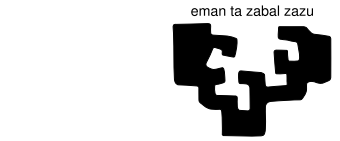
University of the Basque Country
UPV/EHU



Doctoral Thesis

Intrinsic Motivation Mechanisms for a Better Sample Efficiency in Deep Reinforcement Learning applied to Scenarios with Sparse Rewards

Author:
Alain Andres Fernandez

Supervisors:
Dr. Esther Villar-Rodriguez
Prof. Dr. Javier Del Ser

A Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in the

Department of Communications Engineering

June 28, 2023

"*We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run.*"

Roy Amara

"*Most people overestimate what they can achieve in a year and underestimate what they can achieve in ten years*"

Bill Gates

"*The complex line that delimits the short-sighted and long-term decisions for happiness. The γ parameter that governs and rules our lives. The motivations behind each decision. The uncertainty of the environment that surrounds us. There is no "optimal" path to follow; the answer for a worth living life is unique and subjective for each human being.*"

Alain Andres, myself.

UNIVERSITY OF THE BASQUE COUNTRY UPV/EHU

Abstract

Engineering School of Bilbao
Department of Communications Engineering

Doctoral Degree

Intrinsic Motivation Mechanisms for a Better Sample Efficiency in Deep Reinforcement Learning applied to Scenarios with Sparse Rewards

by Alain Andres Fernandez

Driven by the quest to create intelligent systems that can autonomously learn to make optimal decisions, Reinforcement Learning has emerged as a powerful branch of Machine Learning. Reinforcement Learning agents interact with their environment, learning from trial and error, guided by feedback signals shaped in the form of rewards. However, the application of Reinforcement Learning is often hampered by the complexity associated with the design of such rewards. Creating a dense reward function, where the agent receives immediate and frequent feedback from its actions, is often a challenging task. This challenge arises from the difficulty of specifying the *correct* behavior for every possible state-action pair. This issue parallels the challenges faced in human learning where educators often grapple with identifying the best way to teach a certain skill or subject, given that learning styles can vary dramatically among individuals. As a consequence, it is common to formulate the problems with sparse rewards, where the agent is only rewarded when it accomplishes a significant task or achieves the final goal, thus aligning more directly with the objective of the problem. The sparse reward formulation does not require the anticipation of every possible scenario or state, making it more tractable for complex environments and real-world scenarios, where feedback is often delayed and not immediately available.

However, sparse reward settings also introduce their own challenges, most notably, the issue of exploration. In the absence of frequent rewards, an agent can struggle to identify beneficial actions, making learning slow and inefficient. This is where mechanisms such as Intrinsic Motivation come into play, encouraging more effective exploration and improving sample efficiency, despite the sparsity of extrinsic rewards.

In this context, the overall contribution of this Thesis is to delve into how Intrinsic Motivation can boost the performance of Deep Reinforcement Learning approaches in environments with sparse rewards, aiming

to enhance their sample efficiency. To this end, we first stress on its application with concurrent heterogeneous agents, aiming to establish a collaborative framework to make them explore more efficiently and accelerate their learning process. Furthermore, an entire chapter is devoted to analyzing and discussing the impact of certain design choices and parameter settings on the generation of the Intrinsic Motivation bonuses. Last but not least, the Thesis proposes to combine these explorative techniques with Self-Imitation Learning, demonstrating that they can be used jointly towards achieving faster convergence and optimal policies.

All the analyzed scenarios suggest that Intrinsic Motivation can significantly speed up learning, reducing the number of interactions an agent needs to perform, and ultimately, leading to more rapid and efficient problem-solving in complex environments characterized by sparse rewards.

Acknowledgements

It seems like yesterday when I was doing my Master's and began my internship at the Aula Tecnalia in San Mames. Although my research at the time was oriented towards cybersecurity due to its relation to my studies, I had always been curious about the potential of Artificial Intelligence and its possibilities to create solutions that lead us, humans, to a better environment. Unbeknownst to me, I was working alongside a group of high-quality researchers in AI (JRL group)... and one day, I approached them and expressed my interest in their work, not knowing that it would be the first step that propelled me into the world of research.

