward, the agent must give exactly the number of coins *that the thief can see*. The thief field of view is a 5×5 square, i.e. a smaller version than the agent's. In addition of its initial orientation facing the agent, the thief also "look around" in another direction, either left or right (selected at random for each episode). Episodes are aborted without reward if the agent use the *move forward* action (the thief wants the agent not to move), or if the maximum number of steps ($t_{max} = 20$) is reached. Solving the CoinThief environment requires *Theory of Mind* as the agent must understand that the thief holds *false belief* over the agent's total number of coins and must infer how many coins he actually sees. To infer how many coins the thief sees, the agent must learn the thief's field of view and use memory to remember the thief's two view directions.

D.1.7 DiverseExit

The goal of the agent is to exit the room using one of four doors in $t_{max} = 50$ steps. One NPC is present in the environment. Colors and the initial positions of the NPC and the doors are randomized each episode. To find out which door is the correct one, the

agent has to ask the NPC. To talk to the NPC, the agent has to introduce himself by saying one of two possible *introductory utterances* ("Where is the exit?" or "Which is the correct door?").There are 12 different NPC types, one of which is randomly selected to be present in the episode. Each of the 12 NPCs prefer to be introduced to differently. More precisely, when the agent utters one of the two *introductory utterances* for the first time in the episode, the *introductory configuration* is saved. The *introductory configuration* is manifested as the tuple of the following four elements: (is the agent next to the NPC, was the NPC poked, is eye contact established, which *introductory utterance* was used). Each NPC must be asked with its preferred *introductory configuration*. This enables us to create twelve different NPC and their corresponding *introductory configurations*. Those twelve configurations are listed in the table D.4. If the *introductory configuration* is the one corresponding to the present NPC, the NPC will give the agent the directions (ex. "go to the green door") every time they establish eye contact. However, if the *introductory configuration* was not the right one, the NPC will not give the directions in this episode (a once saved introductory state cannot be overwritten in the same episode).

To solve *DiverseExit*, the agent must learn a large diversity of different frames (12). Furthermore, it must learn to differentiate between them and infer which frame to use with which NPC.

Table D.4: Twelve *introductory configurations* corresponding to twelve possible different NPCs in DiverseExit.

npc_type	is next to the NPC	was the NPC poked	eye contact	required intro utterance
0	next to	poked	Yes	"Where is the exit"
1	next to	not poked	Yes	"Where is the exit"
2	not next to	not poked	Yes	"Where is the exit"
3	next to	poked	Yes	"Which is the correct door"
4	next to	not poked	Yes	"Which is the correct door"
5	not next to	not poked	Yes	"Which is the correct door"
6	next to	poked	No	"Where is the exit"
7	next to	not poked	No	"Where is the exit"
8	not next to	not poked	No	"Where is the exit"
9	next to	poked	No	"Which is the correct door"
10	next to	not poked	No	"Which is the correct door"
11	not next to	not poked	No	"Which is the correct door"

D.1.8 ShowMe

The goal of the agent is to exit the room through the door placed at the top wall of the environment in $t_{max} = 100$ steps. At the beginning of the episode the door is locked and can be unlocked by activating the correct switch (one out of three) on the bottom wall. The switches can be activated using the *toggle* action, however once a switch is activated it cannot be deactivated. This means that the agent must press the correct switch from the first try. The information about which switch is the correct one can be inferred by looking at the NPC. Once eye contact is established with the NPC, the NPC

will say "Look at me" and proceed to press the switch and exit the room. After this, the switches are reset and the door is locked once again. As the switch doesn't change whether it's activated or not (it looks the same) the agent must infer what switch was pressed by looking at the NPCs movement and infer which switch was activated. The reward is given once both the NPC and the agent have left the room, i.e. if the agent leaves the room before the NPC no reward is given.

To solve ShowMe, the agent must learn to imitate the NPCs goals (pressing the correct switch) from its behavior (a facet of ToM). Furthermore, it has to infer the *Teaching pragmatic frame* where the *slot* is the pressed button.

D.1.9 Help

The environment consists of two roles: The Exiter and the Helper. The shared goal of both the participants it for the Exiter to exit the environment in $t_{max} = 20$ steps. It can do so using any of the two doors on the right wall of the environment. At the beginning of the episode both doors are locked and can be unlocked by pressing the corresponding switch on the left wall. The environment is separated by a wall of lava in the middle, disabling movement from the left to the right side of the room. The Exiter is placed on the right side of the environment (next to the doors) and the Helper on the left side (next to the switches). As the episode ends without reward if both switches are pressed, the two participants have to agree on which door to use.

The purpose of this environment is to train the agent in the Exiter role and test in the Helper role. When the agent is in the Exiter role (training phase), the NPC is in the Helper role. Then the NPC acts as follows. It moves towards the switch corresponding to the door that is the closest to the agent. Once in front of the switch, it looks at the agent and waits for eye contact. Once eye contact has been established, the NPC activates the switch. The agent, therefore, needs to learn to choose a door and confirm this choice by establishing eye contact. When the agent is in the Helper role (testing phase), the NPC is in the role of the Exiter. Then the NPC chooses a door and moves in front of it. Once there, it looks at the agent and waits for eye contact. Once eye contact has been established, the NPC attempts to exit the room by the door.

To solve Help, the agent must learn the whole pragmatic frame just from seeing its own perspective of it. It must learn to infer the shared goal and which actions from both the agent and the NPC lead to the achievement of this goal.

D.1.10 SocialEnv

SocialEnv is a *meta-environment*, i.e. it is a multi-task environment in which, for each new episode, the agent is randomly spawned into one of the 6 previously discussed environments. The agent's grammar is a set of all previous grammars (see table D.1). t_{max} is set to the original t_{max} of each environment.

Solving SocialEnv requires to infer what is the current social scenario (i.e. the current environnment) the agent is spawned in. This can be achieved by leveraging pragmatic information collected through interaction, i.e. differentiating environments from social

interaction footprints. For instance, a proficient agent could reliably detect it is in the TalkItOut environment by observing that there are 3 NPCs (1 wizard-type and 2 guide-type). Given that this environment detection is mastered, the agent still has to be proficient in *all of the core social skills* we proposed, to be able to solve each environment.

D.2 Experimental Details

D.2.1 Baselines Details

PPO baseline – In this work we use a PPO-trained (Schulman et al., 2017) DRL architecture initially designed for the BabyAI benchmark (Chevalier-Boisvert et al., 2019). More precisely, we used the improved policy architecture presented in their follow-up technical report Hui et al. (2020) (their *original_endpool_res* model). See figure D.1 for a visualization of the complete architecture. First, symbolic pixel grid observations are fed into two convolutional layers (LeCun et al., 1989; Krizhevsky et al., 2012) (3×3 filter, stride and padding set to 1), while dialogue inputs are processed using a Gated Recurrent Unit layer (Chung et al., 2015). The resulting image and language embeddings are combined using two FiLM attention layers (Perez et al., 2018). Max pooling is performed on the resulting combined embedding before being fed into an LSTM (Hochreiter & Schmidhuber, 1997) with a $128D$ memory vector. The LSTM embedding is then used as input for the navigation action head, which is a two-layered fully-connected network with tanh activations and has an 8D output (i.e. 7 navigation actions and no_op action).

In order for our agent to be able to both move and talk, we add to this architecture a talking action head, which is composed of three networks. All of them are two-layered, fully-connected networks with tanh activations, and take the LSTM's embedding as input. The first one is used as a switch: it has a one-dimensional output to choose whether the agent talks (output > 0.5) or not (output < 0.5). If the agent talks, the two other networks are used to respectively sample the template and the word.

Note that the textual input given to the agent consists of the full dialogue history (without the "empty string" indicator) as we found it to work better than giving only current utterances (it simplifies memory requirements).

Table D.5: Training hyperparameters

Hyperparameter	value
learning rate	$1e^4$
GAE λ	0.99
clip ϵ	$1e^5$
batch size	1280
γ	0.99
recurrence	10
epochs	4

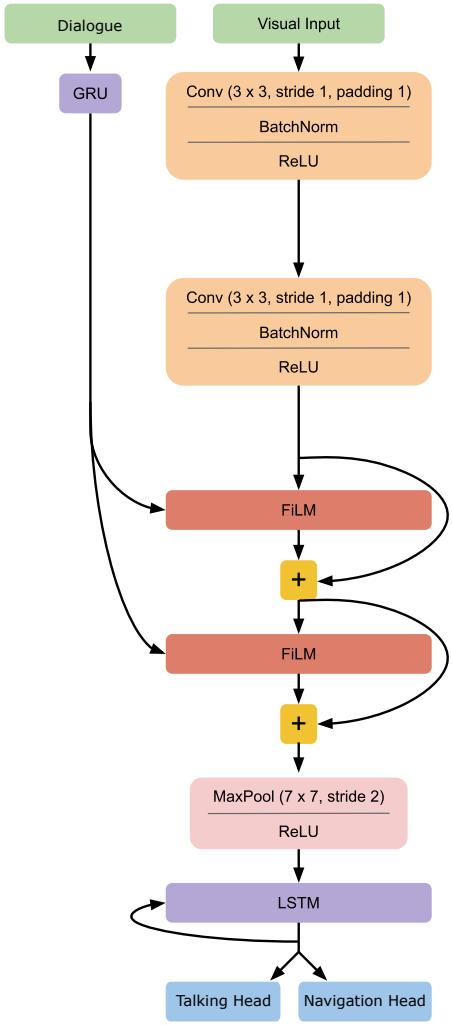


Figure D.1: Our Multi-Headed PPO baseline DRL agent. Architecture visualization is a modified version of the one made by Hui et al. (2020). We perform two modifications: 1) Instead of fixed instruction inputs our model is fed with NPC's language outputs (if the agent is near an NPC), and 2) We add a language action head, as our agent can both navigate and talk.

