Intrinsically Motivated and Interactive Reinforcement Learning: a Developmental Approach

Pierre Fournier

Université Pierre et Marie Curie

École doctorale informatique,
télécommunications et électronique de paris

PhD Thesis

Intrinsically Motivated and Interactive Reinforcement Learning: a Developmental Approach

defended by

Pierre Fournier

supervised by

Olivier Sigaud and Mohamed Chetouani

defended on November 29, 2019

Jury:

Alain Dutech	INRIA Nancy	REVIEWER
Michèle Sebag	INRIA Saclay	REVIEWER
Nicolas Maudet	Sorbonne Université	EXAMINER
Pierre-Yves Oudeyer	INRIA Bordeaux	EXAMINER
Mohamed Chetouani	Sorbonne Université	SUPERVISOR
Olivier Sigaud	Sorbonne Université	SUPERVISOR

Abstract

Reinforcement learning is a learning paradigm in which the agent maximizes the accumulation of a reward signal. It has met with a growing interest since its combination with deep learning, but today, it confronts a paradox known as the Moravec's paradox: reproducing child-level sensorimotor abilities, like locomotion or object manipulation, is more intricate than learning to play abstract games like go or poker. Many recent evolutions of reinforcement learning aim to change this status quo, and take inspiration from human learning behaviors. In doing so, they focus on challenges close to the objectives of another child of the quest for Artificial Intelligence: developmental robotics. The latter aims at modelling and reproducing the development of natural cognitive processes in artificial agents, and can be a rich source of inspiration for reinforcement learning. On the other hand, deep reinforcement learning provides a framework to combine these processes, with strong representational power to tackle high-dimensional environments. Following these observations, bridges between reinforcement learning and developmental sciences are explored more and more, but there is still much to do.

In this dissertation, we study the intersection of two crucial topics in developmental sciences, and their application to reinforcement learning: social learning and intrinsic motivation. Interactive and intrinsically motivated reinforcement learning have already been studied, separately, but with a focus on improving existing agents performance rather than learning in a developmental fashion. Instead, we address this second objective. Precisely, we propose several agents allowing to study how reinforcement learning, intrinsic motivations and simple forms of social learning can interact to produce learning behaviors that are sound from a developmental standpoint, and richer than those produced by each mechanism alone.

Developmental studies show that interactive capabilities emerge incrementally through the progressive interaction of simple attentional and exploratory mechanisms. The first agent that we propose combines two such mechanisms: novelty search and gaze-following. We show that these mechanisms expectedly speed up the learning of a task defined by standard reward signals. More unexpectedly, the two mechanisms together lead the agent to task completion in absence of such rewards, albeit sub-optimally. The agent thus recreates a typical interaction between a learner and a tutor, where the latter guides the former's attention towards objects that pique its curiosity, in order to make it discover new action possibilities. However, this preliminary work shows that the agent lacks a form of intentionality to learn fully without rewards from the environment, as humans do. We thus go on to describe two agents endowed with goal-directed behaviors.

We use goal-conditioned reinforcement learning as the base mechanism for building goal-directed agents, and we study how to interface goal selection with the learning mechanism to automatically produce a sequence of learning situations of growing difficulty, also known as a curriculum. Our survey of the literature on automatic curriculum generation in reinforcement learning suggests that the limited results obtained are due to the poorly understood generalization properties of the agents. From this analysis, we propose a strategy for curriculum generation based on two components: the Learning Progress Hypothesis described in developmental sciences, which suggests that humans organize their learning situations into trajectories by maximizing their learning progress; and a task definition and parametrization that fosters inter-task transfer.

We apply this strategy to the continuous control of a simulated arm, with tasks based on a notion of precision in control. We show that the resulting agent finds a sound developmental trajectory, where it learns control with low accuracy requirements first, and then generalizes to learn faster finer-level control.

Beyond body control, developmental sciences have shown how important the abilities to manipulate objects and imitate others manipulating them are for infant development. In a second time, we thus propose to combine our curriculum generation process with imitation learning for object manipulation. Precisely, we describe and model a new learning situation, where a simulated discrete environment contains several objects, and where an expert, also simulated, manipulates some of the objects with its own goals and without acting as a demonstrator. This scenario mimics that of a caregiver not always acting as a teacher. Experiments in discrete and simulated environments show that the interaction of goal-directedness, reinforcement learning, curriculum generation and imitation leads the agent to display two new learning competences: a form of selective imitation, ignoring the expert when it displays behaviors already mastered or too difficult to master; and a teacher-learner interaction, where the agent follows the expert entirely in the precise case where the latter acts as a teacher, adapting its behaviors to the agent's level of mastery. This last competence is to our knowledge the first example of a reinforcement learning agent imitating a tutor out of intrinsic motivations.

In the end, the developmental approach we describe in this dissertation brings fruitful tools, and enables the emergence of new learning behaviors on the basis of simple initial ones. There is room for improvements in our experiments, especially in the direction of continuous rather than discrete environments, and towards more realistic non-simulated interactions. The approach also bears many interesting connections with recent works in the meta-reinforcement learning subfield, which we discuss. However, we suggest that the developmental ingredient that offers the most promising perspectives is the integration to our work of a capacity to act and interact in a hierarchical and observational way, and we propose a few ideas in this direction.

Résumé

L'apprentissage par renforcement est un paradigme d'apprentissage dans lequel l'agent maximise l'accumulation d'un signal de récompense. Ce paradigme rencontre un intérêt croissant depuis sa combinaison avec l'apprentissage profond, mais se confronte aujourd'hui à un paradoxe connu sous le nom de paradoxe de Moravec : reproduire les capacités sensorimotrices d'un enfant, telles la locomotion ou la manipulation d'objets, se révèle plus difficile qu'apprendre à jouer à des jeux abstraits difficiles comme le go ou le poker. Plusieurs évolutions récentes de l'apprentissage par renforcement cherchent à changer cet état de fait, et prennent leur inspiration dans les comportements d'apprentissage humain. Ce faisant, elles abordent des défis proches des objectifs d'un autre domaine de la recherche en Intelligence Artificielle : la robotique développementale. Celle-ci a pour but de modéliser et reproduire le développement de processus cognitifs naturels chez des agents artificiels, et ouvre des possibilités d'inspiration riches pour l'apprentissage par renforcement. D'un autre côté, l'apprentissage par renforcement profond fournit un cadre pour combiner ces processus, avec de fortes capacités représentationnelles pour traiter des environnements de haute dimension. Suivant ces observations, de nombreuses passerelles entre les deux domaines sont exploitées à l'heure actuelle, mais il reste beaucoup à explorer.

Dans ce manuscrit, nous étudions l'intersection de deux sujets cruciaux en sciences du développement, et leur application à l'apprentissage par renforcement : l'apprentissage social et la motivation intrinsèque. L'interaction et la motivation intrinsèque ont déjà été étudiées, séparément, en combinaison avec l'apprentissage par renforcement, mais avec l'objectif d'améliorer les performances d'agents existants plutôt que d'apprendre de manière développementale. Nous concentrons donc à l'inverse notre étude sur l'aspect développemental de ces deux sujets. Précisément, nous proposons plusieurs agents permettant d'étudier comment l'apprentissage par renforcement, la motivation intrinsèque et des formes simples d'apprentissage social peuvent interagir pour produire des comportements d'apprentissage sensés d'un point de vue développemental, et plus riches que ceux produits par chaque mécanisme séparément.

Les sciences développementales montrent que les capacités d'interaction sociale émergent de manière incrémentale à travers l'interaction progressive de mécanismes simples, notamment attentionnels et exploratoires. Le premier agent que nous proposons combine deux tels mécanismes : recherche de la nouveauté dans l'environnement et propension à suivre le regard d'autrui. Nous montrons que ces mécanismes, séparément comme ensemble, accélèrent de manière attendue l'apprentissage d'une tâche définie par des signaux de récompense. Plus étonnamment, les deux mécanismes ensemble amènent l'agent à effectuer la même tâche en l'absence de telles récompenses, toutefois de manière sous-optimale. L'agent recrée ainsi une interaction typique entre un apprenant et un tuteur, où le second guide l'attention du premier vers des objets qui suscitent sa curiosité pour lui faire découvrir de nouvelles possibilités d'action. Ce travail préliminaire montre cependant qu'il manque à l'agent une forme d'intentionnalité pour apprendre complètement sans récompense de l'environnement extérieur, comme des humains. Nous décrivons donc ensuite deux agents munis d'une forme primitive d'intentionnalité.

Nous utilisons l'apprentissage par renforcement dit *goal-conditioned* comme mécanisme de base pour le design d'agents intentionnels, et nous étudions comment interfacer la sélection de buts avec le mécanisme d'apprentissage pour produire automatiquement une séquence de

situations d'apprentissage de difficulté croissante, autrement dit un *curriculum*. Notre étude de la littérature sur la génération automatique de curriculum en apprentissage par renforcement suggère que les résultats limités obtenus sont dus aux capacités de généralisation encore mal comprises des agents. Partant de ce constat, nous proposons pour la construction du curriculum une stratégie fondée sur deux ingrédients : la *Learning Progress Hypothesis* proposée en sciences développementales, suggérant que les humains organisent leurs situations d'apprentissages en trajectoires qui maximisent leurs progrès ; et une définition et paramétrisation des tâches qui promeut le transfert inter-tâches. Nous appliquons cette stratégie au contrôle continu d'un bras articulé simulé, avec des tâches basées sur une notion de précision dans le contrôle. Nous montrons que l'agent identifie ainsi une trajectoire développementale sensée, où il apprend les tâches faciles de contrôle peu précis d'abord pour ensuite généraliser ses compétences et apprendre un contrôle fin plus rapidement.

Au delà du contrôle du corps, les sciences développementales ont montré l'importance pour l'apprentissage chez l'enfant de la manipulation d'objets et de la capacité d'imitation de telles manipulations. Dans un second temps, nous proposons donc de combiner la génération de curriculum proposée avec de l'apprentissage par imitation de manipulation d'objets. Précisément, nous décrivons et modélisons une nouvelle situation d'apprentissage, où un environnement simulé discret contient plusieurs objets, et où un expert également simulé manipule certains de ces objets avec ses propres objectifs et sans nécessairement agir comme un démonstrateur. Ce scénario reproduit celui d'un parent n'agissant pas toujours comme un éducateur. Les expériences montrent que l'interaction de l'intentionnalité, de la génération de curriculum et de l'imitation avec l'apprentissage par renforcement amène l'agent à déployer deux nouvelles compétences : une imitation sélective, ignorant l'expert lorsque ses comportements de manipulation sont déjà maîtrisés ou trop difficiles ; et une interaction de type tuteur-apprenant, où l'agent suit entièrement l'expert dans le cas précis où ce dernier agit comme un éducateur, adaptant son comportement au niveau de l'agent. Cette dernière compétence est à notre connaissance le premier exemple d'un agent d'apprentissage par renforcement imitant un tuteur par motivation intrinsèque.

En conclusion, l'approche développementale que nous adoptons au long de cette thèse apporte des outils féconds, et permet l'émergence de comportements d'apprentissage nouveaux sur la base de mécanismes initiaux simples. Nos expériences peuvent être améliorées, en particulier dans la direction d'environnements continus plutôt que discrets, et vers des interactions plus réalistes et non simulées. L'approche est aussi riche de connexions avec des travaux récents de meta-apprentissage par renforcement, que nous discutons. Cependant, nous suggérons que l'ingrédient développemental offrant les perspectives les plus riches est l'intégration à nos agents d'une capacité à agir et interagir de manière *hiérarchique et observationnelle*, et nous proposons plusieurs pistes dans cette direction.

Contents

Chapter 1

Introduction

The quest for artificial intelligence has been lasting for more than six decades now, and has proved a much more difficult journey than expected at its beginning, when researchers were predicting fully intelligent machines by the 1980s (McCorduck 2004). Along the path, difficulties led the AI community to split into as many and diverse communities as approaches to these difficulties. In this dissertation, we are interested in two of them — approaches and communities: Reinforcement Learning (RL) and developmental learning.

1.1 Motivations

RL is one of the three basic machine learning paradigms, along with supervised learning and unsupervised learning. We will formalize RL in Section 2.1, but for now let us say that it corresponds to the general idea of learning by trial-and-error how to act in an environment so as to maximize a cumulative reward signal (Sutton and Barto 2018, chap. 1). The basic RL agent-environment interaction loop can be seen in Figure 1.1.

1.1.1 A short history of RL

RL has had a rich and fairly autonomous history, with roots in psychology and animal behavior studies. The observation of trial-and-error learning dates back to the beginning of the 20th century (Thorndike 1898), and the idea of using this mechanism as a computational engineering principle has been first seen in the mid-1950s (Farley and Clark 1954; Minsky 1954). However, proper RL was not studied much until the 1970s, as research focused on supervised learning which was mistakenly seen as also trial-and-error learning at the time.

In 1972, ideas proposed by Klopf (1972) related trial-and-error learning to the massive empirical database of animal learning psychology. He made connections between trial-and-error and temporal-difference learning, a smaller thread of RL history rooted in animal learning psychology. Klopf's ideas were developed by Sutton and colleagues to produce models of psychological conditioning based on temporal-difference learning (Sutton 1978; Sutton and Barto 1981), and

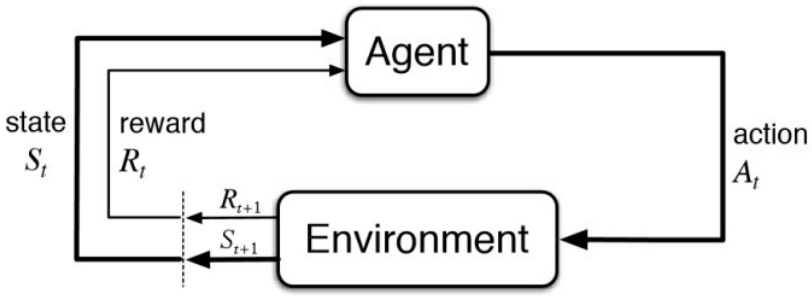

Figure 1.1 – The general reinforcement learning framework (Sutton and Barto 2018, chap. 3)

led to an architecture that truly combined trial-and-error learning and temporal-difference learning, called *actor-critic* (Barto et al. 1983; Sutton 1984).

Finally, trial-and-error learning and temporal-difference learning were brought together with the until then separated field of optimal control (Bellman 1966) by the development of Q-learning by Watkins and Dayan (1992). All along this simplified history of RL, many researchers not cited here contributed to the growth of the field and since then RL has remained active and greatly evolved to be applied to various domains, among which robotics (Kober et al. 2013).

A challenge for RL has long been to scale up to high dimensional environments, in particular using function approximation as a way to compute the utility of actions in a continuous environment. In the late 2000s, the separate machine learning field of connectionism, left out of AI in the 1980s and now called Deep Learning, provided other research communities with trainable function approximators more powerful than ever. It has since then gained much popularity and a part of the research on RL attempted to adopt it either. Indeed, in 2013, Mnih et al. (2015) showed that stable RL could be combined with the representational power of Deep Learning. Since then, the Deep Learning "frenzy" has gained the field of RL and made possible impressive achievements, among which mastering boardgames like go (Silver et al. 2016), real-time strategy games like Dota (OpenAI 2018) and Starcraft (Vinyals et al. 2019), or multi-player games like poker (Brown and Sandholm 2019).

1.1.2 Limitations of RL and deep RL

While deep RL appears very promising, a number of voices have insisted on its current limitations, some of them being limitations of Deep Learning in general, others specific to deep RL (Lake et al. 2017). A strong and recurring concern about Deep Learning and by extension deep RL is its need for tremendous amounts of samples (millions to hundreds of millions in deep RL) and for high computational power to process these samples in a reasonable time (Irpan 2018). This flaw partially explains why so few people have tackled the challenge of getting deep RL to work on real robots, with some exceptions like the work at Berkeley by the teams of Pieter Abbeel and Sergey Levine (Devin et al. 2017; Levine et al. 2016; Wang et al. 2019a).

Another concern specific to RL is the difficulty of exploration in large state spaces. A long-standing challenge in RL is indeed to deal with the *exploration-exploitation trade-off*: either obtain new knowledge to improve performance (exploration), or use the existing knowledge to do

so (exploitation) (Berger-Tal et al. 2014). It seems that deep RL algorithms are good candidates for strong (but sample inefficient) exploitation, but are not equipped to deal with exploration better than their shallow counterparts, as many also rely on simple methods for exploration such as "ϵ-greedy" policies (Sutton and Barto 2018, chap. 2).

Several approaches to address this difficulty have been proposed in RL and deep RL. Some are rooted in mathematical analysis of the problem, like regret minimization in the Multi-Armed Bandits formalism (Auer et al. 2002; Bubeck and Cesa-Bianchi 2012; Gittins 1979; Tang et al. 2016) or Bayesian uncertainty minimization (Osband et al. 2016). On the other hand, more and more works take instead inspiration from how human explore spontaneously in open-ended environments and propose heuristics for intrinsically motivated RL (Chentanez et al. 2004; Colas et al. 2018; Eysenbach et al. 2018; Schmidhuber 1991). A third natural idea to both help RL agents with sample efficiency and exploration consists in relying on interaction with humans, that led to the very wide domain of interactive RL (Knox and Stone 2009; MacGlashan et al. 2017; Warnell et al. 2017).

Finally, RL and deep RL, as most machine learning techniques, are suited to single task learning, but lack good capacities for learning reusable skills, transferring knowledge or competence from one setting to another, and multi-tasking in general. This limitation has been recently alleviated with different approaches: the research on learning reusable skills has mainly produced the rich framework of options and more generally has led to hierarchical RL (Barto et al. 2004; Kulkarni et al. 2016; Sutton et al. 1999a), while transfer and multi-task RL have become a central question of recent deep RL (Gamrian and Goldberg 2018; Hessel et al. 2018; Plappert et al. 2018; Taylor and Stone 2009; Weinshall et al. 2018).

Some of these approaches to RL are discussed in more details all along this dissertation, but for now a global observation is that most of them either directly take inspiration from the way human and in particular children learn, or simply reproduce this way because it is so natural: intrinsic motivation, interactive learning, skill learning, transfer learning are all intuitive mechanisms to learn and are known to play an important role in human cognitive development. It so happens that these ideas have already been the focus of an entire separate research community for some time: developmental robotics.

1.1.3 Definition and challenges of developmental robotics

In cooperation with developmental psychology, neuroscience and dynamical system theory, developmental robotics (also known as epigenetic or ontogenetic robotics) has been modeling the development of cognitive processes in natural and artificial systems since the beginning of the 2000s, in order to understand how these process emerge through physical and social interaction in the world (Lungarella 2007).

Developmental robotics differ from traditional AI, robotics and machine learning by two aspects: "a strong emphasis on body structure and environment as causal elements in the emergence of organized behavior and cognition" and "[realizing] artificial cognitive systems not by simply programming them (e.g. to solve a specific task), but rather by initiating and maintaining a developmental process during which the systems interact with their physical environments (i.e. through their bodies, tools, or other artifacts), as well as with their social environments (i.e.

with people, other robots, or simulated agents)" (Lungarella 2007).

To reach these goals, developmental robotics has identified a number of challenges, among which identifying and providing agents with core knowledge and built-in autonomous and social abilities to bootstrap the learning process, core motives such as creativity, curiosity, intrinsic motivation, and capacities for collecting relevant learning material and achieve self-determined evolution (Lungarella 2007). These different challenges are not entirely separated but advances on each one are often difficult to combine.

1.1.4 The origins of developmental robotics

The idea that human development might be a good avenue to understand and construct cognition actually dates back to early in the study of AI, when in the 1950s Alan Turing already suggested exactly that: "It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. That process could follow the normal teaching of a child. Things would be pointed out and named, etc." (Turing 1950). Unfortunately, the idea took shape only a lot later.

The field of developmental robotics is indeed an indirect product of a division in the AI community that occurred in the late 1980s, when Hans Moravec and Rodney Brooks proposed an alternative to high-level reasoning. Until then, intelligence was characterized as "the things that highly educated male scientists found challenging", like maths or complex games, while "[The] things that children of four or five years could do effortlessly, such as visually distinguishing between a coffee cup and a chair, or walking around on two legs, or finding their way from their bedroom to the living room were not thought of as activities requiring intelligence" (Brooks 2002). But after the first successes at using high-level logic or playing games like chess, researchers found out that these supposedly simpler problems like vision or motion were incomparably more difficult than high level game playing or mathematical reasoning.

This went on to be known as the Moravec's paradox: "it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility" (Moravec 1988). This realization led several researchers to propose a new approach to AI based on robotics, and argued that demonstrating real intelligence requires to have a body and to interact with an environment, taking inspiration from the natural evolution of complex behaviors in animal species.

These theories of *situated* and *embodied cognition* led robotics to split into many approaches from the early 1990s (Khamassi and Doncieux 2016), and among other bio-inspired approaches, developmental robotics started to focus on copying human development in the early 2000s (Lungarella et al. 2003; Weng et al. 2001).

1.1.5 Developmental robotics and RL

These brief descriptions of developmental robotics and its time-line show several interesting things. First we could consider modern deep RL to be coarsely at the same stage in its development as researchers were when developmental robotics emerged. On one side, deep RL seems to suffer

from the Moravec's paradox exactly as old approaches to AI did: mastering the most difficult game for humans appears more straightforward than exploring a room and learning to manipulate objects in a reasonable amount of time. On the other side, many of the improvements over standard RL that appear promising today focus on challenges that are individually especially close to those that constitute the objectives of developmental robotics.

Unfortunately, RL and developmental robotics are two fields of researchers with their communities, which evolved almost entirely separately and for a long time with different objectives: like most machine learning tools, RL was initially more focused on single task learning, while developmental robotics was by principle oriented towards general learning capabilities. As a result, now that (deep) RL and developmental robotics objectives have at least partially started to align, the two communities do not benefit from one another as much as they could, even if things have evolved in the right direction since the beginning of this work.

On one side, RL in general benefits from very promising properties from a developmental standpoint. First, its very idea of having an agent interact with its environment in order to learn clearly matches the requirement for the agent to be situated as advocated by developmental robotics. Then, Deep Learning brought RL the ability to perform end-to-end RL in several environments, meaning that the entire process from sensors to actions in an agent only involves a single neural network. This capacity offers a new interesting possibility from a developmental robotics standpoint: building agents whose development is directly related to the sensors and motor capacities that they are endowed with, coupled with strong computational capacities.

On the other side, developmental robotics has already performed in-depth, psychologically and neuroscientifically grounded studies of many tools the deep RL community is starting to add to its toolbox, like intrinsic motivation, core knowledge, social interaction and incremental learning.

In summary, the developmental robotics standpoint provides interesting insights on modern evolutions of RL, while RL provides a promising framework for combining several mechanisms deemed necessary by developmental robotics to build embodied and situated agents with cognition.

1.2 Dissertation focus

We have identified that several subfields of RL are facing today new problems close to those developmental robotics has been addressing since its birth. This observation is a strong motivation to approach those problems in RL with ideas from developmental sciences, as some researchers recently advocated (Sigaud and Droniou 2015), and this dissertation falls within this methodology. This approach to RL is still in its exploratory phase, and leads to a vast set of questions. In this dissertation our exploration focuses on the intersection of two important aspects of developmental sciences and their application to RL: interactive (or social) learning and intrinsic motivations.

1.2.1 Towards developmental interactive RL

Relying on interaction to speed up learning is a fairly natural idea, and has been the focus of many strands of research, both in standard RL (see Najar 2017, chap. 3 for a complete review)

and in deep RL (Christiano et al. 2017; Hester et al. 2018; Vecerik et al. 2017; Warnell et al. 2017).

However, building human-like learning interactive agents raises several challenges that are yet to be overcome. Generally speaking, social interaction encompasses the perception and interpretation of a variety of social signals from other agents, as well as the ability to respond to these signals appropriately. For example, collaboratively building a multi-part objects requires the ability from all parties to use, perceive and interpret properly interactive gestures like pointing, as well as the ability to synchronize actions with one another, imitate each other or build plans that involve each other. In the current state of research, almost none of these abilities has been successfully and reliably reproduced in an artificial agent, be it an RL agent or not.

Interaction protocols

To bypass these difficulties, most interactive agents rely on predefined interaction protocols (Vollmer et al. 2016), whether they rely on demonstrations, advice or evaluative feedback. In the case where social signals like words, gestures or other signals are providing advice, their meaning is usually given to the agent (Chernova and Veloso 2009; Clouse and Utgoff 1992; Cruz et al. 2015; Sivamurugan and Ravindran 2012; Thomaz and Breazeal 2007). In the case of imitation, the question of how to recognize when a teacher is providing a good demonstration is rarely tackled and the agent instead knows in advance that it should imitate what it is shown, or how to address sub-optimal demonstrations (Argall et al. 2009). Finally feedback interpretation is also often known in advance, as reward shaping (Knox and Stone 2009; Tenorio-Gonzalez et al. 2010) or policy shaping (Cederborg et al. 2015; Griffith et al. 2013).

These protocols are constraints for humans by restricting their possibilities to interact and provide the social signals they deem the most useful or appropriate. Also they may interfere with the assumptions of the learning algorithms used by the agent. For example, adapting the restricted framework of RL to rely on human-provided rewards, as in Knox and Stone 2009, 2010 or Warnell et al. 2017, confronts with the empirical observation that people use reward to provide guidance for future actions instead of feedback for past actions, as required by the theoretical assumptions of RL (Thomaz and Breazeal 2006). Another type of interference takes place when the human teacher is not an expert whereas RL algorithms are not initially equipped to deal with noisy reward signals or suboptimal demonstration data (Brys et al. 2015; Taylor et al. 2011).

Unlabeled instructions

Several attempts to relax the constraints implied by fixed interaction protocols have already been made. A possible route is based on the notion of pragmatic frames (Vollmer et al. 2016), inherited from studies on human development, described as recurrent patterns of interaction comprising multiple sequentially organized actions, like the sequence of gestures and words used by a caregiver to teach new words from images in a book to a toddler.

Another more studied relaxation is the use of unlabeled guidance signals, in the sense that they are not pre-associated to any meaning: a smile or a thumb-up from an external agent is not

initially associated with a positive outcome for example. To address these learning conditions, a potential direction consists in grounding social interaction within the task learning process and in turn using the learned meaning of instructions to boost the learning process. The idea is to learn simultaneously the task and the meaning of the related social interaction, the same way an infant may infer both knowledge of his environment and the meaning of a warning from his parent by trespassing this warning when alone and subsequently receiving a negative reward in the environment.

The use of unlabeled guidance signals has been studied in detail theoretically (Cederborg 2017) and in practice: experiments in virtual environments and on real robots have shown its efficacy to let human teachers shape a real robot behavior, through unlabeled instructions or teaching signals (Branavan et al. 2010; Najar et al. 2015, 2016).

In several other works, the agent has access to a set of hypotheses about possible tasks and infers which one is the target from low-level contextual instructions (Grizou et al. 2014; MacGlashan et al. 2014). In most cases, the agent needs some pre-existing signal of known meaning — like a reward from the environment — to be able to ground social signals. We let the reader refer to Najar 2017, chap. 3 for a complete review of this literature.

A developmental approach through joint attention

Researchers in developmental robotics have observed that the difficulties faced by many artificial agents to interact naturally with humans come from the fact that they are not endowed with any of the basic capabilities that underlie human-like social interaction. In particular, they have insisted on the importance on a specific category of such abilities: attentional behaviors.

Attention can be defined as "the temporally-extended process whereby an agent concentrates on some features of the environment to the (relative) exclusion of others" (Kaplan and Hafner 2006). This choice of features is determined by both bottom-up passive responses to salient events in the environment and active monitoring of parts of the environment. A specific attentional behavior appears crucial for interactions to be successful: joint attention. Following the definition of Kaplan and Hafner (2006), joint attention is "a coordinated and collaborative coupling between intentional agents where the goal of each agent is to attend to the same aspect of the environment".

A developmental treatment of interactive RL suggests to focus first on mimicking in agents the primitive forms of interaction demonstrated by infants, among which joint attention is crucial.

1.2.2 Towards task-independent RL

We have said in introduction that an important limitation of RL lied in its mono-task nature. Indeed, in standard RL, the reward signal comes from the environment of the agent, and takes the form of a mathematical quantity hand-crafted so that the maximization of the expected sum of reward signal coincide with the achievement of a human-defined task (Dewey 2014). This process is called reward engineering.

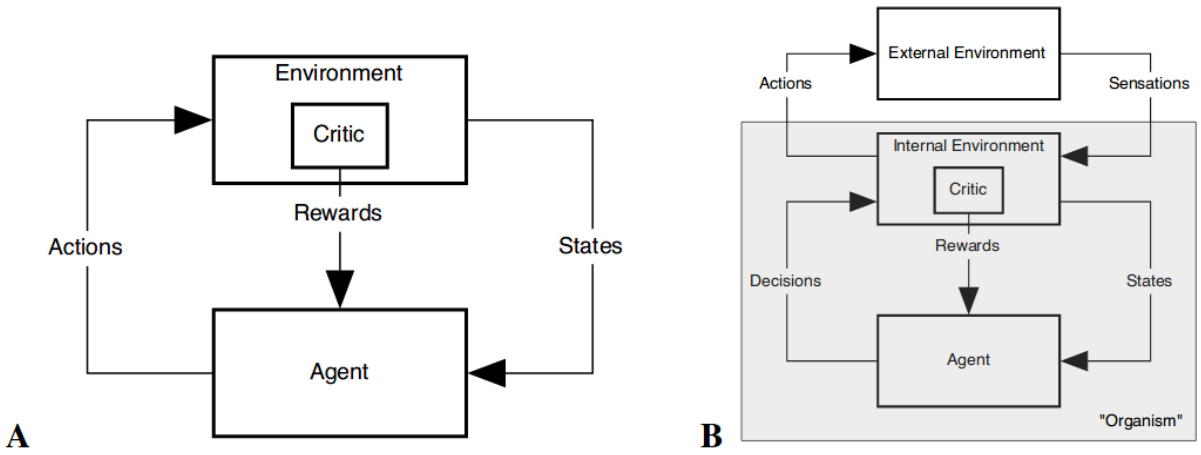

*Figure 1.2 – Agent-Environment Interaction in RL: **A**. the usual view; **B**. the intrinsically motivated standpoint. (Figure reproduced from Chentanez et al. 2004)*

This task-oriented view of RL that it leads to has been standard since the beginning of the field, as in many other machine learning fields that did not pursue general AI. However, it appears limited for the objectives the RL and developmental robotics communities seek to address now, as it goes against the possibility for the agent to learn an unbounded sequence of tasks, on which it may or may not have a degree of control, and for which rewards functions cannot be built at design time. Indeed, in real life, environments do not provide rewards: an infant who manipulates an object alone has no obvious incentive coming from the environment to manipulate it in one way compared to another.

As a consequence, the standard RL scheme is not fit for developmental learning. A possibility to replace ad hoc rewards from the environment is to rely on the guidance of others, as we have mentioned already. Another possibility consists in rethinking the origin of rewards in RL and consider that they should come from the agent itself instead of the environment. In some approaches to learning tasks in sequence, even the state and action spaces change between tasks and must be learned by the agent on-the-fly (Doncieux et al. 2018), but we do not address this additional difficulty in this dissertation.

Intrinsic motivations

A possible approach to rethink reward definition is to use intrinsic motivations. Psychologists indeed distinguish two types of motivation to act in an environment: intrinsic and extrinsic. Quoting Ryan and Deci (2000): "Intrinsic motivation is defined as the doing of an activity for its inherent satisfaction rather than for some separable consequence. When intrinsically motivated, a person is moved to act for the fun or challenge entailed rather than because of external products, pressures or reward."

On the other hand, "Extrinsic motivation is a construct that pertains whenever an activity is done in order to attain some separable outcome. Extrinsic motivation thus contrasts with intrinsic motivation, which refers to doing an activity simply for the enjoyment of the activity itself, rather than its instrumental value".

These definitions are not entirely formal and a more computational definition of intrinsic motivation has been proposed: intrinsic motivation is the search for certain "properties of the flow of sensorimotor values and of its relation to the knowledge and know-how of the system independently of the meaning of the sensorichannels that are involved" (Oudeyer and Kaplan 2009). From this computational definition, Oudeyer and Kaplan (2009) establish a typology of existing intrinsic motivation mechanisms within the reinforcement learning framework, and describe different types of intrinsic motivations and their associated reward computation — in RL, everything boils down to rewards in the end.

To understand how such intrinsic rewards fit in traditional RL, one must abandon the "standard view" of RL where the reward comes from the environment that evaluates the agent's behavior, and instead consider that rewards come from the agent itself (see Figure 1.2): "an animal's reward signals are determined by processes within its brain that monitor not only external state but also the animal's internal state. The critic is in an animal's head." (Chentanez et al. 2004).

1.2.3 Conclusion

We have identified two possibilities among others to make better RL agents: 1) focusing on using low-level social behaviors rooted in developmental sciences to interact, instead of high-level externally defined ones; and 2), replacing task-dependent extrinsic motivations in RL with purely task-independent intrinsic motivations. The guideline of this dissertation is that these two choices have never been considered together in RL, and that we thus experiment with their interaction in several contexts. Precisely, we follow from start to finish the general idea of proposing agents whose developmental trajectories can be influenced positively by properly designed combinations of social interaction and intrinsic motivation mechanisms.

1.3 Outline of the dissertation

The path for experimentation we just described still leaves open the study of diverse possibilities for both social behaviors and intrinsic motivations, and our approaches to design each of these mechanisms as well as their interactions have greatly evolved over time, mainly in reaction to the progressively found results.

Also, although our initial approach was focused on improving RL with ingredients from developmental sciences, writing this document led us to progressively reach a more global understanding of our works, and to see RL more as an other ingredient, admittedly with specific properties, inside a potential recipe for more capable agents. Finally, the evolution in our point of view that occurred while writing this dissertation has also been reflected in our understanding of the connections between our approaches and many subfields of RL that are progressively gaining interest.

In the next three chapters, we present chronologically the works we have accomplished, and we place importance on making the train of thought behind our choices as clear as possible all along their presentation. Then, in order to reflect the evolutions of our point of view and insights during the writing of this document, a discussion chapter completes our main contributions and details the aforementioned connections, right before a global conclusion.

An important side note is that this dissertation is positioned at the intersection of several fields of RL that have almost not been combined before, so that it would be difficult to present a true global "related work" chapter that is relevant for all our works. Instead, we have simply globally motivated our interest for developmental interactive learning and intrinsic motivations, and we describe the context and related work specific to each of our works in the introduction of their corresponding chapters. Likewise, the final discussion chapter describes several subfields of RL, and contains references to their literature in order to make their connection with our work clearer, but the objectives pursued in these references are always strongly different from ours and are thus not directly related work.

In the following paragraphs, we introduce the motivations behind each of our works, as well as the employed methods and their contributions.

1.3.1 Attention coordination and novelty search for task solving

In Chapter 2, we present a preliminary attempt at integrating gaze-following and novelty search with discrete model-based RL, in order to learn simple simulated pick-and-place tasks. We first keep task-dependent extrinsic rewards and then we attempt to remove them. The main objective is to identify the challenges this integration raises, and the kind of learning behavior the combination of these mechanisms can lead to, in presence and absence of extrinsic rewards. This work has been presented to the second workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics (BAILAR) at IEEE RO-MAN 2017 (Fournier et al. 2017).

Overall, this first work shows that an RL agent can learn faster a standard RL task by watching where its tutor watches, and that this mechanism combined with novelty search is a first step to get rid of the non-natural extrinsic rewards traditionally needed with RL. However, experiments in this last case lead to the conclusion that an essential ingredient lacks for the agent to really benefit from the interaction of attention coordination and intrinsic motivation: intentionality.

1.3.2 Intentionality and competence progress for control

Following our conclusion, the work presented in Chapter 3 puts temporarily aside interactive learning, and focuses on using goal-conditioned RL to make agents capable of goal-directed behaviors. In particular, we study the new possibility for the agent, offered by goal-directed learning, to discover a curriculum of learning situations better than random goal selection, and thus learn faster.

Our contributions in this chapter are the following:

- the survey of existing curriculum learning agents in RL and how they relate to competence-based intrinsic motivations as formulated in (Oudeyer and Kaplan 2009),
- an explanation of why curriculum learning in RL is especially difficult, leading to the main contribution:
- a multi-goal RL algorithm that successfully performs progress maximization based on explicit competence measurements.

This latter algorithm has been described in detail in Fournier et al. (2018).

1.3.3 Competence progress and imitation learning for object manipulation

In Chapter 4, we go back to combining intrinsic motivation and interactive learning, this time with an intentionality component in place. The work in this chapter is motivated by several observations. First infants are intrinsically motivated to manipulate objects in their environment, and they actively select objects to play with. Second, infants are able to observe their caregivers including when they are not teaching, and selectively imitate them nonetheless. Third, objects in the environment are not all equally learnable at a given point in development, and some can be related to others.

Following these observations, we propose in Chapter 4 a novel and original RL setting, where:

- the environment contains interrelated objects of variable learnability,
- an other unconstrained agent manipulates the objects for its own purposes, not necessarily to provide demonstrations,
- the learner RL agent explores how to control objects separately in the environment, with no task predefined,
- and relies on a Competence Progress (CP)-based curriculum to identify which object to practice and which object to imitate the other agent practice.

We first propose a discrete-state discrete-action model of multi-objects structured environments. Then we augment goal-conditioned RL with automatic curriculum learning studied in Chapter 3 to deal with single object control in multi-objects environments, and we combine it with the imitation learning algorithm Deep Q-Networks from Demonstrations (DQNfD) to perform multi-goal imitation learning. The resulting algorithm is called Curriculum Learning and Imitation for Control (CLIC), and is evaluated in multiple instances of multi-objects environments with our environment model.