This journey would not have been possible without Javier Del Ser, a.k.a *el señor mayor* or *deidad del ser*... my professor, director, and supervisor throughout this long and challenging journey. I remember the first time we met during a class, but it wasn't until some time later that I realized how *research-aholic* you were(are) when I discovered you held, not one, but two PhDs! I will always be grateful for the time you took to answer my inquiries and explain what doing research is, introduce me to the entire research group, and encourage me to pursue my PhD despite my fears and not being familiar with the field. You provided guidance when I felt lost and demotivated, offering invaluable tips that have shaped my research career up to this point. Without your support, I definitely would not have embarked on this path.

I am also indebted to Tecnalia, which was initially the research partner of the Bikaintek funding program that granted my PhD. Despite the fact that other employer was involved (with more weight and more interest in what respects to my research), when the latter decided to withdraw from the project, Tecnalia took a step forward, assumed the proportional financial aspects of my grant, and continued to support the project and myself, recognizing its value. I want to thank my superiors at that time: Isidoro Cirion, Iñigo Arizaga, Elena Urrutia and Joseba Laka. However, I must emphasize the critical role played by both my directors during this period when we had no results, papers or indicators guaranteeing the viability of such an investment. We had to shift our focus away from the problem we were addressing and start from scratch again (due to the other partner leaving), which posed a real challenge for us. Even in that circumstance, both of you convinced everyone, solely with your words, to continue trusting in me and put yourselves in a complex position. I am at a loss for words to express my gratitude..

I can not forget the most significant pillar during these years, my other director, Esther. Even at this point, I struggle to find the right words to express myself adequately. I could highlight *different* (numerous, various, multiple...) technical aspects that commend such a brilliant brain which have been crucial for the successful development of this thesis. However, without intending to diminish these professional attributes, I want to use my words to emphasize your *humanity*. We have discussed, argued and conversed about various topics for hours, much like a child does with their mother (I think that is one of the reasons behind some co-workers saying

that you were my figurative mom). You have always listened to me, not only during our work hours but also outside of work, offering your perspective and advice in whatever the problem was. I can not enumerate how many calls we have had, and how thankful I felt to have your support, specially in those situations were I was unmotivated due to several reasons that are not relevant at this moment. I can not forget when you said something like: "It is about the person and its values, not just the work or the results. You should be proud of what you are; any team would undoubtedly be lucky to have you". I have repeated those words to myself and use them as a compass during this journey. This thesis and the person I have become, both professionally and personally, owe a great deal to you. Thank you.

Last but not least, I have to thank my friends, but more importantly, my family – both my parents, Txomin and Marijo, and my sister Goretti – who have always support me unconditionally, not only during these past 4 years, but throughout my entire life. I would not be who I am without you, without your patience, without your efforts, without the values you have instilled in me, and without all the trust you placed in me even when I lost myself. I hope I can return everything I got, and to be, at some point in time, for other people, what you have been for me.

Contents

List of Figures

List of Tables

List of Abbreviations

General	
SOTA	**S**tate **O**f **T**he **A**rt
ANN	**A**rtificial **N**eural **N**etwork
DL	**D**eep **L**earning
SL	**S**upervised **L**earning
UL	**U**nsupervised **L**earning
RL	**R**einforcement **L**earning
DRL	**D**eep **R**einforcement **L**earning
MDP	**M**arkov **D**ecision **P**rocess
POMDP	**P**artially **O**bservable **M**arkov **D**ecision **P**rocess
MARL	**M**ulti-**A**gent RL
CLDE	**C**entralized **L**earning with **D**ecentralized **E**xecution
IM	**I**ntrinsic **M**otivation
IL	**I**mitation **L**earning
self-IL	**Self I**mitation **L**earning (generic)
LfD	**L**earning **f**rom **D**emonstrations
IRL	**I**nverse **R**einforcement **L**earning
PCG	**P**rocedurally **C**ontent **G**enerator
KL	**K**ullback-**L**eibler
SR	**S**uccess **R**ate
LSTM	**L**ong **S**hort-**T**erm **M**emory

Reinforcement Learning	
$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{R}$	Reward space
$\mathcal{P}$	Transition probability function
$\mathcal{G}$	Return
$\mathcal{O}$	Observation function
Ω	Observation space
γ	Discount factor
π	Policy
V	Value function
Q	Action-Value function
TD	Temporal Difference