Table D.6: Exploration bonus hyperparameters

Hyperparameter	value						
	TalkItOut	Dance	CoinThief	DiverseExit	ShowMe	Help	SocialEnv
type	lang	vision	vision	lang	vision	vision	vision
T	0.6	0.6	0.6	0.6	0.6	0.6	0.6
C	7	3	2	20	3	3	2
M	50	50	50	50	50	50	50

Exploration bonus – The exploration bonus we use is inspired by recent works in intrinsically motivated exploration (Pathak et al., 2017; Savinov et al., 2019; Tang et al., 2017). These intrinsic rewards estimate the novelty of the currently observed state and add the novelty based bonus to the extrinsic reward. The novelty is estimated by counting various aspects of the state. We make our reward episodic by resetting the counts at the end of each episode.

In this work we study two different techniques for computing the exploration bonus (counting), and we use the one that was more suitable for a given environment. Which reward was used for which environment and the corresponding hyperparameters are visible in table D.6. The two different techniques are: language-based and vision-based.

In the language-based intrinsic reward we count how many times was each utterance observed and compute an additional bonus based on the following equation:

$$r_{intr} = tanh(\frac{C}{(N(s_{lang}) + 1)^M) * T)}) \tag{D.2}$$

, where M, C, and T are hyperparameters and $N(s_{lang})$ is the number of times the utterance s_{lang} was observed during this episode. In the current version of the environment the agent cannot hear his own utterances and the NPCs speak only when spoken to. Therefore, this exploration bonus can be seen as analogous to social influence (Jaques et al., 2019) in the language modality, as the reward is given upon *making the NPC respond*.

In the vision-based intrinsic reward, we reward the agent for seeing a new encoding. An encoding is the 4D representation of a grid (object_type, color, additional_information, orientation). At each step, a set of encountered encodings is created by removing the duplicates, and then the reward computed by the following equation:

$$r_{intr} = tanh(\sum_{e \in E(s)} \frac{C}{(N(e) + 1)^M) * T}) \tag{D.3}$$

, where M, C, and T are as in equation D.2, $E(s)$ is a set of unique encodings visible in state s, and $N(e)$ is the number of times an encoding e was encountered.

D.2.2 Computational Resources

To perform our experiments, we used a slurm-based cluster. Producing our final results require to run 16 seeds of 3 different conditions on each of our 10 environments (7 environments and 3 modified environments for our case-studies), i.e. 480 seeds. Each of these experiments takes approximately 42 hours on one CPU and one 32GB Tesla V100 GPU (one GPU can serve 4 experiments in parallel), which amounts to 20160 CPU hours and 5040 GPU hours.

Bibliography

Abramson, J., Ahuja, A., Brussee, A., Carnevale, F., Cassin, M., Clark, S., Dudzik, A., Georgiev, P., Guy, A., Harley, T., Hill, F., Hung, A., Kenton, Z., Landon, J., Lillicrap, T. P., Mathewson, K., Muldal, A., Santoro, A., Savinov, N., Varma, V., Wayne, G., Wong, N., Yan, C., and Zhu, R.
Imitating interactive intelligence.
arXiv, abs/2012.05672, 2020.

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O.
Grounding Language to Autonomously-Acquired Skills via Goal Generation.
In *ICLR 2021 - Ninth International Conference on Learning Representation*, Vienna / Virtual, Austria, May 2021.

Akalin, N. and Loutfi, A.
Reinforcement learning approaches in social robotics.
Sensors, 21(4), 2021.

Ammanabrolu, P., Urbanek, J., Li, M., Szlam, A., Rocktäschel, T., and Weston, J.
How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds.
In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 807–833, Online, June 2021. Association for Computational Linguistics.

Anderson, C. W.
Learning and Problem-Solving with Multilayer Connectionist Systems.
PhD thesis, University of Massachusetts Amherst, 1986.
AAI8701132.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W.
Hindsight experience replay.
In *NeurIPS*, 2017.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B.
A survey of robot learning from demonstration.
Robotics and Autonomous Systems, 57(5):469–483, 2009.

Arjovsky, M., Chintala, S., and Bottou, L.
Wasserstein GAN.
arXiv:1701.07875 [cs, stat], December 2017.
arXiv: 1701.07875.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A.
Deep reinforcement learning: A brief survey.
IEEE Signal Processing Magazine, 34(6):26–38, 2017.

Asada, M., MacDorman, K. F., Ishiguro, H., and Kuniyoshi, Y.
Cognitive developmental robotics as a new paradigm for the design of humanoid robots.
Robotics and Autonomous systems, 37(2-3):185–193, 2001.

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., and Yoshida, C.
Cognitive developmental robotics: A survey.
IEEE Transactions on Autonomous Mental Development, 1(1):12–34, 2009.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E.
The nonstochastic multiarmed bandit problem.
SIAM journal on computing, 32(1):48–77, 2002.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, S. A., Kohli, P., and Grefenstette, E.
Learning to understand goal specifications by modelling reward.
In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Baillargeon, R., Scott, R. M., and He, Z.
False-belief understanding in infants.
Trends in Cognitive Sciences, 14(3):110–118, 2010.

Bain, M. and Sammut, C.
A framework for behavioural cloning.
In *Machine Intelligence 15*, pp. 103–129, 1995.

Baird, L.
Reinforcement learning in continuous time: advantage updating.
In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 4, pp. 2448–2453 vol.4, 1994.

Baker, B., Kanitscheider, I., Markov, T. M., Wu, Y., Powell, G., McGrew, B., and Mordatch, I.
Emergent tool use from multi-agent autocurricula.
In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Baldassarre, G. and Mirolli, M.
Intrinsically Motivated Learning in Natural and Artificial Systems.
Springer, 2013.

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I.
Emergent complexity via multi-agent competition.
In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Baranes, A.
Motivations intrinsèques et contraintes maturationnelles pour l'apprentissage sensori-moteur.
PhD thesis, Bordeaux 1, 2011.

Baranes, A. and Oudeyer, P.
R-IAC: robust intrinsically motivated exploration and active learning.
IEEE Trans. Autonomous Mental Development, 1(3):155–169, 2009.

Baranes, A. and Oudeyer, P.-Y.
Intrinsically motivated goal exploration for active motor learning in robots: A case study.
In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1766–1773. IEEE, 2010a.

Baranes, A. and Oudeyer, P.-Y.
Maturationally-constrained competence-based intrinsically motivated learning.
In *2010 IEEE 9th International Conference on Development and Learning*, pp. 197–203. IEEE, 2010b.

Baranes, A. and Oudeyer, P.-Y.
The interaction of maturational constraints and intrinsic motivations in active motor development.
In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pp. 1–8. IEEE, 2011.

Baranes, A. and Oudeyer, P.-Y.
Active learning of inverse models with intrinsically motivated goal exploration in robots.
Robot. Auton. Syst., 2013.

Barros, P., Tanevska, A., Cruz, F., and Sciutti, A.
Moody learners-explaining competitive behaviour of reinforcement learning agents.
In *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 1–8. IEEE, 2020.

Barto, A. G., Singh, S., Chentanez, N., et al.
Intrinsically motivated learning of hierarchical collections of skills.
In *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19. Piscataway, NJ, 2004.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.
Relational inductive biases, deep learning, and graph networks.
arXiv preprint arXiv:1806.01261, 2018.

Baxter, J., Tridgell, A., and Weaver, L.
Reinforcement learning and chess.
In *Machines that learn to play games*, pp. 91–116. Nova Science Publishers, 2001.

Bazhydai, M., Twomey, K., and Westermann, G.
Curiosity and Exploration, pp. 460–470.
Elsevier, 2nd ed. edition, March 2021.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S.
Deepmind lab.
arXiv, abs/1612.03801, 2016.

Beauchamp, M. and Anderson, V.
Social: An integrative framework for the development of social skills.
Psychological bulletin, 136:39–64, 01 2010.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R.
Unifying count-based exploration and intrinsic motivation.
In *NeurIPS*, 2016.

Bellemare, M., Candido, S., Castro, P., Gong, J., Machado, M., Moitra, S., Ponda, S., and Wang, Z.
Autonomous navigation of stratospheric balloons using reinforcement learning.
Nature, 588:77–82, 12 2020.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M.
The arcade learning environment: An evaluation platform for general agents.
Journal of Artificial Intelligence Research, 47:253–279, Jun 2013.

Bellemare, M. G., Dabney, W., and Munos, R.
A distributional perspective on reinforcement learning.
In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 06–11 Aug 2017.

Bellman, R.
Dynamic programming.
Princeton University Press, 1957.