The results show that CLIC features independent reusable control of the environment objects when possible, and effectively leverages its curriculum to identify then ignore non-reproducible or already mastered behaviors, both when imitating and when exploring. Additionally, the combination of automatic curriculum learning and imitation learning enables CLIC to truly learn to follow demonstrations from a tutor as long as they make it progress, which is thus the first demonstrated case in RL of a successful feedback loop between intrinsically motivated learning and interactive learning. The behavior obtained, albeit simple, is not obvious for an agent which chooses what to learn, and closely matches the way a human caregiver would influence the order in which an infant discovers some toys, for example.

This work also proposes to our knowledge the first algorithm combining several essential mechanisms from a developmental standpoint — goal-oriented learning, imitation learning and automated curriculum learning — and has been the focus of an article currently in press for the Special Issue on Continual Unsupervised Sensorimotor Learning at IEEE-TCDS 2019 (Fournier et al. in press).

1.3.4 Discussion and conclusion

In Chapter 5, we provide an alternative and more global point of view on our set of experiments, based on a "complex system" approach to the analysis of intelligent agents properties. Precisely, we synthesize our different contributions by insisting on how our agents' learning capabilities result from the interaction of both several mechanisms for learning and interacting, and specific properties from the environments they evolve in.

Then, we detail our recent understanding of the connections between our developmental approach to the RL paradigm and several other, new or older, evolutions of the RL paradigm: hierarchical RL, observational RL and meta-RL. Finally, we complete this analysis with a discussion on usual models for objects in environments, as the latter are present in most of the works we describe and in many studies in the related literature.

Chapter 2

Combining novelty search and gaze-following

We focus in this chapter on combining gaze-following and novelty-search, and we test the resulting agent in a discrete model-based RL task learning scenario.

We have said in introduction that a specific attentional mechanism called joint attention played a significant role in allowing for successful human-human interactions. In detail, Kaplan and Hafner (2006) identify four essential prerequisites for joint attention to happen:

1. attention detection, that is the ability to track the attentional behavior of other agents,
2. attention manipulation, that is the ability to manipulate the attention behavior of other agents,
3. social coordination, that is the ability to engage in coordinate interaction with other agents,
4. intentional understanding, that is the knowledge that oneself and other agents are intentional.

These prerequisite for joint attention are not fully present from birth and instead co-develop, starting from mutual gaze and early identification with others, towards gaze detection or pointing, and then gaze-following or understanding of action plans. While these mechanisms are still being studied in humans, the developmental robotics community has already proposed models of joint attention, often centered around one of its isolated elements. Unfortunately, as stated in (Kaplan and Hafner 2006), "although clear milestones can be identified, the precise developmental route that leads to mastering the necessary skills for joint attention is largely unknown", and no system has achieved true joint attention in the sense defined above.

Recreating full joint attention in artificial agents is still too intricate a task, even though it would be interesting to observe its uses for RL. In this first work, we simply make the hypothesis that the agent considers it rewarding to be in synchrony of gaze with its tutor.

This hypothesis can be interpreted either as saying the utility of gaze-following behaviors has been learned before, with a method similar to (Jasso et al. 2006) for example, or more generally as a translation of the idea that infants tend to favor situations in which they are in

synchrony with their caregivers (Feldman 2007). Overall, this approach to let the agent impact its developmental trajectory is mainly inspired by an interaction scheme that can be seen in adult-infant interactions: the caregiver directs their attention towards an object new to them, and upon recognizing that it is new, they start getting interested into it and practicing it.

2.1 Background

To combine a motivation based on social gaze to autonomous intrinsic motivation in an RL agent, we build upon the existing intrinsically motivated RL agent TEXPLORE-VANIR described in Hester and Stone 2012, which is an intrinsically motivated version of the TEXPLORE model-based reinforcement learning framework (Sutton et al. 1999b). First we introduce the basics of RL and model-based RL, and then we go on to detail the inner workings of TEXPLORE. Readers familiar with RL can skip the next subsection.

2.1.1 Reinforcement learning background

Markov Decision Processes In standard RL, the environment is formulated as a Markov Decision Process (MDP), which consists of a tuple $\{\mathcal{S}, \mathcal{A}, T, R\}$, where:

- $\mathcal{S}$ is the set of states s,
- $\mathcal{A}$ is the set of actions a available to the agent,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function of the environment, such that $T(s, a, s') = P(s'|s, a)$ is the probability of reaching s' from state s by executing action a, conditionally independent of states visited before s (Markov property),
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function of the environment such that the agent receives $R(s, a, s')$ when it went from state s to state s' with action a.

Policies Learning in an MDP means finding an action selection strategy maximizing the amount of rewards received. A function $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ allowing to choose an action in a given state is called a *policy*, and is such that $\pi(a|s)$ represents the probability of choosing a in s, with $\int_{\mathcal{A}} \pi(a|s)da = 1$.

Returns If the sequence of rewards received after time step t is $r_{t+1}, r_{t+2}, \ldots$, we call *discounted return* the quantity

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.1}$$

where γ is called the *discount factor* and determines how much value is placed on future rewards. The discount factor is often included in the tuple defining an MDP: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$.

Policy evaluation The sequence of rewards obtained by the agent is a consequence of its choice of actions, and thus of its policy. Consequently, a policy π can be evaluated in terms of the average discounted amount of reward the agent receives following it:

$$J(\pi) = \int_{\mathcal{S}} T_0(s_0)\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| \pi, s_0\right] ds_0, \tag{2.2}$$

where $0 \leq \gamma < 1$ and T_0 is the probability density over the initial state. The purpose of an RL algorithm is to find an optimal policy $\pi^\star \in \arg\max_\pi J(\pi)$.

Q-value functions To identify good states or good actions in states, most RL algorithms involve estimating *value functions*, that estimate how much discounted return can be expected from a given state, or from a given action in a given state. Since Widrow et al. 1973, the term *critic* has also been used for value functions. The rewards the agent can expect depend on the actions it will take and thus a value function is defined with respect to a policy. In this work, we only use state-action value functions, also called Q-value functions, so we let the reader refer to Sutton and Barto 2018, chap. 3 for the definition of the state value function V. Given a policy π, for any state $s \in \mathcal{S}$, the state-action value function Q^π is defined as

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | \pi, s_t = s, a_t = a\right], \tag{2.3}$$

and evaluates the goodness of selecting action a in state s and then following policy π.

Greedy and optimal policies Given a state-action value function Q^π, we can define a *greedy policy* with respect to Q^π by

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q^\pi(s,a). \tag{2.4}$$

Finally we can define the optimal Q-value function from the optimal policy, writing $Q^\star(s,a) = Q^{\pi^\star}(s,a) = \max_\pi Q^\pi(s,a)$ (Puterman 2014).

Q-learning In this dissertation, we will heavily rely on the Q-learning algorithm by Watkins and Dayan 1992 to learn optimal policies in MDPs, which is reproduced in Algorithm 1. We let the reader refer to Sutton and Barto 2018, chap. 6 for complete explanations and proofs of the algorithms. In practice, a general proof of convergence to the optimal value function is available under the conditions that the state action pairs are represented discretely, and that all actions are repeatedly sampled in all states in order to ensure sufficient exploration.

Algorithm 1 Q-learning for estimating $\pi \approx \pi^\star$

1: **Input:** step size $0 < \alpha \leq 1$, ϵ
2: **Initialize:** $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, except that $Q(terminal, \cdot) = 0$
3: **Loop** for each episode
4: Initialize s
5: **Repeat** for each step
6: Choose a from policy derived from Q (*e.g.*, ϵ-greedy)
7: Take action a, observe r, s'
8: $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_a Q(s', a))$
9: $s \leftarrow s'$
10: **Until** s is terminal

Model-based RL In model-based RL, the agent estimates its transition and reward function from experience. The agent can use its models to produce *simulated experience* and improve its policy on this base. A standard model-based algorithm is presented in Algorithm 2. In essence, lines 9 to 12 update the Q-value function with simulated experience before acting anew.

Algorithm 2 Model-based Q-learning for estimating $\pi \approx \pi^\star$

1: **Input:** step size $0 < \alpha \leq 1$, ϵ, model M of the environment
2: **Initialize:** $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, except that $Q(terminal, \cdot) = 0$
3: **Loop** for each episode
4: **Repeat** for each step
5: $s \leftarrow$ current state
6: Choose a from policy derived from Q (*e.g.*, ϵ-greedy)
7: Take action a, observe r, s'
8: Train model M with tuple (s, a, s', r)
9: **Loop** n times
10: $s, a \leftarrow$ random state and action
11: $r, s' \leftarrow M(s, a)$
12: $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_a Q(s', a))$
13: **Until** s is terminal

2.1.2 Texplore

The full algorithm we used in this chapter is based on components from the TEXPLORE and TEXPLORE-VANIR algorithms. TEXPLORE is an advanced algorithm designed to be applicable to a broad range of robot control tasks. In particular, it relies on rapid domain model building with decision trees, deals with continuous domains with linear regression trees and can act in real time using parallel MCTS. This combination of characteristics made the TEXPLORE algorithm appear as a good architecture for easy study of our objectives, and for future applications to real robots. In the following, we describe its minimal building blocks for our requirements.

Model-learning and planning

The TEXPLORE algorithm follows a model-based RL scheme, where the environment is modeled as a factored Markov Decision Process (MDP) (Boutilier et al. 1999; Degris and Sigaud 2013). The current factored state $s = (s_1, \ldots, s_n)$ of the environment, and the action a are the inputs of the model, and the next state s' and the obtained reward r are its output. The factorization property of the model enables learning separate predictions for each state features s_i (and for the reward) and recombining them thereafter into a full predicted state. This is only valid under the assumption that all features transition independently. In TEXPLORE, each separate feature predictor is a random forest comprising multiple univariate C4.5 decision trees, trained on different subsets of the experiences.

Tabular models of the reward and transitions are re-built from the newly trained predictors when outdated, and the planning part of the model-based algorithm (corresponding to lines 9-12 in the generic description of model-based RL of Algorithm 2) queries these models. TEXPLORE relies on an advanced planner based on the UCT algorithm for exploration (Kocsis and Szepesvári 2006). In TEXPLORE, $R(s,a) = r_{pred}(s,a)$ as rewards only comprise the reward predictions by the reward model given input state and action.

The obtained simulated experience provides action-reward-state sequences $a_t, r_{t+1}, s_{t+1}, \ldots, r_{t+M}, s_{t+M}$, where M is the maximum simulation depth. All along the simulated sequence, Q-value updates for state-action pair $(s_{t'}, a_{t'})$ write:

$$Q(s_{t'}, a_{t'}) \xleftarrow{\alpha_{UCT}} (1-\lambda) \sum_{i=1}^{M+t-t'} \lambda^{i-1} \times \left[\left(\sum_{k=1}^{i} \gamma^{k-1} R(s_{t'+k-1}, a_{t'+k-1}) \right) + \gamma^i \max_{a'} Q(s_{t'+i}, a') \right]. \quad (2.5)$$

The notation $x \xleftarrow{\alpha} y$ is short for $x \leftarrow (1-\alpha)x + \alpha y$, λ is the eligibility trace parameter (Sutton and Barto 2018, chap. 12) and γ is the discount factor. α_{UCT} is a learning rate specific to the UCT algorithm (Sutton et al. 1999b).

Novelty reward

In TEXPLORE-VANIR, the reward R is enriched by two intrinsic motivations favoring 1) high-uncertainty areas of the environment and 2) high-novelty areas of the state-action space. The present work only uses the second one based on novelty.

Novelty-search is a form of knowledge-based intrinsic motivation. Knowledge-based intrinsic motivations are based on "measures of dissonances (or resonances) between the situations experienced by [the agent] and the knowledge and expectations that [the agent] has about these situations" (Oudeyer and Kaplan 2009). In RL and deep RL, knowledge-based models for intrinsic motivation have been used to drive exploration in environments, in combination with sparse extrinsic rewards (Hester and Stone 2012; Tang et al. 2016) or without extrinsic rewards (Burda et al. 2018; Pathak et al. 2017), but never combined with social interaction.

For a given state-action pair $(s, a) = (s_1, \ldots, s_n, a)$, this additional NOVELTY-REWARD $N(s, a, V)$ is based on the normalized distance between (s, a) and the set V of known state-action pairs kept up to date. It writes:

$$N(s, a, V) = \underset{(s_V, a_V) \in V}{\arg\min} \|(s, a) - (s_V, a_V)\|_1 . \tag{2.6}$$

2.2 Methods

We augment the TEXPLORE-VANIR algorithm in several ways: 1) a tutor is present in the environment, 2) both the tutor and the agent are attentional and can gaze at objects in the environment, 3) the agent's action achievements are now conditioned to their coordination with its gaze, and 4) a second reward shaping mechanism is added, in order to favor states in which the agent's gaze follows the tutor's.

2.2.1 Environment and attention-action coordination

We designed a virtual pick-and-place task in a grid world environment, shown in Figure 2.1 to evaluate our future agent. Two infinite sources of blocks are located in two corners of the grid, while two boxes are located in the remaining corners. A move cannot be made against a wall, and the agent (the hand in Figure 2.1) is positioned at the center of the grid at the start of an experiment. The agent state consists of 11 different features: its x_a and y_a coordinates in the room, whether it carries a block, and the x_i and y_i coordinates of all four objects in the grid.

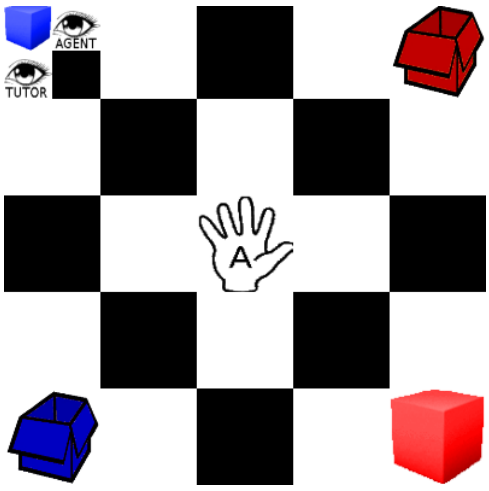

Figure 2.1 – The environment is a 5 by 5 grid with two sources of red and blue blocks (the cubes) and a box of each color. The agent is defined by its position (the hand) and its actions need to be coordinated with its gaze (its eye). The tutor only exists through its gaze, and looks where it is best for the agent to also look at.

In order to add a basic attentional mechanism, the agent and a tutor's gaze at specific points in the grid (the eyes in Figure 2.1). The coordinates of the agent's gaze is thus added to its state, so that we now have $s = (s^{env}, s^{gaze})$ where s^{env} is comprised of pure environment observations

from the agent while s^{gaze} is its gaze position. Ten actions are available: the agent can move one step in each cardinal direction, look at all four objects, pick a block and put one in a box. The pick action (resp. put action) requires the agent to be located at a block position (resp. a box position), while holding nothing (resp. holding a block).

Finally, an attention-action coordination requirement is added to the experiment: apart from looking at an object which is always possible, an action realization is conditioned to the agent's gaze. A move must now be in the direction of the agent's gaze, while the pick (resp. the put) action are now successful only when the agent's gaze is directed towards the block (resp. the box). All actions are deterministic and when one is not successful, the agent state remains unchanged.

The tasks on which the agent is evaluated during training and the policy of the tutor are defined at the beginning of each corresponding results descriptions in the next section.

2.2.2 Gaze following motivation

To induce gaze-following behaviors, a reward J is given if the agent's gaze matches the tutor's: for a given state $s = (s^{env}, s^{gaze})$, and a given observation by the agent of the current tutor's gaze σ^{obs}, we write:

$$J(s, \sigma^{obs}) = \begin{cases} 1, & \text{if } s^{gaze} = \sigma^{obs}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

2.2.3 Final algorithm

The final learning algorithm is described in Algorithm 3, where additional steps with respect to the initial components of TEXPLORE-VANIR are highlighted.

Lines 7-13 comprise building experience that includes the tutor's reactions and training the feature predictors from this experience. Lines 14-18 perform the tabular models updates from the predictors. The reward computations thus include incentives for novelty and gaze-following behaviors by shaping R in (2.5) with (2.6) and (2.7), writing:

$$\begin{aligned} R(s_{t'+k-1}, a_{t'+k-1}) = & \, r_{pred}(s_{t'+k-1}, a_{t'+k-1}) \\ & + \nu N(s_{t'+k-1}, a_{t'+k-1}, V) \\ & + \mu J(s_{t'+k-1}, \sigma^{obs}_t), \end{aligned} \tag{2.8}$$

where ν and μ are tunable parameters that determine the importance granted to novelty and gaze-following behaviors respectively.

It is important to note that $R(s_{t'}, a_{t'})$ relies on σ^{obs}_t for all t' considered in (2.5): the tutor's gaze is not part of the agent state and thus no prediction is made about its future. Thus updates at simulated step t' use the value σ^{obs} at step t. Finally, line 19 corresponds to the UCT-PLANNING algorithm where the Q-table is updated following (2.5) and (2.8).

Figure 2.2 shows the full model-based learning framework with explicit use of the tutor's gaze signals to modulate the tabular reward model.

Algorithm 3 TEXPLORE-VANIR with guidance by gaze-following

Input: An actor, a tutor and an environment
Initialize $Q(s, a) = 0$, $\forall s, a$
Environment model M $\leftarrow$ empty model
Starting state $s \leftarrow s_0$, known states $V \leftarrow \varnothing$
$\pi_{tutor} \leftarrow$ predefined policy, tutor state $\sigma \leftarrow \sigma_0$
Loop for each episode
    Repeat for each step
        $a \leftarrow \arg\max_{a'} Q(s, a')$
        Actor takes action a, observes r, s'
        Tutor updates gaze σ following $\pi_{tutor}(s')$ ▷
        TRAINPREDICTORS($\langle s, a, s', r \rangle, M$)
        $V \leftarrow V \cup (s, a)$
        $\sigma^{obs} \leftarrow \sigma$ ▷
        for all state s, action a **do**
            $T(s, a), R(s, a) \leftarrow$ UPDATEENVMODEL(s, a, M)
            $R(s, a)$ += $N(s, a, V)$
            $R(s, a)$ += $J(s, \sigma^{obs})$ ▷
        end for
        UCTPLANNING(Q, T, R)
        $s \leftarrow s'$
    Until s is terminal

Hyperparameters The parameters λ, γ and M are those of TEXPLORE-VANIR and remain constant in all experiments: $\lambda = 0.1$, $\gamma = 0.9$ and $M = 100$. Q-values are initialized uniformly at random around zero. All experiments consist of 30 trials starting in the exact same conditions (parameters, starting position, etc.). Each trial comprises 800 learning steps. In addition to the accumulated reward, we store for each action a taken in state s the proportions of each of the three rewards in $Q(s, a)$.

2.3 Experiments on task-oriented exploration

Our first set of experiments aims at finding out whether there can be some synergy between intrinsic motivation and gaze following, and in particular studies how the weights of the incentive for gaze-following and that of curiosity impact the discovery and the achievement of a classically defined RL task in a new environment.

Task definition The task here is called ANY: the agent must repeatedly go to *any* block, pick it up, carry it to *any* box, and place it inside. The tutor's policy in this environment is fixed for each task. Each time the agent ends up holding nothing during exploration, the tutor picks a source of blocks, choosing randomly. Then the tutor keeps looking at the source until the agent takes a block from one of the two sources. When the agent picks a block, irrespective if it is the one the tutor looked at, the tutor chooses a box randomly and looks at it until the agent has

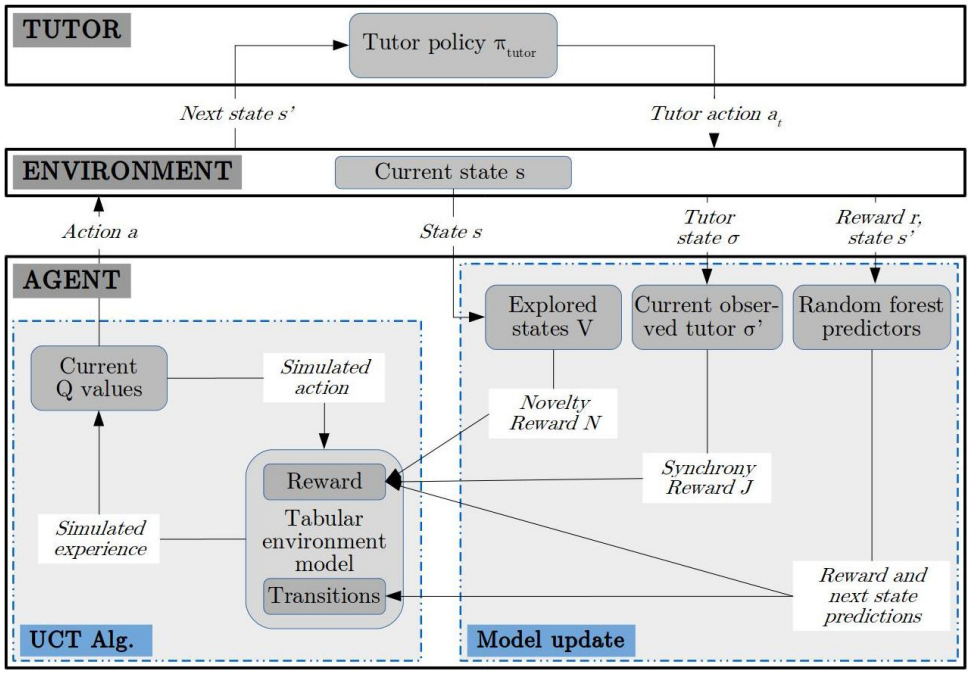

Figure 2.2 – The agent model is an extension of the classical model-based reinforcement learning scheme, where the tutor behavior is explicitly taken into account to product virtual experience. The agent exploits the accumulated experience to train predictors, from which the tabular environment model is built. Novelty and gaze-based motivations modulate the tabular reward model to favor specific state-action pairs. The agent obtains the best action by computing Q values from simulations based on the environment model.

placed the block in it. Of course, this policy is an oversimplification of natural gaze mechanisms. It can be understood as a way to model how a caregiver could point towards an object for a child to interact with, and keep sending attentional signals in the direction of the object until the child has indeed interacted with it.

First, the contribution of the tutor for learning the task ANY is examined. Performances of the agent on the pick-and-place task with and without taking the tutor's gaze into account ($\mu = 1$ and $\mu = 0$) are compared. As we are mainly interested in guiding exploration, we focus on the first few hundreds iterations. Figure 2.3 plots the accumulated rewards versus the number of steps taken by the agent, averaged over 30 trials. The figure shows results for different intrinsic motivation weights: $\mu = 0$ and $\mu = 1$ on one side, $\nu = 0$, $\nu = 1$, $\nu = 5$ and $\nu = 10$ on the other.

Experiments where the agent takes the tutor's gaze into account appear more efficient at discovering the task than their counterparts with intrinsic motivation only. Differences between results for $\nu = 5$ and $\nu = 10$, be it with or without a tutor, are not meaningful on this plot and require further investigation.

Figure 2.3 shows different performance gains for a constant appeal to joint attention ($\mu = 1$) depending on the importance granted to curiosity through ν. This suggests a coupling between intrinsic motivation and gaze-following, which we analyze in Figure 2.4. To this end, we make ν vary between 0 and 20 with a fixed $\mu = 1$ and focus on the final reward accumulated at step 800 only. For each value of ν, Figure 2.4 shows on a vertical line the distribution of results obtained

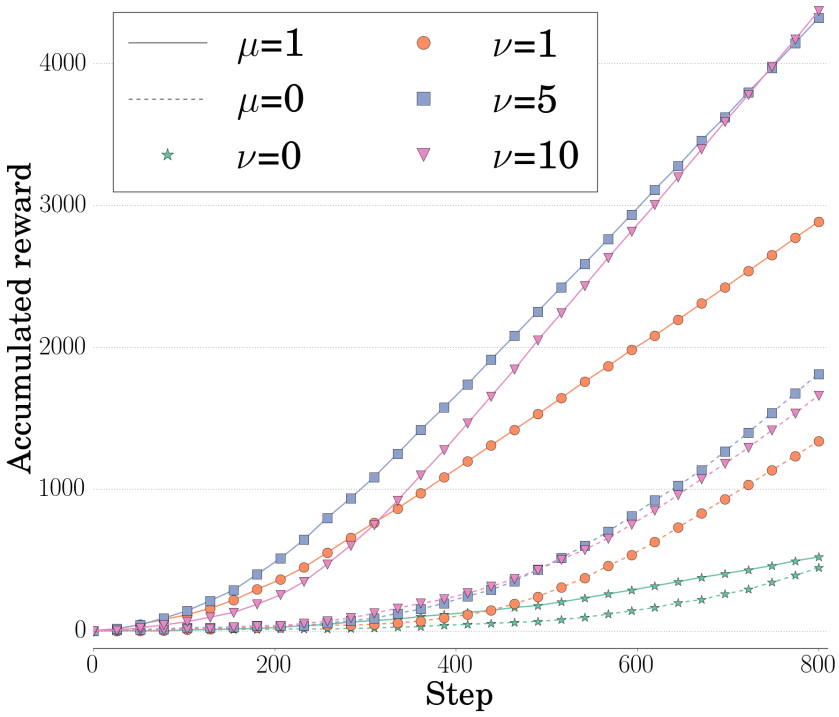

Figure 2.3 – Accumulated reward for the task ANY *versus number of steps taken, averaged over 30 trials, with and without a tutor ($\mu = 0$ or $\mu = 1$) for different parameters ν. The incentive to follow the gaze of the tutor clearly leads to better policies. Performances are best for $\mu = 1$ and $\nu = 10$.*

over the 30 trials. To account for trials giving identical results, these 30 values are binned in intervals $[0, 1000]$, $[1000, 2000]$, ..., $[5000, 6000]$ and each bin is shown as a circle centered at the average value of the bin, and of size proportionate to the number of elements in the bin. For instance, with no curiosity at all ($\nu = 0$, on the left), most trials have not discovered the task nor obtained any reward, hence the large dot on the no-reward horizontal axis.

A significant proportion of the trials for $\nu \leq 10$ still obtains few to no reward at all, as illustrated by a remaining circle close to zero; on the other hand, the majority of other trials reach high end results as shown by large dots for high intervals. Between these two behaviors lie very few samples. These results indicate that either the agent discovers the task early enough and then exploits its discovery to reach a high final accumulated reward, or it remains stuck in a form of inefficient exploration and receives no reward at all. The absence of intermediate end-result (few values between 1000 and 4000 for $0.5 \leq \nu \leq 10$ in particular) is a consequence of this alternative. If we evaluate the chosen parameters based on consistency over trials and on the actual 800 step accumulated reward, performances are best for $\nu = 10$ and $\mu = 1$. The 10/1 ratio between both motivations approximately compensates for the fact that the bonus obtained from novelty is always rather small compared to that from the gaze (the right side of (2.6) is rarely close to 1).

The observed behavior is easily interpreted in the light of the role played by each reward mechanism during exploration. To measure these roles, we use the proportions of each reward/motivation mechanism inside the Q-value corresponding to the chosen action at each

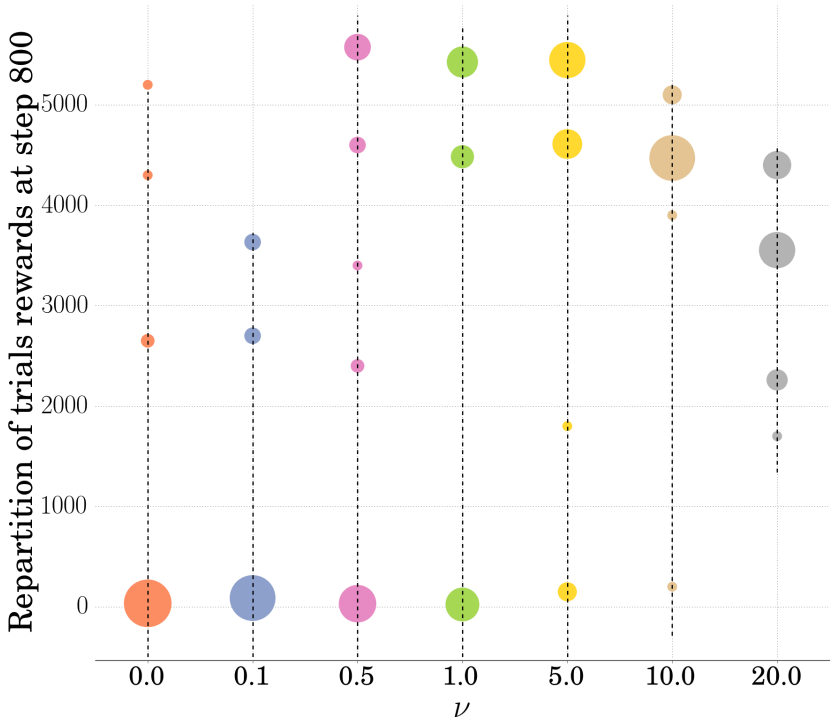

Figure 2.4 – Distributions of the accumulated reward at step 800 over the 30 trials for different values of ν, each corresponding to a vertical line, for a fixed $\mu = 1$. On each vertical line, the 30 trial results are binned in the six $[0, 1000]$, $[1000, 2000]$, ..., $[5000, 6000]$ intervals. Each bin is then displayed as a circle at height equal to the average value of the bin and of size proportionate to the number of results in the bin. The best results in term of both accumulated reward and consistency over trials are obtained for $\nu = 10$: few low reward results and many high reward results.

iteration, which we store all along the trials. They indicate how much each reward/motivation mechanism is responsible for the action of the agent. The evolution of these proportions, shown in Figure 2.5 on the left for two very different behaviors (weak versus strong curiosity), indicates that low performances on the left of Figure 2.4 correspond to the agent only looking where the tutor looks, without searching to discover the environment and with growing but weak interest for the task.

The decrease in performance observed for high values of ν stems from the opposite behavior where the agent's goal is the complete discovery of the environment, independently of the task or the tutor. The benefit of this strategy is that it almost guarantees random task execution and avoids no-reward trials. A successful trade-off between curiosity, joint attention and external reward exploitation is reached for $\nu = 10$ (Figure 2.5, right): novelty-based motivation appears critical at the beginning of learning and then gives way to external reward signal discovered, while the tutor acts as a permanent discrete guidance. This combination leads to a more effective task-oriented exploration of the environment than any of the drives taken separately.

We can conclude by saying that gaze following and curiosity can complement each other as exploration mechanisms, with gaze-based guidance leading the agent to favor some *states* and curiosity favoring the choice of unexplored *actions*.

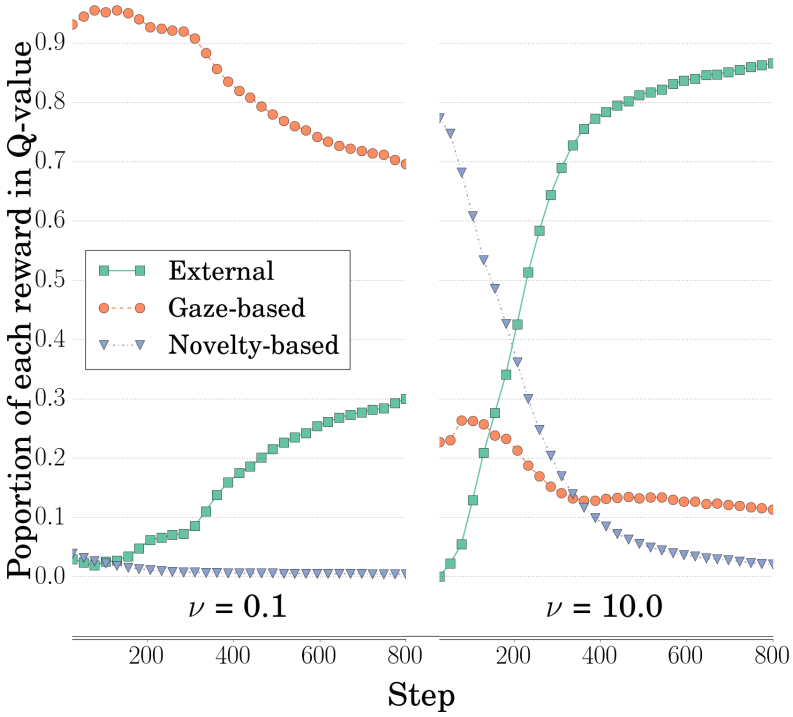

Figure 2.5 – Importance of each reward mechanism in action-selection during the first 800 steps, for the two combinations $[\mu = 1, \nu = 0.1]$ *(left) and* $[\mu = 1, \nu = 10]$ *(right). The evolution of the proportion of each member of* (2.8) *in the Q-value defining the next action is displayed. Successful combinations of motivations (right) enable novelty to play the main role at the very beginning and give way to environment rewards when they are discovered, while the tutor guidance impacts the agent with a continuous moderate intensity. An insufficient curiosity leads to following the tutor's gaze only (left).*

2.4 Reward-free environment

The previous experiments were conducted in the context of learning how to achieve a specific task, defined by rewards provided by the environment itself. As seen in introduction, this kind of rewards is becoming a limitation for RL agents to address open-ended learning. On top of that, predetermined reward functions have a few undesirable side effects.

Undesirable side effects of reward engineering A first issue appears when an agent with strong learning capabilities ends up hacking flaws in the reward design to accumulate reward without actually performing the corresponding task (Irpan 2018). This phenomenon rose recently with the increasing use of powerful function approximators in deep RL and often leads the authors to define overly complicated reward functions that prevent failure modes (Popov et al. 2017). A second more common issue encountered is when its definition leads the agent to fall and stay trapped in poor local optima during exploration (Irpan 2018). While such behaviors are mainly due to the difficulty of the exploration-exploitation dilemma, they also show that the fixed motivation coming from a reward signal designed once and for all by an engineer may be suboptimal for exploration, compared to a more adaptive motivation signal as advocated by

developmental approaches.

In part to avoid these effects, and overall because extrinsic rewards are flawed, we now carry out experiments with our intrinsically motivated gaze-following agent on a setup where there is no reward coming from the environment, in order to evaluate whether it can still reproduce the task the tutor is indicating with its social signals.

Several RL agents which do not receive any standard human-defined reward from the environment have been recently proposed, often called “unsupervised" or “self-supervised" RL agents. We defer their discussion to the final discussion of this dissertation as they stem from wildly different approaches to RL than ours for now: we do not seek to identify any task, but simply explore if the conjunction of curiosity and gaze-based guidance can lead the agent to achieve the task reliably, as we would expect in some way of a child.

2.4.1 Experiments

For now, the task remains ANY, but the reward obtained from the environment is only used for evaluation purposes and learning only relies on the two last terms of (2.8). The agent is only driven by curiosity and the tutor's gaze.

Learning without environment rewards

Figure 2.6 shows the impact of the attention-based guidance on the accumulated reward corresponding to the amount of blocks put inside boxes over time. With curiosity only, the agent achieves the task by chance on rare occasions over the 800 steps. By contrast, in presence of a tutor, it discovers the task earlier. More importantly, it also achieves the task expected by the tutor with regularity. This ability is obtained through the combination of the two reward mechanisms and the attention-action coordination constraint: schematically, the agent attention is first driven to the parts of the state space indicated by the tutor; because of the action-attention coordination constraints, the agent itself is then more likely to actually end up in those states the tutor deems useful; finally, once in those potentially rewarding states, the curiosity mechanism ensures that the agent will find the right action, among those it has not tried yet.

The plot also shows that the binary behavior described in the previous section has been reduced. Indeed, the 25-75 interquartile ranges drawn for each experiment over the 30 corresponding trials indicate that there is far less uncertainty in the end-results without external rewards. Also, the coupling between parameters ν and μ still exists without reward: $[\nu = 5, \mu = 1]$ performs significantly better than both $[\nu = 10, \mu = 1]$ and $[\nu = 1, \mu = 1]$ as shown by the non-overlapping interquartile ranges. A comparison with Figure 2.3 shows that learning without external reward logically remains less efficient than with it.

Switching tasks

In addition to being more developmentally sound, no-reward approaches tackle a difficult problem for classical reinforcement learning: changing environment and tasks. Essentially, if state-action

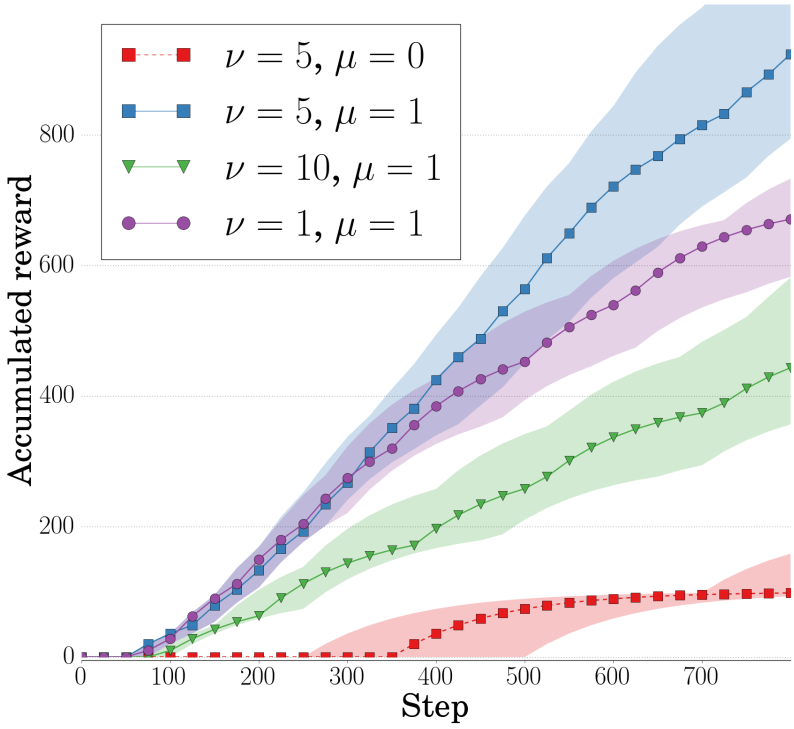

Figure 2.6 – Evolution of the reward accumulated for putting blocks in boxes over 800 steps (median and 25-75 interquartile range), for different combinations of gaze-based and curiosity-based motivations. Proper combinations of curiosity and interaction drives lead to the task being achieved regularly without using dedicated external rewards.

pairs that used to be rewarding become neutral, while others become rewarding, a standard Q-learning algorithm will need some time to forget its learned policy and start exploring again. Figure 2.7 evaluates the no-reward system tested before ($R = 0$, $\nu > 0$ and $\mu > 0$) against its counterpart with external reward ($R = 100$).