Algorithmic approaches

EA	**E**volutionary **A**lgorithms
UCB	**U**pper **C**onfidence **B**ound
SARSA	**S**tate-**A**ction-**R**eward-**S**tate**A**ction-
DQN	**D**eep **Q**-**N**etwork
PPO	**P**roximal **P**olicy **O**ptimization
TRPO	**T**rust **R**egion **P**olicy **O**ptimization
GAE	**G**enerealized **A**dvantage **E**stimator
A3C	Asynchronous Advantage Actor-Critic
IMPALA	**Imp**ortance Weighted **A**ctor-**L**earner **A**rchitecture
DPG	**D**eterministic **P**olicy **G**radient
DDPG	**D**eep **D**eterministic **P**olicy **G**radient
TD3	Twin Delayed DDPG
SAC	**S**oft **A**ctor-**C**ritic
NGU	**N**ever **G**ive **U**p
PER	**P**rioritized **E**xperience **R**eplay
ICM	**I**ntrinsic **C**uriosity **M**odule
RND	**R**andom **N**etwork **D**istillation
RIDE	**R**ewarding **I**mpact **D**riven **E**xploration
RAPID	**Ra**nk the E**pi**so**d**es
BeBold	**Be**yond the **Bo**undary of Expl**o**re**d** Regions
MADE	Exploration via **Ma**ximizing **De**viation from Explored Regions
BeBold	**Be**yond the **Bo**undary of Expl**o**re**d** Regions
NovelD	**Novel**ty **D**ifference
FaSo	**Fa**st and **S**l**o**w intrinsic curiosity
AGAC	**A**dversarially **G**uided **A**ctor-**C**ritic
DoWhaM	Don't **Do Wha**t Doesn't **M**atter
D&E	**D**ivide-and-**E**xplore
SIL	**S**elf-**I**mitation **L**earning
DTSIL	**D**iverse **T**rajectory-conditioned **S**elf-**I**mitation **L**earning
UVFA	**U**niversal **V**alue **F**unction **A**pproximator
BC	**B**ehavior **C**loning
DAGGER	Dataset Aggregation

Chapter 1

Introduction

Artificial Intelligence (AI) is one of those topics in everyone's lips in these days. Although multiple definitions can be found in the literature laid out by how a system should think and act taking into account both the rational and human aspects, a wide and more generalist definition was set in (Russell & Norvig, 2022), which characterized AI as:

"*The study of agents that receive percepts from the environment and perform actions*".

AI's popularity has raised with the irruption of Industry 4.0 (and the upcoming and more sustainable Industry 5.0) where it has been considered one of the main Key Enabling Technologies, being in the own words of the European Commission a *game-changer* due to its potential to increase the efficiency and productivity across multiple sectors[1]. More concretely, Machine Learning (ML) has drawn the attention due its potential to make a computer-system learn from examples (*data*) without explicit supervision of a human-being, getting the necessary information by analyzing patterns. By resorting to ML to automate tasks, people can spend time carrying out other duties (*productivity*) and also rely on the solutions provided by systems with better performance that overcome natural human limitations (*efficiency/optimality*), ultimately improving overall people's welfare. Regarding ML, three subgroups can be distinguished:

- **Supervised learning (SL):** learns from labeled data in order to generalize the knowledge to upcoming new inputs.
- **Unsupervised learning (UL):** learns from unlabeled data so that the information can be compressed and accordingly segmented into classes.
- **Reinforcement learning (RL):** learns through the interaction (trial and error) with an environment where the aim is to solve a defined task.

This thesis gravitates around RL and, although its fundamentals are going to be more deeply explained in Chapter 2, it is important to notice the

[1] https://research-and-innovation.ec.europa.eu/knowledge-publications-tools-and-data/publications/all-publications/ai-research-and-innovation-europe-paving-its-own-way_en.

differences with respect to the other two categories, specially between RL and SL, which are similar and often confused with each other. On the one hand, SL assumes the data to be independent and identically distributed (i.i.d) and requires a priori knowledge about the ground truth (also referred to as *true label* or *annotation*) of the training data. Contrarily, in RL previous decisions influence future inputs (i.e., data are not independent, it is a sequential paradigm) whereas the ground truth answer is not known (correct actions/labels are not provided). Instead, the reward is used as an estimator to guide the learning.