Bender, E. M. and Koller, A.
Climbing towards NLU: On meaning, form, and understanding in the age of data.
In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5185–5198, Online, July 2020. Association for Computational Linguistics.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S.
On the dangers of stochastic parrots: Can language models be too big?
In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J.
Curriculum learning.
In *ICML*, 2009.

Benson, J. B.
Encyclopedia of infant and early childhood development.
Elsevier, 2020.

Bentley, J. L.
Multidimensional binary search trees used for associative searching.
Commun. ACM, 18(9):509–517, 1975.

Benureau, F. C. and Oudeyer, P.-Y.
Behavioral diversity generation in autonomous exploration through reuse of past experience.
Frontiers in Robotics and AI, 3:8, 2016.

Berlyne, D.
Conflict, Arousal and Curiosity.
McGraw-Hill, 1960.

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

Berthouze, L. and Kuniyoshi, Y.
Emergence and categorization of coordinated visual behavior through embodied interaction.
Autonomous Robots, 5(3):369–379, 1998.

Billard, A. and Dautenhahn, K.
Grounding communication in autonomous robots: An experimental study.
Robotics and Autonomous Systems, 24(1):71 – 79, 1998.
Scientific Methods in Mobile Robotics.

Billard, A. and Dautenhahn, K.
Experiments in learning by imitation - grounding and use of communication in robotic agents.
Adaptive Behavior, 7(3-4):415–438, 1999.

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A., and et al.
Experience grounds language.
Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.

Bjorklund, D. F.
The role of immaturity in human development.
Psychological bulletin, 122(2):153, 1997.

Blank, D., Kumar, D., Meeden, L., and Marshall, J. B.
Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture.
Cybernetics and Systems: An International Journal, 36(2):125–150, 2005.

Bozdogan, H.
Model selection and akaike's information criterion (aic): The general theory and its analytical extensions.
Psychometrika, 52(3):345–370, Sep 1987.

Breazeal, C. and Aryananda, L.
Recognition of affective communicative intent in robot-directed speech.
Autonomous robots, 12(1):83–104, 2002.

Breazeal, C. and Scassellati, B.
Robots that imitate humans.
Trends in cognitive sciences, 6(11):481–487, 2002.

Breazeal, C., Brooks, A., Gray, J., Hoffman, G., Kidd, C., Lee, H., Lieberman, J., Lockerd, A., and Chilongo, D.
Tutelage and collaboration for humanoid robots.
International Journal of Humanoid Robotics, 1(02):315–348, 2004.

Breazeal, C., Dautenhahn, K., and Kanda, T.
Social robotics.
Springer handbook of robotics, pp. 1935–1972, 2016.

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

Brooks, R., Breazeal, C., Marjanovic, M., Scassellati, B., and Williamson, M.
The cog project: Building a humanoid robot.
Lecture Notes in Artificial Intelligence, 1562, 03 2002.

Brooks, R. A., Breazeal, C., Irie, R., Kemp, C. C., Marjanovic, M., Scassellati, B., and Williamson, M. M.
Alternative essences of intelligence.
AAAI/IAAI, 1998:961–968, 1998.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D.
Language models are few-shot learners.
NeurIPS 2020, 2020.

Bruner, J.
Child's talk: Learning to use language.
Child Language Teaching and Therapy, 1(1):111–114, 1985.

Burda, Y., Edwards, H., Pathak, D., Storkey, A. J., Darrell, T., and Efros, A. A.
Large-scale study of curiosity-driven learning.
In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O.
Exploration by random network distillation.
ICLR, 2019b.

Busch, B., Toussaint, M., and Lopes, M.
Planning ergonomic sequences of actions in human-robot interaction.
In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1916–1923. IEEE, 2018.

Butler, R.
Discrimination learning by rhesus monkeys to visual-exploration motivation.
Journal of Comparative and Physiological Psychology, 1953.

Butler, R. and Harlow, H.
Discrimination learning and learning sets to visual exploration incentives.
The Journal of general psychology, 1957.

Bäck, T., Hoffmeister, F., and Schwefel, H.-P.
A survey of evolution strategies.
In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann, 1991.

Cai, Q., Yang, Z., Jin, C., and Wang, Z.
Provably efficient exploration in policy optimization.
In *International Conference on Machine Learning*, pp. 1283–1294. PMLR, 2020.

Cambria, E. and White, B.
Jumping nlp curves: A review of natural language processing research.
IEEE Computational intelligence magazine, 9(2):48–57, 2014.

Campbell, C.
Buoyancy - Box2D tutorials - iforce2d, 2013.

Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J. B., Rocktäschel, T., and Grefenstette, E.
Learning with amigo: Adversarially motivated intrinsic goals.
In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Cangelosi, A. and Schlesinger, M.
Developmental Robotics: From Babies to Robots.
The MIT Press, 2015.

Cangelosi, A., Metta, G., Sagerer, G., Nolfi, S., Nehaniv, C., Fischer, K., Tani, J., Belpaeme, T., Sandini, G., Nori, F., et al.
Integration of action and language knowledge: A roadmap for developmental robotics.
IEEE Transactions on Autonomous Mental Development, 2(3):167–195, 2010.

Celemin, C. and Ruiz-del Solar, J.
Coach: Learning continuous actions from corrective advice communicated by humans.
In *2015 International Conference on Advanced Robotics (ICAR)*, pp. 581–586, 2015.

Chevalier-Boisvert, M., Willems, L., and Pal, S.
Minimalistic gridworld environment for openai gym.
https://github.com/maximecb/gym-minigrid, 2018.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y.
Babyai: A platform to study the sample efficiency of grounded language learning.
In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Choi, J., Sharma, A., Lee, H., Levine, S., and Gu, S. S.
Variational empowerment as representation learning for goal-conditioned reinforcement learning.
In *International Conference on Machine Learning*, pp. 1953–1963. PMLR, 2021.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y.
Gated feedback recurrent neural networks.
In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2067–2075. JMLR.org, 2015.

Cideron, G., Seurin, M., Strub, F., and Pietquin, O.
Self-educated language agent with hindsight experience replay for instruction following.
In *Visually Grounded Interaction and Language (ViGIL), NeurIPS 2019 Workshop, Vancouver, Canada, December 13, 2019*, 2019.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K.
Better exploration with optimistic actor critic.
In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1785–1796, 2019.

Clark, A.
Being There: Putting Brain, Body, and World Together Again.
MIT Press, Cambridge, MA, USA, 1st edition, 1996.

Clément, B., Roy, D., Oudeyer, P.-Y., and Lopes, M.
Multi-Armed Bandits for Intelligent Tutoring Systems.
Journal of Educational Data Mining (JEDM), 7(2):20–48, June 2015.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J.
Quantifying generalization in reinforcement learning.
In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J.
Leveraging procedural generation to benchmark reinforcement learning.
ICLR, 2020.

Colas, C., Sigaud, O., and Oudeyer, P.-Y.
How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments.
arXiv:1806.08295 [cs, stat], July 2018.
arXiv: 1806.08295.

Colas, C., Oudeyer, P.-Y., Sigaud, O., Fournier, P., and Chetouani, M.
CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning.
In *ICML*, 2019.

Colas, C., Karch, T., Lair, N., Dussoux, J., Moulin-Frier, C., Dominey, P. F., and Oudeyer, P.
Language as a cognitive tool to imagine goals in curiosity driven exploration.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *NeurIPS 2020*, 2020a.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.
Intrinsically motivated goal-conditioned reinforcement learning: a short survey.
arXiv, abs/2012.09830, 2020b.

Côté, M., Kádár, Á., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M. J., Asri, L. E., Adada, M., Tay, W., and Trischler, A.
Textworld: A learning environment for text-based games.
In Cazenave, T., Saffidine, A., and Sturtevant, N. (eds.), *Computer Games - 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers*, volume 1017 of *Communications in Computer and Information Science*, pp. 41–75. Springer, 2018.

Csikszentmihalyi, M.
Flow: The Psychology of Optimal Experience.
Harper Perennial, New York, NY, March 1991.

Czarnecki, W. M., Pascanu, R., Osindero, S., Jayakumar, S. M., Swirszcz, G., and Jaderberg, M.
Distilling policy distillation.
In Chaudhuri, K. and Sugiyama, M. (eds.), *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1331–1340. PMLR, 2019.

Dahl, F. A.
A reinforcement learning algorithm applied to simplified two-player texas hold'em poker.
In De Raedt, L. and Flach, P. (eds.), *Machine Learning: ECML 2001*, pp. 85–96, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D.
Embodied question answering.
In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 1–10. Computer Vision Foundation / IEEE Computer Society, 2018.

Dautenhahn, K. and Billard, A.
Studying robot social cognition within a developmental psychology framework.
In *1999 Third European Workshop on Advanced Mobile Robots (Eurobot'99). Proceedings (Cat. No. 99EX355)*, pp. 187–194. IEEE, 1999.

de Carvalho, E. M.
Social affordance.
Encyclopedia of Animal Cognition and Behavior, 2020.

Deci, E. and Ryan, M.
Intrinsic Motivation and self-determination in human behavior.
Plenum Press, New York, 1985.