For this experiment, we define two additional tasks in the environment: in the MATCHING task, the agent still has to pick-and-place blocks, but with the additional constraint of matching the colors of the block and the box to place it into; the OPPOSITE task corresponds to placing blocks in boxes of the opposite color. The policy of the tutor is adapted to each task at hand, so that it picks boxes of the right color to look at until the agent achieves the task. Trials consist of a training phase on the task MATCHING for 800 iterations. At step 800, the task becomes OPPOSITE and the numbers of blocks placed in boxes of matching and opposite color are monitored. Figure 2.7 plots the evolution of the proportion of blocks put in the correct box, for various groups of parameters.

Experiments without external reward, during training as well as after the inversion, exhibit greater reactivity to the task change (plain lines in Figure 2.7). Also higher values of parameter ν (independently of the reward value) facilitate adaptation, as expected. It is worth noticing that the combination of joint attention, intrinsic motivation and external reward remains quite flexible as only a thousand iterations seem necessary for $[R = 100, \nu = 10, \mu = 1]$ to perform

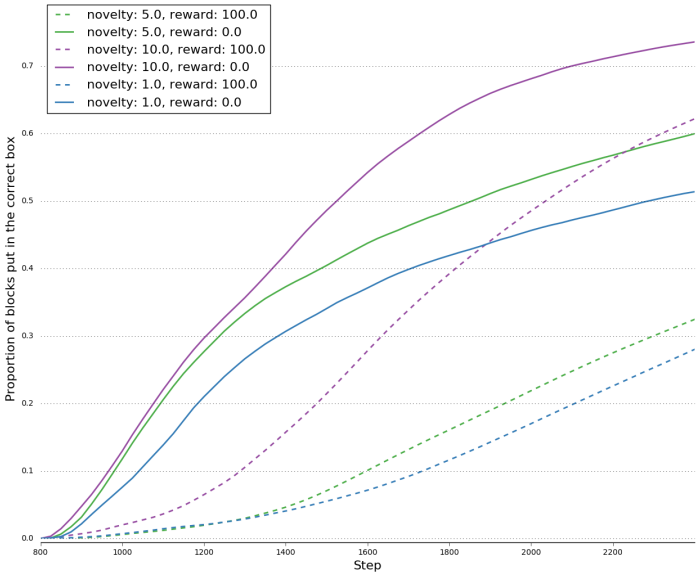

Figure 2.7 – Proportion of blocks put in the correct box for the task OPPOSITE *after training for 800 steps on the task* MATCHING*, with and without environment rewards ($R = 100$, $R = 0$), for three different values of ν, and $\mu = 1$. The agent is more reactive to the task switch without rewards coming from the environment.*

the task OPPOSITE more often than MATCHING. In all cases, the increase in reactivity comes from the joint attention naturally taking the agent to parts of the environment where many state-action pairs are motivating, thus triggering the intrinsic motivation system.

2.5 Analysis of the results

In this section we provide a partial analysis of our results, mainly to underline the elements that motivated our second work. A more in-depth discussion on the topics addressed in this chapter is proposed in Chapter 5. We first review the few existing works that bear resemblance to what we propose, and most importantly, we explain why the approach we chose lacks a crucial element.

2.5.1 Globally related work

There has been very few to no work with the same motivations and approaches as ours. Some existing algorithms combine intrinsic motivation for autonomous exploration and interactive learning though: in Chu et al. 2016, the agent also combines autonomous (self-)exploration and learning from demonstrations to learn object affordances in presence of external users. They compare learning with each source of information separately and together, and come to the same conclusion that the agent usually benefits from intelligently combining both.

More importantly, one of the first work, to our knowledge, to combine intrinsic motivation and social learning is the Socially Guided Intrinsic Motivation by Demonstration (SGIM-D) algorithm (Nguyen et al. 2011). This last work and its later variations are a lot closer in nature to one of our later algorithms, CLIC, and are discussed in detail in the corresponding Section 4.6.1.

In Khetarpal and Precup 2018, an agent in a 3D navigation maze environment is given original images of its environment plus augmented versions of them with saliency maps identified, and takes this information as an implicit indication of what is salient for decision making. The saliency maps are computationally found and not inferred from any interaction with humans, and the results show few to no improvement over the baseline without attentional guidance.

2.5.2 A first limitation

The attention-based social motivation we designed does not drive task achievement as we could have desired, because it lacks a fundamental block: intentionality. For the attentional guidance to be enough to drive the agent towards consistently achieving the task desired by the tutor, the agent should either be able to infer the goal of its tutor and try to achieve it, or set itself its own objectives and let them be influenced by the tutor's behavior. Our standard RL agent cannot do any of this, and thus can only reproduce the desired behaviors by exploring possible actions once in attentional synchrony with the tutor, and finding the right ones in sequence out of luck, which explains the much lower success rate. In other words, the agent lacks mechanisms for goal-directedness and goal inference — the latter corresponding to viewing other agents as goal-directed too.

The idea that an agent needs to infer the goals of another agent to act has already been studied, and is not restricted to teaching contexts. The concept of assistance, where the agent is instead tasked with helping other human agents, or that of collaboration in general, also heavily relies on the ability to estimate goals and select assistive appropriate actions with respect to these goals. In the literature, several models have been used to tackle this problem, from POMDP with hidden states corresponding to human goals (Fern et al. 2007; Javdani et al. 2015), to simulation theoretic approaches (Gray et al. 2005), hierarchical Bayesian learning (Diaconescu et al. 2014) and control sharing (Reddy et al. 2018).

2.6 Conclusion

For this first work, we started from three observations: intrinsic motivation and interaction have not been combined in RL previously but could gain to be; the applicability of most interactive RL works to real world scenarios is limited by a difficulty to interpret interaction; and finally, developmental studies of social interaction advocate for the use of more basic interactive mechanisms, like gaze-following, to lay the building blocks of more advanced communication channels.

From these observations, we proposed an agent that deems rewarding when it looks at the same objects as a tutor, that seeks novelty in its state-action space, and that must coordinate its actions to its attention. We have shown that there can indeed be synergies between attention-based

social interaction and intrinsic motivation, as the former can lead the agent to focus on objects while the latter guarantees the agent explores its action space on these objects.

Following the argument made in introduction against extrinsic human-defined rewards in RL, we evaluated whether the combination of gaze-following and novelty search alone could lead the agent towards the discovery and reliable achievement of the task. Results have shown that removing human-defined rewards enables a better reactivity to task change as expected, and that the combination above provides more frequent task achievement than with no gaze-following. Gaze-following leads the agent to attend to the good objects in the environment, action-attention coordination favors actions on these objects, and curiosity ensures the right action is found to achieve progress in the task.

However, the framework is clearly not sufficient to address task learning without rewards, unsurprisingly in retrospect. Indeed, this first work was limited in one crucial aspect: attention and gaze-following lacked an intentionality component.

Chapter 3

Goal-directed learning

In the previous chapter, we described a first agent whose developmental trajectory is influenced by a motivation to gaze where a tutor gazes and a motivation towards novelty, and we experimented with it a simple pick-and-place task, with and without extrinsic rewards. In retrospect, we concluded that the agent lacked an important component to replace human-defined rewards in RL: intentionality.

It is unclear whether it is innate or acquired, but children display intentional behaviors by the age of nine months (Piaget and Cook 1952), and from there develop the ability to view others as intentional agents too as detailed in Kaplan and Hafner 2006. In the latter reference, an *intentional action* is defined as "an action taking place in an initial state S, oriented towards a goal G, and using a particular temporally-extended action plan P to reach it", as illustrated on Figure 3.1, where an attention process also follows the advancement of the action plan.

The ability to have intentions and see others as intentional agents, with plans that they follow and attend to, is the base for the emergence of strong attention sharing behaviors, leading the infant to engage in triadic exchange with others, acting on objects in the way they are acting on them for example.

In this chapter, we forget about interaction altogether, including attention, and focus on giving an RL agent a capacity for intentional behaviors in some sense. To do so, we propose goal-conditioned RL as a intention directed learning paradigm, and we study its potential in terms of having the agent self-determine its developmental trajectory.

3.1 Goal-conditioned RL

Before delving into goal-conditioned RL, we must present what is widely called deep RL, as all experiments with goal-conditioned agents from now on will rely on its terminology and algorithms. In a first part of the background, we describe deep RL, and then we go on to defining goal-conditioned RL.

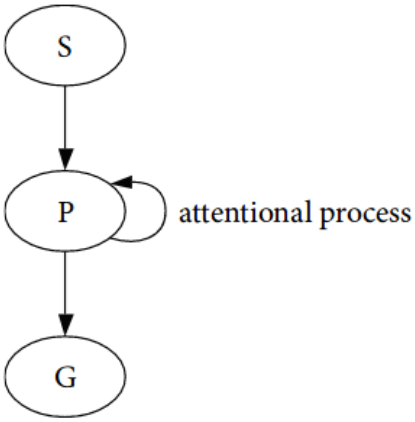

Figure 3.1 – Initial state S, goal G, action plan P and attentional processes following (Kaplan and Hafner 2006)

3.1.1 RL and deep RL

As we have explained in Section 2.1, the learned state-action value function Q in Q-learning (Algorithm 1) approximates the optimal Q-value function $Q^\star$. When the state space is high-dimensional or/and continuous, it is infeasible to rely a discrete representation of state-action pairs and update all their associated approximate Q-values separately. A solution consists in replacing the Q-value table with a parameterized Q-value function $Q(s, a; \theta)$, with θ a set of parameters.

Fitted Q-learning To learn the parameters θ, fitted Q-learning was proposed in Gordon 1996. First, it gathers experiences in a dataset in the form of tuples (s, a, r, s'), with r and s' drawn from the actual transition and reward functions of the MDP. Then, starting with random initial parameters θ_0 giving Q-values close to 0, an approximation of the Q-values at the k-th iteration $Q(s, a; \theta_k)$ is updated towards a *target value*:

$$Y_k^Q = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta_k), \tag{3.1}$$

where θ_k are the parameters defining the Q-function values at iteration k.

Neural-Fitted Q-learning In the neural fitted Q-learning algorithm described in Riedmiller 2005, the Q-values are parameterized with a neural network $Q(s, a; \theta_k)$ with weights θ_k at iteration k, updated by stochastic gradient descent by minimizing the square loss:

$$L = \left(Q(s, a; \theta_k) - Y_k^Q\right)^2. \tag{3.2}$$

Consequently the update of parameters θ_k writes:

$$\theta_{k+1} = \theta_k + \alpha(Y_k^Q - Q(s, a; \theta_k))\nabla_{\theta_k} Q(s, a; \theta_k), \tag{3.3}$$

where α is the learning rate.

Instabilities When the weights are updated, the target value also changes. With function approximators like neural networks, this approach can result in large errors in some states of the state space, which are known experimentally to propagate with the update rule. As a consequence, this form of Q-learning may be slow or unstable (Baird 1995). This drawback has been limiting the performances of combining Q-learning with function approximation (and in particular with neural networks approximators) for some time, and the conditions for its manifestation have been identified (Sutton 2015; van Hasselt et al. 2018). Several heuristic approaches have been proposed (Maei et al. 2010; Sutton et al. 2009, 2008), but one called Deep Q-Network (DQN) became prominent (Mnih et al. 2015).

DQN The DQN algorithm uses two heuristics to limit the instabilities:

1. The target Q-network in Equation 3.1 is replaced by $Q(s', a'; \theta_k^-)$, where the parameters θ_k^- are updated following a slower schedule than the parameters θ_k, for example setting $\theta_k^- = \theta_k$ every m iterations. Keeping the target values fixed during batches of iterations reduces the risk of divergence.

2. An experience replay buffer is used, as first described in Lin 1992: it keeps all the environment transitions tuples (s, a, r, s') collected by the policy for the last N_{replay} time steps. The weights update are then made on a batch of such tuples taken randomly in the buffer. This technique enables to replay transitions several times without being dependent on the policy to go through them several times. Also minibatch updates have less variance compared to a single tuple update.

The original DQN algorithm, which obtained strong performance on a variety of Atari 2600 games by learning directly from the pixels, relies on a few other important heuristics (*e.g.* frame preprocessing, frame stacking, reward clipping...) that are quite specific to the environments used and that will not be detailed here. The basic version of the DQN algorithm is shown in Algorithm 4.

Algorithm 4 The DQN algorithm

Initialize replay memory $\mathcal{D}$ to capacity N_{replay}
Initialize critic network $Q(s, a; \theta^Q)$ with random weights θ^Q
Initialize target networks Q' with weights $\theta^{Q'} \leftarrow \theta^Q$
Loop for each episode
    $t = 0$, initialize $s_t = s_0$
    Repeat for each step t
        Choose a_t following the ϵ-greedy policy w.r.t. $Q(s_t, \cdot)$
        Take action a_t, observe r_{t+1}, s_{t+1}
        Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
        Sample minibatch of N transitions $(s_j, a_j, r_{j+1}, s_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j, & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} Q'(s_{j+1}, a'; \theta^{Q'}), & \text{otherwise} \end{cases}$
        Update critic by minimizing $\frac{1}{N} \sum_j (y_j - Q(s_j, a_j; \theta))^2$
        Update the target networks: $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$
    Until s is terminal or timeout

Continuous control Continuous control refers to the situation where the agent samples its action in a continuous space, instead of a discrete one. As a result, the maximization step present in discrete algorithms (see line 11 of Algorithm 4) requires an optimum search which would be too costly to be repeated constantly during training. Instead, a few algorithms specifically dedicated to continuous control have been proposed, among which the one we use in this chapter: the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al. 2016).

DDPG DDPG is an actor-critic algorithm with deterministic actions, where a second neural network plays the role of the actor and is trained jointly with the value function approximators, using the deterministic policy gradient estimator provided by (Silver et al. 2014). The full algorithm is shown in Algorithm 5, with this last gradient being computed at line 13.

Algorithm 5 The DDPG algorithm

1: Initialize replay memory $\mathcal{D}$ to capacity N_{replay}
2: Initialize critic and actor networks $Q(s, a; \theta^Q)$ and $\pi(s; \theta^\pi)$ with random weights θ^Q and θ^π
3: Initialize target networks Q' and π' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\pi'} \leftarrow \theta^\pi$
4: **Loop** for each episode
5: $t = 0$, initialize $s_t = s_0$
6: **Repeat** for each step t
7: Choose $a_t = \pi(s_t; \theta^\pi) + \text{noise}$
8: Take action a_t, observe r_{t+1}, s_{t+1}
9: Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
10: Sample minibatch of N transitions $(s_j, a_j, r_{j+1}, s_{j+1})$ from $\mathcal{D}$
11: Set $y_j = \begin{cases} r_j, & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma Q'(s_{j+1}, \pi'(s_{j+1}; \theta^{\pi'}); \theta^{Q'}), & \text{otherwise} \end{cases}$
12: Update critic by minimizing $\frac{1}{N} \sum_j (y_j - Q(s_j, a_j; \theta))^2$
13: Update the actor with the sampled policy gradient:

$$\frac{1}{N} \sum_j \frac{\partial Q(s, a; \theta^Q)}{\partial a}\Big|_{s=s_j, a=\pi_\theta(s_j)} \frac{\partial \pi(s; \theta^\pi)}{\partial \theta}\Big|_{s=s_j}$$

14: Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\pi'} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'}$$

15: **Until** s is terminal or timeout

Deep reinforcement learning In this dissertation, we will rely on DQN and DDPG, sometimes with modifications required by our approaches. The algorithms will be described in detail when necessary. We abusively call deep RL every use of multi-layer neural networks as function approximators in reinforcement learning algorithms, including when these neural networks are not deep at all.

Contrary to many traditional approaches, deep RL can deal with high dimensional data by learning representations of smaller dimensions through back-propagation and it can be trained incrementally to make use of new samples acquired during learning. In the RL community, its use has steadily grown during the last few years.

3.1.2 Goal-conditioned RL

Goal-conditioned RL is simply the idea of taking the goal as part of the input. The first goal-conditioned RL algorithm has been proposed in Kaelbling 1993, long before deep RL, under the name Dynamic Goal Learning.

Dynamic Goal Learning

Kaelbling (1993) proposed this algorithm with a technique called *all goals updating* for the problem of navigating in a discrete grid: the agent computes Q-values for tuples $(s, a, g), g \in \mathcal{S}$ instead of only (s, a) pairs, and acts by selecting a goal g and then following a policy produced by the Q-values for this goal $Q(\cdot, \cdot, g)$. Thus the policy is said to be goal-conditioned.

An important part of the algorithm lies in the all goals updating methods, whereby any transition (s, a, s') experienced can be used to update the Q-values for all goals g' independently of what g was being sought when the action was taken in the transition. If the goal space is too large, it only performs such updates for a random selection of goals. Importantly, the reward is easy to recompute for every goal in the navigation problem: $r = 0$ if $s = g'$, else $r = -1$. We show the full procedure in Algorithm 6. Importantly, the all-goals updating method strongly relies on a property of Q-learning: it is an off-policy algorithm.

Algorithm 6 Q-learning with all goals updating

Input: step size $0 < \alpha \leq 1$, ϵ
Initialize: $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, except that $Q(terminal, \cdot) = 0$
Loop for each episode
    Initialize s, g
    Repeat for each step
        Choose a from policy derived from the $Q(s, a, g)$
        Take action a, observe r, s'
        for all goal g' **do**
            $Q(s, a, g') \leftarrow (1 - \alpha) \cdot Q(s, a, g') + \alpha \cdot (r + \gamma \cdot \max_a Q(s', a, g'))$
        end for
        $s \leftarrow s'$
    Until s is terminal

Off-policy learning

Off-policy learning comes in opposition to on-policy learning. In on-policy learning, the policy being evaluated and/or improved during learning is the same as that used to acquire the data in

the first place, whereas both are different in the off-policy case. In dynamic goal learning with all goals updating, experience acquired by trying to achieve a goal g, i.e. following the policy associated to goal g, is used to update Q-values for *all* other goals g', and is thus off-policy experience with respect to the policies associated to these other goals.

An off-policy algorithm must not compute updates for a policy under the assumption that the observed experience has been gained from the same policy. Q-learning is naturally an off-policy algorithm: whatever may be the policy providing training samples, the Q-learning updates assume the next action taken follows the greedy policy at the next state (see line 8 in Algorithm 1), which indeed corresponds to the policy for which Q-values are computed. In general, the policy used to acquire data is called the *behavior policy*, while the policy being learned is called the *target policy*. An off-policy method can use experience from the behavior policy to evaluate and improve the target policy without additional correction needed.

Universal Value Function Approximators

Dynamic Goal Learning has been proposed with eligibility traces under the name Concurrent Q-Learning (Ollington and Vamplew 2005), but has been relatively absent from RL studies otherwise. The interest of the community in this paradigm was revived by the work on Universal Value Function Approximators (UVFAs) in Schaul et al. 2015. The latter actually takes it inspiration from a separate algorithmic architecture that learns simultaneously and off-policy multiple value functions to act in an environment, called Horde (Sutton et al. 2011), but simplifies it to the extent that it becomes essentially similar to the multi-goal learning with all goals updating of Kaelbling 1993, despite not citing it.

In Sutton et al. 2011, Sutton and colleagues propose the rich notion of General Value Function. In essence, they explain that the Q-value $Q^{\pi,r,z}(s,a)$ for a policy π, a reward function r and a termination function z can describe *knowledge about the environment*. For example, if the reward is 1 when the agent successfully advances one step in one direction, if the policy is to always go in this direction and the stopping criterion takes effect whenever the agent cannot advance anymore in this direction, then $Q^{\pi,r,z}$ is the distance to the first obstacle in this direction. The authors then propose to define and learn multiple such General Value Functions simultaneously with an off-policy algorithm, in an all goals updating style.

Simplifying these General Value Functions, Schaul et al. (2015) propose to learn Q-value functions for state-action-goal triplets only, and propose to approximate such Q-values with neural networks, writing $Q(s,a,g;\theta) \approx Q^\star_g(s,a)$ the approximate optimal value function for accomplishing goal g from state s and action a. Finally, they describe an algorithm to learn these Q-values called Direct Bootstrapping, but which is actually entirely Dynamic Goal Learning.

Another contribution of Schaul et al. (2015) is that they propose several approaches to process the goal and state inputs, with the objective of identifying and exploiting structure across both s and g, as in many cases the state and goal spaces are related. For example, in the case where the goal space *is* the state space, so that the agent seeks to reach some state, then there is as much regularity in the value of goals as there is in the value of states. In the end, the agent takes the concatenation of a state s and a goal g as a joint input, and outputs $\gamma^{\|s-g\|_2}$, taking advantage of the fact that goal reaching amounts to shortest path finding in their case.

The first results of UVFAs for RL were not especially impressive, mainly because they were obtained before any methods to deal with instability in RL with function approximation were solidly installed in the field. In theory, following the spirit of General Value Functions, the parameter g in UVFAs can represent more forms of possible goals than simple states to reach, from a discrete set of objectives, or their power set, to a vector representation of parameterized reward functions.

3.1.3 Goal-conditioned RL as multi-task RL

We have presented deep and goal-conditioned RL, and we now propose to think about the possibilities offered by this augmented RL paradigm. First, we want to relate goal-conditioned RL to multi-task RL. Essentially, intention-directed learning is multi-task learning where the task is explicitly and internally represented by the agent, and where the agent actions are conditioned on this representation. Let us have a look at this claim in detail.

Multi-task RL

Multi-task RL is the application of multi-task learning to reinforcement learning tasks. Following Caruana 1997: "Multi-task Learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. It does this by learning tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better". Inductive transfer (or transfer learning) corresponds to using experience gained in learning to perform one task to improve learning performance in a related but different task (Taylor and Stone 2009).

The intuition behind the usefulness of multi-task learning for RL is that seeing multiple tasks at train time can have a regularizing effect on single-task performance. RL agents are indeed known to strongly overfit their environments and distributions of inputs when training on too restricted problems, resulting in not being able to generalize to small changes in input space or input distributions, nor fine-tune their networks to adapt to these changes (Gamrian and Goldberg 2018).

To understand the exact connection between goal-conditioned RL and multi-task RL, we must make a distinction between two settings for multi-task RL.

Seemingly different transition functions

In several works experimenting on games from the Atari suite, or in the Malmo platform for Minecraft playing, the agent must learn to solve a set of games in the environment with a single neural network, so that all parameters are shared between different tasks learning situations. In this case, in the vocabulary used to describe the experiments, each game is a task, and all tasks obviously share their state space (raw pixel observations in Atari for example) and action space. Do they share the same MDP for all that?

Intuitively, we could say that each game is characterized by a different transition function and a different reward function, so that each game is associated with a single distinct MDP. However,

formally speaking and from the point of view of the agent, this is not entirely true. Indeed, we can formally say that there is only one transition function and one reward function for all games simultaneously, which are both defined in a piecewise fashion of the different subspaces of the pixel space that characterize each game. For example, the transition function programs the cluster of pixels defining a ball in Pong to bounce off the cluster of pixels defining the bar, and programs the cluster of pixels defining a space invader to vanish when hit by the cluster of pixels defining a bullet in space invaders.

The important hypothesis for these games to allow for such common transition and reward functions to be defined is that two different games cannot make the agent go through the same state. Otherwise, an agent with fully shared parameters for all games learning is confronted with perceptual aliasing, where states that differ by the task being pursued appear identical (Whitehead and Ballard 1991). With the state space being the pixel space and the definition of the games from the Atari 2600 suite for example, this hypothesis holds true.

Despite being rarely mentioned, it is essential for multi-task RL approaches with fully shared parameter sets to work, although in theory, limited perceptual aliasing can be mitigated by using recurrent neural networks. As a result of this hypothesis, all games should be formally considered in the same global MDP, and differ instead by the state and action distributions the agent encounters when playing them. More precisely, each game is defined by a specific initial state distribution, and the global transition function is such that for any policy of the agent, the steady state distribution of the policy is entirely separated from that of another game.

Clearly identical transition functions

Now we consider a second case for multi-task RL, where the transition functions associated to the task are the same, this time *clearly* and not after formal considerations, but where each task is associated with a specific reward function while visited state distributions clearly overlap between tasks, contrary to the previous case.

This way to describe the setting can seem opaque, but it describes most multi-task continuous control environments: a simulated arm for example can learn to achieve two different tasks like pushing and picking, or try to reach two different goals, but in all cases the different learning situations are defined by different reward functions while nothing prevents the agent from going through the same state when pursuing different tasks. To avoid the perceptual aliasing problem mentioned in the previous case, we *must* differentiate in some way the two learning situations.

Goal-conditioned RL operates this differentiation by parameterizing the learning situation, over goals for example, and by augmenting the state space of the problem with this parameterization. In doing this, goal-conditioned RL transforms a multiple overlapping MDPs setting into a standard multi-task setting *rigorously formally identical to that of the first case.*

Indeed, once the state space is augmented with a parameterization p of the learning situation, we can define a single reward function of the augmented state space $r(s, p)$ (the transition function was already common) like in Dynamic Goal Learning and UVFAs, and the different tasks once again only differ by the initial augmented state distribution, while the steady state distribution of any policy remains entirely separated from that of another task with another parameterization. The main difference this time is that this last property is more due to the different nature of

task parameterization with respect to states (they do not change under the agents action) than to the real non-augmented transition function of the environment.

A seemingly different approach to this case of multi-task setting can be found, and relies on models with multiple heads, one for each task (Cabi et al. 2017; Riedmiller et al. 2018). Writing properly the value function in this approach shows that it is only goal-conditioned RL with a gated architecture.

3.2 Curriculum learning

We have shown in the previous section that goal-conditioned RL and multi-head multi-task models are rigorously identical in formal nature to multi-task learning algorithms with fully shared parameters, where tasks are defined by MDPs with different initial state distributions. An important difference remains: a goal-conditioned RL agent has control over its representation of the goal that defines its current learning situation all along the episode (formally this control translates to a control over a part of its state distribution), while standard multi-task models must simply adapt their actions to the state distribution they are in.

To allow for cross-goal or cross-task learning, approaches of this kind rely on the generalization capabilities of the value function approximation to perform transfer learning: in games from the Atari 2600 benchmark for example, some cluster of pixels have similar behaviors across games and the agent could benefit from observing this regularity across multiple full state distributions; in goal-conditioned RL, the agent can infer the appropriate behavior in one learning situation by generalizing the effect of its parameterization from previously seen and differently parameterized situations. In the end, these agents must approximate value functions on constantly changing state-action distributions.

Goal-conditioned RL, as well as some works in multi-task RL, leaves the possibility to select the sequence of learning situations, by sampling a goal at the beginning of each episode for example. This makes it possible for the agent to have a direct impact on its own learning trajectory, and in particular it can perform curriculum learning.

Curriculum learning is not a strongly formalized concept in the RL literature, but broadly refers to the idea of a "learning regime inspired by the learning process of humans and animals that gradually proceeds from easy to more complex samples in training" (Bengio et al. 2009; Jiang et al. 2015). Some authors consider that curriculum learning refers to the situation where the curriculum is predetermined by prior knowledge and fixed (Jiang et al. 2015), and works proposing agents finding the curriculum by themselves during learning often say they perform “automated” or “automatic” curriculum learning (Florensa et al. 2017b; Graves et al. 2017).

3.2.1 The Learning Progress Hypothesis

We are especially interested in automatic curriculum learning for RL because it constitutes a truly developmental approach to learning in environments with multiple potential tasks or goals to reach. As a matter of fact, the curriculum learning problem is clearly an active one

in developmental sciences, and is close in nature to those of "self-determined evolution" or "autonomous development" often mentioned (Lungarella 2007).

This problem has been mainly approached with intrinsic motivations and the aim to use them to drive the organization of learning trajectories on long time-scales. This approach is the one we intend to use in this second work to drive RL agents in multi-goal environments. Precisely, we focus on an hypothesis called the Learning Progress Hypothesis, which has been formulated multiple times in developmental robotics, cognitive neurosciences and educational technology studies Kaplan and Oudeyer 2007; Oudeyer 2018; Oudeyer et al. 2016. It suggests that learning progress itself generates intrinsic rewards.

This idea can be applied to knowledge-based intrinsic motivations, and gives Information Gain Maximization and Prediction Progress Motivation (sometimes called generically Learning Progress Motivation) (Oudeyer et al. 2007; Schmidhuber 1991), but also and more usefully for us, to competence-based intrinsic motivations.

Competence-based intrinsic motivations

Competence-based intrinsic motivations consist in orienting one's learning by using measures of one's competence on self-determined skills (Oudeyer and Kaplan 2009). In particular, competence-based intrinsic motivation assumes that the agent can plan actions to reach self-determined goals, exactly as goal-conditioned RL does, and computes its level of achievement of said goals all along learning to derive measures of interestingness for each goal.

Oudeyer and Kaplan (2009) distinguish between several types of competence-based intrinsic motivation: maximizing incompetence (selecting hardest skills for the agent), maximizing competence (selecting easiest ones), intermediate incompetence (selecting skills of intermediate difficulty), while applying the Learning Progress Hypothesis gives the motivation of maximizing CP (selecting skills were the agent is currently making the more progress).

Advantages of the hypothesis

Learning progress maximization and intermediate difficulty selection are related, but the latter has been more discussed (Berlyne 1960; Czikszentmihalyi 1990). Both are compatible with a number of observations in animal studies, like simultaneous neophilia and neophobia (Oudeyer et al. 2016), or in cognitive science, like the Goldilocks effect, that refers to infants' preference to attend to events and sequences of events that are neither too simple nor too complex (Kidd et al. 2012).

Compared to intermediate difficulty selection, learning progress intrinsic motivation comes with several advantages, detailed in (Oudeyer et al. 2016), two of which being important for us. First, it does not rely on thresholds to evaluate situations and simply needs to evaluate its own knowledge and competence during learning, so that the agent can pick whatever situation leads to the more progress. From a goal-reaching point of view, competence progress alone is enough to discriminate between goals in the way of being mastered, already mastered, that cannot be mastered, and that the agent already "partially" masters but cannot master more.

Indeed, if the agent is in the way of mastering an objective behavior, then it means that its self-evaluated competence on this behavior is increasing with training, which means its progress is strictly positive. On the contrary, when the objective cannot be mastered at all or is fully mastered, the agent competence never rises or does not vary anymore, leading to zero progress. Noticeably, if the objective is partially mastered in some sense but practicing it does not make the agent master it more at all, the progress stays zero too. All these different situations thus lead to variations in learning progress that are exploitable by the agent to drive skill acquisition.

Second, it introduces a new relation between learning and curiosity: instead of simply leading to "curiosity situations" that favor learning, learning itself in a situation now influences the intrinsic motivation mechanism by making the situation motivating. Compared to other models, this reinforcing feedback loop is new and contrary to other models, it can help the learner organize its learning experience on longer time scales (Oudeyer et al. 2016).

Applications

In the field of developmental robotics, but out of RL in general, the learning progress hypothesis has been applied to devise both knowledge-based and competence-based intrinsic motivations, and used to enable agents to select incrementally and on-line learning situations, including the automated selection of policy parameters (Oudeyer et al. 2007), goals (Baranes and Oudeyer 2013), models (Forestier and Oudeyer 2016), and learning strategies or teachers in an interactive context (Nguyen and Oudeyer 2014; Oudeyer 2012).

In particular, the SAGG-RIAC algorithm described in (Baranes and Oudeyer 2013) proposed an active goal babbling strategy for learning sensorimotor mappings, and addressed the difficulty of computing meaningful competence measurements in continuous goal spaces. It dynamically and incrementally learns regions of the goal space with sufficiently distinct average competence, uses a sliding window to compute progress measures on-line and samples the regions in proportions to these measures to propose goals to reach. The algorithm was the basis for a number of subsequent algorithms (Nguyen and Oudeyer 2014), and adapted to high dimensional goal space where the region learning strategy became impractical by instead computing competence in a modular fashion in presence of objects (Forestier and Oudeyer 2016). Most of these works rely on non-parametric models to learn inverse kinematics, like Approximate Nearest Neighbors algorithms.

3.2.2 SAGG-RIAC plus goal-conditioned RL

In a preliminary work, we attempted to use the SAGG-RIAC algorithm to replace random goal selection at the beginning of episodes in standard goal-conditioned RL with CP-maximizing goal selection, as the algorithm has never been used with RL to our knowledge.

We performed experiments in the simplistic reacher environment, shown in Figure 3.2, which is a two degree-of-freedom arm that must reach a target in its plane, making it a good and very simple environment to start studying principled ego-motion learning. In this environment, the agent state contains the arm angles and angular velocities, its end-effector position p_{finger} and

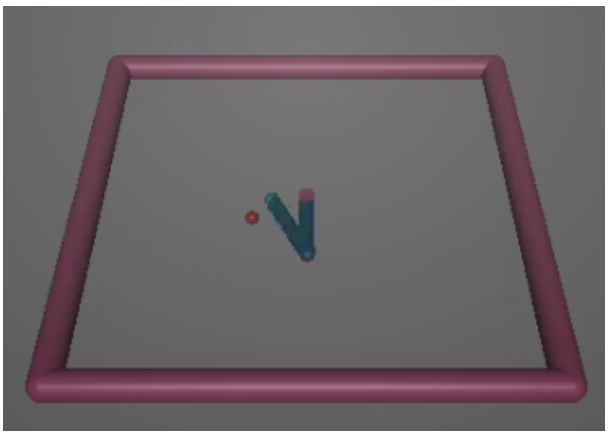

Figure 3.2 – The Reacher environment.

the target position p_{target}. We used its sparse reward function:

$$r = \begin{cases} 0, & \text{if } |p_{finger} - p_{target}| \leq \epsilon \\ -1, & \text{otherwise} \end{cases} \tag{3.4}$$

with the parameter ϵ determining the accuracy required from the agent for being successful.

The goal space was the set of states in the square arena defined by the borders of the gym environment, and we used the standard DDPG algorithm (Algorithm 5). In this first attempt at principled goal selection, we counted on the SAGG-RIAC algorithm (Baranes and Oudeyer 2013) to iteratively identify regions of this goal space where the agent makes progress at different speed, and favor high competence progress regions for goal selection, with the hope that better performance obtained in these regions would slowly generalize to other regions and result in faster global reaching capability acquisition.

Unfortunately, we never succeeded in demonstrating any gain of this goal selection algorithm over the random baseline, and we proceed to analyze this failure. First, we survey other attempts in RL to perform principled goal or task selection, and then we discuss the merits and flaws of the methods, as well as the origin of the difficulty of the problem in general.

3.2.3 Curriculum learning: related work

Curriculum learning has been studied from a developmental learning standpoint, but has also been addressed separately in multi-task RL. To tackle the problem, several approaches have been proposed.

A hierarchical control view

Some works have modeled the learning process with the Multi-Armed Bandit or the RL formalisms, with different reward schemes leading to different global objectives.

CP-based rewards Matiisen et al. (2017) frame task selection as a partially observable MDP with the parameters of the learner being the hidden state, the tasks being its actions, and rewards being the changes in evaluation score compared to the last evaluation of the current task. Stout and Barto (2010) use the framework of options to address multi-task learning and off-policy Q-learning to learn them in parallel. The mean value function increase for an option is used as a reinforcement signal for a higher-level RL algorithm to find the skill with highest expected competence progress and practice that skill. In both cases, the accumulation of a reward based on CP leads to the maximization of a certain form of final competence.

Incompetence-based rewards Sharma et al. (2018) explicitly provide the level of difficulty of each task to the agent through objective scores from state-of-the-art literature, and the active task selection mechanism seeks to favor tasks that "lag behind" the desired scores. It does so with an ad hoc sampling heuristic or with ad hoc human-engineered rewards, in UCB or a full RL framework.

Competence-based rewards Finally, the SAC-X agent of Riedmiller et al. 2018 must achieve a main task, for which it is given a fully human-engineered extrinsic reward, but is also given many auxiliary tasks and their ad hoc rewards to maximize in parallel. SAC-X uses an off-policy algorithm with as many heads as tasks and selects the next task to practice via a scheduling module. The scheduler is learned with an algorithm that maximizes competence on the main task with a tabular approximation of the accumulated return on the main tasks of choosing each task after a sequence of tasks.

Goal-generation processes

A second category of methods relies on defining a process that provides goals “at the right difficulty level” and can often be seen as favoring learning situations of intermediate difficulty.