Although the RL field has been under study since the 20th century, it did not come to the fore until the last decade due to advances in Deep Learning (DL) and computational capabilities that ease their application. DL involves using non-linear function approximators – typically Artificial Neural Networks (ANN) – so that ML algorithms can ingest unstructured data and automate the feature extraction process. Regarding computational capabilities, the processing units have experienced significant advances in efficiency enabling the deployment of larger and more complex models while exponentially decreasing the time devoted to train them. By the virtue of this progress, RL can leverage ANNs to handle more complicate and diverse problems unapproachable in the past, which gives name to the field where this dissertation is contextualized, Deep Reinforcement Learning (DRL), Figure 1.1.

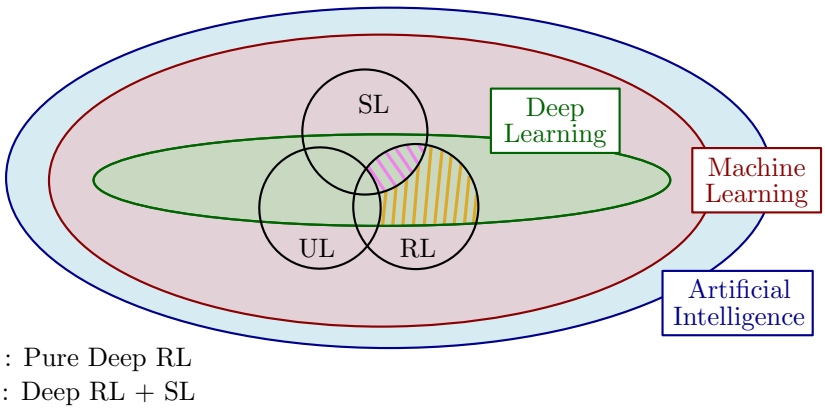


FIGURE 1.1: Artificial Intelligence taxonomy: Supervised Learning (SL), Unsupervised Learning (UL) and Reinforcement Learning (RL). This dissertation is focused on the areas highlighted in orange, Pure DRL, and pink, DRL+SL.

1.1 Motivation

Despite the premises stated above, state-of-the-art (SOTA) ML methods are not mature enough to solve the vast majority of the problems without human presence. Behind the very basic idea of learning from a reward,

RL has to deal with multiple challenges derived from its demanding setup requirements (Dulac-Arnold et al., 2021) (e.g., lack of an available-good simulator, delayed feedback signals, learning from poorly specified reward functions) as well as other difficulties inherent to these techniques(Osband et al., 2020) (e.g., exploration-exploitation dilemma, credit assignment problem, generalization to unseen experiences). However, this has not been an obstacle to begin applying RL to real-world problems when possible(Li, 2019) and see outstanding results in fields like:

- Industry/robotics (supply chain, manufacturing)(Ibarz et al., 2021; Nian et al., 2020)
- Healthcare (treatment recommendation)(Gottesman et al., 2019)
- Energy (power consumption)(Fu et al., 2022)
- Finance (portfolio management)(Filos, 2019)
- Communications and Networking Systems (network access and security, adaptive rate control)(Luong et al., 2019)

Motivated by the exciting journey of RL in those fields, the research-driven interest have been oriented towards narrowing the gap between real-world problem requirements and experimental RL setups, so that more problems become tractable. With all this in mind, multiple high level challenges can be identified (Dulac-Arnold et al., 2021):

- **Sparse rewards**: in RL a feedback signal (reward) is needed to guide the learning so that the agent can distinguish whether the decisions made were actually good/bad. Informative rewards are not necessary right after every single interaction as long as the credit of each action can be deduced. Nevertheless, determining if a decision is better/worse than another, without considering a whole sequence of events, is complex – even when having access to the whole state information and the objective to attain – as there are a large amount of possible sequential combinations that exponentially grow with the extension of the action space and the required number of steps up to the goal, which can lead to very different outcomes. Thus, sparse rewards can be used to evaluate a sequence of decisions. In fact, sparse feedback signals are one of the main challenges present in real-world setups: system delays and difficulties in modeling reward functions in complex problems. However, the more sparse the rewards, the more arduous becomes to determine which actions are useful. Furthermore, the exploration becomes more troublesome. Therefore, sparsity remains as one of the main concerns to be solved in real-world RL problems.
- **Partial observability**: the RL-framework is commonly formalised as a Markov Decision Process (MDP), where a state must contain all the necessary information to make a decision. In practice, this rarely holds true due to the lack of critical information needed in each

time step. Hence, it is common that the agent gets an observation rather than a state, which obviously limits the comprehension of the environment that surrounds it. That context is formally referred to as a Partially Observable Markov Decision Process (POMDP) and exposes difficulties regarding generalization, credit assignment and long-term consequences[2], being a challenge present in large number of real-world scenarios.