Dempster, A. P., Laird, N. M., and Rubin, D. B.
Maximum likelihood from incomplete data via the em algorithm.
Journal of the royal statistical society, Series B, 39(1):1–38, 1977.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A. M., Russell, S., Critch, A., and Levine, S.
Emergent complexity and zero-shot transfer via unsupervised environment design.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Devlin, J., Chang, M., Lee, K., and Toutanova, K.
BERT: pre-training of deep bidirectional transformers for language understanding.
In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Dinh, L., Sohl-Dickstein, J., and Bengio, S.
Density estimation using real NVP.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Dohare, S., Mahmood, A. R., and Sutton, R. S.
Continual backprop: Stochastic gradient descent with persistent randomness.
arXiv preprint arXiv:2108.06325, 2021.

Dominguez, M. and Jacobs, R. A.
Developmental constraints aid the acquisition of binocular disparity sensitivities.
Neural Computation, 15(1):161–182, 2003.

Edelman, G. M.
Neural Darwinism: The theory of neuronal group selection.
Basic books, 1987.

Ellenberger, B.
Pybullet gymperium.
https://github.com/benelot/pybullet-gym, 2018–2019.

Elman, J. L.
Learning and development in neural networks: the importance of starting small.
Cognition, 1993.

Eppe, M. and Oudeyer, P.-Y.
Intelligent behavior depends on the ecological niche.
KI-Künstliche Intelligenz, 35(1):103–108, 2021.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al.
Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures.
In *International Conference on Machine Learning*, pp. 1407–1416. PMLR, 2018.

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

Fang, M., Zhou, T., Du, Y., Han, L., and Zhang, Z.
Curriculum-guided hindsight experience replay.
In *NeurIPS*, 2019.

Fedus, W., Zoph, B., and Shazeer, N.
Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.
arXiv preprint arXiv:2101.03961, 2021.

Flavell, J. H.
Cognitive development: Children's knowledge about the mind.
Annual Review of Psychology, 50(1):21–45, 1999.

Florensa, C., Held, D., Wulfmeier, M., and Abbeel, P.
Reverse curriculum generation for reinforcement learning.
CoRL, 2017.

Florensa, C., Held, D., Geng, X., and Abbeel, P.
Automatic goal generation for reinforcement learning agents.
In *ICML*, 2018.

Fodor, J. A.
Representations: Philosophical Essays on the Foundations of Cognitive Science. MIT Press, 1981.

Fong, T., Nourbakhsh, I., and Dautenhahn, K.
A survey of socially interactive robots.
Robotics and autonomous systems, 42(3-4):143–166, 2003.

Forestier, S. and Oudeyer, P.-Y.
Curiosity-driven development of tool use precursors: a computational model.
In *Proceedings of the 38th Annual Meeting of the Cognitive Science Society*, 2016a.

Forestier, S. and Oudeyer, P.-Y.
Modular active curiosity-driven discovery of tool use.
In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 3965–3972. IEEE, 2016b.

Forestier, S. and Oudeyer, P.-Y.
Overlapping waves in tool use development: a curiosity-driven computational model.
In *Sixth Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2016c.

Forestier, S. and Oudeyer, P.-Y.
A unified model of speech and tool use early development.
In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, 2017.

Forestier, S., Mollard, Y., and Oudeyer, P.
Intrinsically motivated goal exploration processes with automatic curriculum learning.
arXiv, 2017.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S.
Noisy networks for exploration.
In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Fournier, P., Sigaud, O., Chetouani, M., and Oudeyer, P.
Accuracy-based curriculum learning in deep reinforcement learning.
arXiv, 2018.

French, R. M., Mermillod, M., Chauvin, A., Quinn, P. C., and Mareschal, D.
The importance of starting blurry: Simulating improved basic-level category learning in infants due to weak visual acuity.
In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 24, 2002.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S.
From language to goals: Inverse reinforcement learning for vision-based instruction following.
In *International Conference on Learning Representations*, 2019.

Fujimoto, S., van Hoof, H., and Meger, D.
Addressing function approximation error in actor-critic methods.
In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A.
Born-again neural networks.
In *ICML*, pp. 1602–1611, 2018.

Gaier, A. and Ha, D.
Weight agnostic neural networks.
In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Gibson, K. R., Gibson, K. R., and Ingold, T.
Tools, language and cognition in human evolution.
Cambridge University Press, 1993.

Glorot, X. and Bengio, Y.
Understanding the difficulty of training deep feedforward neural networks.
In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gonçalves, A., Abrantes, J., Saponaro, G., Jamone, L., and Bernardino, A.
Learning intermediate object affordances: Towards the development of a tool concept.
In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pp. 482–488. IEEE, 2014.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.
Generative Adversarial Nets.
In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Goodrich, M. A. and Schultz, A. C.
Human-robot interaction: a survey.
Now Publishers Inc, 2008.

Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., and Farhadi, A.
IQA: visual question answering in interactive environments.
In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4089–4098. IEEE Computer Society, 2018.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K.
Automated curriculum learning for neural networks.
In *ICML*, pp. 1311–1320. JMLR. org, 2017.

Grizou, J., Iturrate, I. n., Montesano, L., Oudeyer, P.-Y., and Lopes, M.
Interactive learning from unlabeled instructions.
In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pp. 290–299, Arlington, Virginia, USA, 2014. AUAI Press.

Grollman, D. H. and Billard, A.
Donut as i do: Learning from failed demonstrations.
In *2011 IEEE International Conference on Robotics and Automation*, pp. 3804–3809, 2011.

Ha, D.
Generating abstract patterns with tensorflow.
blog.otoro.net, 2016.

Ha, D.
Reinforcement learning for improving agent design.
Arti. Life, 2019.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S.
Reinforcement learning with deep energy-based policies.
In *International Conference on Machine Learning*, pp. 1352–1361. PMLR, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S.
Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.
ICML, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S.
Soft actor-critic algorithms and applications, 2018b.

Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., and Levine, S.
Learning to walk via deep reinforcement learning.
In Bicchi, A., Kress-Gazit, H., and Hutchinson, S. (eds.), *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

Hacohen, G. and Weinshall, D.
On the power of curriculum learning in training deep networks.
In Chaudhuri, K. and Salakhutdinov, R. (eds.), *ICML*, 2019.

Hämäläinen, P., Babadi, A., Ma, X., and Lehtinen, J.
Ppo-cma: Proximal policy optimization with covariance matrix adaptation.
In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2020.

He, K., Zhang, X., Ren, S., and Sun, J.
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.
In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, X., Zhao, K., and Chu, X.
Automl: A survey of the state-of-the-art.
Knowledge-Based Systems, 212:106622, 2021.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., Wainwright, M., Apps, C., Hassabis, D., and Blunsom, P.
Grounded language learning in a simulated 3d world.
arXiv, abs/1706.06551, 2017.

Hernandez, D., Denamganaï, K., Gao, Y., York, P., Devlin, S., Samothrakis, S., and Walker, J. A.
A generalized framework for self-play training.
In *IEEE CoG*, 2019.

Hernandez-Leal, P., Kartal, B., and Taylor, M. E.
Is multiagent deep reinforcement learning the answer or the question? a brief survey.
learning, 21:22, 2018.

Herrmann, J. M., Pawelzik, K., and Geisel, T.
Learning predictive representations.
Neurocomputing, 32:785–791, 2000.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D.
Rainbow: Combining improvements in deep reinforcement learning.
In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and Van Hasselt, H.
Multi-task deep reinforcement learning with popart.
Proceedings of the AAAI Conference on Artificial Intelligence, 33:3796–3803, Jul 2019.

Hessel, M., Danihelka, I., Viola, F., Guez, A., Schmitt, S., Sifre, L., Weber, T., Silver, D., and van Hasselt, H.
Muesli: Combining improvements in policy optimization.
In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4214–4226. PMLR, 2021.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al.
Deep q-learning from demonstrations.
In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A.
Environmental drivers of systematicity and generalization in a situated agent.
In *International Conference on Learning Representations*, 2020a.

Hill, F., Mokra, S., Wong, N., and Harley, T.
Human instruction-following with deep reinforcement learning via transfer-learning from text.
arXiv, abs/2005.09382, 2020b.

Hochreiter, S. and Schmidhuber, J.
Long short-term memory.
Neural Comput., 9(8):1735–1780, 1997.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., and Silver, D.
Distributed prioritized experience replay.
In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Huang, X. and Weng, J.
Novelty and reinforcement learning in the value system of developmental robots.
Cogprints (pre-print), 94:47–55, 2002.

Huang, X. and Weng, J.
Motivational system for human-robot interaction.
In *International Workshop on Computer Vision in Human-Computer Interaction*, pp. 17–27. Springer, 2004.

Hui, D. Y.-T., Chevalier-Boisvert, M., Bahdanau, D., and Bengio, Y.
Babyai 1.1.
arXiv, 2020.

Hutchins, E.
Cognition in the Wild (Bradford Books).
The MIT Press, September 1996.

Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., and Hofmann, K.
Generalization in reinforcement learning with selective noise injection and information bottleneck.
In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S.
Transient non-stationarity and generalisation in deep reinforcement learning.
In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Ivanovic, B., Harrison, J., Sharma, A., Chen, M., and Pavone, M.
Barc: Backward reachability curriculum for robotic reinforcement learning.
ICRA, 2018.