Adversarial goal generation In Sukhbaatar et al. 2017, a goal-conditioned RL agent is decomposed into a teacher Alice and a learner Bob, and Alice models curriculum design as a reinforcement learning problem, which rewards Hindsight Experience Replay (HER) in proportion to HER capacity to provide Bob with the simplest goal states it cannot reach. The agent alternates periods of self-play between Alice and Bob and periods of standard learning for Bob. The reward scheme employed during self play is close in nature to maximizing competence progress, or at least follows the idea of proximal development. In Florensa et al. 2017b, some intermediate difficulty level goals are manually labeled with external knowledge on the environment, and taking inspiration from GANs (Goodfellow et al. 2014), a combination of goal-generator and discriminator networks learns to propose goals of the same level of difficulty as those labeled to the agent. A UVFA-based agent is then trained on continuous control tasks.

Perturbative generation Florensa et al. (2017c) propose to make an agent learn to reach its goal state from start states such that the average reward from them is comprised between two thresholds. To accomplish this, the algorithm chooses close states at the beginning, and

then adds Brownian perturbations around states from which it is already successful to define new starting states.

Hindsight generation In the HER algorithm of Andrychowicz et al. 2017, the authors observe that during an episode aiming at reaching a goal g, a UVFA agent that does not succeed in reaching g still goes through many state s' from start state s. All these traversed states are thus such that the episode would have been successful had they been its goal. Building on this observation, HER proposes to exchange some real goals g in the transitions (s, a, s', g, r) the agent trains on with such *virtual goals* s' seen during the episodes, and the authors explore several virtual goal sampling strategies to do so. By biasing the transitions used as training samples towards goals actually reached by following policies obtained from recent estimates of the UVFA, the agent trains on reaching goals that are guaranteed to be reachable with its current network parameters and for which it has rewarding experience available.

Explicit measures of interest

Finally, some methods rely on explicit "task interest" measurements and use them to bias task selection. Eppe et al. (2018) propose a UVFA agent that aims at reaching separately the different coordinates of its goal space, and focuses on the dimension for which the current performance is close to a certain optimal difficulty level found empirically. In Veeriah et al. 2018, a part of the experiments uses the decreases in squared Bellman error as measures of interest to sample goals observed in the past in a UVFA setting. This strategy thus falls into the category of knowledge-based learning progress maximization, and not competence-based learning-progress maximization as competence measures are never computed.

3.2.4 Limitation of an analysis

Rigorously comparing the proposed methods is difficult and global conclusions must be taken with care, for a number of reasons that we detail now. To put it shortly, evaluating curriculum designs against one another requires these designs to be compared *under the same conditions*, and it is almost never the case.

Comparing deep RL algorithms

First, in the current state of the literature, independently of curriculum learning, comparing deep RL algorithms in general is difficult. Algorithms are brittle because highly sensitive to hyperparameters, architectures, and implementation, and their results are not always reproducible, because of a lack of rigorous statistical performance evaluation, and sometimes of exhaustive, clear descriptions (Colas et al. 2019; Henderson et al. 2018; Islam et al. 2017).

As a result of these issues with the field, deep RL algorithms properties in terms of generalization, learning capacities, overfitting or transfer are still poorly understood (Cobbe et al. 2018; van Hasselt et al. 2018; Witty et al. 2018; Zhang et al. 2018), and both the conditions and necessary components for them to be advantageous in comparison with previously existing approaches are

not always clear (Mania et al. 2018; Rajeswaran et al. 2017). Curriculum learning, where we capitalize on many properties of the agent to build better sequences of task, is thus not really based on solid foundations.

Different environments

Most of the algorithms we presented are evaluated on different sets of tasks and in different environments: games from the Atari 2600 benchmark (Sharma et al. 2018), Minecraft (Matiisen et al. 2017) or visual grid world (Veeriah et al. 2018) game playing from raw pixel-level states, or various different continuous control environments, in openAI gym (Andrychowicz et al. 2017; Eppe et al. 2018; Florensa et al. 2017b,c), rllab (Sukhbaatar et al. 2017), or other simulated control platforms (Riedmiller et al. 2018). This lack of a common benchmark environment suite makes comparisons of performance difficult as we must rely on the likely false assumption that environment choice does not influence the impact of curriculum strategies.

3.2.5 A theoretical standpoint

In the curriculum learning problem for RL, the agent must approximate its value function on a discrete or continuous set of subspaces of its state space corresponding to its different tasks or goals. As we have seen, in the case of goal-conditioned RL — including multi-head networks —, these subspaces have a constant component defined by the goal representation, while in traditional multi-task RL, they simply correspond to the different part of the state space visited by the agent when accomplishing the task from its initial state distribution.

In this context, curriculum learning consists for the agent in choosing better than randomly which subspace to train on at each occasion it is given the choice. This problem appears especially hard in the case of RL.

Evaluating the agent competence

A first obstacle is that in RL, the performance of an agent at a precise point of its training, meaning its ability to accumulate reward by following its approximated value function at this point, is counter-intuitively not given by its value function approximation. In other words, the agent cannot evaluate the quality of its function approximation and thus cannot directly evaluate its competence.

At first sight, one could say that RL agents are never given the *true* value function to learn, unlike in supervised learning, so that the property above is expected. But this is only a superficial view of the problem. Indeed, in Q-learning, the value function of the agent is explicitly defined and computed so as to represent the performance (the expected value) of the agent *when it follows the greedy policy with respect to the Q-value function*, which is what the agent actually needs to evaluate. The true reason for which it cannot do so is actually that the training regime used to iteratively compute Q-values only guarantees that the definition of the Q-value function holds after the Q-value approximation has converged, and at no point before.

Like all model-free RL algorithms, Q-learning is achieved through Generalized Policy Iteration, which is the idea of letting two processes alternate and interact: "making the value function consistent with the current policy (policy evaluation), and making the policy greedy with respect to the current value function (policy improvement)" (Sutton and Barto 2018, chap. 4). The problem comes from the fact that none of these two processes has the time to converge between each training step. For example, if the agent just trained on one subspace S_1, the Q-values for state-action pairs in another subspace S_2, given by the newly obtained weights, do not reflect the performance the agent would reach by following the greedy policy with respect to these Q-values on S_2. To obtain such an information, the agent would need to freeze the current weights and Q-values to define the static policy corresponding to them, and then achieve many updates of separate and non-frozen Q-values following the frozen policy.

Obviously, such a scheme is intractable and the agent can only rely on its past episodes results to evaluate its competence at one point. Unfortunately, when there are many possible tasks or with a continuous set of goals, it is likely that for a given task or goal, the agent has not attempted to achieve it in its recent experience, or may have even never done so.

The non-regularity of the agent competence evolution

By chance, the agent may have indeed carried out an attempt at a goal of interest in its recent experience, and could hope to use this recent experience to evaluate its own competence at its current point in learning. However, it so happens that even the small recent changes in the weights of the agent since this experience, and thus the small changes in its Q-values, can result in large deviations in terms of performance, for the value function is not the only component having an impact on accumulated rewards.

Indeed the performance of new Q-values is the result of the policy these Q-values define, which is based on a highly non-regular "arg max" function, and then on the accumulated reward this policy leads to. In both of these relations, small changes in input can have heavy consequences: a small change in Q-value can lead the best action in one state to change, and a small change in an action sequence can lead to a totally different behavior.

As a consequence, methods that try to constraint updates in one subspace not to be destructive in other subspaces still cannot guarantee that the gained regularity in Q-values evolution translates to a regularity in terms of performance.

3.2.6 Conclusions

Despite the mentioned limitations of a comparison of the algorithms presented in the related work, we can still make a few general observations and relate them to the theoretical properties of curriculum learning described above. If we sum up the previous section, the agent can neither evaluate easily its competence on its goals or tasks, nor make sure the evolution of this competence is controlled in any way.

In other words, the agent cannot really access its own "learning state", and the evolution of the latter is especially hard to evaluate and anticipate. In this light, the use of the formalisms of bandits or RL with partial observation appears natural, as they require none of these pieces of

information. However, these formalisms share a downside with adversarial methods: they are known to be difficult and costly to train, unstable and sensitive to hyperparameters.

On the contrary, methods based on explicit measurements of competence or other metrics have the advantage of being computationally cheap compared to adversarial methods or hierarchical RL methods. Their main flaw is related to what we have seen on theoretical nature of curriculum building: if the agent cannot access its real learning state, nor anticipate its future learning state, the explicit measurements used are bound to give ill-informed indications in many situations.

As a result of these respective downsides, a recurrent observation in the works analyzed above, either explicitly stated or simply visible in results, is that the random goal selection baseline is actually hard to beat (Matiisen et al. 2017; Riedmiller et al. 2018; Sharma et al. 2018; Veeriah et al. 2018). This conclusion seems to remain true for curriculum learning applied to multi-task learning of non RL tasks, as in Graves et al. 2017, where prediction gain maximization is compared to several other predictive and complexity-based learning progress measures on the ordering of tasks encountered by a LSTM network. The authors also observe that none of the progress signals tried provide significant benefits for all the tasks evaluated, and that they were in average only weak improvements over the uniform sampling baseline.

To compensate for this difficulty, many works in each category implicitly or explicitly integrate external knowledge to help the agent design its curriculum, relying on externally-given environment-dependent thresholds (Florensa et al. 2017b), good solutions for the problem to start with (Florensa et al. 2017c), relevant auxiliary tasks in the environment (Riedmiller et al. 2018) or state-of-the-art scores for task (Sharma et al. 2018).

In terms of re-usability, one specific method stands out from others in this study: HER has demonstrated benefits of non-random training goal ordering in multiple environments (Nachum et al. 2018; Nair et al. 2017, 2018; Plappert et al. 2018), it is conceptually very intuitive, computationally cheap, does not require any external knowledge and is very easy to implement. These appealing properties have led it to become a standard tool in the multi-task deep RL toolbox, used more and more often.

However, HER falls into the category of methods that address sparse rewards scenarios and densify reward observation with multi-goal learning, as the SAC-X agent of Riedmiller et al. 2018. In these works, it is unclear how much of the good properties demonstrated are due to the capacity to order training goals, and how much are due to the simple increase in rewards seen during training.

3.3 Accuracy-based curriculum learning in RL for control

In the light of our analysis of curriculum learning in RL, the failure of our first attempt based on SAGG-RIAC can be explained. As we have seen, a small change in the network made to optimize for one specific goal can result in great changes in terms of other goals achievement competence. Thus training on a specific region of the state space leads to immediate changes in performances on other regions that are difficult to predict and the true competence of the agent on each goal is changing constantly with training, so that periodic explicit competence measurements are

bound to reflect them inaccurately. As a consequence, our principled approach to curriculum building relied on meaningless CP measures and could not beat the random baseline.

In the previous experiments, as in many continuous control environments, goals were parameterized by a target position to reach in some way, and within a radius fixed for all experiments. In this section, we propose to include in the goal parameterization the value of this radius, and perform competence progress based on the accuracy within which the agent can reach its goals.

3.3.1 Motivations

The reward functions used in experiments in manipulation environments all rely on a distance threshold for evaluating success of episodes, that depends on the actual dimensions of the environment and of the precision of the agent actuators (Brockman et al. 2016; Plappert et al. 2018). We argue that this hidden parameterization can be leveraged to build an accuracy-based curriculum for the agent.

First, many robotics tasks can be made more or less difficult by requiring different degrees of accuracy from the agent: learning to bring its end-effector within 10cm of a point may be easier than within 10mm. A task which was easy with loose accuracy requirements can become difficult if the accuracy constraint becomes tighter. The impact of accuracy requirement on learning efficiency in the context of curriculum learning can be particularly powerful, since progress in RL is made by finding rewarding experiences, and it is harder to find a source of reward when the accuracy requirement is stronger.

From another point of view, this parameterization can be compared to the target position parameter in terms of transfer between different values. The amount of transfer between tasks corresponding to different values for the accuracy required from the agent has actually more straightforward properties, than when the target position is the parameter. Indeed, if we consider two values $\epsilon_1 \leq \epsilon_2$ defining two decreasing accuracy requirements, then any policy that is successful for ϵ_1 is successful for ϵ_2, and a policy successful for ϵ_2 is simply likely to be good for ϵ_1. In other words, this parameterization defines a hierarchy over tasks, in the sense that being able to complete some tasks is required to be able to complete some other tasks.

Following these observations, we propose to consider the accuracy parameter as being part of the goal instead, so that a goal is now "reaching a certain target position within a certain radius ϵ". Then we define Accuracy Based Curriculum Learning (ABCL) as a specific form of curriculum learning which acts on the value of ϵ to improve learning efficiency, and we study the impact of ABCL on the learning efficiency of a multi-goal deep RL agent trying to learn a reaching task.

Our contributions are the following. First, we show that being trained with various accuracy requirements is beneficial to learning efficiency even when the required accuracies are sampled in a random order. Then we show that using ABCL to maximize CP as defined in Baranes and Oudeyer 2013 enables to automate the sampling of accuracy constraints in order of increasing difficulty, and that such ordering results in a better learning efficiency than random sampling. Finally, we compare this idea to the existing literature.

3.3.2 Methods

In the standard setting, the ϵ parameter is set at the beginning of training to a user-defined value and left untouched afterwards, whereas here ϵ changes from one training episode to the next. But changing ϵ during the reward/termination calculation without adding its value to the state would remove the MDP structure of the problem: a given state could be either terminal or not depending on ϵ, leading to incoherent updates and perceptual aliasing (Whitehead and Ballard 1991) for DDPG. Thus we extend the UVFA framework to contain both states, goals and accuracies using a $V(s, g, \epsilon; \theta)$ representation. The MDP structure of the environment is not changed by this addition: ϵ simply keeps constant whatever the transition, and is only used to compute the termination condition.

As in the first curriculum learning experiment, we use DDPG as the learning algorithm once the accuracy for the episode has been chosen. We simply augmented the basic DDPG implementation with two techniques from the literature: the gradient inversion technique from Hausknecht and Stone 2015 to better deal with the bounded action space, and target clipping from Andrychowicz et al. 2017 to mitigate target critic overestimation. Finally, we do not use any exploration noise, as we have found that such exploration noise is detrimental to performance in the environment, due to the fact that picking a new target position randomly at the beginning of each episode constitutes a strong enough form of exploration.

All the other parameters of DDPG used in this chapter are taken from baselines used in systematic studies (Henderson et al. 2018; Islam et al. 2017): we use a minibatch of size 64, a replay buffer of size $1e6$, the critic and actor learning rates are respectively 0.001 and 0.0001, $\tau = 0.001$ and $\gamma = 0.99$. Both critic and actor networks have two hidden layers of 400 and 300 neurons with ReLU activations and are densely connected. They are trained with the Adam optimizer.

Two training strategies are studied in this chapter. The first, called RANDOM-ϵ, consists in sampling a training accuracy ϵ uniformly from $E = \{0.02, 0.03, 0.04, 0.05\}$ at the beginning of each episode, and adding it to the input state of the UVFA for all the duration of the episode.

The second, called ACTIVE-ϵ, consists in measuring the agent competence progress for each $\epsilon \in E$ all along training, and sampling an accuracy at the beginning of each episode proportionally to its current competence progress measurement. The way we measure learning progress is directly taken from the SAGG-RIAC algorithm (Baranes and Oudeyer 2013), and described now.

For each possible value of $\epsilon \in E$, the agent is tested every 1000 steps on its ability to reach 10 randomly sampled target positions with this accuracy, providing a competence score between 0 and 1. From the ordered list of scores obtained this way during training, we extract a measure of competence progress as follows: if c_k^i is the score measured for ϵ_i at measurement time t_k, then the competence progress cp_i at time T is the discrete absolute derivative of competence scores for ϵ_i over a sliding window of the $2N$ more recent evaluations:

$$cp_i = \frac{\left| \left(\sum_{j=T-N}^{T} c_j^i \right) - \left(\sum_{j=T-2N}^{T-N} c_j^i \right) \right|}{2N}. \tag{3.5}$$

The use of an absolute value guarantees that if competence starts dropping for a given ϵ, then its associated progress will be negative with high absolute value and thus will be sampled in

priority over other slowly progressing accuracies. This avoids catastrophic forgetting. For all experiments, $N = 3$ is used to compute progress on sliding windows containing 3000 steps. Given this cp_i for ϵ_i, we define the probability of sampling ϵ_i as

$$P(\epsilon_i) = \frac{cp_i^\beta}{\sum_k cp_k^\beta}. \tag{3.6}$$

The exponent β determines how much prioritization is used, with $\beta = 0$ corresponding to the uniform sampling case (RANDOM-ϵ), and $\beta = \infty$ to only sampling the accuracy value ϵ resulting in the maximum competence progress. In the experiments, we determined by sampling β in the range $[0, 8]$ that using a value of $\beta = 4$ was providing the best results.

Both strategies are compared to the baseline that corresponds to the DDPG algorithm being trained to reach target positions with rewards obtained from the hard accuracy only $\epsilon = 0.02$.

All strategies are evaluated on the high accuracy reaching task: every 1000 steps, the agent is run for ten 50-step episodes on randomly sampled target positions, and recording the proportion of successful episodes for $\epsilon = 0.02$. Thus the baseline is always trained with accuracy constraints corresponding to that used for evaluation. By contrast, both ACTIVE-ϵ and RANDOM-ϵ are designed to sometimes train on easier requirements, and thus train less with the evaluation accuracy constraint.

3.3.3 Results

In this section, we validate the hypotheses that: (1) sampling ϵ randomly from E is more efficient for learning high accuracy behaviors than always using the strongest accuracy requirement ($\epsilon = 0.02$), and this despite seeing less experience with such a high accuracy; (2) sampling ϵ using competence progress on each accuracy level results in ordering accuracy requirements from the easiest to the most difficult; (3) the resulting curriculum improves learning efficiency over random selection.

Figure 3.3 shows the benefits of training with multiple accuracies as well as that of prioritizing values of ϵ depending on the level of competence progress of the agent.

The RANDOM-ϵ strategy already provides a clear gain over the baseline: one can observe a significant increase in accuracy as well as a reduction of the variance across different runs. The ACTIVE-ϵ strategy is shown with $\beta = 4$. The resulting curriculum provides even better learning performance than RANDOM-ϵ: progress is faster at the beginning of the learning curve, the agent is more accurate in the end and shows less variability.

Figures 3.4 and 3.5 focus on the curriculum obtained with $\beta = 4$ and parallels the evolution of competence progresses for each value of ϵ with their sampling frequencies. We observe on Figure 3.4 that lower precisions lead to quicker progress at the beginning of learning compared to the most demanding task with $\epsilon = 0.02$. After about 150K steps, the opposite trend is observed: the agent competence on low precisions starts to reach a plateau as it masters the reaching task with these accuracies, leading to a decrease in progress; instead high accuracy remains challenging and the associated competence progress stays higher until later during training.

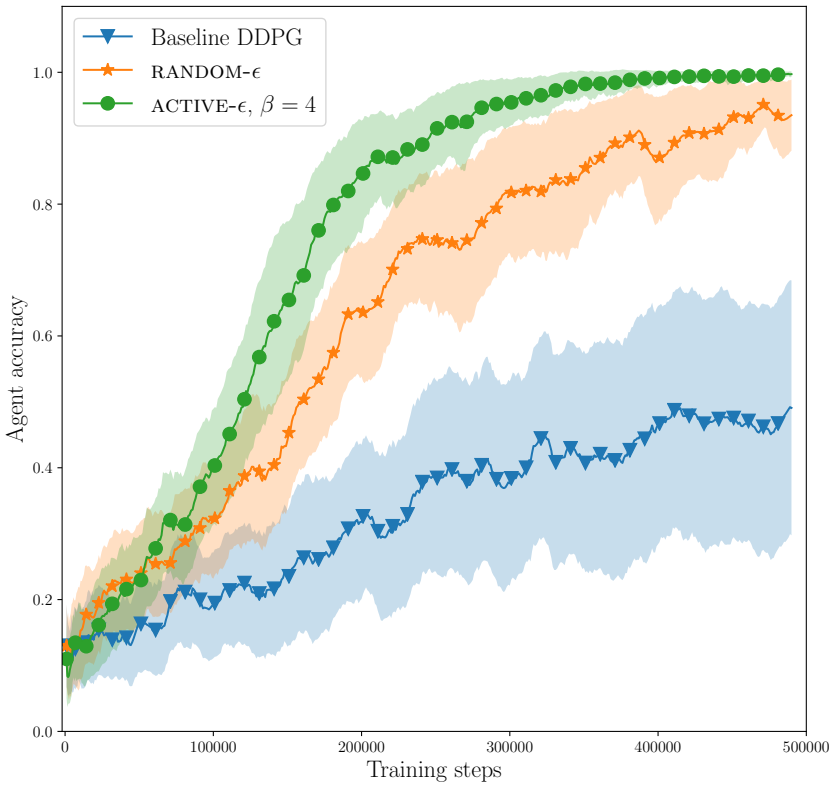

Figure 3.3 – Average observed accuracy of the agent on 10 randomly sampled target positions across the environment with a required accuracy of 0.02 versus number of steps taken, averaged over 10 independent runs. The RANDOM-ϵ *and* ACTIVE-ϵ *(with $\beta = 4$) strategies are compared to the baseline algorithm alone. Colored areas cover one standard deviation around the mean of the 10 runs.*

In parallel, Figure 3.5 shows the proportions of all ϵ sampled represented by each specific value of ϵ, and reflects how their sampling frequency changes with training. During the first 150K steps, priority is given to low accuracy objectives with high competence progresses, and the situation is reverted afterwards, with the agent almost only sampling the strongest accuracy requirement in the end, for which it is still making progress.

3.3.4 Discussion

Our method is related to several existing reward schemes in RL, but comes with fewer external knowledge and better properties.

Related work

The idea of incorporating the accuracy of the arm movement in the reward scheme of the agent is reminiscent of the reward shaping method used in standard versions of Reacher-like environments. Reward shaping consists in providing heuristic knowledge by an additional reward at each transition (s, a, s'), so that r becomes $r + F(s, a, s')$. An interesting property of reward

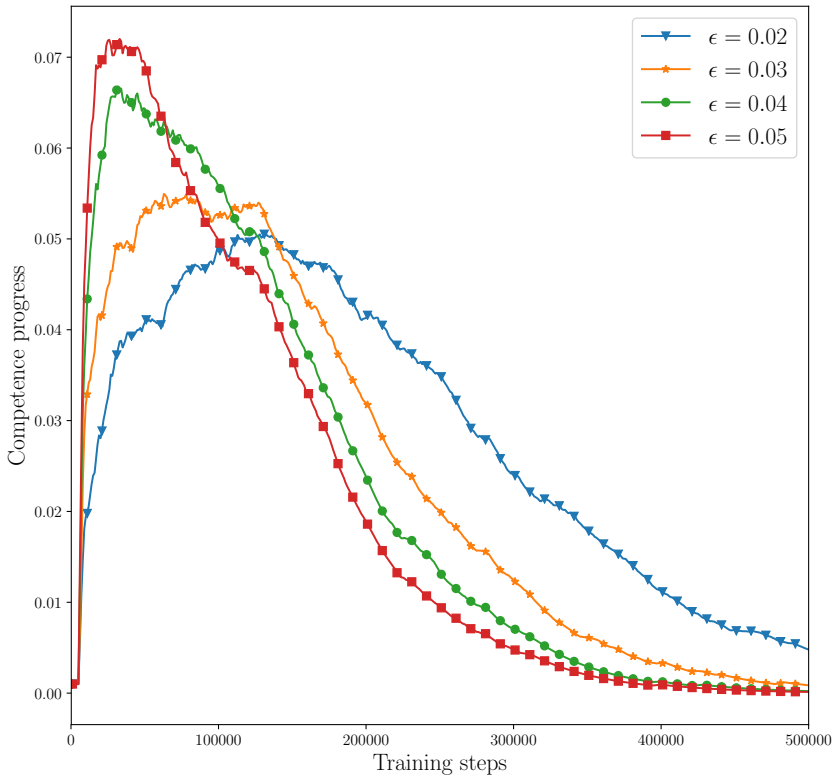

Figure 3.4 – Evolution of competence progress for each $\epsilon \in E = \{0.02, 0.03, 0.04, 0.05\}$ *with training steps, averaged over 10 runs, for the* ACTIVE-ϵ *strategy with* $\beta = 4$.

shaping is that when it is potential-based, namely that the added shaped reward function can be written $F(s, a, s') = \gamma\phi(s') - \phi(s)$, then the optimal policy under the full reward $r + F(s, a, s')$ is the same as under the initial reward. However, with a good choice of shaping function, the time taken to learn the optimal policy may be significantly lower (Ng et al. 1999).

For all that, defining a shaping function F that is potential-based and truly reduces the learning time requires much knowledge about the environment. Instead, our setup *reduces* the amount of external knowledge given to the agent. Indeed, by augmenting the goal with ϵ and letting the agent choose the level of accuracy adapted to its level, we remove the need for the engineer to specify this accuracy on the basis of the environment properties. This comes at the cost of making the problem slightly more difficult for the agent, as the latter now has a larger input space.

Our work also differs from the reward engineering schemes that are sometimes used in control tasks, where the agent is rewarded to "get closer" to the goal, by setting $r = -|p_{target} - p|$. This form of reward definition does not add much external knowledge and is a way to densify a potentially sparse goal-conditioned reward function. However, such rewards can be deceptive: we have no guarantee that two target positions close in the Euclidean space can be reached with close policies for a robotic agent. For example, a position can very well be reachable with a certain policy and within a large radius, but becomes unreachable with the same policy start within a smaller radius, because of a limited joint or something blocking the path in the

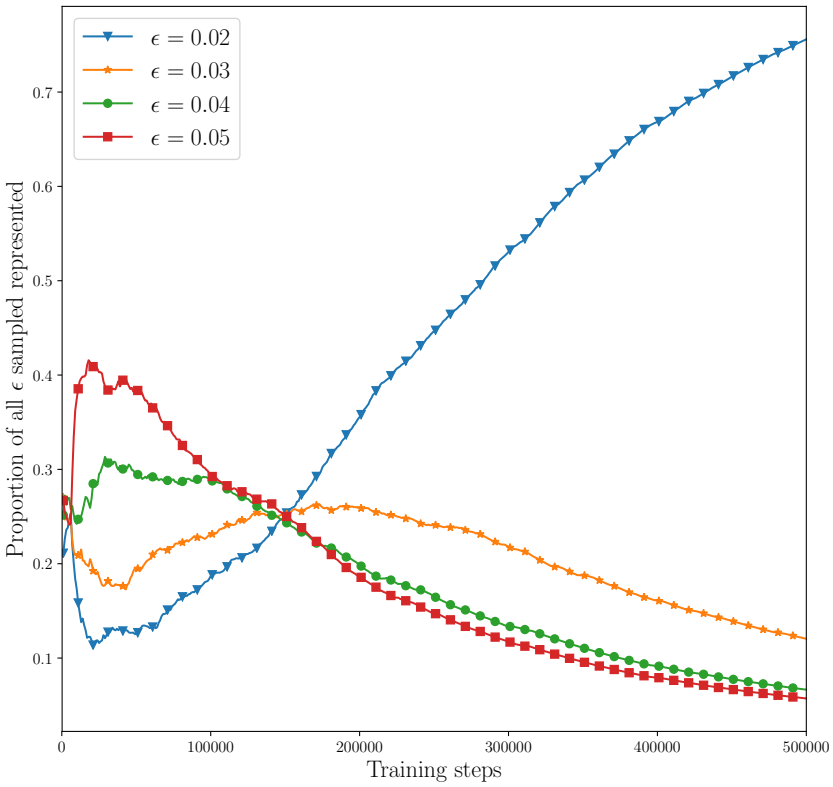

Figure 3.5 – Proportions of all ϵ sampled during training represented by each specific value in E, versus number of steps taken, averaged over 10 runs, for the ACTIVE-ϵ *strategy with* $\beta = 4$.

environment. In mathematical words, distances in the Euclidean space have no reason to be meaningful in the policy space. As a result, the form of reward shaping described is misleading as it would strongly reinforce getting closer to the objective by following a path that actually ends up being a dead end.

On the contrary, by integrating the desired accuracy in the input, our agent never reinforces the wrong actions: if it succeeds in reaching some position with a low accuracy that it could not reach with a higher accuracy, then the actions taken are reinforced *in the states that comprise the low accuracy input*. These reinforcements say nothing about states with high accuracy objectives in them. Of course, the agent might still wrongly generalize from a low accuracy to a higher one, but this is inevitable in such environments without adding external knowledge.

To our knowledge, the work described here is the first use of the ϵ parameter as more than a hyperparameter in control tasks. Kerzel et al. (2018) adopt an approach that ends up being similar to ours: they rely on artificially augmenting the size of the target goal zone, through a manually set curriculum of sizes that involves thresholds. Also, by not integrating the size of the goal reaching zone into the input of the controller, the agent needs to be given additional memory storage to avoid perceptual aliasing between experience acquired for different sizes. Finally, the agent does not feature any type of automatically found developmental trajectory, as the management of the different objective sizes is human-engineered.

3.4 Conclusions

In this chapter, we have focused on how the recent goal-conditioned RL paradigm for multi-task RL can be used to build agents that perform self-determined learning and curriculum learning.

We have first explained how goal-conditioned RL relates to multi-task RL, which has been more studied. Both paradigms for RL offer the agent the possibility to choose its learning situations in an adaptive fashion, and thus to perform curriculum learning. Following our objective of improving RL with developmental sciences, we have proposed an approach explicitly centered around a developmental hypothesis called the Learning Progress Hypothesis (Kaplan and Oudeyer 2007; Oudeyer 2018; Oudeyer et al. 2016).

The latter suggests that human agents organize their learning situations into developmental trajectories by globally aiming at situations where they maximize their learning progress, or CP in the case of goal-reaching behaviors. We attempted to apply this hypothesis to the continuous control of a two degree-of-freedom arm with RL, by reproducing the work of (Baranes and Oudeyer 2013) initially performed with lazy learning based on Approximate Nearest Neighbor, but this attempt was unsuccessful.

In order to improve our approach, we conducted a survey of existing works on curriculum learning for multi-task RL and we sought to identify the obstacles to curriculum building that are specific to RL and could have made our initial attempt unsuccessful. The survey showed that methods used to build curricula in RL often implicitly build on ideas related to competence-based intrinsic motivations (Oudeyer and Kaplan 2009) or to the Learning Progress Hypothesis that we use explicitly. However, they often fail to provide general and interpretable gains (over random baselines) without external knowledge, because the deep RL algorithms they rely on are used in MDPs in which they cannot have good generalization properties between goals or tasks, so that the explicit or implicit self-evaluation of their competence in these environment is bound to be unstable and inaccurate.

From this analysis, we adapted our first attempt by augmenting the goal representation, and thus the MDP to improve the stability of the agent competence evaluation: instead of aiming at reaching states within a fixed accuracy, the agent we have proposed aims at reaching states within a chosen accuracy, and we added the desired accuracy in the agent goal representation. As a consequence, the task space contains a natural hierarchy that the agent can represent meaningfully.

In these conditions, we have shown that simply training on multiple accuracies and thus seeing *less* experience with $\epsilon = 0.02$ in favor of larger ϵ values provides a substantial gain over the baseline. This suggests that the agent is able to take advantage of rewarding experience acquired with large ϵ values and then properly generalize the policy learned from these values to smaller ones. Finally, adding CP maximization to this setting provides another performance gain, and results in developmental trajectories where the agent starts practicing low-accuracy control and then refines this control. As a consequence, we have shown that CP is an effective metric to build a curriculum in the context of deep reinforcement learning, provided the agent is endowed with representations of its goals or tasks that enable it to generalize properly between them.

Though we have validated the potential of explicit competence progress measures to drive the selection of learning situations, our experiments were made in a simplistic environment, and

the difficulties encountered in building a natural curriculum, similar to some observed in the literature, led us to manually design an artificial set of tasks with good properties inside a single "true" reaching task. In our the chapter, we seek to apply the Learning Progress Hypothesis to build curriculum learning agents in more realistic and complex settings.

Chapter 4

Curriculum Learning for Imitation and Control

In the previous chapter, we used accuracy as a variable to build a curriculum on, but for a single goal-parameterized control task. Now we would like to show uses of CP maximization for building curricula over multiple tasks.

4.1 Multi-task, multi-goal learning

The problem becomes that of acquiring multiple qualitatively different skills, some of them still potentially parameterized by continuous goals, like *pushing* an object *somewhere*: pushing defines the meaning of the task, while the goal position is its parameterization. With several goal-parameterized tasks, only adding a parameterization goal g to the state input would lead the agent to be unable to make the difference between reaching position g, pushing an object in position g or holding an object in position g. A naive solution consists in defining the goal space for the agent as the product space of the goal spaces of the different tasks, and using UVFA as it is in this goal space. We will see later that this solution does not work well.

The difficulty in this multi-task multi-goal case is that unlike in our previous work, distinguishing qualitatively distinct tasks in the input through different values of a single parameter is likely to be a bad idea. Adding a single real-valued task parameter t to the state-goal pair (s, g) as input to the network is theoretically enough for the neural network to output highly different behaviors for two different values t_1 and t_2, but t as a parameter would then bear *absolutely no meaning* and in practice, the agent would wrongly generalize between different values of t and thus be extremely slow to learn strongly different behaviors.

Instead, a possible parameterization of such tasks is through one hot vectors t with length the number of tasks in the environment. This requires to set in advance a finite number of skills to acquire, which is a limitation from a developmental point of view. This idea combined with competence progress maximization presented in the previous sections led Cédric Colas at INRIA Bordeaux and me to develop in parallel two multi-task multi-goal deep RL agents with automated curriculum generation capacities: Continual Universal Reinforcement learning with

Intrinsically mOtivated sUbstitutionS (CURIOUS) in the continuous control case in Colas et al. (2018), and CLIC in the discrete control case with a different focus in Fournier et al. (in press).

4.1.1 CURIOUS

The experiments by Colas et al. with CURIOUS firmly establish that the change in architecture to integrate one hot encoded tasks is necessary as the naive product goal space strategy proposed above does not work at all. This result is simple to explain: an agent based on the combination of goal-conditioned policies and UVFAs in their standard form is forced to learn how to reach single points in the goal space from points in the state space; if the goal space is the product space of the goal spaces of each task, then a point in this space describes a goal for *every task*, so that the agent rewarding itself for reaching points in the goal space is thus learning to achieve goals for every task *simultaneously*. This latter objective is of course both much harder than achieving tasks separately, and overall not what we are interested in.

From a developmental standpoint, the results showcase several good properties of competence progress maximization. First, it leads to idiosyncratic learning trajectories, meaning that the sequence of skills the agent acquires and its order are affected by its random exploration of the environment and what it discovers during the latter, so that two agents in the same open-ended environment will learn potentially different tasks or will learn tasks in different orders, if no natural constraint in the tasks definition fosters specific learning trajectories. Then, CP-maximization allows to reduce the negative impact on learning of several common situations in a realistic environment, in a statistically significant way: the CURIOUS agent demonstrates a resilience to sensor perturbation, catastrophic forgetting — thus confirming the benefits of using absolute values to compute CP —, and resilience to distracting tasks.

The last point is of particular interest for the rest of our work. In a realistic environment, an agent is indeed likely to be confronted to many situations where it cannot learn to control anything more than it already does. In such situations, the competence progress maximization algorithm proves tremendously useful, as it serves as a binary classifier between learnable and not learnable.

4.1.2 Limitations

On the negative side, results also confirm the difficulty of building curricula that improve global speed at environment control acquisition: Figure 7 of Colas et al. 2018 shows that, in presence of only tasks that can be learned, the CP maximization does not bring any speed up in global success rate across all skills. Compared to our results on the much simpler reaching task, the difference can be explained by the fact that this time, tasks are not organized following an inclusion-based hierarchy where achieving a difficult one necessarily means also achieving some easier one.

For all that, the tasks definitions are not exempt of all forms of hierarchy. Indeed, the defined tasks are *reach*, *push*, *pick-and-place* and *stack*, so that at first glance, being able to reach any point of the 3D space should help the other two tasks, and being able to pick-and-place a block should help the agent stack a block on the other. As already mentioned, even if easier forms of

control seem to be discovered before as intuition suggests, it does not completely help the agent achieve mastery of more complicated forms of control.

A second limitation of the multi-task approach is that the tasks themselves are externally provided to the agent in the form of standard ad hoc reward functions (lightly parameterized though), and do not necessarily correspond to skills that would naturally be discovered by a fully autonomous agent in a multi-object context.

4.2 CLIC

In CURIOUS, the tasks represented with one-hot vectors correspond to "tasks" from a human point of view (e.g. pushing), that come with their human-defined reward function. From a developmental point of view, there is no real reason for pushing, or picking or any behavior rewarded to actually be rewarding for an agent placed in autonomy in such an environment. In this section, we present our work on the combination of multi-task multi-goal learning and CP maximization in a different context.

4.2.1 Motivations

Precisely, we seek to apply the Learning Progress Hypothesis studied in the previous chapter to a new interactive and developmental learning context, where an agent is in an environment with a number of objects, and another agent manipulates them for its own purposes. Our goal is to evaluate the possible use of CP maximization to let the agent pick both what it manipulates in autonomy and what it imitates the other agent on. This context is motivated by observations on both object manipulation and imitation by children.

Object-oriented learning

Most environments contain numerous separately evolving objects, that change states when acted upon. Drawing a parallel with how infants learn by manipulative exploration of the objects in their surroundings, it seems sensible to explore such environments by attempting to act on and control these objects (Ruff 1984). This is the approach we adopt in this work, and a control objective corresponds to being able to set an object individual state to a desired value.

In appearance, this approach entails providing the agent with external knowledge in the form of objects identification, separation and tracking. Actually, the idea of focusing on objects instead of tasks is founded on the observations from developmental studies of children that human seem to be born with innate "core knowledge" about objects, that includes to some degree the ability to track them and expectations regarding their behavior: "cohesion (objects move as connected and bounded wholes), continuity (objects move on connected, unobstructed paths), and contact (objects do not interact at a distance)" (Spelke and Kinzler 2007). A naive strategy to learn in such a context would be to randomly choose an object at the start of each episode and explore how to act upon it during the episode. Two issues arise with such a strategy.