- **High dimensional continuous states spaces**: among the different possibilities to model a problem, one of the big issues is how to represent the state (or observation) in such a way that the agent can learn. This implies selecting the type of data and the dimensions to be used as input, where an inappropriate criteria can downgrade dramatically the expected results. This may cause that the agent is unable to model the correlation between the input features, the selected action and their utility. Thank to advances in DL and assuming an agent can understand/infer the world similarly to how humans do, it has become popular to model problems taking into account, for example, images, as input. Therefore, high dimensional inputs are related to generalization issues which are also present in real-world problems.

- **Evolution-Adaptation to action space modifications**: the modification and the consequence adaptation of the agent to either state and/or action spaces can bring new behaviors. Instead of re-training from scratch, the previous knowledge can be reused with techniques like Transfer Learning or by the virtue of using Expert Demonstrations. In such context, how heterogeneous agents should be trained is not clear, as they are supposed to learn different policies. The challenge resides in how to exploit the knowledge gained by other agents.

- **Real-time inference**: in order to deploy any ML-based solution into a production system, the algorithm has to be designed according to the system's capabilities and constraints. While large and complex artificial neural network (ANN) architectures have achieved remarkably good results in various applications, their high computational costs often hinder their adoption in real-world systems. Therefore, striking a balance between performance and costs becomes a practical criterion. Sometimes, achieving high performance can be accomplished by reducing the complexity of the network while introducing complementary, yet lighter, procedures from algorithmic development into an extended ML pipeline.

[2] As the agent only manages to understand the impact of the decisions that modify parts of state that are measurable in its observation, the credit of each action is usually hard to determine (credit assignment). This problem can be minored if such effects can be correlated within a narrow sequence of interactions (long-term consequences), which could ultimately affect the capacity to act in new or similar observations (generalization capacity).

The Thesis aims to develop novel strategies to cope proficiently with all these aspects, which are the facets that most faithfully reproduce realistic scenarios.

1.2 Outline and Contributions of the Thesis

In light of the aforementioned objectives, the core problem to be addressed can be entitled as **sample-efficiency in POMDPs with sparse rewards**, covering exploration-exploitation dilemma in multiple scenarios while attempting to use the minimum samples to get an optimal policy. Therefore, the Thesis is structured in chapters with different use-cases. A brief summary of each chapter is introduced below.

Chapter 2

This chapter – **Background** – aims to introduce and condense all the needed information to understand the technical contributions. Besides the fundamentals of RL and the benchmarks/environments that can be found into the literature, the reasons why sparse reward problems have become popular are highlighted. At the same time, the incoming challenges of adopting such sparse paradigm are explained altogether with the most popular techniques adopted to face the major drawbacks. Along this section a wide review of related research works are presented in order to provide the reader with the fundamental concepts, which are indeed transversal for the following chapters.

Chapter 3

In this chapter – **Collaborative training between hetereogeneously skilled agents in environments with sparse rewards** – we focus on how to carry out a collaborative learning framework between heterogeneous agents with different action spaces yielding different optimal policies. Unlike multi-agent systems, in which agents operate in the same scenario and are typically evaluated based on a team-reward function, we analyze how to learn more efficiently when agents' rewards are independent and each of them interact with distinct instances of the environment. This is also known as the *concurrent learning* paradigm, which lies somewhere between single- and multi- agent problems. Besides the heterogeneity, this chapter also delves into the challenges of **POMPDs**, **sparse rewards** and **high-dimensional state spaces** by learning how to navigate directly from pixels.