Jabri, A., Hsu, K., Gupta, A., Eysenbach, B., Levine, S., and Finn, C.
Unsupervised curricula for visual meta-reinforcement learning.
NeurIPS, 2019.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K.
Reinforcement learning with unsupervised auxiliary tasks.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Jaques, N., Lazaridou, A., Hughes, E., Gülçehre, Ç., Ortega, P. A., Strouse, D., Leibo, J. Z., and de Freitas, N.
Social influence as intrinsic motivation for multi-agent deep reinforcement learning.
In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, pp. 3040–3049. PMLR, 2019.

Jara-Ettinger, J.
Theory of mind as inverse reinforcement learning.
Current Opinion in Behavioral Sciences, 29:105–110, 2019.
Artificial Intelligence.

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., and Rocktäschel, T.
Replay-guided adversarial environment design.
Advances in Neural Information Processing Systems, 34, 2021a.

Jiang, M., Grefenstette, E., and Rocktäschel, T.
Prioritized level replay.
In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021b.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B.
CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning.
In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1988–1997. IEEE Computer Society, 2017.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D.
The malmo platform for artificial intelligence experimentation.
In *IJCAI*, 2016.

Justesen, N., Torrado, R. R., Bontrager, P., Khalifa, A., Togelius, J., and Risi, S.
Illuminating generalization in deep reinforcement learning through procedural level generation.
NeurIPS 2018 Deep RL Workshop, 2018.

Kaelbling, L. P., Littman, M. L., and Moore, A. W.
Reinforcement learning: A survey.
Journal of Artificial Intelligence Research, 4:237–285, 1996.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al.
Model-based reinforcement learning for atari.
ICLR, 2020.

Kaplan, F. and Oudeyer, P.-Y.
Motivational principles for visual know-how development.
Cogprints (pre-print), 2003.

Kaplan, F. and Oudeyer, P.-Y.
In search of the neural circuits of intrinsic motivation.
Frontiers in neuroscience, 2007.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.-M., Lam, V.-D., Bewley, A., and Shah, A.
Learning to drive in a day.
In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254. IEEE, 2019.

Kim, K., Sano, M., Freitas, J. D., Haber, N., and Yamins, D.
Active world model learning with progress curiosity.
In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5306–5315. PMLR, 2020.

Kipf, T. N., van der Pol, E., and Welling, M.
Contrastive learning of structured world models.
In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P.
Deep reinforcement learning for autonomous driving: A survey.
IEEE Transactions on Intelligent Transportation Systems, 2021.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T.
A survey of generalisation in deep reinforcement learning.
arXiv preprint arXiv:2111.09794, 2021.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R.
Overcoming catastrophic forgetting in neural networks.
Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017.

Klink, P., D'Eramo, C., Peters, J., and Pajarinen, J.
Self-paced deep reinforcement learning.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Klopf, A.
Brain function and adaptive systems: A heterostatic theory.
Technical Report Air Force Cambridge Research Laboratories, 1972.

Klopf, A. H.
A comparison of natural and artificial intelligence.
SIGART Newsl., 52:11–13, 1975.

Klopf, A. H.
The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence.
Hemisphere, 1982.

Koedinger, K. R., Brunskill, E., de Baker, R. S. J., McLaughlin, E. A., and Stamper, J. C.
New potentials for data-driven intelligent tutoring system development and optimization.
AI Magazine, 34(3):27–41, 2013.

Kohl, N. and Stone, P.
Policy gradient reinforcement learning for fast quadrupedal locomotion.
In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pp. 2619–2624. IEEE, 2004.

Kormushev, P., Calinon, S., and Caldwell, D. G.
Robot motor skill coordination with em-based reinforcement learning.
In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pp. 3232–3237. IEEE, 2010.

Kovac, G., Portelas, R., Hofmann, K., and Oudeyer, P.
Socialai 0.1: Towards a benchmark to stimulate research on socio-cognitive abilities in deep reinforcement learning agents.
Visually Grounded Interaction and Language (ViGIL) Workshop, NAACL 2021, abs/2104.13207, 2021.

Kovács, I.
Human development of perceptual organization.
Vision research, 40(10-12):1301–1310, 2000.

Kovač, G., Laversanne-Finot, A., and Oudeyer, P.-Y.
Grimgep: Learning progress for robust goal sampling in visual deep reinforcement learning, 2020.

Kozima, H. and Yano, H.
A robot that learns to communicate with human caregivers.
In *Proceedings of the First International Workshop on Epigenetic Robotics*, volume 2001. Citeseer, 2001.

Krichmar, J. L. and Edelman, G. M.
Machine psychology: autonomous behavior, perceptual categorization and conditioning in a brain-based device.
Cerebral Cortex, 12(8):818–830, 2002.

Krizhevsky, A., Sutskever, I., and Hinton, G. E.
Imagenet classification with deep convolutional neural networks.
Advances in neural information processing systems, 25:1097–1105, 2012.

Kroemer, O., Ugur, E., Oztop, E., and Peters, J.
A kernel-based approach to direct action perception.
In *2012 IEEE international Conference on Robotics and Automation*, pp. 2605–2610. IEEE, 2012.

Kröger, B. J., Kannampuzha, J., and Neuschaefer-Rube, C.
Towards a neurocomputational model of speech production and perception.
Speech Communication, 51(9):793–809, 2009.

Krueger, K. A. and Dayan, P.
Flexible shaping: How learning in small steps helps.
Cognition, 2009.

Kumar, M., Husain, M., Upreti, N., and Gupta, D.
Genetic algorithm: Review and application.
Available at SSRN 3529843, 2010.

Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al.
Google research football: A novel reinforcement learning environment.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4501–4510, 2020.

Lair, N., Colas, C., Portelas, R., Dussoux, J., Dominey, P. F., and Oudeyer, P.
Language grounding through social interactions and curiosity-driven multi-goal learning.
In *Visually Grounded Interaction and Language (ViGIL), NeurIPS 2019 Workshop, Vancouver, Canada, December 13, 2019*, 2019.

Lakoff, G., Johnson, M., et al.
Philosophy in the flesh: The embodied mind and its challenge to western thought, volume 640.
Basic books New York, 1999.

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

Langton, C. G.
Artificial Life: An Overview.
The MIT Press, 07 1995.

Lapeyre, M., Rouanet, P., Grizou, J., Nguyen, S., Depraetre, F., Le Falher, A., and Oudeyer, P.-Y.
Poppy project: open-source fabrication of 3d printed humanoid robot for science, education and art.
In *Digital Intelligence 2014*, pp. 6, 2014.

Laversanne-Finot, A., Pere, A., and Oudeyer, P.-Y.
Curiosity driven exploration of learned disentangled goal spaces.
CoRL, 2018.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D.
Backpropagation applied to handwritten zip code recognition.
Neural computation, 1(4):541–551, 1989.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P.
Gradient-based learning applied to document recognition.
In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

LeCun, Y., Bengio, Y., and Hinton, G. E.
Deep learning.
Nat., 521(7553):436–444, 2015.

Lehman, J., Stanley, K. O., et al.
Exploiting open-endedness to solve problems through the search for novelty.
In *ALIFE*, pp. 329–336. Citeseer, 2008.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Rodríguez, N. D.
Continual learning for robotics.
arXiv, 2019.

Levine, S., Finn, C., Darrell, T., and Abbeel, P.
End-to-end training of deep visuomotor policies.
The Journal of Machine Learning Research, 17(1):1334–1373, 2016.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D.
Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection.
The International Journal of Robotics Research, 37(4-5):421–436, 2018.

Li, C., Zhang, M., and He, Y.
Curriculum learning: A regularization method for efficient and stable billion-scale gpt model pre-training.
arXiv preprint arXiv:2108.06084, 2021.

Li, W. and Fritz, M.
Teaching robots the use of human tools from demonstration with non-dexterous end-effectors.
In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 547–553. IEEE, 2015.

Li, Y., Wen, Y., Tao, D., and Guan, K.
Transforming cooling optimization for green data center via deep reinforcement learning.
IEEE transactions on cybernetics, 50(5):2002–2013, 2019.

Li, Z., Lin, X., Zhang, Q., and Liu, H.
Evolution strategies for continuous optimization: A survey of the state-of-the-art.
Swarm and Evolutionary Computation, 56:100694, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D.
Continuous control with deep reinforcement learning.
In *ICLR*, 2016.

Lin, L.-J.
Self-improving reactive agents based on reinforcement learning, planning and teaching.
Mach. lear., 1992.

Lindblom, J. and Ziemke, T.
Social situatedness of natural and artificial intelligence: Vygotsky and beyond.
Adaptive Behavior, 11(2):79–96, 2003.

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., and Bachem, O.
A sober look at the unsupervised learning of disentangled representations and their evaluation.
J. Mach. Learn. Res., 21:209:1–209:62, 2020.

Loewenstein, G.
The psychology of curiosity: A review and reinterpretation.
Psychological Bulletin, 116:75–98, 07 1994.

Lopes, M. and Oudeyer, P.-Y.
The strategic student approach for life-long exploration and learning.
In *ICDL*, 2012a.

Lopes, M. and Oudeyer, P.-Y.
The Strategic Student Approach for Life-Long Exploration and Learning.
In *IEEE Conference on Development and Learning / EpiRob 2012*, San Diego, United States, November 2012b.