First, an environment can contain too many objects to interact with for random exploration to be sample efficient. For example, if the agent plays with a shape-sorting cube toy, it is likely that the agent would gain from focusing on this object instead of switching to another one instantly after a first failed attempt at manipulation. Switching objects too often may lead the agent to never capitalize on its attempts to explore better options.

Second, environments exhibit structure in the sense that some objects may be more difficult to control than others, not controllable at all, or interdependent. Again, in the example of the shape sorting cube, manipulating objects of each shape can be an objective at first, before trying to insert one shape in the sorting cube, then another, and so on. Intuitively, picking random objects to practice is likely to be suboptimal: some may not be controllable, or become so only after having mastered others, while training too much on already controlled ones may slow down progress on new ones.

In other words, random exploration in an environment with many objects possibly in a structure is probably sample-inefficient. The agent is once again in the situation where it should learn and follow a curriculum, on objects instead of tasks this time, both to focus long enough to learn and to leverage the environment structure.

Imitation learning

In chapter 2, we have explored a first way infants interact with their caregivers with low-level social signals. We focus in this new chapter on a second form of low-level interaction that is paramount to human development, even though its innate nature is still debated (Jones 2009; Meltzoff and Moore 2005): imitation.

In an environment with several objects, each associated to many possibilities to interact with them, imitation is a natural and effective strategy to improve learning sample efficiency, by narrowing the exploration space (Huang and Charman 2005). We choose to focus on imitation because its combination with RL has been the focus of intensive research for several years, under the Learning from Demonstrations (LfD) paradigm (Schaal 1997).

Usually, several hypotheses are made in LfD: 1) an expert provides demonstrations with the intent to guide the agent, 2) and it does so for one task only, that the agent does not have to identify and knows it should imitate. Following the observations in Chapter 2, we keep our initial objective of not relying too much on predefined interaction protocols. In this case, the hypotheses above are not necessarily met in a realistic environment setting. Precisely, the other agent can be unconstrained, meaning that they act independently of the learner instead of providing demonstrations, and they can perform behaviors with various intentions, that are not readily observable.

Although this other agent is not teaching, its behaviors may result in coincidental demonstrations for various ways to control objects: cleaning up the room will likely result in demonstrations for how to manipulate toys, for instance. Importantly, the learner may want to reproduce behaviors from this other agent that are only accessory, and do not constitute its final intentions.

For all that, the learner may not benefit from trying to reproduce *all* the behaviors it observes: the forms of control shown may be too hard/easy to reproduce given the current level of mastery

of the agent. Consequently, on top of helping focus on promising objects and leverage structure in the environment, curriculum learning can also be used by the learner to identify *which effects* to imitate among those caused by the other agent.

As a final remark, we do not attempt in the work presented next to limit the amount of demonstrated behaviors, unlike other works (Kang et al. 2018). The reason is that we consider an external artificial agent providing non intentional demonstrations by acting independently, a context in which many demonstrations can be considered available, contrary to that of a human intentionally teaching the agent with demonstrations of varying abilities, as is usually the case in interactive reinforcement learning (Argall et al. 2009).

4.2.2 Outline

In the two previous paragraphs, we have described a reinforcement learning setting that strongly differs from that of CURIOUS, and which is original. In this setting:

- the environment does not contain any predefined task and does not provide external rewards, but contains a number of separately available objects, potentially organized into a structure of relationships and dependences,
- an other unconstrained agent interacts independently with the objects, coincidentally providing demonstrations for the control objectives of the learner,
- the learner agent must explore autonomously and imitate the other agent, by following a learned curriculum so as to gain maximum control of surrounding objects and do so faster than with random exploration and imitation.

The challenges for the learner agent in this setting are three-fold: (i) purposefully controlling parts only of the state space (namely objects), (ii) extracting from an other agent's behaviors the control objectives they help learn (objects for which they contain fortuitous control demonstrations), and (iii) devising a curriculum learning strategy to leverage the structure of the environment and improve sample-efficiency.

We first describe our model of environments with objects, then we propose an agent named CLIC which aims at single object control and that uses CP maximization and imitation learning to learn faster in our environment model. This agent is then evaluated on multiple instances of this model and we discuss its results in comparison with existing work.

4.3 Environments

To carry out experiments in this challenging setting, we need to build an environment with interrelated objects of tunable difficulties, as well as a third party agent capable of acting to follow its own objectives. While continuous state environments are more realistic from a developmental point of view, they are more difficult to engineer in such a way that they have predetermined properties, and require the agent too much time to learn the kind of behavior we want (object manipulation with continuous actions is a very difficult topic (Jang et al. 2018; Plappert et al. 2018; Popov et al. 2017; Wang et al. 2019a)). For these reasons, we choose in this

work to model the type of environments described in our learning setting with a discrete state, discrete action environment. It is easier this way to impose specific relations between objects, as well as to define a good policy for the third party agent. We will see that such an environment can still model accurately important properties of objects.

Let us describe our environment model, its different instances used for the experiments, and the second unconstrained agent.

4.3.1 Modeling objects

In a realistic environment, stimuli associated to identical objects can only vary or be controlled together. Instead of perceiving the environment with raw sensors measurements and having to identify these objects from scratch, this work makes the simplifying but developmentally sound assumption that the environment is readily perceived as a collection of separate objects. It focuses on learning how to control them independently and quickly by leveraging the environment structure and an external agent.

The given objects in the environment may have various dynamics and be of distinct difficulties. In this work, we take the stance that the difficulty of an object is related to the amount of intermediate states it needs to go through to be controlled. To illustrate this, take the example of a door: to open it, an agent should first grasp the handle, rotate it enough in some direction, and draw or pull. This action sequence is necessary for the object “door" to go through three states: “handle grasped", “handle rotated" and “door open". A finer model of the dynamics of a door could lead to defining a more complex series of intermediate states, but the idea of intermediate states remains.

To model this form of difficulty, we use grid environments, and model each object in the environment as a list of locations in the grid. For each object, this list defines the locations the agent should go through in the grid to make the object state change, the same way the agent has to go through the three mentioned states to open a door. The atomic moves in the grid model correspond to atomic actions in reality. The same way reaching the intermediate state of an object can be achieved with different sequences of atomic actions, reaching a location in the grid can be achieved with different sequences of atomic moves. This model enables us to define a notion of difficulty, as an object modeled by a longer list of locations (corresponding to an object with more intermediate states to go through) requires a longer sequence of moves.

Formally, the environment is a $N \times N$ grid world containing n objects O_i, where the agent can take five actions — RIGHT, LEFT, UP, DOWN and ACT. As explained above, each object O_i is associated to a finite list of l_i positions in the grid $L_i = (p_i^1, \cdots, p_i^{l_i})$. The object O_i goes through several abstract states o_i as the agent goes through the list of positions. Precisely, object O_i state o_i takes its values k in $\{0, \cdots, l_i\}$ and under the actions of the agent, this state value transitions from k to $k+1$ if the agent selects ACT on position p_i^{k+1}. In other words, object O_i is in state k when the agent has gone through the k first locations of the list of locations associated to object O_i but not yet through the $k+1$-th one. The full environment state is the agent position in the grid along with the internal state of each object: $s_A = (x_A, y_A, o_1, \ldots, o_n)$.

This model enables to reproduce the environment properties we are interested in. For example, objects with close dynamics can simply have close or largely overlapping sequences of intermediate

positions. A more difficult object to master is simply defined by a longer list of positions. Last, the hierarchy between two objects O_1 and O_2 can be modeled by the inclusion of L_1 in L_2. For example, in the previous case, one can argue that the handle and the door should be separate objects to maximize control, and that opening the door implies rotating the handle first (following positions in L_{handle}), and only then drawing the door properly to set open it: $L_{handle} \subset L_{door}$.

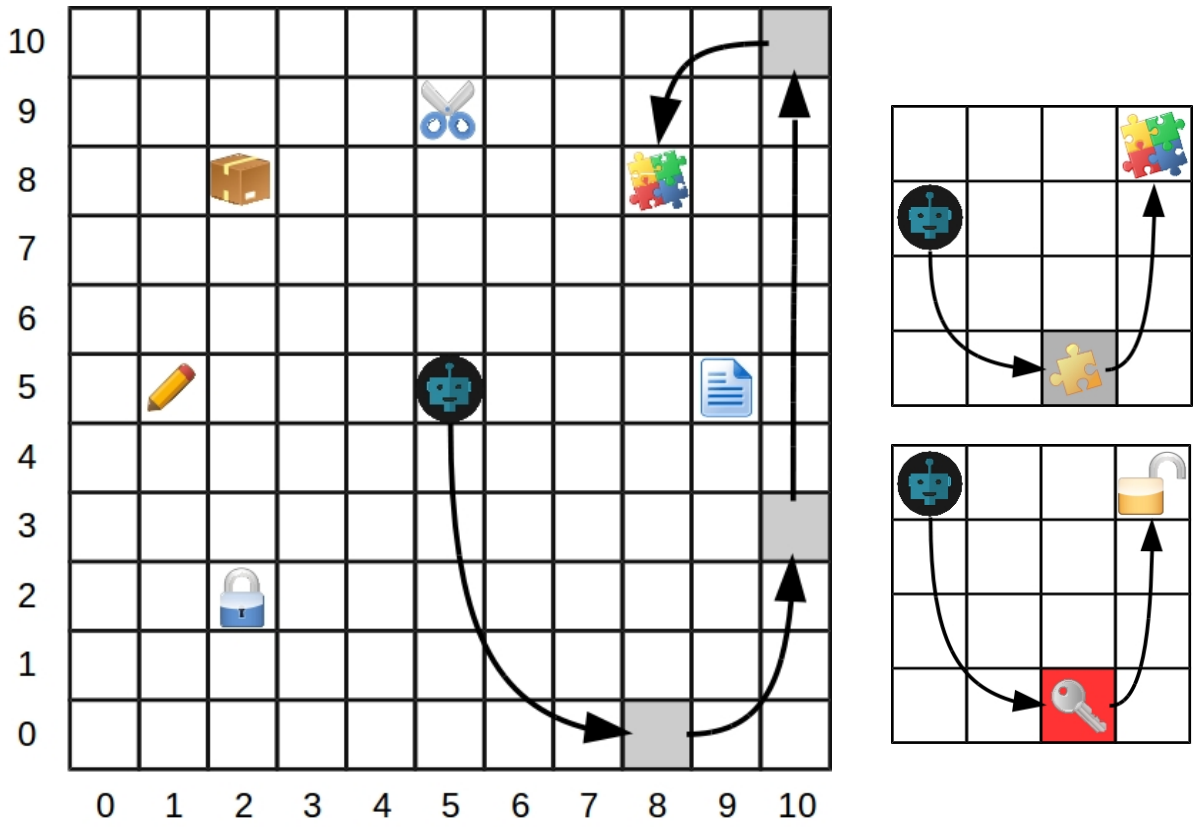

Figure 4.1 – ***Left.*** $\mathbf{E_6}$*, with six objects. To change the internal state of an object, an agent, be it the learner or Bob, has to go through an unknown predefined ordered list of positions, shown here in gray for the jigsaw puzzle. These lists constitute an abstract representation of the necessity for the agent to go through specific intermediate states in order to control objects.* ***Top right.*** *Example of hierarchical relationship between two objects: the jigsaw piece is both an object and an intermediate position for the jigsaw puzzle, so that controlling the internal state of the puzzle requires controlling that of this individual piece.* ***Bottom right.*** *Example of partial control over features: one can imagine that Bob has a key that the agent does not have, and thus can use it to open the lock and continue his actions, while the agent cannot: the position in the grid corresponding to the non-reproducible action of using the key is shown in red.*

4.3.2 A second agent Bob

To model the presence of other agents in the environment, a second agent called Bob can act in the grid, and CLIC may imitate him. The agent Bob is in every way similar to our learning agent, and can move across the grid as well as act on objects. Every F_{demo} steps, Bob achieves d trajectories in the environment (Figure 7, line 12), choosing an object O_i, a value $k \leq l_i$

and setting O_i in state k. To obtain these trajectories, we rely on the property of our object model that the optimal sequence of actions to set object i in state $k \in \{0, \cdots, l_i\}$ is defined by navigating through all p_i^1 to p_i^k in the right order. All objects can be controlled by Bob, but not all can be controlled by the agent. To model this, some objects change internal states only when Bob goes through their list of positions, and not when the agent does so. Bob's choice of objects to act on depends on the experiment and is detailed for each experimental study.

4.3.3 Environments instances

We carry out experiments in four 11×11 grids worlds where some objects can be out of control for CLIC:

- $\mathbf{E_6}$, with six objects all controllable by CLIC. Objects are independent, meaning that for two objects O_i and O_j, the two lists of positions defining their intermediate states for controlling them have no position in common: $L_i \cap L_j = \varnothing$. In other words, the tasks of controlling the two objects cannot contain a common subtask. Also, objects are of the same difficulty, meaning that they require going through the same number of intermediate object states: $l_i = l_j$ for all i, j.

- $\mathbf{E_1}$, $\mathbf{E_3}$, two variations of $\mathbf{E_6}$ with the same objects, but where CLIC can control respectively one and three of the six objects only.

- $\mathbf{E_h}$ with six fully controllable but hierarchically related objects: $L_1 \subset L_2 \subset \cdots \subset L_6$. In this case, Objects 4, 5 and 6 are more difficult to master than objects in $\mathbf{E_6}$, while Objects 1 and 2 are easier and Object 3 is of the same difficulty.

Both CLIC and Bob always start in the center of the grid, and stochasticity is introduced in the environment through a *sticky action* phenomenon: they repeat their last action with probability 0.25 instead of following their policy (Machado et al. 2018). CLIC perceives and stores the states visited by Bob as $s_{Bob} = (x_{Bob}, y_{Bob}, o_1, \ldots, o_6)$, and we assume that the only source of non-optimality of Bob's actions comes from sticky actions.

4.4 Methods

We now detail the CLIC agent, which is based on three components: (1) it seeks object-level purposeful control of the environment by combining goal-conditioned RL and Double Deep Q-Network (DDQN) (Section 4.4.1); (2) it observes and imitates the way an unconstrained agent Bob acts and changes individual objects states in the environment (Section 4.4.3) with an adaptation of the DQNfD algorithm; and (3) it builds and follows a curriculum over objects based on absolute CP maximization to choose what to learn and imitate (Section 4.4.4), similar to that of the CURIOUS algorithm. The outline of the full algorithm is given in Algorithm 7, which is detailed in the next paragraphs.

Algorithm 7 CLIC: Curriculum Learning and Imitation for Control

Input: transition function T, empty replay buffer RB
Initialize: state s, $O_i \sim p_{k,\epsilon}$, step $k = 0$
loop
  $a \sim softmax_a Q^{\boldsymbol{w^i}}(s, a)$
  $s' \sim T(s, a)$
  $RB \leftarrow (s, a, s', \boldsymbol{w^i})$
  Minimize (4.2) on batch from RB
  if terminal or timeout **then**
    $O_i \sim p_{k,\epsilon}$
  end if
  if k % $F_{demo} = 0$ **then**
    $D \leftarrow (\boldsymbol{s_B}, a_B, \boldsymbol{s'_B})$ ▷ Bob's demonstrations (see experiments)
    for all $(\boldsymbol{s_B}, a_B, \boldsymbol{s'_B}) \in D$ **do**
      Augment $(\boldsymbol{s_B}, a_B, \boldsymbol{s'_B})$ with $(\boldsymbol{w^i})$ ▷ 4.4.3
      $RB \leftarrow (\boldsymbol{s_B}, a_B, \boldsymbol{s'_B}, \boldsymbol{w^i})$
    end for
    for N_{imit} steps **do**
      $O_i \sim p_{k,\epsilon}$
      Sample batch B of $(\boldsymbol{s_B}, a_B, \boldsymbol{s'_B}, \boldsymbol{w^i}) \in D$
      Minimize (4.2) + (4.3) on B
    end for
    Empty D
  end if
  $k = k + 1$
end loop

4.4.1 Multi-object control

Common full-state goal parameterization of tasks is not expressive enough to tackle *object-level control* as we wish. Indeed, in our setting, the state space of the agent is the concatenation of the states of the objects in its environment, as in most multi-object environments (Colas et al. 2018; Nair et al. 2017; Popov et al. 2017; Tassa et al. 2018). In this case, the usual goal for goal-conditioned RL is a point in the state space to reach, it cannot achieve a control objective *for one object among several*, the exact same way it could not learn to achieve a goal for *one task among several* in the CURIOUS algorithm (see Section 4.1.1).

Let us give an example in our setting with multiple objects: take an environment with two Objects 1 and 2, where the environment state (s_1, s_2) contains the two object states; then the agent cannot use standard goal-conditioned policies to aim at a specific state for Object 2 without having to also aim at a specific state for Object 1, because the goal state added to the input is of the form (s_1^g, s_2^g); but if s_1^g is removed from the goal input, the agent cannot control Object 1 later any more.

As a result, to control objects individually, the agent must aim at sets of states where objects are in a desired configuration, for example "all states where Object 2 is in some state". So the

agent must add *goal state sets* to its input to reach the desired control, which is done through an adaptation of UVFAs.

Formally, let $\mathcal{M}$ be an MDP without reward function, with state space $\mathcal{S}$, and discrete action space $\mathcal{A}$. With standard goal-conditioned policies, the agent learns to reach $g = (g_1, \dots, g_n)$ from $s = (s_1, \dots, s_n)$, by maximizing R_g, where $R_g(s) = 0$ if $\|g - s\|_2 \leq \epsilon$, -1 otherwise. This parameterization does not let the agent focus on certain aspects of the environment and ignore others; instead it is forced to set $s_i = g_i$ for all i.

To alleviate this restriction, we modify the reward function family to weight each feature s_i and write

$$R_{g,w}(s) = \begin{cases} 0, & \text{if } |w \cdot (g - s)| \leq \epsilon \\ -1, & \text{otherwise,} \end{cases} \tag{4.1}$$

where $\cdot$ denotes the dot-product and $w = (w_1, \dots, w_n)$ is a normalized weight vector with values between 0 and 1. It describes how much setting each feature to its goal value matters to the agent when it comes to rewarding itself. For example if $w_1 = w_2 = 1$ and all others w_i are zeros, then the agent is aiming at any state where $s_1 = g_1$ and $s_2 = g_2$, independently of other s_i. At the present step of the model description, an agent must thus augment it standard environment state with two additional inputs: a goal g and a weight vector w.

4.4.2 Single-object control

In this work, we focus more on the interaction between objects in a structure, imitation learning and curriculum learning, and less on complex tasks involving many objects together. As a consequence, we make several assumptions that do not change the nature of the problem but simplify learning.

First, in our grid-world environments, we are interested in the objective of *individual object control* only, while the family of reward functions defined by Equation 4.1 enables to work with any weight vectors and control any number of objects in the environment. We keep the study of this multi-object control for future work. For now, if the agent state writes $s_A = (x_A, y_A, o_1, \dots, o_n)$, for each object O_i, we define w^i the one-hot encoded vector with a one at the i-th position and zeros elsewhere. So instead of working with any weight vector, we only work with this restricted set tailored for individual object control.

Then, in CURIOUS, such one-hot vectors are used to represent human-defined task in the environment, and the model concatenates them with goals and states directly in its inputs. In preliminary experiments, we observed that this goal-conditioned architecture with one-hot inputs from the start was prone to over-generalization between tasks/objects: Q-values for objects that the agent cannot control would rise quickly with Q-values for controllable objects. This overestimation has a simple explanation: the zero inputs of one-hot vectors do not have a strong enough impact on further layers outputs to enforce a very large difference between the final Q-values of the different objects/tasks. To alleviate this issue, we switch to a gated goal-conditioned architecture, as done in the SAC-X algorithm (Riedmiller et al. 2018), where the networks comprises multiple heads — one per object — and the one-hot vector is used at a gate after the last layer to select which head gives the sought Q-value. As we have seen in Section 3.1.3, this architecture is still goal-conditioned but makes it easier to reach Q-values

with stronger difference between tasks, as the parameters in each head have a direct instead of diluted impact on the network outputs.

Finally, when sampling a goal state for object O_i, the agent can theoretically pick a value $g \in \{0, \dots, l_i\}$, with increasing values corresponding to reaching more and more advanced states in the realization of the full sequence of actions associated to the manipulation of this object (just grabbing the handle of a door, and then rotating it, and so on). Instead, we make the simplifying choice of always aiming at the realization of full sequence of actions, and the agent thus always samples $g = 1$, so that it does not need to augment the state with a goal.

These simplifications are made at the cost of certain capacities of the agent, but once again, this work focuses more on the interaction of multi-object control learning, curriculum learning and imitation learning, which has not been affected by these simplifications, and is intricate enough to study by itself. In the end, the agent boils down to a standard multi-task RL agent with one head for each object control task, and at the start of each episode the agent follows its curriculum to select an object O_i to practice on (Section 4.4.4, Algorithm 7, line 2 and 9), and acts to maximize $R_{l_i,\boldsymbol{w^i}}$.

To this end, CLIC uses UVFAs and approximates the object-related action-value function $Q_\pi^{g,\boldsymbol{w^i}}(s,a)$ with a neural network, taking s, $\boldsymbol{w^i}$ and g as inputs. The agent acts following a softmax exploration strategy and stores the transitions in a standard replay buffer. At each step, it minimizes the double DQN loss J_{DQ} (Hasselt 2010; Hasselt et al. 2016):

$$J_{DQ}^{\boldsymbol{w^i}}(Q) = \left[R_{\boldsymbol{w^i}}(\boldsymbol{s'}) + \gamma \tilde{Q}^{\boldsymbol{w^i}}(\boldsymbol{s'}, \arg\max_a Q^{\boldsymbol{w^i}}(\boldsymbol{s'}, a)) - Q^{\boldsymbol{w^i}}(s,a)\right]^2 \tag{4.2}$$

where $\tilde{Q}$ is the target network.

4.4.3 Imitation

As explained in the introduction, a single trajectory from Bob may contain fortuitous executions of several control objectives. In this work, CLIC simply observes and tries to reproduce Bob's actions that change the states of objects CLIC seeks to control, no matter whether these changes were the true goals of the actions or not. Precisely, each time Bob sets the state of an object O_i of interest for the learner to its value l_i, Bob's trajectory to reach state l_i can be considered a demonstration for this goal-directed control, even if setting O_i state to value l_i is not Bob's goal when he acts.

Each transition $(\boldsymbol{s_B}, a_B, \boldsymbol{s'_B})$ from the d instances of Bob trajectories is augmented with $(l_i, \boldsymbol{w^i})$ pairs identified this way, and CLIC can compute the associated rewards at training time for all such pairs. The augmented transitions are stored both in the same replay buffer as the agent's own experience (Algorithm 7, line 15), and in a separate set D for imitation learning (line 12).

After trajectories from Bob are observed, the agent imitates him for N_{imit} steps (Algorithm 7, lines 18-20). Adapting DQNfD (Hester et al. 2018) to the multi-goal case, we define a large margin classification loss for our general Q-values $J_I^{\boldsymbol{w^i}}(Q)$ that writes

$$J_I^{\boldsymbol{w^i}}(Q) = \max_{a \in \mathcal{A}} \left[Q^{\boldsymbol{w^i}}(\boldsymbol{s_B}, a) + l(a_B, a)\right] - Q^{\boldsymbol{w^i}}(\boldsymbol{s_B}, a_B) \tag{4.3}$$

where $l(a_B, a)$ is a margin function that is 0 when $a = a_B$ and 1 otherwise. This loss ensures that the values for the agent of Bob's actions will be at least a margin above Q-values of other actions.

At each of the N_{imit} imitation step, CLIC selects an object following Section 4.4.4, and minimizes $J_{DQ} + J_I$ on batches of transitions observed for this object. The continual imitation regime proposed here, where the agent regularly tries to selectively imitate Bob all along its autonomous exploration, is different from the pretraining imitation regime described in DQNfD, where the agent imitates Bob once and for all before starting to act in the environment with well initialized Q-values. A consequence of this change is that the margin loss can cause problems around convergence. Indeed, in many cases, the true value function is such that the Q-value of the wrong action in one state is not much lower than the Q-value of the right one, as the agent can for example undo the wrong action next with a well chosen action and be back in the considered state right after. In an even more clearly problematic context, two courses of action can be simultaneously optimal to reach a goal.

In all these situations, true Q-values must be close, or even equal, whereas the large margin loss enforces that the one imitated be separated by at least a margin from the others. So, if the margin is not tuned properly, or if there are multiple optimal trajectories, the large margin loss minimization may lead to incoherent updates. In our experiments, we used 1 for the margin so that only the multiple optimal trajectories case remains problematic, and we did not observe any clear convergence issue.

4.4.4 Curriculum learning

CLIC uses its own CP to select which object to practice on (Section 4.4.1, Algorithm 7, lines 2, 9), and which object to imitate Bob on (Section 4.4.3, Algorithm 7, line 18). To that end, it tracks its CP on each object of interest, and samples preferably those for which it is maximal.

As in CURIOUS, we define a notion of competence for the agent's, $C_k(O_i) \in [0, 1]$ at step k for object O_i, as the average success over a window of l tentative episodes at controlling O — success meaning $R_{g,\boldsymbol{w_i}}(s) = 0$ at the end of episode, with desired value g for o_i. The CP then writes $LP_k(O_i) = |C_k(O_i) - C_{k-l}(O_i)|$, and from now on we use the expression "competence progress" as a synonym of CP. These CPes take real and continuous values that we use to derive for each object a sampling probability at step k:

$$p_{k,\epsilon'}(O_i) = \epsilon' \times \frac{1}{N} + (1 - \epsilon') \times \frac{LP_k(O_i)}{\sum_f LP_k(O_i)}$$

with ϵ' controlling the sampling randomness: $\epsilon' = 1$ means pure random sampling, while $\epsilon' = 0$ means pure proportional sampling. An object for which the agent makes more progress will have a higher probability of being sampled. To ensure a minimum amount of exploration of objects, CLIC uses $\epsilon' = 0.2$. We call CLIC-RND the version of CLIC that samples objects randomly without maximizing progress, using $\epsilon' = 1$. The use of absolute values in LPs ensures a competence drop on a feature leads to training again on this feature (Baranes and Oudeyer 2013; Colas et al. 2018).

4.4.5 Summary and parameters

In summary, the agent starts an episode by sampling an object to act upon and a goal value to set its state to. CLIC-RND performs this sampling randomly while CLIC chooses an object that it is likely to control better after practicing on it. From there, it alternates phases of autonomous learning using the double DQN algorithm with phases of imitation learning using the DQNfD algorithm, both extended to the multi-goal setting. When observing Bob's actions, CLIC simply stores the transitions that lead to objects states changes, to train on them later.

All CLIC parameters are as follows: we take $N_{imit} = 100$, $F_{demo} = 5000$, $d = 25$, $l = 100$, an episode timeout is reached after 200 steps, the batch size is 64, the temperature for the softmax exploration is 1 and the replay buffer size is $1e^5$. Q-values are approximated with a neural network with two hidden layers of 32 neurons with RELU activations.

4.5 Results

In all figures, $C(k) = \sum_i C_k(O_i)/N_c$ is the average normalized competence at step k, with N_c the number of controllable objects. Unless stated otherwise, when Bob is said to act on a set of objects, he samples randomly one of these objects O_i at the start of all his d trajectories and acts to set $o_i = 1$. Plain curves correspond to the median of 10 runs and shaded areas to their interquartile ranges.

In these experiments we examine: 1) if CLIC can identify behaviors in Bob's actions worth reproducing to achieve its control objective, despite Bob never intentionally providing such ad hoc demonstrations; 2) to what extent CLIC's learning trajectory can be influenced by Bob's behaviors; and 3) if CLIC can identify and ignore behaviors that should not be reproduced, because it will not be able to do so, or already knows how to do so. As explained in Section 4.4.4, the curriculum learning agent CLIC uses proportional sampling with $\epsilon' = 0.2$ whereas the basic agent CLIC-RND uses random sampling with $\epsilon' = 1$.

4.5.1 Imitating Bob

First, we analyze the impact of observing and imitating Bob, both in $\mathbf{E_6}$ (Figure 4.2 A) and $\mathbf{E_h}$ (Figure 4.2 B). In $\mathbf{E_6}$, Bob either does nothing, or acts on Objects 1, 2 and 3, or acts on all objects. It is clear that CLIC learns faster when it sees Bob acting on more objects.

In $\mathbf{E_h}$, Bob either does nothing, or controls the intermediate Object 4 or the two hardest of the hierarchy, 5 and 6. This environment is harder to control in autonomy as it requires more exploration to discover the hard objects. Results show that CLIC uses Bob's behavior to gain control over the objects Bob seeks to control, but also over those Bob controls as intermediate steps.

For example, when Bob controls Objects 5 and 6, the hierarchy implies that he has to also act on all other objects before. In this case, results show that CLIC can imitate Bob's actions to control more than only what Bob intended to achieve. When Bob controls only Object 4, CLIC gains

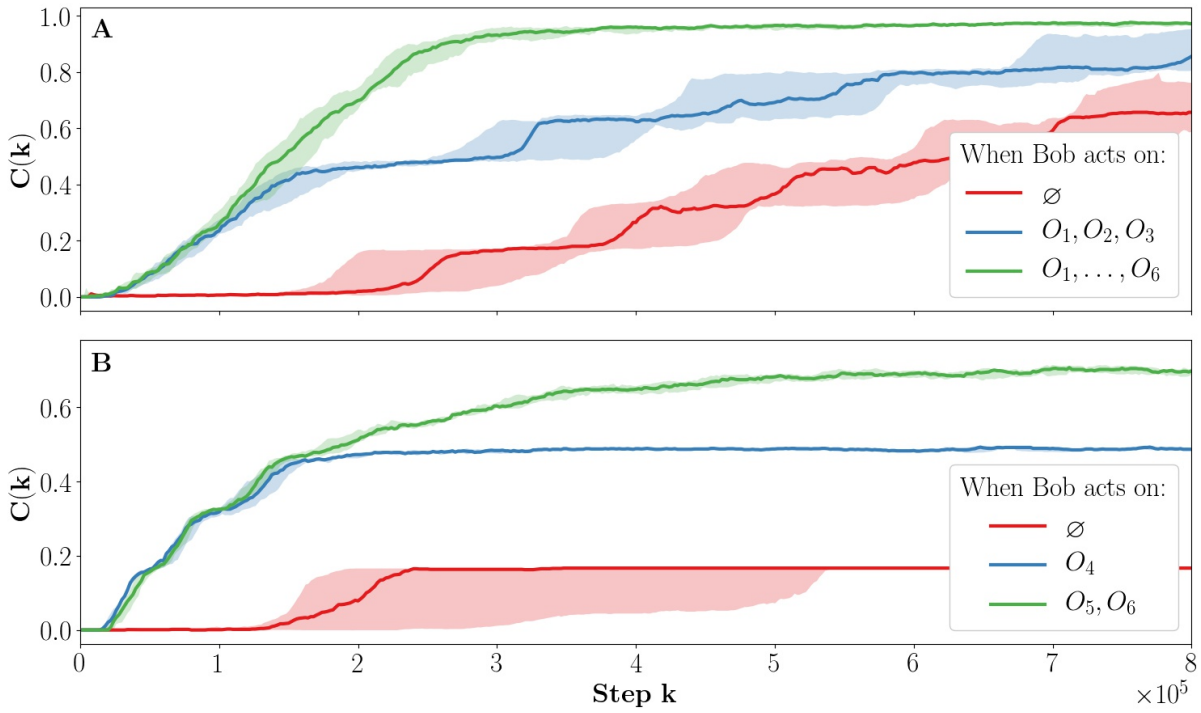

*Figure 4.2 – **A**. Average competence of* CLIC *in* $\mathbf{E_6}$*, when Bob does nothing or controls three or six objects.* ***B**. Average competence of* CLIC *agent in* $\mathbf{E_h}$*, when Bob does nothing or controls one intermediate object or the two hardest of the hierarchy.*

control over Objects 1 to 4 as well, but discovering autonomously harder ones is too difficult (the lists of positions that define them is too long).

4.5.2 Following Bob's teaching

In Figure 4.2 A, when Bob controls only a subset of $\mathbf{E_6}$ objects, CLIC first imitates him to reach an intermediate global competence $C \approx 0.5$, which corresponds to mastering the objects controlled by Bob, and then explores the rest of the objects autonomously. From another point of view, these results prove that Bob can influence CLIC's learning trajectory by only showing it how to control some selected objects. We now show that, acting as a mentor, Bob can completely control this learning trajectory.

In Figure 4.3, instead of choosing randomly an object to control, Bob shows how to control Object 1 until the agent's competence for it is above 0.9. Then Bob demonstrates control over Object 2, and so on. If the agent's competence on a previously mastered object falls below 0.9, Bob provides demonstrations for it again. The figure compares the performances of CLIC (Figure 4.3 B) and CLIC-RND (Figure 4.3 A), in $\mathbf{E_6}$ (where objects are of the exact same difficulties) and with Bob acting as specified.

Without curriculum learning, CLIC-RND's learning trajectory starts to follow Bob's demonstrations order but does not do so up to the end: as CLIC-RND learns autonomously in parallel of imitating

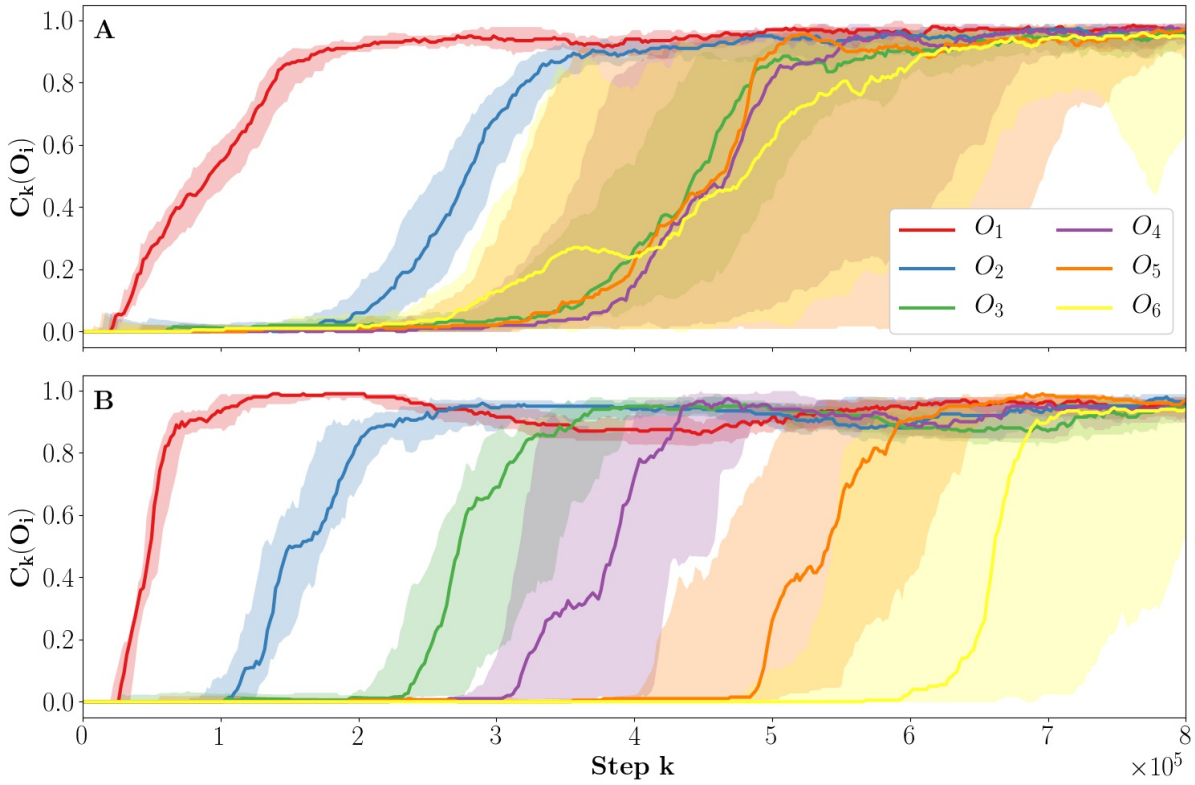

Figure 4.3 – Competence for each object in $\mathbf{E_6}$*, when Bob provides demonstrations in order, for* CLIC-RND *(**A**) and* CLIC *(**B**). CP maximization enables the agent to follow Bob's demonstrations order.*

Bob, and randomly chooses what to learn in autonomy, nothing prevents it from learning to control Objects 4, 5 and 6 before Bob shows how to do so.

By contrast, CLIC strictly follows the curriculum taught by Bob, learning almost always one object at a time. This time, during its autonomous learning phase, CP maximization pushes CLIC to stick to the objects Bob demonstrated, until it does not make progress on them anymore. In other words, CLIC can be taught control over the environment in a desired order by simply showing it demonstrations in this order.