Chapter 4

Motivated by the great success and advances of Intrinsic Motivation (IM) techniques, Chapter 4 – **An Evaluation Study of Intrinsic Motivation Techniques applied to Reinforcement Learning over Hard Exploration Environments** – presents an empirical study to assess and

compare the most popular methods for generating intrinsic rewards to gain a better intuition about what actually matters when implementing IM approaches. For that purpose, we establish 3 types of criteria intended to shed light on which scenarios these methods are advantageous. In addition, we conduct our research on procedurally-generated (PCG) environments, which as it is going to be seen in Chapter 2, they impose generalization properties into the learning. These scenarios exhibit previously mentioned challenges in the form of **POMDPs and sparse rewards**. What is more, we analyze the impact of using different ANN architectures regarding **real-time inference challenges**.

Chapter 5

This chapter – **Towards Improving Exploration in Self-Imitation Learning using Intrinsic Motivation** – combines the aforementioned IM ideas with Self-Imitation Learning in order to augment the sample-efficiency in PCG environments. Imitation Learning (IL) has been shown to be effective to tackle hard exploration problems, but requires to have an expert (or at least expert demonstrations) to learn from. Contrarily, Self-Imitation Learning does not demand any expertise and is the agent itself who collects and ranks the experiences in a buffer for a posterior replay. In that context, how to gather good trajectories in first instance is critical. This chapter analyzes how IM can be useful in overcoming that exploration barrier in PCG environments where a good decision (and learning) in an episode may not be transferable to subsequent episodes. To that end, a novel ranking system is adopted in order to replay those samples that enhance the agent's exploration(and ultimately, its performance). Akin to chapter 4, within this chapter the agent partially perceives the environment that surrounds it (**POMPD**) and only gets a non-zero reward when the task is successfully accomplished (**sparse rewards**).

Chapter 6

Last but not least, Chapter 6 – **Concluding Remarks** – summarizes the insights obtained after these years of research. The results in terms of contributions to conferences and journals are listed here too. Moreover, future research lines aligned with the interests of this Thesis are mentioned, which are at the moment being addressed despite the fact that their corresponding publications are out of the scope of this manuscript.

1.3 Reading this Thesis

The contents of this Thesis can be read in a non sequential fashion. Even though the structure follows a logical order, a reader familiar with the background can jump to the experimental chapters (3,4,5) and refer to the background chapter (2) when the related work included in the chapters is insufficient to fully understand the contribution. Having said this, it is strongly recommended to skim the information about environments and

exploration strategies in chapter 2, both cornerstone of main design-related choices along the rest of the chapters.

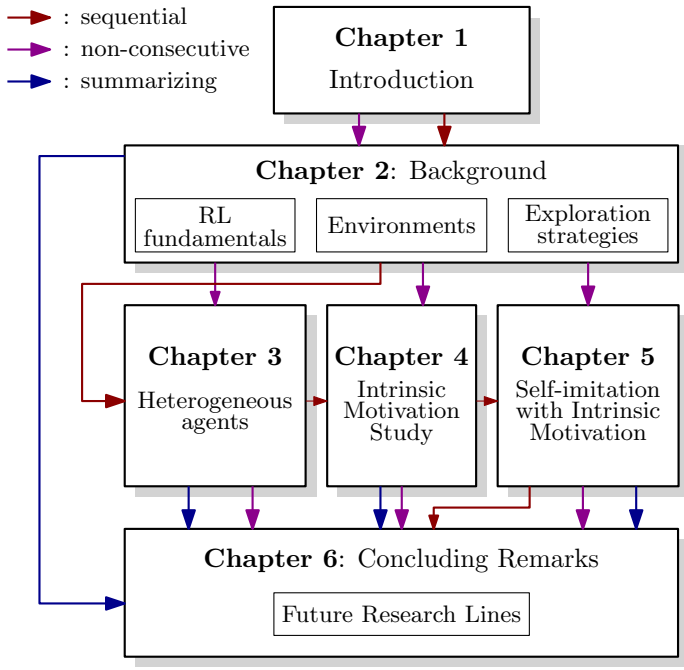


FIGURE 1.2: Block diagram of the structure and reading flow of the Thesis.

Finally, chapter 6 remarks the obtained conclusions in combination with the possible extensions based on the outcomes of each chapter. Besides, future research lines are also elicited. This last chapter summarizes the results in layman's terms, so that no specific technical knowledge is required.