Lowe, R., Gupta, A., Foerster, J. N., Kiela, D., and Pineau, J.
On the interaction between supervision and self-play in emergent communication.
In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T.
A survey of reinforcement learning informed by natural language.
In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 6309–6317, 7 2019.

Lungarella, M. and Berthouze, L.
Adaptivity through physical immaturity.
Cogprints (pre-print), 2002a.

Lungarella, M. and Berthouze, L.
On the interplay between morphological, neural, and environmental dynamics: a robotic case study.
Adaptive Behavior, 10(3-4):223–241, 2002b.

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G.
Developmental robotics: a survey.
Connect. Sci., 15(4):151–190, 2003.

Lynch, C. and Sermanet, P.
Grounding language in play.
arXiv, abs/2005.07648, 2020.

Maeda, G., Ewerton, M., Osa, T., Busch, B., and Peters, J.
Active incremental learning of robot movement primitives.
In *Conference on Robot Learning*, pp. 37–46. PMLR, 2017.

Markey, K. L.
The sensorimotor foundations of phonology: a computational model of early childhood articulatory and phonetic development.
PhD thesis, University of Colorado at Boulder, 1994.

Marshall, J., Blank, D., and Meeden, L.
An emergent framework for self-motivation in developmental robotics.
Proceedings Of The 2004 International Conference On Development And Learning, 2004.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J.
Teacher-student curriculum learning.
IEEE TNNLS, 2017.

Mazoure, B., des Combes, R. T., Doan, T., Bachman, P., and Hjelm, R. D.
Deep reinforcement and infomax learning.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

McCarthy, J.
From here to human-level ai.
Artificial Intelligence, 171(18):1174–1182, 2007.

Mehta, B., Diaz, M., Golemo, F., Pal, C. J., and Paull, L.
Active domain randomization.
CoRL, 2019.

Meltzoff, A. N. and Prinz, W.
The imitative mind: Development, evolution and brain bases, volume 6.
Cambridge University Press, 2002.

Metta, G. and Fitzpatrick, P.
Early integration of vision and manipulation.
Adaptive behavior, 11(2):109–128, 2003.

Metta, G., Sandini, G., and Konczak, J.
A developmental approach to visually-guided reaching in artificial systems.
Neural networks, 12(10):1413–1427, 1999.

Mirolli, M. and Parisi, D.
Towards a vygotskyan cognitive robotics: The role of language as a cognitive tool.
New Ideas in Psychology, 29(3):298–311, 2011.
Special Issue: Cognitive Robotics and Reevaluation of Piaget Concept of Egocentrism.

Misra, D. K., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y.
Mapping instructions to actions in 3d environments with visual goal prediction.
In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2667–2678. Association for Computational Linguistics, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M.
Playing atari with deep reinforcement learning.
NeurIPS Deep Learning Workshop 2013, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K.
Asynchronous methods for deep reinforcement learning.
In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Moerland, T. M., Broekens, J., and Jonker, C. M.
Model-based reinforcement learning: A survey.
arXiv preprint arXiv:2006.16712, 2020.

Moore, A. W. and Atkeson, C. G.
Prioritized sweeping: Reinforcement learning with less data and less time.
Mach. learn., 1993.

Mordatch, I. and Abbeel, P.
Emergence of grounded compositional language in multi-agent populations.
In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 1495–1502. AAAI Press, 2018.

Moulin-Frier, C., Nguyen, S. M., and Oudeyer, P.-Y.
Self-organization of early vocal development in infants and machines: The role of intrinsic motivation.
Frontiers in Psychology (Cognitive Science), 4(1006), 2014.

Mysore, S., Platt, R., and Saenko, K.
Reward-guided curriculum for robust reinforcement learning.
preprint, 2018.

Nagabandi, A., Konolige, K., Levine, S., and Kumar, V.
Deep dynamics models for learning dexterous manipulation.
In *Conference on Robot Learning*, pp. 1101–1112. PMLR, 2020.

Nagai, Y., Asada, M., and Hosoda, K.
Developmental learning model for joint attention.
In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pp. 932–937. IEEE, 2002.

Nagai, Y., Asada, M., and Hosoda, K.
Learning for joint attention helped by functional development.
Advanced Robotics, 20(10):1165–1181, 2006.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P.
Overcoming exploration in reinforcement learning with demonstrations.
In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

Narasimhan, K., Kulkarni, T. D., and Barzilay, R.
Language understanding for text-based games using deep reinforcement learning.
In Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., and Marton, Y. (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 1–11. The Association for Computational Linguistics, 2015.

Narvekar, S. and Stone, P.
Generalizing curricula for reinforcement learning.
In *4th Lifelong Learning Workshop at the International Conference on Machine Learning (ICML 2020)*, July 2020.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P.
Curriculum learning for reinforcement learning domains: A framework and survey.
J. Mach. Learn. Res., 21:181:1–181:50, 2020.

Ndousse, K. K., Eck, D., Levine, S., and Jaques, N.
Emergent social learning via multi-agent reinforcement learning.
In *International Conference on Machine Learning*, pp. 7991–8004. PMLR, 2021.

Newell, A. and Simon, H. A.
Computer science as empirical inquiry: Symbols and search.
Commun. ACM, 19(3):113–126, March 1976.

Ng, A. Y., Russell, S. J., et al.
Algorithms for inverse reinforcement learning.
In *Icml*, volume 1, pp. 2, 2000.

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E.
Autonomous inverted helicopter flight via reinforcement learning.
In *Experimental robotics IX*, pp. 363–372. Springer, 2006.

Nguyen, K., Misra, D., Schapire, R. E., Dudík, M., and Shafto, P.
Interactive learning from activity description.
In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8096–8108. PMLR, 2021.

Nguyen, S. M. and Oudeyer, P.-Y.
Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner.
Paladyn, 3(3):136–146, 2012.

Nguyen-Tuong, D. and Peters, J.
Local gaussian process regression for real-time model-based robot control.
In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 380–385. IEEE, 2008.

Nguyen-Tuong, D. and Peters, J.
Model learning for robot control: a survey.
Cognitive processing, 12(4):319–340, 2011.

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J.
Gotta learn fast: A new benchmark for generalization in RL.
arXiv, 2018.

Nisioti, E. and Moulin-Frier, C.
Grounding artificial intelligence in the origins of human behavior.
arXiv preprint arXiv:2012.08564, 2020.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q.-M., Zaremba, W., and Zhang, L.
Solving rubik's cube with a robot hand.
arXiv, 2019.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Roy, B. V., Sutton, R., Silver, D., and Hasselt, H. V.
Behaviour suite for reinforcement learning.
ICML, 2020.

Oudeyer, P.-Y. and Kaplan, F.
What is intrinsic motivation? a typology of computational approaches.
Frontiers in neurorobotics, 1:6, 2007.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V.
Intrinsic motivation systems for autonomous mental development.
IEEE trans. on evolutionary comp., 2007a.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V.
Intrinsic motivation systems for autonomous mental development.
IEEE transactions on evolutionary computation, 11(2):265–286, 2007b.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D.
Assessing generalization in deep reinforcement learning.
arXiv, 2018.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T.
Curiosity-driven exploration by self-supervised prediction.
In *ICML*, 2017.

Pathak, D., Gandhi, D., and Gupta, A.
Self-supervised exploration via disagreement.
In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071. PMLR, 2019.

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M.
Deepmimic: Example-guided deep reinforcement learning of physics-based character skills.
ACM Transactions on Graphics (TOG), 37(4):1–14, 2018a.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P.
Sim-to-real transfer of robotic control with dynamics randomization.
In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 1–8. IEEE, 2018b.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C.
Film: Visual reasoning with a general conditioning layer.
In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3942–3951. AAAI Press, 2018.

Pfeifer, R. and Scheier, C.
Understanding intelligence.
MIT press, 2001.

Pham, T.-H., De Magistris, G., and Tachibana, R.
Optlayer-practical constrained optimization for deep reinforcement learning in the real world.
In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6236–6243. IEEE, 2018.

Piaget, J.
The origins of intelligence in children.
W W Norton & Co, 1952.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A.
Robust adversarial reinforcement learning.

In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2817–2826. PMLR, 2017.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al.
Multi-goal reinforcement learning: Challenging robotics environments and request for research.
arXiv preprint arXiv:1802.09464, 2018.

Pong, V., Gu, S., Dalal, M., and Levine, S.
Temporal difference models: Model-free deep RL for model-based control.
In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

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

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y.
Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments.
CoRL, 2019.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y.
Automatic curriculum learning for deep rl: A short survey.
IJCAI, 2020a.

Portelas, R., Hofmann, K., and Oudeyer, P.-Y.
Trying again instead of trying longer: Prior learning for automatic curriculum learning.
Beyond "Tabula Rasa" in Reinforcement Learning (BeTR-RL) workshop, ICLR, 2020b.

Portelas, R., Romac, C., Hofmann, K., and Oudeyer, P.
Meta automatic curriculum learning.
arXiv, abs/2011.08463, 2020c.

Portelas, R., Kovač, G., Hofmann, K., and Oudeyer, P.-Y.
Socialai: Benchmarking socio-cognitive abilities in deep reinforcement learning agents.
arXiv preprint arXiv:2107.00956, 2021.