The last result can seem trivial: engineering a system that can carry processes in an order by demonstrating this order is indeed easy. What is not trivial however is to make sure that such a system is not simply always copying what it observes, for such a system would be especially bad at learning each time it observes behaviors it already knows or cannot learn, as it would lose samples and time attempting to reproduce them. To ensure this does not happen, we enabled the agent to discriminate in the behaviors it observes what should be exploited to learn and what should not through the curriculum. Now an agent that chooses itself what to imitate on the basis of an internal criterion is not guaranteed anymore to learn behaviors in the order in which they are demonstrated by a second agent. What this last result shows is that by defining this criterion as the maximization of competence progress, the learner is still endowed with the ability to follow the learning trajectory demonstrated by another agent, provided the demonstrated tasks all fall in the domain of what the agent can indeed make progress on. As will be shown in

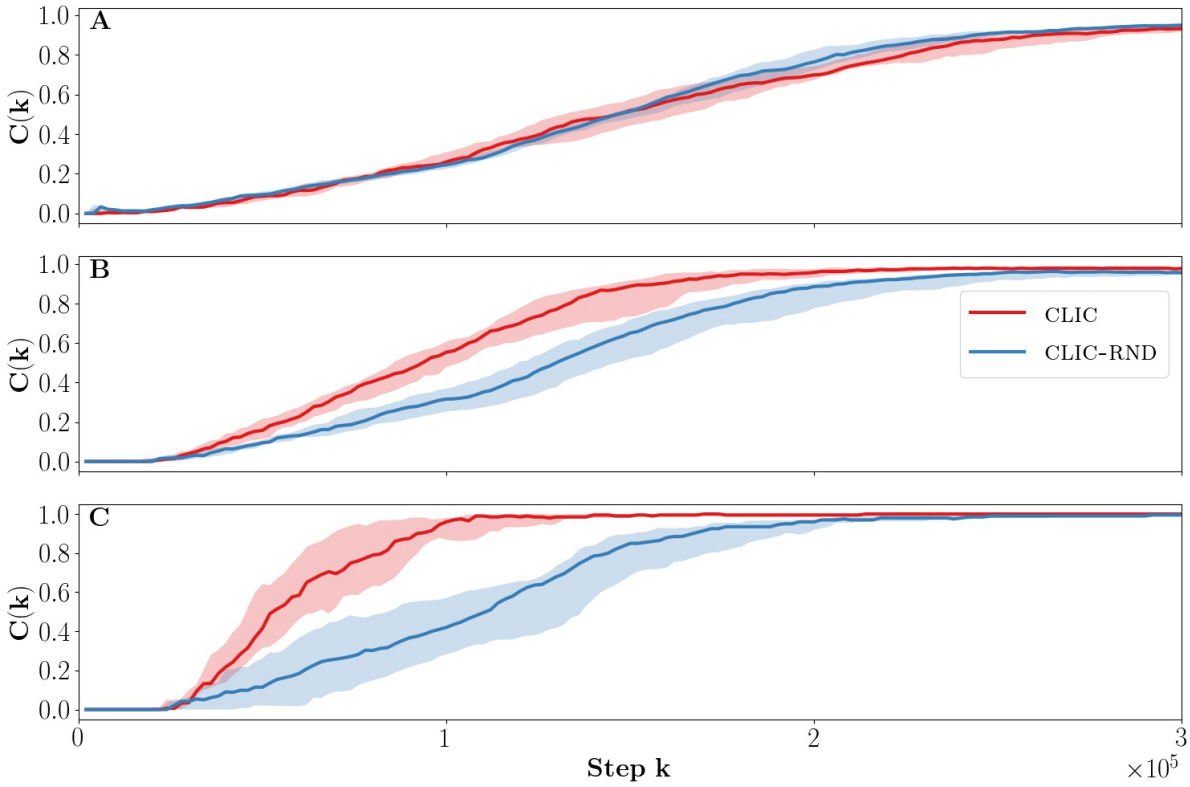

Figure 4.4 – Average competence of CLIC-RND *and* CLIC *over all controllable objects in* ***A***. $\mathbf{E_6}$, ***B***. $\mathbf{E_3}$ *and* ***C***. $\mathbf{E_1}$. *The benefits of CP maximization increase with the number of uncontrollable aspects of the environment.*

the next experiments, when the agent is shown non-reproducible behaviors, it can ignore the demonstrations and focus on other behaviors, while a standard imitation learning agent would be stuck trying to reproduce these behaviors that are useless to achieve its global control objective.

We observe that following Bob's order does not speed up global control of the environment. In $\mathbf{E_6}$, all objects are independent, so there is no transfer between learning the different objects. The ability to focus on learning them one by one is balanced by the fact that the network overfits when it trains on one only at a time.

4.5.3 Ignoring non-reproducible behaviors from Bob

When Bob controls objects that the agent cannot, imitating Bob can be a waste of time. Yet, if the agent knew in advance the subset of objects it cannot control and imitate Bob on, learning would be simpler as the agent could focus on fewer aspects of the environment. But in this work, CLIC does not have this knowledge and relies on curriculum learning to discover what to learn and imitate.

In Figure 4.4, we perform experiments in (A) $\mathbf{E_6}$ with all objects controllable by the agent, (B) $\mathbf{E_3}$ with half of them controllable, and (C) $\mathbf{E_1}$ with only one controllable. In each environment,

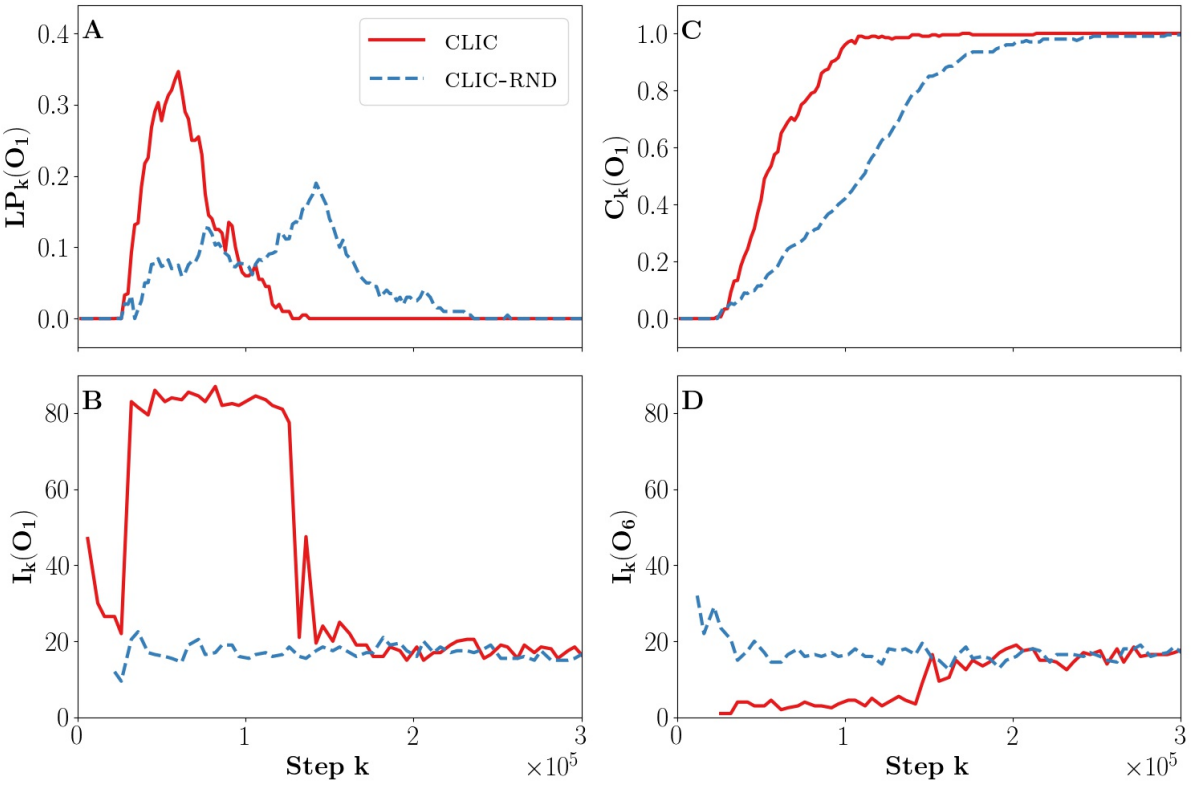

Figure 4.5 – Effect of CP maximization in environment $\mathbf{E_1}$ *with one controllable object* O_1*: **A**. CP* $LP_k(O_1)$*; **B**. Steps spent imitating Bob controlling* O_1*,* $I_k(O_1)$*; **C**. Competence* $C_k(O_1)$ *(same as Figure 4.2.c); **D**. Steps spent imitating Bob controlling* O_6*,* $I_k(O_6)$*, with* O_6 *not controllable by* CLIC*.*

Bob acts on all objects, including those that the agent has no control on, and we study the impact of CP maximization on the agent performance.

Comparing the three blue curves shows that, without curriculum learning, having fewer objects to master boosts only slightly CLIC-RND performances on global control. As expected, without a mechanism to tell between what can and cannot be learned, much of the advantage of having fewer to learn is lost. Also, we note that the variability of CLIC-RND increases with the number of uncontrollable objects: achieving global control becomes dependent on the probability that it discovers the only controllable objects more or less early.

In $\mathbf{E_6}$, when all objects are controllable, CP maximization does not bring any benefit and CLIC does not outperform CLIC-RND. Indeed objects are independent and of equal difficulty, so there can only be a poor form of transfer between them, and a low gain from practicing them and imitating Bob on them in any specific order, as CP maximization pushes the agent to do.

In $\mathbf{E_1}$ and $\mathbf{E_3}$, on the contrary, the red curves show that CP maximization helps achieving maximum control faster in presence of uncontrollable objects: when three or five objects are uncontrollable, CLIC learns faster than CLIC-RND. Besides, the impact of curriculum learning is greater when there are more uncontrollable objects: the gap between CLIC-RND and CLIC performances is wider in $\mathbf{E_1}$ than it is in $\mathbf{E_3}$.

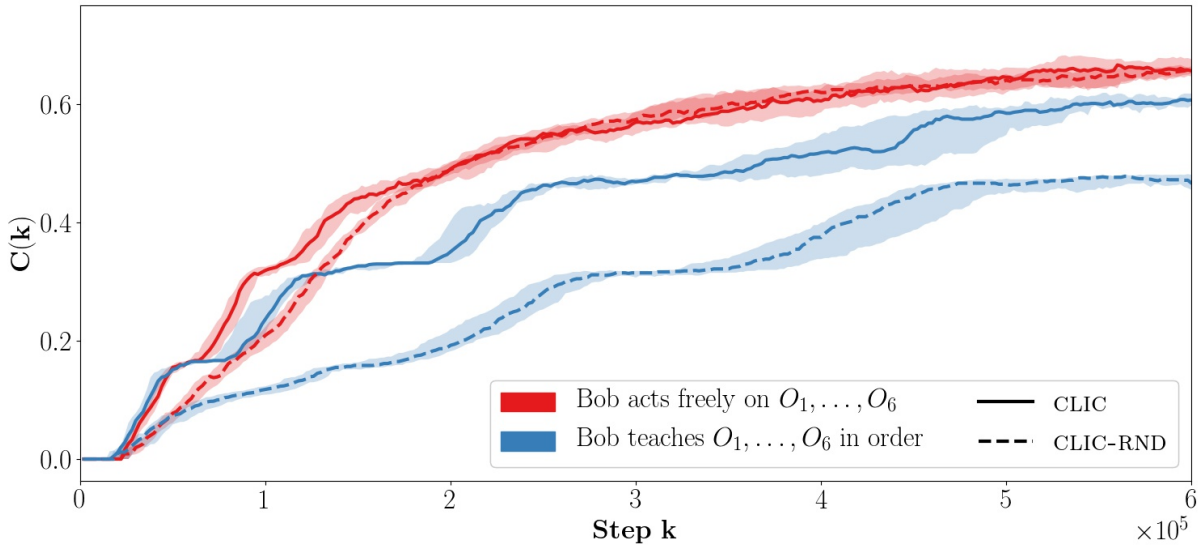

Figure 4.6 – Average competence of CLIC-RND *and* CLIC *in E_h, depending on Bob's policy: selecting randomly objects to control, or teaching the agent in order.*

Reasons for this gain are found in Figure 4.5, which shows the impact of CP maximization when only O_1 is controllable. Figure 4.5 displays the evolution of the agent CP (A) and competence (B) on O_1, the amount of steps spent imitating Bob controlling O_1 (C), and the amount of steps uselessly spent imitating Bob controlling O_6 (D).

When it does not maximize CP (dashed blue), the agent ignores the CP bump (Figure 4.5 A) resulting from gaining some control over the object (Figure 4.5 C). So it randomly selects what to train on, and above all what to imitate: the agent tries to reproduce Bob's behavior when it can as often as when it cannot, as shown by the blue curves at the same level on Figure 4.5 B and Figure 4.5 D. This is sub-optimal as performances on uncontrollable objects will never increase.

Instead, when the agent maximizes CP (plain red), the bump in CP at 50k steps (Figure 4.5 A) results in more imitation for this object (Figure 4.5 B) and more episodes trying to control it. Simultaneously, the agent stops focusing on other objects for which its CP is too small, among which uncontrollable ones (Figure 4.5 D). This focus results in a faster learning pace on controllable objects (Figure 4.5 C). When the controllable object is fully controlled so that the agent does not make progress anymore, all objects appears equally interesting so that the agent starts exploring them all equally again, hence the two red curves finishing at the same level on Figure 4.5 B and Figure 4.5 D.

4.5.4 Ignoring Bob's demonstrations for mastered objects

By contrast with $\mathbf{E_1}$ to $\mathbf{E_6}$, in $\mathbf{E_h}$, all objects are controllable but some are easier than others. In particular, controlling Object i is easier than controlling Object $i+1$, and any demonstration for Object $i+1$ contains a demonstration for Object i.

As a consequence, in such an environment, when Bob acts on random objects, he ends up providing more demonstrations for the easiest objects. If Bob chooses to guide the agent, he initially focuses on these easy objects, and the bias towards them is even stronger.

Figure 4.6 shows the global performance of CLIC-RND and CLIC in these two scenarios where actions from Bob are biased towards easy objects. In both scenarios, curriculum learning helps ignoring Bob's behavior affecting already mastered objects. The effect of curriculum is greater when Bob guides the agent, as the bias is stronger in this case. But the same effect holds when Bob chooses randomly what to do, at least early in learning.

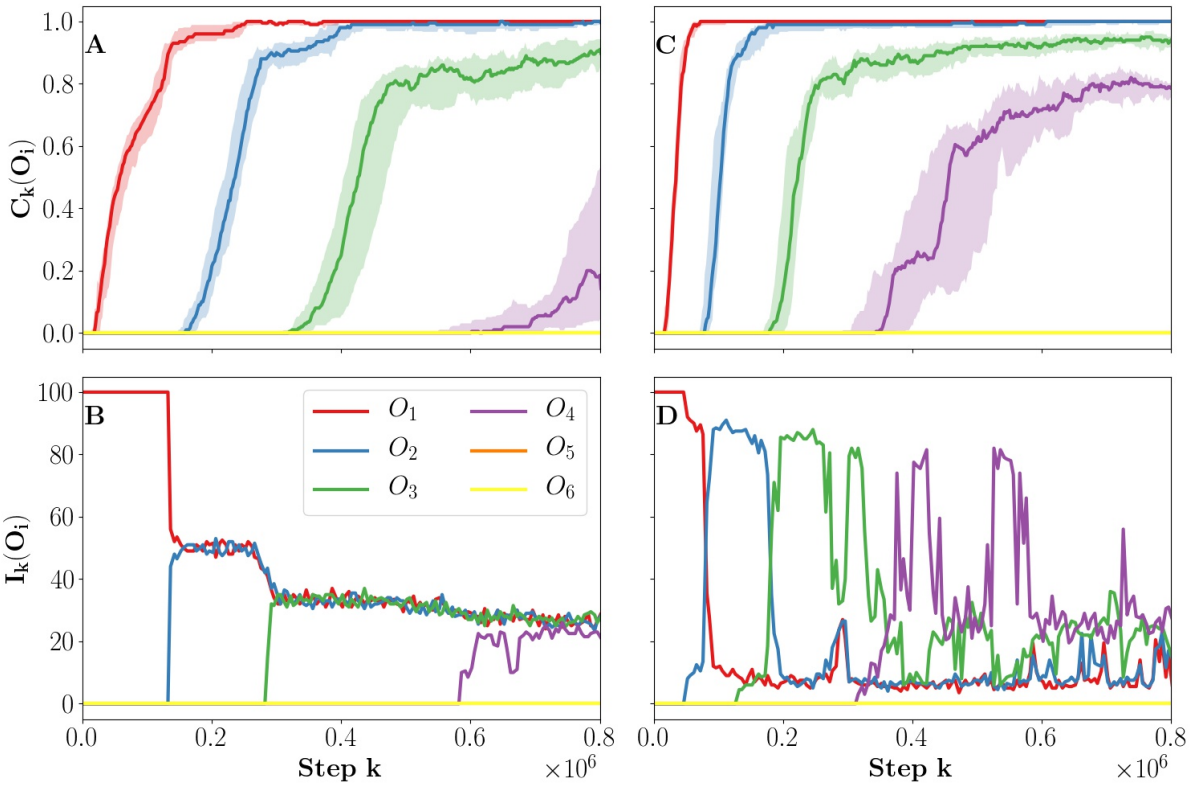

Figure 4.7 – In $\mathbf{E_h}$*, when Bob teaches objects* O_1 *to* O_6 *in order:* ***A****. Competence of* CLIC-RND *on each object.* ***B****. Steps spent by* CLIC-RND *imitating Bob controlling each object.* ***C****. Competence of* CLIC *on each object.* ***D****. Steps spent by* CLIC *imitating Bob controlling each object.*

The fact that curriculum learning enables CLIC to stop imitating Bob on easy objects can be clearly seen in Figure 4.7. Here Bob guides the agent, and the bias towards easy objects is strong. CLIC-RND chooses randomly what to imitate among what it is shown, so it imitates Bob mainly on quickly mastered objects.

For instance, at step 400K, Object 3 is not yet mastered by CLIC-RND so Bob demonstrates it, along with its necessary intermediate steps Objects 1 and 2; CLIC-RND, observing these three objects being controlled, imitates Bob on all of them (Figure 4.7 B), whereas it already knows how to control Objects 1 and 2. Thus it loses imitation steps to make progress on Object 3.

Instead, using CP maximization, CLIC stops imitating Bob on Object 1 (Figure 4.7 D) as soon

as its competence on it reaches 1 and stops rising (Figure 4.7 C). This mechanism enables CLIC to learn more and faster by focusing on non-mastered objects only.

4.6 Analysis

This work proposes to our knowledge the first agent positioned at the intersection of goal-conditioned RL, imitation learning and automated curriculum learning.

4.6.1 Related work: Socially Guided Intrinsic Motivation

The algorithm closest to our work is the Socially Guided Intrinsic Motivation by Demonstration (SGIM-D) algorithm by Nguyen et al. 2011, even more than its different augmentations or variations: SGIM with Active Choice of Teacher and Strategy (SGIM-ACTS) (Oudeyer 2012), SGIM-ACTS for Curriculum Learning (SGIM-ACTSCL) (Duminy and Duhaut 2016), and SGIM with Procedure Babbling (SGIM-PB) (Duminy et al. 2018). All these algorithms are designed to explore the sensorimotor space of real robotic agents.

Following the observations on the challenges of learning in real sensorimotor spaces described in (Oudeyer et al. 2013) — unlearnability, high-dimensionality and unboundedness (see Section 2.5.2) —, the authors of the SGIM-D algorithm propose to improve intrinsically motivated exploration based on the existing SAGG-RIAC algorithm with demonstrations. They describe different "teaching by demonstration" modes, several timing possibilities for alternating intrinsically motivated exploration and imitation, and several demonstration strategies. Results on an experimental setup with a continuous high-dimensional space demonstrate the interest and the efficiency of combining intrinsic motivations with social guidance to discover unknown spaces and to discriminate between interesting spaces and uninteresting ones.

The SGIM-D algorithms has then been extended to provide the agent with several additional capacities: in SGIM-ACTS, it can actively choose between several teachers and strategies (mimicry, emulation and intrinsically motivated exploration); in SGIM-ACTSCL, several tasks share parts of an observation space and are organized hierarchically, so that exploration for one task can be used for another one; and in SGIM-PB, the agent can chain (two) skills through a learning mode where it samples (two) goals to reach consecutively without resetting the environment.

Compared to this series of works, the strongest difference with CLIC lies in the low-level learning paradigm. In the SGIM algorithms, inverse models are learned with non parametric Approximate Nearest Neighbors methods, while we focused on deep parametric model-free RL. As a consequence, we faced very different technical challenges. Crucially, single-feature level control with RL is not a standard machine learning paradigm. Also, with Approximate Nearest Neighbors, imitating a few times a demonstration for goal g is enough to both be able to infer an action to achieve goals close to g later, and to update measures of CP for the goal space region containing g. In deep RL on the contrary, imitating once a demonstration does not guarantee at all that the parameters of the model are then "attuned" to be able to reproduce the imitated goal, and the agent likely needs to train several times on the demonstration samples to reach

this objective. Due to this complexity, the CP of the agent does not evolve in the same way with imitation as in SGIM.

Another difference lies in our focus on object-directed learning and imitation, and in particular on the influence of the interrelations between objects on the importance of curriculum learning to select what to learn and imitate. Finally, we removed the usual assumption in RL of a teacher that willingly provides "useful" demonstrations and showed that CP enables the agent to avoid spending samples imitating unlearnable behaviors, much in line with the active choice of teachers and strategies of SGIM-ACTS.

4.6.2 Related work: other approaches

In this work, we assume that: 1) the agent possibly acts independently of the learner, without the role of explicitly demonstrating helpful behaviors that are reproducible and correspond to the objective of the learner, and 2) its actions have unknown intentions but result in coincidental demonstrations for various interactions with objects in the environment. Some works have departed from the traditional LfD setting in close ways.

Breaking the assumption of a helpful teacher has been approached from a game theory point of view, where the teacher and the learner learn to trade information optimally to learn from each other, by estimating the cost and gain of providing and observing demonstrations (Shon et al. 2007). The idea of unlabeled supervision in general, where the learner must infer what the teacher wants, has been studied in detail theoretically and in practice (see Section 1.2.1).

Our work is more focused on the idea of fortuitous demonstrations from a single third party agent, with no notion of trade or supervision involved, and with the assumption that behaviors are close to optimal with respect to their unknown goals. It is actually closer in spirit to the idea of “Learning from Demonstration in the Wild" (Behbahani et al. 2019), where the objective is to learn behavior models from data acquired with sensors deployed for other purposes, like traffic camera footage.

Third person imitation learning (Stadie et al. 2017), where demonstrations are not provided in the first-person, adds the issue that the learner may struggle reproducing the behaviors of the teacher. First, the agent can be lacking a correspondence model between its states/actions and the teacher's, leading to the well-known correspondence problem (Nehaniv and Dautenhahn 2002). Also, the agent's observations of the teacher's actions and states can be noisy, so that imitating does not always lead to the same behaviors. In this work, we do not address these possible causes for the learner difficulties and instead model the difficulties by simply making some behaviors of the teacher unlearnable.

A small number of works have proposed agents capable of imitating multiple skills from a tutor, either based on Bayesian inverse reinforcement learning with multiple intentions (Babes et al. 2011; Choi and Kim 2012), or using Generative Adversarial Networks to segment skills and imitate them simultaneously (Hausman et al. 2017). These methods are difficult and heavy to adapt to the control problem described here, so we instead propose a simple heuristic to coarsely identify potential intentions and build upon DQNfD (Hester et al. 2018) to actually perform imitation.

Finally, the idea of building a curriculum of tasks based on dependences between objects has been proposed in the study of reinforcement learning with transfer. Close to our work is indeed the object-oriented curriculum proposed by Silva and Costa (2018), based on the object-oriented RL framework of Diuk et al. (2008). Task definitions are given and explicitly object oriented, so that task descriptors include which objects must be used to achieve the task, and curriculum building is based on choosing next tasks that have high *transfer potential*. This last measure includes how many common objects tasks share, and how close their state distributions are, as computed from the domains of the objects present in each task. However this work has different objectives as it aims at maximizing performance on a target task rather than learning reusable goal-directed behaviors, and relies on different mechanisms for transfer between the successive tasks it faces, like value function transfer or transfer via reward shaping (Gamrian and Goldberg 2018).

4.7 Conclusion

In this chapter, we propose a simple multi-task multi-goal architecture and and present two algorithms using it in very different contexts and with different purposes: CURIOUS, studied in detail in (Colas et al. 2018), and CLIC, that we present here.

In CLIC we make the developmental choice of automatically defining tasks in a multi-object environment as individual control of each object. Precisely, we design a discrete grid-like model of generic environments with multiple objects, where the level of difficulty of each object is defined by a distinct list of locations in the grid that the agent must go through to control the object: the longer the list, the more difficult the object. Each object state takes several values, depending on the advancement of the agent through its list of locations.

As in real life, acting in this model of environment is not rewarding, and the agent should instead set its own objectives. We rely on UVFAs to enable the agent to define at the beginning of each episode a “goal" in the sense of a desired configuration for one or several of the objects. In practice, the agent picks an object and attempts to put it in a desired state. This goal object configuration is then added to the input of the agent to pick actions.

To accelerate learning, we consider the presence in the environment model of a third party agent called Bob, that acts selfishly, with the same action capacities as the learning agent: it picks an object and puts it in some state. Bob may not necessarily teach the learning agent, and his goal object configurations when acting are hidden, but the agent can still observe Bob’s states and actions and imitate them. To account simply for the difficulties of third person imitation learning, some behaviors from Bob are coded to be non-reproducible by the learning agent.

In this environment model with Bob to imitate, the learning agent is given the possibility to select which object to train on, and which object to imitate Bob on, by maximizing its absolute CP: if the learner’s progress on an object is low, then it means either it already mastered it or it constantly fails all its attempts at controlling it. In any case, if their exists another object for which it is making more progress, then the agent should switch its focus to the latter. Otherwise, it can keep on exploring or consolidating its behaviors.

We performed experiments with the CLIC agent that combines these ideas — autonomous goal

object configuration setting, third person imitation learning and absolute CP based curriculum — in several variations of our generic environment model. We have shown that in all forms of environments, CLIC can leverage Bob's actions to make progress on its own goal configurations. In particular, it observes behaviors that accomplish Bob's goals and applies these behaviors to make progress on its own. We also demonstrated that CLIC retains the natural ability to learn to control objects in an order showed by Bob, while also being able to ignore those demonstrations when they slow down or halt learning. We showed that this ability to ignore some objects and focus on others provides the strongest gains when the environment exhibits hierarchy between objects. The version of CLIC that does not rely on curriculum learning fails to follow Bob's demonstrations until the end, and is slower in environments where some objects should be progressively ignored because they are already mastered or impossible.

Chapter 5

Discussion

We focused in this dissertation on building agents whose developmental learning trajectory can be influenced both by their own choices with well designed intrinsic motivations and by other agents through interactive learning. In this chapter, we summarize our contributions with a different point of view than the one used until now, and we discuss the global limitations of our work, as well as the perspectives beyond these limitations. In particular, we analyze the connections of our works with three domains of RL: meta-RL, hierarchical RL, and observational RL.

5.1 Contributions

All along this dissertation, we presented our work and its contributions with the idea of answering the question "how can we make RL better with ideas and tools from developmental sciences?". In this discussion, we seek to summarize the contributions of our work with an alternative point of view, where the agent is seen as a collection of mechanisms interacting in a context with specific properties.

5.1.1 A systemic approach to intelligent agents

Precisely, in a systemic approach to building an "intelligent" agent, we consider that such an agent must be endowed with a number of "building blocks", or mechanisms, aimed at fulfilling important functions, like learning, attention, interaction, motivation and so on. These blocks can interact with one another, and an objective when building the agent is that this interaction, combined with the properties of its environment, leads it to display behaviors that are richer in complexity than those enticed by each mechanism separately. Another objective is that these behaviors be reproducible so that the agent may have its repertoire of action capabilities growing with time and experience. To put it briefly, we propose to survey the contributions of our work with a "complex system" view, where we as researchers/engineers seek to articulate different mechanisms so that behaviors produced by their interaction in an environment have good properties.

To illustrate this approach, we can mention the work of (Jasso et al. 2006). In the latter, an RL agent takes as input a human's head and eye direction, a saliency map of the space, and acts to maximize the saliency of the position it gazes at after the last action is taken. The authors show that this objective leads the agent to follow the gaze of the caregiver, as it often indicates salient points. In this model, the new social behavior learned of gaze-following results from the interplay of several components — RL for learning, saliency search for being motivated and gaze-detection for interacting — in an environment with the crucial property that the other agent gazes towards salient points.

Now, we go on to describe the contributions brought by this dissertation. In the light of the systemic approach introduced, the works presented in this dissertation are to a large extent an exploration of different building blocks, and their interaction with RL in environments with specific properties.

5.1.2 Intrinsic motivation, attention synchrony and RL

In our first work presented in Chapter 2, we have studied the interaction with RL of three building blocks — social motivation towards attention synchrony, intrinsic motivation towards novelty, extrinsic motivation towards task-reward accumulation —, in a specific context: in presence of a helping attentional tutor, and under an attention-action (eye-hand) coordination constraint.

A first contribution of this work has been to show that an agent in this last context discovers and consistently achieves the RL task much faster with the intrinsic and social motivation mechanisms together than without them, or with one of them only. The performance of the agent is highly sensitive to the weights of each motivation mechanism and different behaviors result from different weight combinations. Also, our agent happens to be the first RL agent to combine social and intrinsic motivations, provided that these elements were only ever studied together in non-RL agents before. Finally, the agent features the first use of attention synchrony for speeding up task learning in an RL context.

A second contribution has been to conduct task learning experiments after removing any extrinsic motivation towards environmental rewards accumulation. The objective was to reproduce a seemingly natural learning capability of infants, where a caregiver directs their attention towards objects that should pique their curiosity, and lets this curiosity lead them towards discovering new behaviors with the objects without extrinsic rewards. We have partially reached this objective, as we have shown that the interaction of a social motivation towards gaze synchrony with an intrinsic motivation towards novelty, in presence of a tutor and under an eye-hand coordination constraint, enables the agent to achieve the object-related task regularly, but not to learn the optimal action sequence to do so. This is a new example of how multiple decision making mechanisms can interact to lead an agent towards certain behaviors, while each mechanism could not do so in isolation. For all that, these last experiments also highlighted that the agent lacked a specific building block to go further without environmental rewards: intentionality.

5.1.3 Intention generation, intention selection and RL

From our systemic standpoint, the global contribution of Chapter 3 has been to use goal-conditioned RL to build goal-directed agents. Goal-directedness implies two separate mechanisms: generating a set or space of goals, and selecting goals in this set. At the start of our work in Chapter 3, we simply used the state space as a goal space for generation. For selection, we adopted the Learning Progress Hypothesis, and proposed to use the mechanism of Baranes and Oudeyer (2013) based on CP to build a curriculum of goals for an RL agent learning to control a two degree-of-freedom arm. However, this first attempt was not successful. A survey of curriculum building techniques in RL helped us identify that the RL mechanism does not interact well with a goal selection mechanism based on explicit evaluations of the agent competence, if this competence fluctuates unpredictably during training.

In the light of this observation, we have changed the goal generation mechanism so that the competence per goal fluctuates less. To do so, instead of aiming at reaching states of the state space within an externally predefined accuracy, the agent aims at reaching states within a self-chosen accuracy. Each precision defines a task — controlling the arm within this precision — and this task space contains a natural hierarchy, as being more precise can only be harder than being less so. Thus, the agent competence on low precision control fluctuates less as the agent learns high precision control.

Our main result is that goal selection based on CP maximization in this multi-accuracy learning setting provides a boost in performance, and results in sound developmental trajectories where the agent starts practicing low-accuracy control and then refines this control later during learning. The environment, as all control environments, contain a natural hierarchy between tasks defined by different control accuracies. By allowing manually the agent to perceive this hierarchy with an ad hoc discrete task generation mechanism, we have ensured that the task selection mechanism based on CP maximization indeed produces a sound developmental learning trajectory. The results of the agent proposed in Chapter 3 are thus due to the interactions of the two mechanisms of goal selection and generation in an environment with a natural hierarchy of a specific type.

5.1.4 Object control, intention selection, imitation and RL

In Chapter 4, we have interfaced the goal-directedness of Chapter 3 with an interactive learning mechanism and a richer environment centered around objects. Precisely, we have proposed an agent called CLIC which is equipped with building blocks specifically designed to 1) manipulate individual objects intentionally, 2) observe and imitate a non-necessarily teaching agent manipulating objects, and 3) select in a principled way objects to practice, both for autonomous and imitation learning.

We have also built an environment model allowing to enforce structure between multiple objects, following the observation that real environments have properties regarding their objects that influence how autonomous agents may learn in them. In particular, objects are of variable difficulty and can be related to one another, hierarchically for example, so that a learning agent may benefit from exploring their manipulation in a principled way. Some objects can also be impossible to manipulate and thus unlearnable.

We performed experiments with CLIC in our environment model with objects and we have shown that it can leverage an other agent's actions to make progress in manipulating objects, and that it is better in this respect than its non-curriculum-learning counterpart in environments where some objects should be ignored. In other words, the combination of realistic environment properties (unlearnability) with well-design mechanisms of imitation, principled generation of object-manipulation-oriented intentions have led the agent to display behaviors of selective imitation.

We have also demonstrated that CLIC retains the ability to learn control of objects in an order if shown in this order by a second agent, whereas this capacity could have been lost in the process of making the agent responsible for its choice of objects to learn. Once again, this results from the interaction of several mechanisms, as the learner focuses on the right object in sequence only because it so happens that imitating the teacher leads to progress, while competence progress is what drives its object selection for practice. In other words, this successful learner-teacher interaction is the result of a feedback loop between the intrinsic motivation mechanism of CP maximization and imitation learning, and is to our knowledge the first such loop demonstrated in an RL agent.

5.2 Environments: limitations and perspectives

Our approach in the previous discussion has put forth the fact that both environment properties and the agent's multiple learning or interaction mechanisms have an impact on the behaviors emerging from our experiments. In this section, we address a few limitations of our environments. Indeed, we have made choices regarding which kind of environment to use and which properties to include in them, but in retrospect, some of these choices limited the impact of our results.

Experiments in Chapter 2 took place in a small discrete environment with few truly different explorable behaviors. They were in part motivated by the observation that intrinsic and social motivation had not been combined before in RL, and that their interaction looked promising to favor learning behaviors from a developmental standpoint. In Chapter 4, the environment was also discrete, still with low-dimensional state and action spaces in comparison with existing environments for algorithms like DQN, but allowed for a more diverse set of behaviors as it included multiple distinct objects with their own courses of evolution under the agent's actions. Finally, in Chapter 3, we experimented with the continuous control DDPG algorithm, but in one of the simplest environments from existing benchmarks (Reacher), with few controllable parameters and basic simulated dynamics.

Additionally, in the interactive learning works from Chapter 2 and Chapter 4, we relied on the presence of an explicit tutor, or simply an other agent. In both cases, we used hard coded policies for these agents, each time optimal with respect to their objectives in the studied problem: guiding the agent towards task completion with gaze-based guidance in the first case, achieving one's own goals in the second.

5.2.1 Arguments for ad hoc simple environments

The choices we have just described were made for several reasons. First, we focused on demonstrating precise effects of the interaction of multiple mechanisms in agents with specific properties of environments. These latter properties were necessary for our work, could not be found in existing benchmarks, and thus required the manual design of environments instead. Manually designing continuous control environments with the desired properties would have been much harder than doing so for discrete environments and this strongly accounts for our choice of discrete environments.

A second reason for our choice of either discrete or very simple continuous control environments is the sample requirements of the algorithms used. As mentioned in introduction of this dissertation, a strong limitation of deep RL is its reliance on huge amounts of experience. Our experimental setups required the tuning of numerous mechanisms, and the multi-task setting we worked in led to a strong variance in results requiring numerous identical runs to extract statistically significant conclusions. As a consequence, the long training times that would have resulted from more complex environments would have also strongly limited our capacity to study properly our agents.

The rationale behind the use of hard-coded simulated policies, instead of real humans, or learned policies, providing guidance or coincidental demonstrations, is twofold: time and reliability. First and foremost, we considered our work as more of an exploration of new ideas than as the refinement of existing ones, so that it was likely that experiments would require tinkering with our models and setups, and run a lot of experiments. In such conditions, the cost of having human participate in experiments is prohibitive. The same conclusion applies to a lesser extent to the use of learned policies: deep RL algorithms are long to train and not that robust, so that relying on policies trained with such algorithms would have incurred a certain cost to ensure they perform as desired.

Instead, with hard-coded policies, we had a complete freedom to design the policies we desired and an almost absolute confidence in the results they produce. However, such policies constitute a third reason for the restriction to discrete action space in our interactive learning works: we cannot easily write optimal nor even near-optimal policies for third party agents in a continuous action environment.

Finally, there is a weaker but perhaps more motivated reason for our choice not to experiment in any of the many existing deep RL benchmarks. We have observed by ourselves, and by the account of other researchers, that benchmarks are prone to be overfitted by techniques seeking state-of-the-art results more than meaningful, interpretable and statistically significant ones (Henderson et al. 2018; Islam et al. 2017). In other words, RL has not been able to escape Goodhart's Law: "When a measure becomes a target, it ceases to be a good measure" (Chrystal and Mizen 2003; Manheim and Garrabrant 2018). With these observations in mind, we think that searching for new insights and points of view, that sometimes require experimenting on "toyish" environments before being generalized to realistic ones, does not make first experimental results less interesting.

5.2.2 Limitations

Despite all these arguments, there are a number of reasons to explore our insights in more intricate continuous control cases. First, a more thorough analysis of the literature has shown in hindsight that stronger arguments for combining social guidance and intrinsic motivation exist, and likely need more complex and continuous control environments.