Chapter 2

Background

DRL has been object of research in the last decade (2010-) due to the advances in both DL and RL (Abbeel & Ng, 2004; Goodfellow et al., 2014; He et al., 2015; Hochreiter & Schmidhuber, 1997; Vaswani et al., 2017), which has catapulted the number of problems in which DRL can be adopted. Among the works that have made the field take off, two notable examples are: Bellemare et al., 2013 and Mnih et al., 2015. These studies introduced an approach that successfully used ANNs in an end-to-end framework with RL to play Atari games. The approach achieved comparable superhuman performance levels by using Deep Q-networks (DQN) trained directly from raw input pixels. Right after, policy gradient methods were formulated to embrace deterministic settings (Silver et al., 2014) whereas (Lillicrap et al., 2015) extended DRL to continuous control tasks using an actor-critic framework combining the concepts of both policy gradients(for the actor) and the previously obtained ideas of value based methods (for the critic). In addition to these works, others have had an even larger impact since they developed algorithms capable of beating any human being in classic and highly complex board games, such as chess, shogi, and go (Schrittwieser et al., 2020; Silver et al., 2018; Silver et al., 2017). These algorithms also demonstrated their superiority in e-sports games, such as Starcraft (Vinyals et al., 2019) and Dota II (OpenAI et al., 2019), where they defeated the respective professional champions. More recently, two studies have had an outstanding impact on the RL research community: (Team et al., 2021) and (Reed et al., 2022). These works introduced agents capable of demonstrating good generalization properties across different domains and tasks, even when using the same ANN weights and biases. This represents a significant breakthrough in the field, as it addresses one of the big challenges concerning the application and potential of DRL.

In the previous paragraph the most exceptional works that have paved the way for research in DRL have been mentioned. However, in order to contextualize such advances and this dissertation content, this chapter elaborates on the main and necessary concepts related to the field of DRL, together with exploration related approaches. First of all, RL fundamentals are addressed. Next, some benchmarks/environments that can be found in the literature to evaluate the approaches are highlighted. Finally,

the most popular and adopted exploration techniques related to this Thesis are explained, with special emphasis on Intrinsic Motivation methods.

2.1 Fundamentals of Reinforcement Learning

The main goal of any reinforcement learning algorithm is to select the actions that optimize a given objective. Typically, the problems tackled with RL are modelled via states, actions and rewards in a Markov Decision Process (Section 2.1.1) according to the sequence boundaries in charge of defining how data is processed for training purposes (Section 2.1.2). The design of proper feedback signals to achieve the goals is a crucial factor (Section 2.1.3) in order to learn the functions that govern the decision-making and parameterize the agent's behavior, which subsequently, influences the performance of the agent (Section 2.1.4). The learning process of such functions hinges on the use of the collected data in the training stage (Section 2.1.5). What is more, these functions can play the role of decision-makers or support elements instead, which leads to a broad family of algorithms that can be used to make the *magic happen* (Section 2.1.6). Last but not least, the irruption of DL into RL has had big consequences to extend all these concepts to a wide range of problems that previously were intractable (Section 2.1.7). These concepts are explained in detail below.

2.1.1 Markov Decision Process

Commonly, RL problems are formalized as Markov Decision Processes (MDP), which is used to represent sequential decision making problems in which an agent interacts with a stochastic environment. This framework is subject to the Markov property where the probability of an event occurring depends only on the current and previous event. That is, the future is independent of the past given the present.

$$\mathcal{P}[s_{t+1}|s_t] = \mathcal{P}[s_{t+1}|s_0, s_1, s_2, ..., s_t] \tag{2.1}$$

Formally, a Markov Decision Process is defined by a tuple of 4 elements: $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state-transition probability function and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ stands for the reward function.

Within the RL formulation two main components can be distinguished: the **agent** and the **environment**. The first (agent), also known as the learner or decision maker, is in charge of collecting experiences and optimizing the sequence of decisions (actions) that maximize/minimize the objective, whereas the latter (environment), can be seen as the world with which the agent interacts to build up its knowledge[1].

[1]The terms *agent*,*environment* and *action* are also analogous/can be also interpreted as *controller*, *plant* and *control signal* respectively at engineering control systems.