Prabhumoye, S., Li, M., Urbanek, J., Dinan, E., Kiela, D., Weston, J., and Szlam, A.
I love your chain mail! making knights smile in a fantasy game world: Open-domain goal-oriented dialogue agents, 2020.

Press, O., Smith, N. A., and Lewis, M.
Shortformer: Better language modeling using shorter inputs.
In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pp. 5493–5505. Association for Computational Linguistics, 2021.

Pugh, J. K., Soros, L. B., and Stanley, K. O.
Quality diversity: A new frontier for evolutionary computation.
Frontiers in Robotics and AI, 3:40, 2016.

Puig, X., Shu, T., Li, S., Wang, Z., Liao, Y., Tenenbaum, J. B., Fidler, S., and Torralba, A.
Watch-and-help: A challenge for social perception and human-ai collaboration.
In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X.
Pre-trained models for natural language processing: A survey.
Science China Technological Sciences, pp. 1–26, 2020.

Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S. M. A., and Botvinick, M.
Machine theory of mind.
In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4215–4224. PMLR, 2018.

Racaniere, S., Lampinen, A., Santoro, A., Reichert, D., Firoiu, V., and Lillicrap, T.
Automated curriculum generation through setter-solver interactions.
In *International Conference on Learning Representations*, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.
Language models are unsupervised multitask learners.
OpenAI blog, 1(8):9, 2019.

Raileanu, R. and Rocktäschel, T.
RIDE: rewarding impact-driven exploration for procedurally-generated environments.
In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S.
Epopt: Learning robust neural network policies using model ensembles.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Rasmussen, C. E.
The infinite gaussian mixture model.
In Solla, S. A., Leen, T. K., and Müller, K. (eds.), *Advances in Neural Information Processing Systems 12*, pp. 554–560. MIT Press, 2000.

Riedmiller, M.
Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method.
In *European conference on machine learning*, pp. 317–328. Springer, 2005.

Riedmiller, M. A., Hafner, R., Lampe, T., Neunert, M., Degrave, J., de Wiele, T. V., Mnih, V., Heess, N., and Springenberg, J. T.
Learning by playing solving sparse reward tasks from scratch.
In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4341–4350. PMLR, 2018.

Riggio, R.
Assessment of basic social skills.
Journal of Personality and Social Psychology, 51:649–660, 09 1986.

Risi, S. and Stanley, K. O.
Deep neuroevolution of recurrent and discrete world models.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 456–462, 2019.

Risi, S. and Togelius, J.
Procedural content generation: From automatically generating game levels to increasing generality in machine learning.
arXiv, 2019.

Rohlfing, K. J., Wrede, B., Vollmer, A.-L., and Oudeyer, P.-Y.
An alternative to mapping a word onto a concept in language acquisition: Pragmatic frames.
Frontiers in Psychology, 7:470, 2016.

Romac, C., Portelas, R., Hofmann, K., and Oudeyer, P.
Teachmyagent: a benchmark for automatic curriculum learning in deep RL.
In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9052–9063. PMLR, 2021.

Rosch, E.
Principles of categorization.
Concepts: core readings, 189:312–322, 1999.

Ross, T.
Machines that think.
Scientific American, 148(4):206–208, 1933.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M.
A benchmark for systematic generalization in grounded language understanding.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19861–19872. Curran Associates, Inc., 2020.

Rummery, G. A. and Niranjan, M.
On-line Q-learning using connectionist systems, volume 37.
Citeseer, 1994.

Russell, S.
Learning agents for uncertain environments.
In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998.

Salimans, T. and Chen, R.
Learning montezuma's revenge from a single demonstration.
NeurIPS, 2018.

Samuel, A. L.
Some studies in machine learning using the game of checkers.
IBM Journal of research and development, 3(3):210–229, 1959.

Samuel, A. L.
Some studies in machine learning using the game of checkers. ii—recent progress.
IBM Journal of research and development, 11(6):601–617, 1967.

Savinov, N., Raichuk, A., Vincent, D., Marinier, R., Pollefeys, M., Lillicrap, T. P., and Gelly, S.
Episodic curiosity through reachability.
In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Scassellati, B. M.
Foundations for a Theory of Mind for a Humanoid Robot.
PhD thesis, Massachusetts Institute of Technology, 2001.

Schaal, S.
Is imitation learning the route to humanoid robots?
Trends in cognitive sciences, 3(6):233–242, 1999.

Schaal, S.
Dynamic movement primitives-a framework for motor control in humans and humanoid robotics.
In *Adaptive motion of animals and machines*, pp. 261–280. Springer, 2006.

Schaul, T., Horgan, D., Gregor, K., and Silver, D.
Universal value function approximators.
In *ICML*, 2015.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D.
Prioritized experience replay.
In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Scheier, C. and Lambrinos, D.
Categorization in a real-world agent using haptic exploration and active perception.
In *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior (SAB'96)(Cambridge MA: MIT Press)*, pp. 65–75, 1996.

Schlesinger, M., Berthouze, L., and Balkenius, C.
Heterochrony: It's (all) about time!
In *Proceedings of the Eighth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, pp. 111–117. Citeseer, 2008.

Schmidhuber, J.
Curious model-building control systems.
In *IJCNN*. IEEE, 1991.

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

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P.
Trust region policy optimization.
In *ICML*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.
Proximal policy optimization algorithms.
arXiv, abs/1707.06347, 2017.

Selfridge, O. G., Sutton, R. S., and Barto, A. G.
Training and tracking in robotics.
In *IJCAI*, 1985.

Sharma, P., Pathak, D., and Gupta, A.
Third-person visual imitation learning via decoupled hierarchical controller.
In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 2593–2603, 2019.

Shridhar, M., Yuan, X., Cote, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M.
{ALFW}orld: Aligning text and embodied environments for interactive learning.
In *International Conference on Learning Representations*, 2021.

Shyam, P., Jaśkowski, W., and Gomez, F.
Model-based active exploration.
In *International conference on machine learning*, pp. 5779–5788. PMLR, 2019.

Siekmann, J., Green, K., Warila, J., Fern, A., and Hurst, J. W.
Blind bipedal stair traversal via sim-to-real reinforcement learning.
In Shell, D. A., Toussaint, M., and Hsieh, M. A. (eds.), *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021.

Sigaud, O., Salaün, C., and Padois, V.
On-line regression algorithms for learning mechanical models of robots: a survey.
Robotics and Autonomous Systems, 59(12):1115–1129, 2011.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D.
Mastering the game of Go with deep neural networks and tree search.
Nature, 529(7587):484–489, jan 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D.
Mastering the game of go without human knowledge.
Nature, 2017.

Singh, S., Barto, A. G., and Chentanez, N.
Intrinsically motivated reinforcement learning.
Technical report, Massachusetts Univ Amherst dept of Computer Science, 2005.

Sinha, S., Song, J., Garg, A., and Ermon, S.
Experience replay with likelihood-free importance weights.
arXiv preprint arXiv:2006.13169, 2020.

Song, D. R., Yang, C., McGreavy, C., and Li, Z.
Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge.
In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 311–318, 2018.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., et al.
V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control.
ICLR, 2020.

Stadie, B. C., Abbeel, P., and Sutskever, I.
Third person imitation learning.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Stanley, K. O.
Compositional pattern producing networks: A novel abstraction of development.
Genet. Program. Evolvable Mach., 8(2):131–162, 2007.

Stanley, K. O.
Why open-endedness matters.
Artificial life, 25(3):232–235, 2019.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J.
A hypercube-based encoding for evolving large-scale neural networks.
Artificial life, 15(2):185–212, 2009.

Stanley, K. O., Lehman, J., and Soros, L.
Open-endedness: The last grand challenge you've never heard of.
While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself, 2017.

Steels, L.
The symbol grounding problem has been solved. so what's next?
Symbols, Embodiment and Meaning. Oxford University Press, Oxford, UK, 01 2007.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M.
Decoupling representation learning from reinforcement learning.
In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Stoytchev, A.
Behavior-grounded representation of tool affordances.
In *Proceedings of the 2005 ieee international conference on robotics and automation*, pp. 3060–3065. IEEE, 2005.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R.
Intrinsic motivation and automatic curricula via asymmetric self-play.
In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G.
On the importance of initialization and momentum in deep learning.
In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

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

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y.
Policy gradient methods for reinforcement learning with function approximation.
In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Szepesvári, C.
Algorithms for reinforcement learning.
Synthesis lectures on artificial intelligence and machine learning, 4(1):1–103, 2010.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P.
#exploration: A study of count-based exploration for deep reinforcement learning.
In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2753–2762, 2017.

Tang, Y. and Ha, D.
The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning.
NeurIPS, 2021.

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., and Heess, N.
dm_control: Software and tasks for continuous control.
arXiv, 2020.

Taylor, M. E. and Stone, P.
Transfer learning for reinforcement learning domains: A survey.
JMLR, 2009.

Te Boekhorst, R., Lungarella, M., and Pfeifer, R.
Dimensionality reduction through sensory-motor coordination.
In *Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003*, pp. 496–503. Springer, 2003.

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al.
Open-ended learning leads to generally capable agents.
arXiv preprint arXiv:2107.12808, 2021.

Teh, Y. W., Bapst, V., Czarnecki, W., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N. M. O., and Pascanu, R.
Distral: Robust multitask reinforcement learning.
In *NeurIPS*, 2017.