Indeed, Oudeyer et al. (2013) argue that intrinsic motivations alone cannot deal with the exploration of real world sensorimotor spaces for three theoretically strong reasons:

- they are high-dimensional and redundant, so that exploration is hindered by the curse of dimensionality (Bishop 2006),
- they can contain potentially unlearnable subspaces, that most exploration mechanisms would fail to ignore, and more importantly,
- they can be unbounded, so that even in case where the agent is provided the information of what it can learn, "the set of learnable predictive models and/or skills is infinite and thus much larger than what can be practiced in a life-time" (Oudeyer et al. 2013).

With these properties in mind, the idea of relying on social interaction to learn faster is not solely a developmental concept that addresses a lack of exploratory capabilities and could be done without with better agents. Instead, open-ended real environments are continuous and intrinsically hard to explore, so that from a developmental standpoint, additional mechanisms like social interaction (or embodiment and maturational constraints) are *needed* to deal with their difficulty, whatever the agent and its computational capacities.

In the light of this analysis, it would be interesting to observe the progressive inefficacy of autonomous novelty search in higher and higher dimensional spaces, which without social guidance would explore the space exhaustively and at the expense of task achievement performance, and seek to mitigate this inefficacy with social guidance. In the same spirit, continuous actions make it possible to display a greater diversity of behaviors, and in a setting with much explorable diversity, it would be of interest to study the kind of behavior produced by an intrinsic motivation based on CP maximization in this case.

5.2.3 Perspectives

A second strong argument in favor of experimenting in more continuous control environments is that the transition could be achieved smoothly for CLIC, and to an extent for the work based on TEXPLORE.

Indeed, as explained earlier, the environmental properties present in our work were certainly easier to implement in an ad hoc discrete environment, but the agent itself does not rely on the discrete nature of the environments. The ideas presented in our work with the CLIC agent could easily be extended to the continuous case by replacing DQNfD with Deep Deterministic Policy Gradient from Demonstrations (DDPGfD) (Vecerik et al. 2017), and the difficulty would rather lie in reproducing an environment with separately controllable objects, but we will come back to this point further in the discussion.

Regarding the work accomplished with TEXPLORE-VANIR and its augmentation, the transition from discrete to continuous environments is also possible. Indeed, our attention was initially brought on this algorithmic architecture because of its high sample efficiency and because it was described as working in continuous cases through discretization of the state and action spaces, so that it could be used in further experiments on real robotic setups. However, it seems that relying on discretization is a relatively weaker approach to continuous cases than having dedicated algorithms like DDPG, as it leads to additional difficulties, like the environment becoming non-Markovian (Dutech 2010). With this observation in mind, one could instead simply reproduce the setup of Chapter 2 in a simulated room and with a prehensile arm, and use DDPG with count-based exploration for deep RL (Tang et al. 2016) to experiment and reproduce our results.

Finally, the methods evaluated in the experiments with accuracy-conditioned manipulation learning in Chapter 3 are completely independent from the Reacher environment and could be leveraged to many simulated environments such as the Fetch environments suite from OpenAI or to real robotic setups. Interestingly, we discovered recently that researchers on goal-conditioned RL have explicitly advocated in a request for research to include the ϵ parameter of most continuous control settings in goal definition (Plappert et al. 2018), exactly as we did.

5.3 Agents: limitations and perspectives

We have discussed our choices regarding the environments used in the experiments, and we can go on to address a few limitations of our agents and their algorithmic design.

5.3.1 Off-policy learning biases

As we explained in Chapter 3, dynamic goal learning behind the goal-conditioned RL tool of goal relabeling, present in HER or in our approach to goal agnostic imitation in Chapter 4, is off-policy learning. We also said that off-policy algorithms like Q-learning, DQN or DDPG, in which value functions updates do not assume that the behavior policy and the target policy match, can be used in this case without causing bias.

As a matter of fact, for this last assertion to be theoretically true, another hypothesis must hold: the behavior policy must visit each state-action pair an infinite number of times. This claim can be made intuitive by an extreme example: if the agent observes another agent in a grid environment only ever going left, it cannot use this experience to learn off-policy how to reach goals on the right of the grid. More formally, the issue is that the state-action pair distribution in the replay buffer is heavily dependent on the behavior policy: if d_μ is the state visitation distribution of policy μ, then the state-action pair distribution in the buffer follows $d_\mu(s)\mu(s,a)$. If this distribution is used to learn Q-values for a different target policy, then there is a form of covariate shift: a change in the distribution of the input variables between the training data and the data encountered during a test phase, here when the agent uses the learned Q-values for actual control.

In our experiments as well as in experiments carried out with HER, this covariate shift seems to have little impact. It is probably because the mechanism based on hindsight that both methods

use guarantees that relabeled goals that the agent learns to reach off-policy are never "too off-policy". Indeed, by definition of the hindsight mechanism, they are reached by the behavior policy, so the target policy explicitly searching to reach them should not be too far from the behavior policy in average.

However, in richer environments, for example in the continuous control case, when the agent has fewer chances to have at its disposal state-action pairs from a distribution similar to the one produced by its target policy, this effect may grow and hinder off-policy learning with algorithms that are called off-policy nonetheless. This effect has already been observed and addressed when learning from scratch from expert demonstrations with DDPG in (Fujimoto et al. 2019), and a more theoretically grounded solution is proposed in Emphatic Temporal-Difference Learning algorithms (Sutton et al. 2016).

5.3.2 Interaction limitations and perspectives

In this dissertation, we studied two interaction mechanisms: gaze-following by searching for attention synchrony, and imitation. These mechanisms are limited in some aspects, either because they involve external knowledge, or because they are too simplified.

In Chapter 2, the interaction with the tutor is too simple in several respects. First it is not specifically sound from a developmental point of view to assume from a tutor to constantly gaze at an object until the agent manipulates it. In retrospect, the properties of the gaze behavior of the agent in this work are closer to those of a pointing behavior: continually pointing towards an object makes more sense to provide indications. A good gaze behavior would be to constantly switch from looking at the agent and observe where it looks, to looking at the object where it should look next.

In a similar fashion, the mechanism pushing the agent to follow the gaze of the tutor is too simple et leads the former to follow the latter too often. Precisely, this mechanism assumes that the agent always perceives the tutor's gaze, and then rewards the agent when it is in the same gaze-state as the tutor. This actually amounts to performing imitation of the tutor in the gaze space. Rewarding state visitation to encourage imitation has indeed been used in several existing works adopting the Learning from Demonstrations paradigm (Peng et al. 2018; Reddy et al. 2019). From a developmental point of view, supposing the agent always perceives the tutor's gaze is unrealistic as it means that the agent continually performs gaze detection. Instead, the agent should learn to look at its tutor to know where to look next, but should not perceive its gaze otherwise (and would thus be learning in a partially observable MDP). Also, the agent should seek to infer the hidden goals of the tutor from its gaze among other social cues.

In Chapter 4, we also make a few simplifying hypotheses regarding the imitation mechanism. First, the training regime for the agent alternates between autonomous learning phases and imitation learning phases, whereas a realistic approach would consist in letting the agent choose when to imitate and when to learn alone, as in Nguyen and Oudeyer 2014. Then, the agent perceives Bob's actions without noise and in its own action representation space. As we briefly mentioned in the discussion specific to this chapter, this approach neglects the difficulties brought by a more realistic third party imitation learning possibility, among which the correspondence problem (Nehaniv and Dautenhahn 2002). However, we argued that the case we studied where

the tutor's actions are not reproducible can be considered a simplistic way to model these difficulties.

5.4 A discussion on objects

In this section, we focus on a particular recurring aspect of our environments: modeling objects. Two of the works presented in this dissertation involve manipulating extra-simplistic models of objects, and we even argued in the last chapter that developmental RL agents should explicitly seek object level control of their environments. In this last piece of work, we devised a simple model for environments with potentially interrelated objects of various difficulties, and an agent to learn in this model, based on competence progress maximization intrinsic motivation and imitation in the wild.

The agent in the model has good and expected properties when evaluated on learning multiple objects control, in particular an ability to use competence progress maximization to avoid the pitfalls of interrelations between objects and unlearnability when imitating. For all that, independently of any curriculum and imitation capability, we have come to realize that it may not have good properties regarding single object control.

5.4.1 Object control: desired properties

We expect a properly designed agent to feature some abilities regarding object-based learning: zero-shot transfer between *identical objects* and few-shots transfer between *similar objects*. For example, let us say that a box contains several balls rigorously identical in color, size and so on, and that a teacher presents a first such ball to an agent to play with it. After some play time, the teacher takes back the ball, and presents another rigorously identical one to the agent. What should be its capacities on this new ball?

Mainly, we expect the agent to almost instantly be able to reproduce on the new ball what it learned on the play ball. Now, if all balls are of different color, we still expect the agent to quickly infer that color plays no role in the effects of the agent actions on the balls. How can such transfer capabilities arise?

Such natural generalization abilities could appear if objects are seen in a common perceptual space, where the agent learns features of objects that explain their behaviors and facilitate their control, so that two similar objects have similar representations in the feature space. From a human point of view, these features should be close to shape, texture, size, color and so on, but an agent learning them in autonomy with its own sensors and for its own control can very well learn different features, perhaps providing more control than we have.

5.4.2 Existing models

This way to have agents with good natural generalization capabilities appears very natural, but for some reason it is not the way many multi-objects environments are treated in deep RL works. Instead, there are two main categories of state representations in multi-objects environments in

deep RL. The first one consists of low level sensor measurements, like raw pixels observations in games or robotics (Jang et al. 2018; Mnih et al. 2015; Nair et al. 2018).

The second approach consist in describing each of the m objects o_i by the values taken by a set of feature functions $(f_j^{o_i})_j$ chosen for the object. The features describing all objects are often the same, and each object o state writes $s_o^t = (f_1^o(o^t), \dots, f_n^o(o^t))$, so that the environment state is the concatenation of the objects states: $s^t = (s_{o_1}^t, \dots, s_{o_m}^t)$. When an agent in the environment samples an action a from its action set A, the next environment state comes from the transition distribution of the environment: $s^{t+1} \sim T(s_t, a)$. These states, action set and transition function define an MDP that models the environment in the traditional way.

Our environment model used with the CLIC agent falls into this second category: it does not seek to represent object properties, and objects are described with one feature only, that only describes how far the agent has gone in the course of actions naturally associated with this object; this is justified by our wish to focus on modeling interrelations between objects and difficulty levels, instead of causality in objects behavior.

Many other simulated robotic environments like the Fetch suite (Plappert et al. 2018), the Deepmind control suite (Tassa et al. 2018) or custom environments use this type of representation: objects are represented by their Cartesian position, their rotation using Euler angles, and their linear and angular velocities, but the features used rarely include a notion of shape, or texture that are often crucial to account for objects behaviors (Colas et al. 2018; Nair et al. 2017; Popov et al. 2017; Vecerik et al. 2017). Instead, all objects states are concatenated into a vector, where each object feature set has a predefined position and can dealt with differently.

The rationale behind this second approach to feature selection for objects is that we as engineers 1) should not provide descriptive features for objects to the agent as they are external knowledge in the form of explanatory factors of variations in perceived observations, and 2) we would not know exactly how to provide abstract features like shape or texture as variables to the agent, otherwise than as raw sensor measurements or pixel observations.

Advantages

The state representation described formally above has two advantages. First, the evolution of the features of an object can depend on features of other objects through the environment transition function, meaning that interactions between objects can be dealt with. Second, objects can be described with features that are actually insufficient to account fully for their behaviors, or not even designed to do so. Indeed, the features for all objects being concatenated into one environment state, two different objects have in all cases their features *at different positions in the environment state.*

Let us consider a cube and a ball in an environment, each described by their position, rotation, speed and angular velocities. Considered only in this mono-object state space, the two objects have different behaviors under the same actions, so that a model of Q-values cannot reliably be learned. Now if we concatenate the two object states into one environment state, the agent can completely learn that the first group of features and the second one have different meanings: it is agnostic to their true causal meaning, which happens to be the same, and simply learns for example that when say the first feature of the state is low (the relative position of the ball),

some actions make this feature greatly increase, while these effects may not hold for the fourth feature (the relative position of the cube, that does not change because of friction for instance).

Downsides

In summary, the current preferred approach at multi-object environment modeling and multi-object manipulation learning relies on:

- either incomplete feature description of objects, overcome with object state concatenation that hides all the missing features in the architecture of the network, or
- complete but very high-dimensional low level representation of objects with sensor measurements, and still most of the time all objects being considered simultaneously in an environment state

We know from experimental results that object manipulation in the second case is an especially hard problem, and many works on continuous control in particular have focused on the first approach applied to simulated environments. However, the aforementioned advantages of the first type of state representation come with a strong downside: they limit the agent's transfer capacities.

For example, if several objects have some close high-level features values (same shape, same size, etc.), and if their dynamics depend only on these shared features, they can be manipulated in close ways, and a good learner should be able to leverage their proximity in features to infer their proximity in behaviors. Function approximation in RL could do exactly that: if the Q-values of the agent are a parameterized function of the properties of the object it observes, then two objects with close properties should have close Q-values, and thus close policies can be used to control them. Of course, this reasoning is valid under some hypotheses of smoothness of the Q-value function, or in other words of the dynamics of the objects.

Unfortunately, this generalization capability of the Q-function approximator is lost with the concatenation of the objects features: the agent is deprived of the knowledge that some features of object 1 and some features object 2 have the same meaning, because they are artificially separated by being placed at two different positions of the input. In the environment model where the CLIC agent is evaluated for example, if two objects are rigorously identical, they are associated with the same sequence of intermediate positions in the grid, and we could expect the agent to generalize instantly from one to an other. Instead, the state of each object is at a different location in the input so that nothing guarantees that learning to manipulate one leads to improvements on the other.

Instead of concatenating object states into one environment states, a better idea would be to apply a single perception module to objects *separately*, so that the agent perceive objects *in a common feature space* and can achieve proper generalization.

5.5 Environment control: a meta-RL point of view

In this dissertation, we have presented goal-conditioned RL as a specific form of multi-task RL, similarly to most existing works. We will now discuss it with respect to another RL approach: meta-RL.

5.5.1 Meta-RL

Meta-RL is meta-learning applied to RL algorithms (Duan et al. 2016; Wang et al. 2016; Weng 2019). In practice, meta-RL is characterized by having different sets of train and test tasks that are drawn from the same family of problems, and a meta-RL agent must be able to learn in few shots how to perform well test tasks after having trained only on train tasks. The tasks are MDPs with common state and action spaces, but can differ in reward function or initial state distributions. The main difference with multi-task RL is that the algorithm is evaluated on test tasks unknown in advance, so that the purpose is the design of meta-algorithms that optimize on line and from train tasks the standard RL algorithms so that they generalize between tasks.

Following the presentation by Weng (2019), a meta-RL agent includes several components: a memory for the agent to retain knowledge from one task to another, an meta-algorithm to meta-learn the reinforcement learning algorithm, and task distributions for train and test sets. Most existing meta-RL works focus on building meta-algorithms to train on a given meta-train task distribution. These algorithms optimize weight updating schemes to foster transfer between tasks and avoid catastrophic forgetting (Finn et al. 2017; Nichol et al. 2018), or RL hyperparameters (Xu et al. 2018), loss function (Houthooft et al. 2018) or exploration (Gupta et al. 2018b) at meta-train time.

5.5.2 Goal-conditioned RL as meta-RL?

Recently, some authors have claimed that goal-conditioned RL is related to meta-RL, arguing that it also has an ability to generalize to new goals at test time from ones practiced at train time (Gupta et al. 2018a).

At first sight, an objection to this claim can lie in the fact that meta-RL focuses on multiple *tasks*, while goal-conditioned RL focuses on multiple *goals*. However, in Chapter 3, we have seen that multi-task RL and goal-conditioned RL are actually close in formal nature. In both cases, the agent navigates in a single MDP, with one transition function and one reward function, but tasks or goals are associated with non-overlapping distributions of visited MDP states, whatever the policy followed. The difference between some multi-task RL and goal-conditioned RL works lies in how the non-overlapping nature of the distributions is obtained.

In some multi-task works, like those based on multiple Atari 2600 games, it comes from the fact that given the games initial state distributions, the global MDP transition function makes it impossible for task state distributions to overlap — a spaceship cannot merge into a ladder in the middle of a game. In goal-conditioned works based on simulated physics environments, where tasks or goals correspond to intentions of the agent, the external environment state space is explicitly augmented with an internal state space representing intentions, so as to recover the

non-overlapping distributions property. In the case of the CURIOUS or CLIC agents for instance, intentions correspond to goal-parameterized "tasks", and a representation of these intentions in the form of a goal variable and a task variable are added to the external environment state observation. In the end, tasks and goals both describe non-overlapping "learning situations", or "learning contexts", whether the difference between the situations or contexts is internally generated by the agent, or externally produced by the environment.

A second objection to seeing goal-conditioned RL as meta-RL is that it does not feature any obvious meta-algorithm to "learn how to reinforcement learn". However, it is possible to consider that the meta-algorithm consist in adding the task or goal representation in the networks input along with weights to link this representation to the networks output, and then letting the training phase adjust these weights. In the end, if the task/goal representation is meaningful and if these task/goal-related weights have been properly learned during the training phase, then the agent should be able to do well on test tasks/goals.

This is exactly what happens with our accuracy-based task representation in Chapter 3 for example, where the natural representation of each task by its associated desired precision leads the agent to generalize well to the hardest task from easier ones. Unfortunately, in many cases, defining task parametrization that foster transfer is not as easy. In the CURIOUS agent for instance, the dissimilarities and commonalities between a reach, a push and a pick-and-place task are still difficult to represent in a meaningful way into intention variables, so a simple approach consist in separating them entirely through one-hot representations. For all that, there is a continuum between these tasks, and if we could trade one-hot representations for richer ones where the common need to approach the object is clear for instance, the agent would probably gain a lot in learning and generalization capacities.

We have focused in this dissertation on goal space generation and curriculum building as goal selection. Both topics are relevant from a meta-RL standpoint, separately or together.

5.5.3 Curriculum learning as unsupervised meta-RL

In Chapter 3 and 4, we proposed a curriculum building goal selection mechanism based on CP maximization. Automatic curriculum learning for goal-conditioned RL can be seen from a meta-RL point of view as a process which optimizes the selection of training tasks. Precisely, Gupta et al. (2018a) make a distinction between supervised and unsupervised meta-RL, depending on whether the meta-train task distribution is human-engineered or learned. Automatic curriculum learning can be seen as a way to change from a stationary task distribution to a non-stationary task distribution in order to accelerate further learning.

Indeed, in an environment with many identifiable tasks, the curriculum is not only here to practice tasks at the right level for the agent; it also builds on the idea that the right tasks learned first make learning more tasks easier next, exactly as a good choice of training tasks is supposed to facilitate learning test tasks in meta-RL. In meta-RL, better than random training task acquisition has been proposed with domain randomization (Tobin et al. 2017) or automatic environment generation by evolutionary strategies (Wang et al. 2019b).

5.5.4 Goal space identification

Along with a goal selection mechanism, we have proposed in Chapter 3 and 4 different ways to define goal/tasks. In many goal-conditioned RL works, the goal/task space is the state space (a task consists in reaching a specific goal) (Andrychowicz et al. 2017; Plappert et al. 2018; Schaul et al. 2015). This was the approach we used in our initial attempt at curriculum building with the SAGG-RIAC algorithm in Chapter 3. It is a form of automatic task identification in the environment, that works well in some cases where a good environment state representation is already provided to the agent.

Beyond state reaching tasks

Unfortunately, defining tasks as reaching states of the environment is a bad idea in general. In Chapter 4, state-based representation were identified to be a problem when state-features correspond to disentangled aspects of the environment, like separate objects. More generally, in a high-dimensional state space, it makes no sense to aim at reaching exact states, as it amounts to controlling every perceived feature of the environment simultaneously. Instead of picking random goal states likely unreachable, the agent can learn to produce better task parameterizations. This issue has been considered in several existing works.

A possibility explored in the literature is to focus on generating semantically different tasks in an environment, in an unsupervised fashion and so that the agent may benefit from meta-training on them before learning other behaviors. Such agents rely on reward generation with automatic option discovery (Machado and Bowling 2016), through algebraic approaches to identify eigenpurposes (Machado et al. 2017) or information theoretic approaches to maximize discriminability and/or diversity (Eysenbach et al. 2018; Gregor et al. 2016; Warde-Farley et al. 2018).

Nair et al. (2018) propose Reinforcement learning with Imagined Goals (RIG) to combine unsupervised representation learning with goal-conditioned RL, so that goals can be expressed in a latent space instead of the otherwise too large sensory space. They show that the latent space obtained allows to define meaningful distance-to-goal reward functions contrary to the image space, and that synthetic goal relabeling coupled with an off-policy learning algorithm boost performance, as in other goal-conditioned RL works. Training on such "imagined goals" picked in the latent space, enables the achievement of new user-specified goals at test-time, and RIG is thus a form of unsupervised meta-RL, even if not explicitly presented this way. Pong et al. (2019) propose a generative process called Skew-Fit to produce goals that are close to those already reached by its previous goal-directed behaviors, and attempts to maximize the coverage of the reachable state space.

A series of works based on IMGEPs (Forestier et al. 2017), and thus not completely in the RL literature, has also studied goal representation learning in simulated and real robotic environments. Péré et al. (2018) propose a 2-stage approach to IMGEPs where the agent first learns passively a latent space from environment observations with Variational Auto-Encoders, and then only use IMGEPs to perform goal exploration, by sampling goals from the estimated distribution of the projection of its observations in the latent space. Laversanne-Finot et al. (2018) extend the previous work to learn disentangled goal spaces with β-Variational Auto-Encoders. Once the

representation has been learned, the method is similar to ours in the CLIC agent: modules are defined as subsets of latent variables, and goals are sampled in a modular fashion as achieving certain values for the corresponding latent variables of the representation of the observation. As such β-VAEs could thus replace our hand-made representation of multi-object states in higher-dimensional environments. Finally, the experiments conducted in the two previous works are extended to a real robotic setup by Laversanne-Finot et al. (2019).

Coupling task identification and selection

In this dissertation, we have touched a second use case for better task/goal representation learning, in both Chapter 3 and 4: making curriculum learning easier. In the CLIC agent, by making state features correspond to disentangled factors of variation in the environment (objects), we make it easy for the agent to identify easier forms of control than others. Without disentanglement, feature control would be harder as all forms of control would be interleaved, so that the agent learning them would be generalizing/forgetting in unpredictable ways all the time. This property of disentanglement is also used in IMGEPs to build curricula, either by making the problem space intrinsically modular (Forestier et al. 2017) or by learning modular goal space representations in (Laversanne-Finot et al. 2018). In both cases, they proceed to use modular competence progress maximization to sample which modules to practice next, following the same Learning Progress Hypothesis as we do. In our work on Reacher however, the almost sole purpose of our task representation is facilitating curriculum learning by allowing the agent to perceive a natural hierarchy where it could not before, without any modular learning.

We have shown that goal representations, learned or not, have an impact on the agent capacity to learn a curriculum. In our works, goal representation are set in advance, and in the work of Laversanne-Finot et al. 2018, they are learned before any autonomous interaction of the agent. An interesting development would be to let both goal representation and goal selection learning mechanisms interact, so that the agent really learns representations that foster progress. For example, the agent could learn a curriculum not over point to reach in the state space, but over points to reach in the learned feature space, and optimize the feature space so that it makes competence progress evaluations more clear. To a certain extent this corresponds to identifying features that are disentangled, not as factors of variation in the environment, but as controllable variations in the environment.

This last idea has been studied in Thomas et al. 2017, and the authors make the connection with principled exploration during human development, but not with the corresponding literature in developmental sciences. Generally speaking, this discussion shows that some issues addressed with a machine learning standpoint in meta-RL correspond to existing objectives of developmental sciences: unsupervised training tasks generation corresponds to the autonomous development of a learning trajectory, and latent goal/task representation learning is directly related to intention representation learning. This connection is in line with our global observation that several subfields of RL and developmental sciences would gain from interacting more.

5.6 Observational and hierarchical RL: a global perspective

The agents presented in this dissertation have shown that the interaction of their mechanisms with properties of the environment, including potential other agents, could give way to new learning behaviors. From a developmental standpoint, this result is promising to build agents that learn incrementally, like humans. However, the agents presented lack several crucial abilities to make true incremental learning a reality: identifying in which context these new behaviors can be used, and knowingly reusing them.

For example, the CLIC agent presented in Chapter 4 learns in autonomy to follow in order the demonstrations of an other agent Bob acting like a tutor, simply using its meta-policy for object selection based on CP maximization. In the experiments, Bob gives no indication that he is acting as a demonstrator and that the demonstrations are likely to make the agent progress in its control of the environment. Instead, the agent discovers this on its own, and if the tutor starts teaching a new sequence of object manipulations, the agent must once again count on its CP-based meta-policy to follow it, one object manipulation at a time.

In a more realistic context, it is likely that any agent acting as a tutor or teacher stands out from other agents acting for other purposes, notably by a number of social signals specific to the teacher/tutor role, like eye contact to initiate an interaction, and then sustained attentional signals to maintain the interaction. This suggests that a better interactive learning agent should 1) identify that in presence of these social signals, the global action of imitating the teacher over a period of time leads to some CP, and 2) decide to perform this global action knowing this.

The ability to choose and perform "higher-level" actions, like an entire subpolicy, refers to hierarchical RL, while the ability to observe the state of other agents and let it impact one's own behaviors refers to observational RL. In our opinion, these two forms of RL are of paramount importance for future developmental learning agents, and we detail a few perspectives and existing works on these topics, in relation to our works.

5.6.1 Hierarchical RL

Hierarchical learning is the ability to "conceptualize the world at different granularities and translate from one abstraction level to the others easily"(Zhang and Zhang 2006). In RL, this potential granularity is often considered at the action level: when considering how to attain long-horizon or complex objectives, a natural way to act consists in dividing the problem into smaller subproblems with easier objectives, and potentially repeating the division further for the smaller objectives. The purpose of these divisions is to reach a point where the main problem has been reduced to a sequence of objectives that the agent knows how to achieve.

This approach is supported by examples in neuroscience, that for example show that animals do not create action plans with the sequences of low-level motor controls needed for each movement, but instead decompose their plans into mid-level abstractions (Jing et al. 2004; Mussa–Ivaldi and Bizzi 2000). This mechanism has been adapted in robotics and produced dynamic movement primitives for continuous control (Ijspeert et al. 2003).

Options

In reinforcement learning, the same idea has led to the notion of temporally extended actions, or temporal abstractions, which are subpolicies that can be followed until they reached a termination criterion. These latter extended actions naturally introduce a notion of hierarchy in control, and the subfield of RL that deals with them is called hierarchical RL (Barto and Mahadevan 2003).

The most common approach to hierarchical RL is the framework of options, that we briefly mentioned in the previous section, based on the theory of semi-Markov decision processes (Sutton et al. 1999a), which make it possible to learn reusable skills. An option is a "closed-loop control rule", characterized by the policy it follows, initiation states in which the policy can be called and a termination condition. Early work on automatic learning of such skills has focused on discrete domains (Chentanez et al. 2004; Vigorito and Barto 2010), where a common approach is to use statistics about state transitions to identify bottleneck states (Mannor et al. 2004; Stolle and Precup 2002).

Hierarchical RL has struggled to transpose the option framework to high-dimensional continuous spaces, but recent years have seen a few solutions emerge for the learning of reusable skills in such domains (Bacon et al. 2017; Florensa et al. 2017a; Gregor et al. 2016; Warde-Farley et al. 2018). Many deep hierarchical RL algorithms now use goal-conditioned lower-level policies, with goals being provided by higher level policies (Kulkarni et al. 2016; Levy et al. 2017; Nachum et al. 2018). The low-level policies can be trained with rewards conditioned on the achievement of their goals, while high level policies for goal selection are trained to maximize a standard reward in the RL environment, inducing a natural hierarchy between the two levels.

Goal-conditioned RL and hierarchy

Goal-conditioned RL in its standard form is not a hierarchical process: goals alone are not options that the agent can pick like regular actions. Goal selection is a process disconnected from the agent-environment interaction loop, so that the agent simply chooses a goal for the duration of an episode at a regular frequency. It cannot explicitly decide to reach an intermediate and mastered goal to make its current task easier, so that it cannot entirely rely on reusable skills.

Being able to choose action and goals in a hierarchical way would be useful for curriculum learning in hierarchically structured environments like the ones we presented. Indeed, the agent could quickly identify, in autonomy or with the help of a tutor, that if achieving task 1 requires achieving task 2 first, then it benefits hugely from learning task 2 and then seeking to achieve it to start in a better state for task 1. At the present time, the agent can only hope that the parameters it reached learning task 2 are a good initialization to learn those for task 1, which is a weaker form of transfer.

Also, in Chapter 4, we focused on simple goal object configurations where the agent aimed at controlling one object at a time. Instead, the agent could aim at controlling multiple objects simultaneously as enabled by our framework, with non-one-hot weight vectors, and we could study the impact of curriculum in presence of hierarchies between *configurations of objects* — controlling Objects 1 and 2 separately as a prerequisite to control them together.

In an interactive context, hierarchical learning could give the agent the possibility to select *higher level interactions*. Taking the same example as before, instead of imitating a tutor one object after the other, the CLIC agent could decide to imitate the tutor when the latter display certain cues, and until it displays other cues. Such an approach fits perfectly within the framework of options that we have just described: starting cues would help define the initiation state distribution, and terminal cues the termination conditions.

5.6.2 Observational learning

For the options described above to be identified by an agent, the tutor should display social cues, and the agent should observe them and learn their interpretation on its own. These conditions are not met in our works, simply because at the time we had not identified them formally yet in the context of RL.

In Chapter 2, the gaze of the tutor is indeed observed. However, the possibility of learning an advanced gaze-based interactive behavior is hindered by the already provided interpretation and use for this gaze, as the agent is rewarded for synchronizing its own with it. In Chapter 4, the tutor does not provide any social cues, and is not even truly observed: its transitions in the environment are simply and directly registered in the memory of the agent. This approach is actually standard practice in imitation learning and Learning from Demonstration.

In Chapter 2, we could have instead kept the tutor's gaze in the state representation, removed the synchrony based reward, and observed if the agent discovered that states where its gaze direction and the tutor's are identical have higher values in general. If they did, then it would be interesting to evaluate whether these high values could be leveraged to learn faster new tasks in sequence, either by forgetting and relearning Q-values in mono-task learning setting, or by adding to the framework intentionality as in Chapter 3.

Generally speaking, the idea that an agent may simply observe other agents and consider their state as part of the perceived environment state, without the knowledge that this state has a peculiar meaning, relates to the notion of observational learning. In observational learning, the agent can still learn about the environment from the behaviors of others, for example because they show what can be accomplished in this environment (Heyes 1993).

From a developmental point of view, it is extreme to consider other agents as any other part of the environment because we know humans perceive differently animate and inanimate entities from very early in infancy (Rakison and Poulin-Dubois 2001). In particular, it is likely that other agents states are not seen as mere variables evolving without more meaning behind it. However, the paradigm corresponds to the developmental requirement mentioned above of not attaching any meaning to other agents behaviors ex ante. This lets the agent infer what is useful for its development, and allows it to adapt to the different social signals provided by others.

This approach has been recently used in RL in Borsa et al. (2019). As described above, the authors augment the learner state with the state of another agent present in the environment, but let the learner act with no additional mechanisms in this augmented state. So the agent can observe the other agent change positions, as the agent in Chapter 2 can observe the tutor change gaze directions, except that in contrast to our work, the learner is not given any indication to use these observations.

In this case, the agent cannot seek to reproduce the tutor's behaviors as no correspondence is given between the tutor's and the agent's states and actions (Nehaniv and Dautenhahn 2002). However, the authors show that if the agent is endowed with some form of memory, like a recurrent neural network as an action selector, then its simple additional observations of the learner behaviors can be leveraged to discover tasks faster in several environments.

Chapter 6

Conclusion

In this dissertation, we started with two general observations: first, the RL community is coming to the realization that its paradigm must evolve to address a number of challenges, and second, these challenges are close to those that the separate fields of developmental sciences have been exploring for twenty years. From there, we drew a general orientation for our work: making the RL paradigm better with tools from the developmental sciences communities, while keeping the advances made by the RL community on its own. In particular, these observations led us to pick a certain approach for the dissertation: how can we build RL agents whose developmental trajectories have the sound property of being sensitive to both the agent social interactions and intrinsic motivations?

To answer this question, we have proposed different agents in Chapter 2, 3, 4. Generally speaking, the social interaction and intrinsic motivation mechanisms we implemented in these agents interact both with one another and with the agent environment, and lead to the emergence of new learning behaviors, more elaborate and more effective than those produced by each mechanism separately.

We have seen in the discussion of Chapter 5 that the experiments we carried out on these agents had limitations and could be greatly improved. First, looking at the environments, many ideas presented in this dissertation would gain in strength and utility in continuous state and continuous action environments. In particular, the developmental approach suggests to confront the agents with environments that share some important properties with the real world: it is unbounded, very high-dimensional and many subspaces of the sensorimotor space are unlearnable.

Then, looking at other agents which might be present along with the learner, we have made the simplistic choice of relying on hard-coded policies. The next step is obviously to address more realistic interaction policies, from imperfect simulated agents, if not real humans.

Finally, looking at the mechanisms we provided our agents with, we identified an important obstacle to their fruitful collaboration and co-evolution: the space in which the agent makes its decision should grow as it learns the potential uses of controlling some aspects of its environment. This points to two aspects of learning that we did not address in this dissertation: hierarchical learning to be able to choose goals of progressively higher level, and maturational constraints to

deal with an increase in accessible outcomes and behaviors in the world.

The approach adopted in this dissertation is still exploratory, and the agents built provide as many answers as new questions. However, we can take home a few general messages.

A recurrent observation in recent RL articles is that authors draw on their intuitions on how children learn to reproduce their behaviors in artificial agents, but do not go as far as to delve into the corresponding literature in developmental sciences. However, this literature has gone much further than intuitions and often provides detailed guidelines for reproducing human behaviors, that cannot necessarily be guessed. For example, the components of joint attention presented in Chapter 2 can be the base for several works leading to a behavior which we almost know for sure will be necessary for the future of human-robot interaction. Likewise, even if we have barely touched the subject, the modalities of early imitation in infants has been the focus of intensive studies which resulted in several theoretical models, like the Active Intermodal Mapping hypothesis by Meltzoff and Moore (1977), and these models could serve as strong starting points for building agents with deep and naturalistic imitation capabilities.

Then, our work shows that it may be profitable to acknowledge some limits of current deep RL approaches and move on. Since the first end-to-end learning pipeline on Atari 2600 games by Mnih et al. (2015), this paradigm for RL has been dominant in a part of the community, but reusable and robust advances have been more and more scarce. A potential and partial cause thereof is a lack of rigor and clear formalization of the tackled issues. For instance, we have shown that the state of the art in terms of automated curriculum learning is almost impossible to identify, simply because the many existing works are not achieved in a properly defined common framework. Also, we have mentioned and briefly analyzed the connections between several subfields in the domain, like goal-conditioned RL, multi-task RL and meta-RL, but these connections have not really been the focus of any formally strong study yet. As a result, it appears sometimes difficult for researchers to simply position their work in the related literature, and we have observed many incoherences on this matter between articles presented in this dissertation.

Finally, it could be useful to perceive RL only as a possible form of learning among others, with its specificities and which could be combined with others. We have seen that the convergence properties of RL algorithms make it difficult to find meaningful and usable curricula, whereas several recent approaches focusing less on RL and using alternative learning mechanisms, like lazy learning, seem to have better properties for the matter (Baranes and Oudeyer 2013; Forestier et al. 2017).

This last observation suggest that a Complementary Learning Systems approach could be leveraged (McClelland et al. 1995), as it suggests to combine learning methods working at different time-scales and with different properties, so that RL could be only one of these methods. More generally, the idea that a learning agent must be a system of algorithms, and that its properties emerge from this system, which we have relied on to present our contributions in Section 5.1, has been already used in developmental sciences in general (Lungarella et al. 2003) and holds promise for machine learning.

Glossary

RIG	Reinforcement learning with Imagined Goals.
ABCL	Accuracy Based Curriculum Learning.
CLIC	Curriculum Learning and Imitation for Control.
CURIOUS	Continual Universal Reinforcement learning with Intrinsically mOtivated sUbstitutionS.
DDPGFD	Deep Deterministic Policy Gradient from Demonstrations.
DDPG	Deep Deterministic Policy Gradient.
DDQN	Double Deep Q-Network.
DQNFD	Deep Q-Networks from Demonstrations.
DQN	Deep Q-Network.
HER	Hindsight Experience Replay.
UVFA	Universal Value Function Approximator.
CP	Competence Progress.
IMGEP	Intrinsically Motivated Goal Exploration Process.
MDP	Markov Decision Process.
RL	Reinforcement Learning.

Bibliography

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). "Hindsight experience replay". In: *Advances in Neural Information Processing Systems*, pp. 5048–5058 (pages 44, 45, 49, 94).

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). "A Survey of Robot Learning from Demonstration". *Robotics and Autonomous Systems* 57.5, pp. 469–483 (pages 6, 61).

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). "The nonstochastic multiarmed bandit problem". *SIAM journal on computing* 32.1, pp. 48–77 (page 3).

Babes, M., Marivate, V., Subramanian, K., and Littman, M. L. (2011). "Apprenticeship Learning about Multiple Intentions". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 897–904 (page 77).

Bacon, P.-L., Harb, J., and Precup, D. (2017). "The option-critic architecture". In: *Thirty-First AAAI Conference on Artificial Intelligence* (page 97).