Ten, A., Gottlieb, J., and Oudeyer, P.-Y.
Intrinsic rewards in human curiosity-driven exploration: An empirical study.
In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 43, 2021.

Tesauro, G.
Temporal difference learning and td-gammon.
J. Int. Comput. Games Assoc., 18(2):88, 1995.

Tesauro, G., Gondek, D., Lenchner, J., Fan, J., and Prager, J. M.
Simulation, learning, and optimization techniques in watson's game strategies.
IBM Journal of Research and Development, 56(3.4):16–1, 2012.

Tesauro, G., Gondek, D. C., Lenchner, J., Fan, J., and Prager, J. M.
Analysis of watson's strategies for playing jeopardy!
Journal of Artificial Intelligence Research, 47:205–251, 2013.

Thomaz, A. L., Berlin, M., and Breazeal, C.
An embodied computational model of social referencing.
In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pp. 591–598. IEEE, 2005.

Thorndike, E. L.
Animal Intelligence.
Hafner, Darien, CT, 1911.

Thrun, S.
Exploration in active learning.
Handbook of Brain Science and Neural Networks, pp. 381–384, 1995.

Tikhanoff, V., Pattacini, U., Natale, L., and Metta, G.
Exploring affordances and tool use on the icub.
In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 130–137. IEEE, 2013.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P.
Domain randomization for transferring deep neural networks from simulation to the real world.
In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pp. 23–30. IEEE, 2017.

Todorov, E., Erez, T., and Tassa, Y.
Mujoco: A physics engine for model-based control.
In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Tomasello, M.
The Cultural Origins of Human Cognition.
Harvard University Press, 1999.

Tomasello, M.
Becoming human: A theory of ontogeny.
Harvard University Press, 2019.

Torabi, F., Warnell, G., and Stone, P.
Recent advances in imitation learning from observation.
In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 6325–6331. ijcai.org, 2019.

Turchetta, M., Kolobov, A., Shah, S., Krause, A., and Agarwal, A.
Safe reinforcement learning via curriculum induction.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Turing, A. M.
Computing machinery and intelligence.
Mind, 1950.

Turkewitz, G. and Kenny, P. A.
Limitations on input as a basis for neural organization and perceptual development: A preliminary theoretical statement.

Developmental Psychobiology: The Journal of the International Society for Developmental Psychobiology, 15(4):357–368, 1982.

Ugur, E., Nagai, Y., Sahin, E., and Oztop, E.
Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese.
IEEE Transactions on Autonomous Mental Development, 7(2):119–139, 2015.

Ungerleider, L. G.
Two cortical visual systems.
Analysis of visual behavior, pp. 549–586, 1982.

Urbanek, J., Fan, A., Karamcheti, S., Jain, S., Humeau, S., Dinan, E., Rocktäschel, T., Kiela, D., Szlam, A., and Weston, J.
Learning to speak and act in a fantasy text adventure game.
In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 673–683. Association for Computational Linguistics, 2019.

Van Der Malsburg, C.
Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms.
In Palm, G. and Aertsen, A. (eds.), *Brain Theory*, pp. 245–248, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

Van Hasselt, H., Guez, A., and Silver, D.
Deep reinforcement learning with double q-learning.
In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Vanschoren, J.
Meta-learning: A survey.
arXiv, abs/1810.03548, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I.
Attention is all you need.
In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vélez, N. and Gweon, H.
Learning from other minds: An optimistic critique of reinforcement learning models of social learning.
Current Opinion in Behavioral Sciences, 38:110–115, 2021.

Vie, J.-J.
Cognitive diagnostic computerized adaptive testing models for large-scale learning.
Thesis, Université Paris Saclay (COmUE), 2016.

Vie, J.-J., Popineau, F., Bruillard, É., and Bourda, Y.
Automated Test Assembly for Handling Learner Cold-Start in Large-Scale Assessments.
IJAIED, 2018.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D.
Grandmaster level in StarCraft II using multi-agent reinforcement learning.
Nature, 2019.

Vollmer, A.-L., Wrede, B., Rohlfing, K. J., and Oudeyer, P.-Y.
Pragmatic frames for teaching and learning in human–robot interaction: Review and challenges.
Frontiers in Neurorobotics, 10:10, 2016.

Vuong, Q., Zhang, Y., and Ross, K. W.
Supervised policy update for deep reinforcement learning.
ICLR, 2019.

Vygotsky, L. S. and Cole, M.
Mind in society : the development of higher psychological processes.
Harvard University Press Cambridge, 1978.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S.
GLUE: A multi-task benchmark and analysis platform for natural language understanding.
In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Wang, C. and Ross, K.
Boosting soft actor-critic: Emphasizing recent experience without forgetting the past.
arXiv preprint arXiv:1906.04009, 2019.

Wang, H., Liu, N., Zhang, Y., Feng, D., Huang, F., Li, D. S., and Zhang, Y.
Deep reinforcement learning: a survey.
Frontiers Inf. Technol. Electron. Eng., 21(12):1726–1744, 2020a.

Wang, Q., Li, Y., Xiong, J., and Zhang, T.
Divergence-augmented policy optimization.
In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O.
Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions.
arXiv, 2019b.

Wang, R., Lehman, J., Rawal, A., Zhi, J., Li, Y., Clune, J., and Stanley, K. O.
Enhanced POET: open-ended reinforcement learning through unbounded invention of learning challenges and their solutions.
In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9940–9951. PMLR, 2020b.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N.
Dueling network architectures for deep reinforcement learning.
In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Warlaumont, A. S.
A spiking neural network model of canonical babbling development.
In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pp. 1–6. IEEE, 2012.

Warlaumont, A. S., Westermann, G., Buder, E. H., and Oller, D. K.
Prespeech motor learning in a neural network using reinforcement.
Neural Networks, 38:64–75, 2013.

Watkins, C. J. C. H. and Dayan, P.
Q-learning.
Machine Learning, 8(3):279–292, May 1992.

Wellman, H. M.
The child's theory of mind.
The MIT Press, 1992.

Weng, J.
A theory for mentally developing robots.
In *Proceedings 2nd International Conference on Development and Learning. ICDL 2002*, pp. 131–140. IEEE, 2002.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E.
Autonomous mental development by robots and animals.
Science, 291(5504):599–600, 2001.

White, D. J.
Dynamic programming, volume 1.
Oliver & Boyd Edinburgh, 1969.

White, D. J.
Real applications of markov decision processes.
Interfaces, 15(6):73–83, 1985.

White, D. J.
Further real applications of markov decision processes.
Interfaces, 18(5):55–61, 1988.

White, R.
Motivation reconsidered: the concept of competence.
Psychological review, 66:297—333, September 1959.

Whiten, A.
Primate culture and social learning.
Cognitive Science, 24(3):477–508, 2000.

Whiteson, S., Tanner, B., Taylor, M., and Stone, P.
Protecting against evaluation overfitting in empirical reinforcement learning.
Journal of Urology - J UROL, pp. 120–127, 04 2011.

Widrow, B. and Hoff, M. E.
Adaptive switching circuits.
In *1960 IRE WESCON Convention Record, Part 4*, pp. 96–104, New York, 1960. IRE.

Williams, R. J.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine learning, 8(3-4):229–256, 1992.

Woodward, M., Finn, C., and Hausman, K.
Learning to interactively learn and assist.
In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pp. 2535–2543. AAAI Press, 2020.

Yim, J., Joo, D., Bae, J., and Kim, J.
A gift from knowledge distillation: Fast optimization, network minimization and transfer learning.
CVPR, pp. 7130–7138, 2017.

Yu, X. and Gen, M.
Introduction to evolutionary algorithms.
Springer Science & Business Media, 2010.

Zadeh, A., Chan, M., Liang, P. P., Tong, E., and Morency, L.-P.
Social-iq: A question answering benchmark for artificial social intelligence.
In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8799–8809, 2019.

Zha, D., Lai, K., Zhou, K., and Hu, X.
Experience replay optimization.
In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 4243–4249. ijcai.org, 2019.

Zhang, A., Ballas, N., and Pineau, J.
A dissection of overfitting and generalization in continuous reinforcement learning.
arXiv preprint arXiv:1806.07937, 2018a.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S.
A study on overfitting in deep reinforcement learning.
arXiv, abs/1804.06893, 2018b.

Zhang, F., Leitner, J., Milford, M., Upcroft, B., and Corke, P. I.
Towards vision-based deep reinforcement learning for robotic motion control.
Australasian Conference on Robotics and Automation, abs/1511.03791, 2015.

Zhang, Y., Abbeel, P., and Pinto, L.
Automatic curriculum learning through value disagreement.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Zhao, R. and Tresp, V.
Energy-based hindsight experience prioritization.
In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pp. 113–122. PMLR, 2018a.

Zhao, R. and Tresp, V.
Curiosity-driven experience prioritization via density estimation.
NeurIPS Deep Reinforcement Learning Workshop, 2018b.

Zhao, W., Queralta, J. P., and Westerlund, T.
Sim-to-real transfer in deep reinforcement learning for robotics: a survey.
In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744. IEEE, 2020.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al.
Maximum entropy inverse reinforcement learning.
In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.