Baird, L. (1995). "Residual Algorithms: Reinforcement Learning with Function Approximation". In: *Machine Learning Proceedings 1995*, pp. 30–37 (page 33).

Baranes, A. and Oudeyer, P.-Y. (2013). "Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots". *Robotics and Autonomous Systems* 61.1, pp. 49–73 (pages 41, 42, 48, 49, 54, 68, 83, 102).

Barto, A. G. and Mahadevan, S. (2003). "Recent Advances in Hierarchical Reinforcement Learning". *Discrete event dynamic systems* 13.1-2, pp. 41–77 (page 97).

Barto, A. G., Singh, S., and Chentanez, N. (2004). "Intrinsically Motivated Learning of Hierarchical Collections of Skills". In: *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19 (page 3).

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). "Neuronlike adaptive elements that can solve difficult learning control problems". *IEEE transactions on systems, man, and cybernetics* 5, pp. 834–846 (page 2).

Behbahani, F., Shiarlis, K., Chen, X., Kurin, V., Kasewa, S., Stirbu, C., Gomes, J., Paul, S., Oliehoek, F. A., Messias, J., et al. (2019). "Learning from Demonstration in the Wild". In: *2019 International Conference on Robotics and Automation (ICRA)*, pp. 775–781 (page 77).

Bellman, R. (1966). "Dynamic Programming". *Science* 153.3731, pp. 34–37 (page 2).

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48 (page 39).

Berger-Tal, O., Nathan, J., Meron, E., and Saltz, D. (2014). "The Exploration-Exploitation Dilemma: A Multidisciplinary Framework". *PLOS ONE* 9.4, e95693 (page 3).

Berlyne, D. E. (1960). "Conflict, Arousal, and Curiosity." (page 40).

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning* (page 86).

Borsa, D., Heess, N., Piot, B., Liu, S., Hasenclever, L., Munos, R., and Pietquin, O. (2019). "Observational learning by reinforcement learning". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1117–1124 (page 98).

Boutilier, C., Dean, T., and Hanks, S. (1999). "Decision-theoretic planning: Structural assumptions and computational leverage". *Journal of Artificial Intelligence Research* 11, pp. 1–94 (page 17).

Branavan, S. R. K., Zettlemoyer, L. S., and Barzilay, R. (2010). "Reading between the Lines: Learning to Map High-Level Instructions to Commands". In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1268–1277 (page 7).

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). "OpenAI Gym". arXiv: 1606.01540 [cs] (page 48).

Brooks, R. A. (2002). "Flesh and Machines: How Robots Will Change Us". *New York: Patheon* (page 4).

Brown, N. and Sandholm, T. (2019). "Superhuman AI for multiplayer poker". *Science* 365.6456, pp. 885–890 (page 2).

Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., and Nowé, A. (2015). "Reinforcement learning from demonstration through shaping". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence* (page 6).

Bubeck, S., Cesa-Bianchi, N., et al. (2012). "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". *Foundations and Trends® in Machine Learning* 5.1, pp. 1–122 (page 3).

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). "Large-Scale Study of Curiosity-Driven Learning". arXiv: 1808.04355 [cs, stat] (page 17).

Cabi, S., Colmenarejo, S. G., Hoffman, M. W., Denil, M., Wang, Z., and de Freitas, N. (2017). "The Intentional Unintentional Agent: Learning to Solve Many Continuous Control Tasks Simultaneously". arXiv: 1707.03300 [cs] (page 39).

Caruana, R. (1997). "Multitask Learning". *Machine learning* 28.1, pp. 41–75 (page 37).

Cederborg, T. (2017). "Artificial Learners Adopting Normative Conventions from Human Teachers". *Paladyn, Journal of Behavioral Robotics* 8.1, pp. 70–99 (page 7).

Cederborg, T., Grover, I., Isbell, C. L., and Thomaz, A. L. (2015). "Policy Shaping with Human Teachers". In: *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 3366–3372 (page 6).

Chentanez, N., Barto, A. G., and Singh, S. P. (2004). "Intrinsically Motivated Reinforcement Learning". In: *Advances in Neural Information Processing Systems*, pp. 1281–1288 (pages 3, 8, 9, 97).

Chernova, S. and Veloso, M. (2009). "Interactive Policy Learning through Confidence-Based Autonomy". *Journal of Artificial Intelligence Research* 34, pp. 1–25 (page 6).

Choi, J. and Kim, K.-E. (2012). "Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions". In: *Advances in Neural Information Processing Systems*, pp. 305–313 (page 77).

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). "Deep reinforcement learning from human preferences". In: *Advances in Neural Information Processing Systems*, pp. 4299–4307 (page 6).

Chrystal, K. A. and Mizen, P. D. (2003). "Goodhart's Law: its origins, meaning and implications for monetary policy". *Central banking, monetary theory and practice: Essays in honour of Charles Goodhart* 1, pp. 221–243 (page 85).

Chu, V., Fitzgerald, T., and Thomaz, A. L. (2016). "Learning Object Affordances by Leveraging the Combination of Human-Guidance and Self-Exploration". In: *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, pp. 221–228 (page 27).

Clouse, J. A. and Utgoff, P. E. (1992). "A Teaching Method for Reinforcement Learning". In: *Machine Learning Proceedings 1992*, pp. 92–101 (page 6).

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2018). "Quantifying Generalization in Reinforcement Learning". arXiv: 1812.02341 [cs, stat] (page 44).

Colas, C., Fournier, P., Sigaud, O., Chetouani, M., and Oudeyer, P.-Y. (2018). "CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning". arXiv: 1810.06284 [cs] (pages 3, 58, 65, 68, 78, 90).

Colas, C., Sigaud, O., and Oudeyer, P.-Y. (2019). "A Hitchhiker's Guide to Statistical Comparisons of Reinforcement Learning Algorithms". arXiv: 1904.06979 [cs, stat] (page 44).

Cruz, F., Twiefel, J., Magg, S., Weber, C., and Wermter, S. (2015). "Interactive Reinforcement Learning through Speech Guidance in a Domestic Scenario". In: *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (page 6).

Czikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience* (page 40).

Degris, T. and Sigaud, O. (2013). "Factored markov decision processes". *Markov Decision Processes in Artificial Intelligence*, pp. 99–126 (page 17).

Devin, C., Abbeel, P., Darrell, T., and Levine, S. (2017). "Deep Object-Centric Representations for Generalizable Robot Learning". arXiv: 1708.04225 [cs] (page 2).

Dewey, D. (2014). "Reinforcement Learning and the Reward Engineering Principle". In: *2014 AAAI Spring Symposium Series* (page 7).

Diaconescu, A. O., Mathys, C., Weber, L. A. E., Daunizeau, J., Kasper, L., Lomakina, E. I., Fehr, E., and Stephan, K. E. (2014). "Inferring on the Intentions of Others by Hierarchical Bayesian Learning". *PLOS Computational Biology* 10.9, e1003810 (page 28).

Diuk, C., Cohen, A., and Littman, M. L. (2008). "An Object-Oriented Representation for Efficient Reinforcement Learning". In: *Proceedings of the 25th International Conference on Machine Learning*, pp. 240–247 (page 78).

Doncieux, S., Filliat, D., Diaz-Rodriguez, N., Hospedales, T., Duro, R., Coninx, A., Roijers, D. M., Girard, B., Perrin, N., and Sigaud, O. (2018). "Open-ended learning: a conceptual framework based on representational redescription". *Frontiers in neurorobotics* 12 (page 8).

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). "RL$\hat{2}$: Fast Reinforcement Learning via Slow Reinforcement Learning". arXiv: 1611.02779 [cs, stat] (page 92).

Duminy, N. and Duhaut, D. (2016). "Strategic and Interactive Learning of a Hierarchical Set of Tasks by the Poppy Humanoid Robot". In: *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 204–209 (page 76).

Duminy, N., Nguyen, S. M., and Duhaut, D. (2018). "Learning a set of interrelated tasks by using a succession of motor policies for a socially guided intrinsically motivated learner". *Frontiers in neurorobotics* 12 (page 76).

Dutech, A. (2010). "Apprentissage par Renforcement : Au delà des Processus Décisionnels de Markov (Vers la cognition incarnée)". Habilitation à diriger des recherches. Université Nancy II (page 87).

Eppe, M., Magg, S., and Wermter, S. (2018). "Curriculum Goal Masking for Continuous Deep Reinforcement Learning". arXiv: 1809.06146 [cs, stat] (pages 44, 45).

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). "Diversity Is All You Need: Learning Skills without a Reward Function". arXiv: 1802.06070 [cs] (pages 3, 94).

Farley, B. and Clark, W. (1954). "Simulation of Self-Organizing Systems by Digital Computer". *Transactions of the IRE Professional Group on Information Theory* 4.4, pp. 76–84 (page 1).

Feldman, R. (2007). "Parent–Infant Synchrony: Biological Foundations and Developmental Outcomes". *Current Directions in Psychological Science* 16.6, pp. 340–345 (page 14).

Fern, A., Natarajan, S., Judah, K., and Tadepalli, P. (2007). "A Decision-Theoretic Model of Assistance". In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pp. 1879–1884 (page 28).

Finn, C., Abbeel, P., and Levine, S. (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135 (page 92).

Florensa, C., Duan, Y., and Abbeel, P. (2017a). "Stochastic Neural Networks for Hierarchical Reinforcement Learning". arXiv: 1704.03012 [cs] (page 97).

Florensa, C., Held, D., Geng, X., and Abbeel, P. (2017b). "Automatic Goal Generation for Reinforcement Learning Agents". arXiv: 1705.06366 [cs] (pages 39, 43, 45, 47).

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. (2017c). "Reverse Curriculum Generation for Reinforcement Learning". arXiv: 1707.05300 [cs] (pages 43, 45, 47).

Forestier, S., Mollard, Y., and Oudeyer, P.-Y. (2017). "Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning". arXiv: 1708.02190 [cs] (pages 94, 95, 102).

Forestier, S. and Oudeyer, P.-Y. (2016). "Modular Active Curiosity-Driven Discovery of Tool Use". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3965–3972 (page 41).

Fournier, P., Sigaud, O., and Chetouani, M. (2017). "Combining artificial curiosity and tutor guidance for environment exploration". In: *Workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics at IEEE RO-MAN 2017* (page 10).

Fournier, P., Sigaud, O., Chetouani, M., and Oudeyer, P.-Y. (2018). "Accuracy-Based Curriculum Learning in Deep Reinforcement Learning". arXiv: 1806.09614 [cs, stat] (page 11).

Fournier, P., Sigaud, O., Colas, C., and Chetouani, M. (in press). "CLIC: Curriculum Learning and Imitation for Object Control in Non-Rewarding Environments". *IEEE Transactions on Cognitive and Developmental Systems*: *Special Issue on Continual Unsupervised Sensorimotor Learning*, pp. 1–10 (pages 11, 58).

Fujimoto, S., Meger, D., and Precup, D. (2019). "Off-Policy Deep Reinforcement Learning without Exploration". In: *International Conference on Machine Learning*, pp. 2052–2062 (page 88).

Gamrian, S. and Goldberg, Y. (2018). "Transfer Learning for Related Reinforcement Learning Tasks via Image-to-Image Translation". arXiv: 1806.07377 [cs] (pages 3, 37, 78).

Gittins, J. C. (1979). "Bandit processes and dynamic allocation indices". *Journal of the Royal Statistical Society: Series B (Methodological)* 41.2, pp. 148–164 (page 3).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*, pp. 2672–2680 (page 43).

Gordon, G. J. (1996). "Stable Fitted Reinforcement Learning". In: *Advances in Neural Information Processing Systems*, pp. 1052–1058 (page 32).

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). "Automated Curriculum Learning for Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1311–1320 (pages 39, 47).

Gray, J., Breazeal, C., Berlin, M., Brooks, A., and Lieberman, J. (2005). "Action Parsing and Goal Inference Using Self as Simulator". In: *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.* Pp. 202–209 (page 28).

Gregor, K., Rezende, D. J., and Wierstra, D. (2016). "Variational Intrinsic Control". arXiv: 1611.07507 [cs] (pages 94, 97).

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. (2013). "Policy Shaping: Integrating Human Feedback with Reinforcement Learning". In: *Advances in Neural Information Processing Systems*, pp. 2625–2633 (page 6).

Grizou, J., Iturrate, I., Montesano, L., Oudeyer, P.-Y., and Lopes, M. (2014). "Interactive Learning from Unlabeled Instructions". In: pp. 1–8 (page 7).

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. (2018a). "Unsupervised Meta-Learning for Reinforcement Learning". arXiv: 1806.04640 [cs, stat] (pages 92, 93).

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. (2018b). "Meta-Reinforcement Learning of Structured Exploration Strategies". In: *Advances in Neural Information Processing Systems*, pp. 5302–5311 (page 92).

Hasselt, H. van (2010). "Double Q-Learning". In: *Advances in Neural Information Processing Systems*, pp. 2613–2621 (page 67).

Hasselt, H. van, Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double Q-Learning". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094–2100 (page 67).

Hausknecht, M. and Stone, P. (2015). "Deep Reinforcement Learning in Parameterized Action Space". arXiv: 1511.04143 [cs] (page 49).

Hausman, K., Chebotar, Y., Schaal, S., Sukhatme, G., and Lim, J. J. (2017). "Multi-Modal Imitation Learning from Unstructured Demonstrations Using Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*, pp. 1235–1245 (page 77).

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). "Deep Reinforcement Learning That Matters". In: *Thirty-Second AAAI Conference on Artificial Intelligence* (pages 44, 49, 85).

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. (2018). "Multi-Task Deep Reinforcement Learning with PopArt". arXiv: 1809.04474 [cs, stat] (page 3).

Hester, T. and Stone, P. (2012). "Intrinsically Motivated Model Learning for a Developing Curious Agent". In: *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference On*, pp. 1–6 (pages 14, 17).

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. (2018). "Deep q-learning from demonstrations". In: *Thirty-Second AAAI Conference on Artificial Intelligence* (pages 6, 67, 77).

Heyes, C. M. (1993). "Imitation, Culture and Cognition". *Animal Behaviour* 46.5, pp. 999–1010 (page 98).

Houthooft, R., Chen, Y., Isola, P., Stadie, B., Wolski, F., Ho, O. J., and Abbeel, P. (2018). "Evolved Policy Gradients". In: *Advances in Neural Information Processing Systems*, pp. 5400–5409 (page 92).

Huang, C.-T. and Charman, T. (2005). "Gradations of Emulation Learning in Infants' Imitation of Actions on Objects". *Journal of Experimental Child Psychology* 92.3, pp. 276–302 (page 60).

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2003). "Learning Attractor Landscapes for Learning Motor Primitives". In: *Advances in Neural Information Processing Systems*, pp. 1547–1554 (page 96).

Irpan, A. (2018). *Deep Reinforcement Learning Doesn't Work Yet*. URL: https://www.alexirpan.com/2018/02/14/rl-hard.html (pages 2, 24).

Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. (2017). "Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control". arXiv: 1708.04133 [cs] (pages 44, 49, 85).

Jang, E., Devin, C., Vanhoucke, V., and Levine, S. (2018). "Grasp2Vec: Learning Object Representations from Self-Supervised Grasping". arXiv: 1811.06964 [cs] (pages 61, 90).

Jasso, H., Triesch, J., Teuscher, C., and Deák, G. (2006). "A reinforcement learning model explains the development of gaze following". In: (pages 13, 82).

Javdani, S., Srinivasa, S. S., and Bagnell, J. A. (2015). "Shared Autonomy via Hindsight Optimization". arXiv: 1503.07619 [cs] (page 28).

Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. G. (2015). "Self-Paced Curriculum Learning". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence* (page 39).

Jing, J., Cropper, E. C., Hurwitz, I., and Weiss, K. R. (2004). "The Construction of Movement with Behavior-Specific and Behavior-Independent Modules". *Journal of Neuroscience* 24.28, pp. 6315–6325 (page 96).

Jones, S. S. (2009). "The Development of Imitation in Infancy". *Philosophical Transactions of the Royal Society B: Biological Sciences* 364.1528, pp. 2325–2335 (page 60).

Kaelbling, L. P. (1993). "Learning to Achieve Goals". In: *IJCAI*, pp. 1094–1099 (pages 35, 36).

Kang, B., Jie, Z., and Feng, J. (2018). "Policy Optimization with Demonstrations". In: *International Conference on Machine Learning*, pp. 2474–2483 (page 61).

Kaplan, F. and Hafner, V. V. (2006). "The Challenges of Joint Attention". *Interaction Studies* 7.2, pp. 135–169 (pages 7, 13, 31, 32).

Kaplan, F. and Oudeyer, P.-Y. (2007). "In Search of the Neural Circuits of Intrinsic Motivation". *Frontiers in Neuroscience* 1 (pages 40, 54).

Kerzel, M., Mohammadi, H. B., Zamani, M. A., and Wermter, S. (2018). "Accelerating Deep Continuous Reinforcement Learning through Task Simplification". In: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6 (page 53).

Khamassi, M. and Doncieux, S. (2016). "Nouvelles approches en Robotique Cognitive". *Intellectica* 65.1, pp. 7–25 (page 4).

Khetarpal, K. and Precup, D. (2018). "Attend Before You Act: Leveraging Human Visual Attention for Continual Learning". arXiv: 1807.09664 [cs] (page 28).

Kidd, C., Piantadosi, S. T., and Aslin, R. N. (2012). "The Goldilocks Effect: Human Infants Allocate Attention to Visual Sequences That Are Neither Too Simple Nor Too Complex". *PLOS ONE* 7.5 (page 40).

Klopf, A. (1972). "Brain function and adaptive systems-a heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedfored, MA. A summary appears". In: *Proceedings of the International Conference on Systetms, Man, and Cybernetics, 1974* (page 1).

Knox, W. B. and Stone, P. (2009). "Interactively Shaping Agents via Human Reinforcement: The TAMER Framework". In: *Proceedings of the Fifth International Conference on Knowledge Capture*, pp. 9–16 (pages 3, 6).

Knox, W. B. and Stone, P. (2010). "Combining Manual Feedback with Subsequent MDP Reward Signals for Reinforcement Learning". In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, pp. 5–12 (page 6).

Kober, J., Bagnell, J. A., and Peters, J. (2013). "Reinforcement Learning in Robotics: A Survey". *The International Journal of Robotics Research* 32.11, pp. 1238–1274 (page 2).

Kocsis, L. and Szepesvári, C. (2006). "Bandit Based Monte-Carlo Planning". In: *European Conference on Machine Learning*, pp. 282–293 (page 17).

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, pp. 3675–3683 (pages 3, 97).

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). "Building Machines That Learn and Think like People". *Behavioral and Brain Sciences* 40 (page 2).

Laversanne-Finot, A., Pere, A., and Oudeyer, P.-Y. (2018). "Curiosity Driven Exploration of Learned Disentangled Goal Spaces". In: *Conference on Robot Learning*, pp. 487–504 (pages 94, 95).

Laversanne-Finot, A., Péré, A., and Oudeyer, P.-Y. (2019). "Autonomous Goal Exploration Using Learned Goal Spaces for Visuomotor Skill Acquisition in Robots". arXiv: 1906.03967 [cs, stat] (page 95).

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). "End-to-end training of deep visuomotor policies". *The Journal of Machine Learning Research* 17.1, pp. 1334–1373 (page 2).

Levy, A., Konidaris, G., Platt, R., and Saenko, K. (2017). "Learning Multi-Level Hierarchies with Hindsight". arXiv: 1712.00948 [cs] (page 97).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). "Continuous Control with Deep Reinforcement Learning". In: *ICLR2016-4th International Conference on Learning Representations* (page 34).

Lin, L.-J. (1992). "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching". *Machine learning* 8.3-4, pp. 293–321 (page 33).

Lungarella, M. (2007). "Developmental Robotics". *Scholarpedia* 2.8, p. 3104 (pages 3, 4, 40).

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). "Developmental Robotics: A Survey". *Connection science* 15.4, pp. 151–190 (pages 4, 102).

MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Roberts, D., Taylor, M. E., and Littman, M. L. (2017). "Interactive Learning from Policy-Dependent Human Feedback". arXiv: 1701.06049 [cs] (page 3).

MacGlashan, J., Littman, M., Loftin, R., Peng, B., Roberts, D., and Taylor, M. (2014). "Training an Agent to Ground Commands with Reward and Punishment". In: *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence* (page 7).

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2018). "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents". *Journal of Artificial Intelligence Research* 61, pp. 523–562 (page 64).

Machado, M. C. and Bowling, M. (2016). "Learning Purposeful Behaviour in the Absence of Rewards". arXiv: 1605.07700 [cs] (page 94).

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. (2017). "Eigenoption Discovery through the Deep Successor Representation". arXiv: 1710.11089 [cs] (page 94).

Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. (2010). "Toward Off-Policy Learning Control with Function Approximation." In: *Proceedings of the 27th International Conference on Machine Learning*, pp. 719–726 (page 33).

Manheim, D. and Garrabrant, S. (2018). "Categorizing Variants of Goodhart's Law". arXiv: 1803.04585 [cs, q-fin, stat] (page 85).

Mania, H., Guy, A., and Recht, B. (2018). "Simple Random Search of Static Linear Policies Is Competitive for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*, pp. 1805–1814 (page 45).

Mannor, S., Menache, I., Hoze, A., and Klein, U. (2004). "Dynamic Abstraction in Reinforcement Learning via Clustering". In: *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 71– (page 97).

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. (2017). "Teacher-Student Curriculum Learning". arXiv: 1707.00183 [cs] (pages 43, 45, 47).

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological review* 102.3, p. 419 (page 102).

McCorduck, P. (2004). "Machines Who Think" (page 1).

Meltzoff, A. N. and Moore, M. K. (1977). "Imitation of facial and manual gestures by human neonates". *Science* 198.4312, pp. 75–78 (page 102).

Meltzoff, A. N. and Moore, M. K. (2005). *Imitation et Développement Humain: Les Premiers Temps de La Vie*. 44 (page 60).

Minsky, M. L. (1954). *Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain Model Problem* (page 1).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). "Human-Level Control through Deep Reinforcement Learning". *Nature* 518.7540, pp. 529–533 (pages 2, 33, 90, 102).

Moravec, H. (1988). *Mind Children: The Future of Robot and Human Intelligence* (page 4).

Mussa–Ivaldi, F. A. and Bizzi, E. (2000). "Motor Learning through the Combination of Primitives". *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 355.1404, pp. 1755–1769 (page 96).

Nachum, O., Gu, S. S., Lee, H., and Levine, S. (2018). "Data-Efficient Hierarchical Reinforcement Learning". In: *Advances in Neural Information Processing Systems*, pp. 3303–3313 (pages 47, 97).

Nair, A. V., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017). "Overcoming Exploration in Reinforcement Learning with Demonstrations". arXiv: 1709.10089 [cs] (pages 47, 65, 90).

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). "Visual Reinforcement Learning with Imagined Goals". In: *Advances in Neural Information Processing Systems*, pp. 9191–9200 (pages 47, 90, 94).

Najar, A. (2017). "Shaping Robot Behaviour with Unlabeled Human Instructions". PhD Thesis. Paris 6 (pages 5, 7).

Najar, A., Sigaud, O., and Chetouani, M. (2015). "Social-Task Learning for HRI". In: *Social Robotics*, pp. 472–481 (page 7).

Najar, A., Sigaud, O., and Chetouani, M. (2016). "Training a Robot with Evaluative Feedback and Unlabeled Guidance Signals". In: *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 261–266 (page 7).

Nehaniv, C. L. and Dautenhahn, K. (2002). "The Correspondence Problem". *Imitation in animals and artifacts* 41 (pages 77, 88, 99).

Ng, A. Y., Harada, D., and Russell, S. (1999). "Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping". In: *In Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287 (page 52).

Nguyen, S. M., Baranes, A., and Oudeyer, P.-y. (2011). "Bootstrapping intrinsically motivated learning with human demonstrations". In: *in Proceedings of the IEEE International Conference on Development and Learning* (pages 28, 76).

Nguyen, S. M. and Oudeyer, P.-Y. (2014). "Socially Guided Intrinsic Motivation for Robot Learning of Motor Skills". *Autonomous Robots* 36.3, pp. 273–294 (pages 41, 88).

Nichol, A., Achiam, J., and Schulman, J. (2018). "On First-Order Meta-Learning Algorithms". arXiv: 1803.02999 [cs] (page 92).

Ollington, R. B. and Vamplew, P. W. (2005). "Concurrent Q-Learning: Reinforcement Learning for Dynamic Goals and Environments". *International journal of intelligent systems* 20.10, pp. 1037–1052 (page 36).

OpenAI (2018). *OpenAI Five*. https://blog.openai.com/openai-five/ (page 2).

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). "Deep exploration via bootstrapped DQN". In: *Advances in neural information processing systems*, pp. 4026–4034 (page 3).

Oudeyer, P.-Y. (2012). "Active Choice of Teachers, Learning Strategies and Goals for a Socially Guided Intrinsic Motivation Learner". *Paladyn* 3.3, pp. 136–146 (pages 41, 76).

Oudeyer, P.-Y. (2018). "Computational Theories of Curiosity-Driven Learning". arXiv: 1802.10546 [cs] (pages 40, 54).

Oudeyer, P.-Y., Baranes, A., and Kaplan, F. (2013). "Intrinsically Motivated Learning of Real-World Sensorimotor Skills with Developmental Constraints". In: *Intrinsically Motivated Learning in Natural and Artificial Systems*, pp. 303–365 (pages 76, 86).

Oudeyer, P.-Y., Gottlieb, J., and Lopes, M. (2016). "Intrinsic Motivation, Curiosity, and Learning: Theory and Applications in Educational Technologies". In: *Progress in Brain Research*. Vol. 229, pp. 257–284 (pages 40, 41, 54).

Oudeyer, P.-Y. and Kaplan, F. (2009). "What Is Intrinsic Motivation? A Typology of Computational Approaches". *Frontiers in Neurorobotics* 1 (pages 9, 10, 17, 40, 54).

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). "Intrinsic Motivation Systems for Autonomous Mental Development". *IEEE Transactions on Evolutionary Computation* 11.2, pp. 265–286 (pages 40, 41).

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). "Curiosity-Driven Exploration by Self-Supervised Prediction". arXiv: 1705.05363 [cs, stat] (page 17).

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. (2018). "Deepmimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills". *ACM Transactions on Graphics (TOG)* 37.4, p. 143 (page 88).

Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P.-Y. (2018). "Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration". In: *ICLR2018-6th International Conference on Learning Representations* (page 94).

Piaget, J. and Cook, M. (1952). *The Origins of Intelligence in Children*. Vol. 8. 5 (page 31).

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. (2018). "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research". arXiv: 1802.09464 [cs] (pages 3, 47, 48, 61, 87, 90, 94).

Pong, V. H., Dalal, M., Lin, S., Nair, A. V., Bahl, S., and Levine, S. (2019). "Skew-Fit: State-Covering Self-Supervised Reinforcement Learning". arXiv: 1903.03698 [cs, stat] (page 94).

Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., and Riedmiller, M. (2017). "Data-Efficient Deep Reinforcement Learning for Dexterous Manipulation". arXiv: 1704.03073 [cs] (pages 24, 61, 65, 90).

Puterman, M. L. (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming* (page 15).

Rajeswaran, A., Lowrey, K., Todorov, E. V., and Kakade, S. M. (2017). "Towards Generalization and Simplicity in Continuous Control". In: *Advances in Neural Information Processing Systems*, pp. 6550–6561 (page 45).

Rakison, D. H. and Poulin-Dubois, D. (2001). "Developmental origin of the animate–inanimate distinction." *Psychological bulletin* 127.2, p. 209 (page 98).

Reddy, S., Dragan, A. D., and Levine, S. (2019). "SQIL: Imitation Learning via Regularized Behavioral Cloning". arXiv: 1905.11108 [cs, stat] (page 88).

Reddy, S., Levine, S., and Dragan, A. (2018). "Shared Autonomy via Deep Reinforcement Learning". arXiv: 1802.01744 [cs] (page 28).

Riedmiller, M. (2005). "Neural Fitted Q Iteration–First Experiences with a Data Efficient Neural Reinforcement Learning Method". In: *European Conference on Machine Learning*, pp. 317–328 (page 32).

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. (2018). "Learning by Playing - Solving Sparse Reward Tasks from Scratch". arXiv: 1802.10567 [cs, stat] (pages 39, 43, 45, 47, 66).

Ruff, H. A. (1984). "Infants' Manipulative Exploration of Objects: Effects of Age and Object Characteristics." *Developmental Psychology* 20.1, p. 9 (page 59).

Ryan, R. M. and Deci, E. L. (2000). "Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions". *Contemporary educational psychology* 25.1, pp. 54–67 (page 8).

Schaal, S. (1997). "Learning from Demonstration". In: *Advances in Neural Information Processing Systems*, pp. 1040–1046 (page 60).

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). "Universal Value Function Approximators". In: *PMLR*, pp. 1312–1320 (pages 36, 94).

Schmidhuber, J. (1991). "Curious Model-Building Control Systems". In: *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pp. 1458–1463 (pages 3, 40).

Sharma, S., Jha, A. K., Hegde, P. S., and Ravindran, B. (2018). "Learning to Multi-Task by Active Sampling" (pages 43, 45, 47).

Shon, A. P., Verma, D., and Rao, R. P. (2007). "Active Imitation Learning". In: *AAAI*, pp. 756–762 (page 77).

Sigaud, O. and Droniou, A. (2015). "Towards deep developmental learning". *IEEE Transactions on Cognitive and Developmental Systems* 8.2, pp. 99–114 (page 5).

Silva, F. L. D. and Costa, A. H. R. (2018). "Object-Oriented Curriculum Generation for Reinforcement Learning". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1026–1034 (page 78).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. (2016). "Mastering the Game of Go with Deep Neural Networks and Tree Search". *nature* 529.7587, p. 484 (page 2).

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic Policy Gradient Algorithms". In: *International Conference on Machine Learning*, pp. 387–395 (page 34).

Sivamurugan, M. S. and Ravindran, B. (2012). "Instructing a Reinforcement Learner". In: *Twenty-Fifth International FLAIRS Conference* (page 6).

Spelke, E. S. and Kinzler, K. D. (2007). "Core Knowledge". *Developmental science* 10.1, pp. 89–96 (page 59).

Stadie, B. C., Abbeel, P., and Sutskever, I. (2017). "Third-Person Imitation Learning". arXiv: 1703.01703 [cs] (page 77).

Stolle, M. and Precup, D. (2002). "Learning Options in Reinforcement Learning". In: *Abstraction, Reformulation, and Approximation*, pp. 212–223 (page 97).

Stout, A. and Barto, A. G. (2010). "Competence Progress Intrinsic Motivation". In: *2010 IEEE 9th International Conference on Development and Learning*, pp. 257–262 (page 43).

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. (2017). "Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play". arXiv: 1703.05407 [cs] (pages 43, 45).

Sutton, R. S. (1978). "A unified theory of expectation in classical and instrumental conditioning" (page 1).

Sutton, R. S. (1984). "Temporal credit assignment in reinforcement learning". *PhD thesis, University of Massachusetts* (page 2).

Sutton, R. S. (2015). "Introduction to Reinforcement Learning with Function Approximation". In: *Tutorial at the Conference on Neural Information Processing Systems* (page 33).

Sutton, R. S. and Barto, A. G. (1981). "Toward a modern theory of adaptive networks: expectation and prediction." *Psychological review* 88.2, p. 135 (page 1).

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (pages 1–3, 15, 17, 46).

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009). "Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation". In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 993–1000 (page 33).

Sutton, R. S., Mahmood, A. R., and White, M. (2016). "An emphatic approach to the problem of off-policy temporal-difference learning". *The Journal of Machine Learning Research* 17.1, pp. 2603–2631 (page 88).

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). "Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction". In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pp. 761–768 (page 36).

Sutton, R. S., Precup, D., and Singh, S. (1999a). "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning". *Artificial intelligence* 112.1-2, pp. 181–211 (pages 3, 97).

Sutton, R. S., Precup, D., and Singh, S. (1999b). "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning". *Artificial intelligence* 112.1-2, pp. 181–211 (pages 14, 17).

Sutton, R. S., Szepesvári, C., and Maei, H. R. (2008). "A Convergent O (n) Algorithm for off-Policy Temporal-Difference Learning with Linear Function Approximation". *Advances in neural information processing systems* 21.21, pp. 1609–1616 (page 33).

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). "Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning". arXiv: 1611.04717 [cs] (pages 3, 17, 87).

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. (2018). "DeepMind Control Suite". arXiv: 1801.00690 [cs] (pages 65, 90).

Taylor, M. E. and Stone, P. (2009). "Transfer Learning for Reinforcement Learning Domains: A Survey". *Journal of Machine Learning Research* 10 (Jul), pp. 1633–1685 (pages 3, 37).

Taylor, M. E., Suay, H. B., and Chernova, S. (2011). "Integrating reinforcement learning with human demonstrations of varying ability". In: *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 617–624 (page 6).

Tenorio-Gonzalez, A. C., Morales, E. F., and Villaseñor-Pineda, L. (2010). "Dynamic Reward Shaping: Training a Robot by Voice". In: *Ibero-American Conference on Artificial Intelligence*, pp. 483–492 (page 6).

Thomas, V., Pondard, J., Bengio, E., Sarfati, M., Beaudoin, P., Meurs, M.-J., Pineau, J., Precup, D., and Bengio, Y. (2017). "Independently Controllable Features". *arXiv preprint arXiv:1708.01289* (page 95).

Thomaz, A. L. and Breazeal, C. (2006). "Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance". In: *Aaai*. Vol. 6, pp. 1000–1005 (page 6).

Thomaz, A. L. and Breazeal, C. (2007). "Robot Learning via Socially Guided Exploration". *Development and Learning*, pp. 82–87 (page 6).

Thorndike, E. L. (1898). "Animal Intelligence". *Nature* 58.1504, p. 390 (page 1).

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). "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)*, pp. 23–30 (page 93).

Turing, A. M. (1950). "Computing Machinery and Intelligence". *Mind* 59.236, pp. 433–460 (page 4).

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. (2018). "Deep Reinforcement Learning and the Deadly Triad". arXiv: 1812.02648 [cs] (pages 33, 44).

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. (2017). "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards". arXiv: 1707.08817 [cs] (pages 6, 86, 90).

Veeriah, V., Oh, J., and Singh, S. (2018). "Many-Goals Reinforcement Learning". arXiv: 1806.09605 [cs, stat] (pages 44, 45, 47).

Vigorito, C. M. and Barto, A. G. (2010). "Intrinsically Motivated Hierarchical Skill Learning in Structured Environments". *IEEE Transactions on Autonomous Mental Development* 2.2, pp. 132–143 (page 97).

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., et al. (2019). "AlphaStar: Mastering the real-time strategy game StarCraft II". *DeepMind Blog* (page 2).

Vollmer, A.-L., Wrede, B., Rohlfing, K. J., and Oudeyer, P.-Y. (2016). "Pragmatic Frames for Teaching and Learning in Human–Robot Interaction: Review and Challenges". *Frontiers in neurorobotics* 10 (page 6).

Wang, A., Kurutach, T., Liu, K., Abbeel, P., and Tamar, A. (2019a). "Learning Robotic Manipulation through Visual Planning and Acting". arXiv: 1905.04411 [cs] (pages 2, 61).

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). "Learning to Reinforcement Learn". arXiv: 1611.05763 [cs, stat] (page 92).

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019b). "POET: open-ended coevolution of environments and their optimized solutions". In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 142–151 (page 93).

Warde-Farley, D., Van de Wiele, T., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. (2018). "Unsupervised Control Through Non-Parametric Discriminative Rewards". arXiv: 1811.11359 [cs, stat] (pages 94, 97).

Warnell, G., Waytowich, N., Lawhern, V., and Stone, P. (2017). "Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces". arXiv: 1709.10163 [cs] (pages 3, 6).

Watkins, C. J. C. H. and Dayan, P. (1992). "Q-Learning". *Machine learning* 8.3-4, pp. 279–292 (pages 2, 15).

Weinshall, D., Cohen, G., and Amir, D. (2018). "Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks". In: *International Conference on Machine Learning*, pp. 5235–5243 (page 3).

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. (2001). "Autonomous Mental Development by Robots and Animals". *Science* 291.5504, pp. 599–600 (page 4).

Weng, L. (2019). *"Meta Reinforcement Learning"*. URL: http://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html (page 92).

Whitehead, S. D. and Ballard, D. H. (1991). "Learning to Perceive and Act by Trial and Error". *Machine Learning* 7.1, pp. 45–83 (pages 38, 49).

Widrow, B., Gupta, N. K., and Maitra, S. (1973). "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems". *IEEE Transactions on Systems, Man, and Cybernetics* 5, pp. 455–465 (page 15).

Witty, S., Lee, J. K., Tosch, E., Atrey, A., Littman, M., and Jensen, D. (2018). "Measuring and Characterizing Generalization in Deep Reinforcement Learning". arXiv: 1812.02868 [cs, stat] (page 44).

Xu, Z., van Hasselt, H., and Silver, D. (2018). "Meta-Gradient Reinforcement Learning". In: *Advances in Neural Information Processing Systems*, pp. 2396–2407 (page 92).

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. (2018). "A Study on Overfitting in Deep Reinforcement Learning". arXiv: 1804.06893 [cs, stat] (page 44).

Zhang, L. and Zhang, B. (2006). "Hierarchical machine learning–a learning methodology inspired by human intelligence". In: *Proceedings of the First international conference on Rough Sets and Knowledge Technology*, pp. 28–30 (page 96).