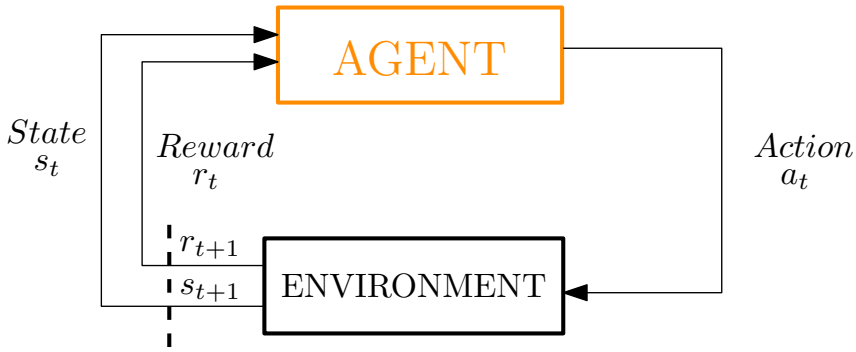


FIGURE 2.1: Reinforcement Learning framework.

The learning process is performed on a trial-and-error basis so that the agent selects an action and the environment provides a feedback signal and a new situation to be faced. More specifically, the agent in each time step, $t = 0, 1, 2, 3, ...$ observes a state $s_t \in \mathcal{S}$ and selects an action, $a_t \in \mathcal{A}$. As a consequence, the environment transitions to a new state $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and the agent receives a reward $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$, that determines how good/bad that decision was (see Figure 2.1). Eventually, all the interactions are summarized in a sequence of states, actions and rewards named *experiences*, which can also be referred as tuples, $< s_t, a_t, r_{t+1}, s_{t+1} >$, containing the information of a complete interaction.

Partially Observable Markov Decision Process

In the case of MDPs, the state of the entire system is always observable, which allows for optimal decisions to be made at each point. The state encompasses all the necessary information to represent the problem at each time step, thus satisfying the Markov Property, Equation (2.1).

Partially Observable Markov Decision Process (POMDP) is a more general framework where instead of states the agent has access only to a restricted part of the environment's information, also known as observations. In this setting, the agent has to make decisions that may cause changes in the whole environment that might remain unnoticed for the agent, which makes the learning process more challenging. Thus, the Markov Property is not fulfilled. Formally, a POMDP updates the previously described MDP to a 6 element tuple: $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega\}$ where Ω represents the observation space and $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow [0, 1]$ the observation function that maps a state and action to a distribution over observations. Consequently, at each time step t the agent only receives an observation $o_t \sim \mathcal{O}(s_t, a_t)$ based on the actual state representation $s_t \in \mathcal{S}$ and the selected action $a_t \in \mathcal{A}$, and conditions its policy on the episodic history of observations.

2.1.2 Sequence Boundaries: Episode & Rollout

The sequence or number of interactions between the agent and the environment can be broken into subsequences which can be referred as *trajectory*, *rollout* and/or *episode*, being their meaning slightly different depending on the boundaries. In this Thesis, we adopt the following taxonomy which is widely used in the literature:

- *Trajectory* is the less restrictive concept and can be used to refer to any of the next two terms.
- An *episode* ends when a maximum number of steps are taken or when the agents achieves the goal (the number of steps required to finish an episode in any of those cases is parameterized by T). As a result, the environment is reset and the agent is brought back to a initial state[2] in order to solve the environment again. Despite the fact that the large majority of problems are of this nature, commonly referred as *episodic tasks*, others are categorized as *continuous tasks* when the goal is never achieved ($T = \infty$) because the task is endless.
- On the contrary, a *rollout* (τ) is not subject to the termination of the episode and is composed by a predetermined number of steps. Consequently, a *rollout* could contain less experiences than an *episode*, or even a multiple amount of them, being the number of such experiences (T) a parameter defined by the user (independently of the environment).

For the sake of clarity, we provide an example in Figure 2.2 where interactions of two different complete episodes can be distinguished. If we considered a rollout of size 15 ($T = 15$), then the rollout would fence the two different episode's information in; on the opposite, if it was set to 5 ($T = 5$), then the rollout will cover less information (e.g., half of an episode in the first example).

Note that an episode's length (number of experiences) depends not only on the environment, but also on the quality of the policy that selects the actions, since an expert agent will be able to accomplish the task with the optimal, i.e. smallest, number of steps[3]. Thus, the defined rollout size (T) ends up containing a variable number of episodes during the training process, which is important in order to balance the bias and variance of the updates generated upon those experiences.

2.1.3 Rewards and Returns

In RL, a reward is a scalar value that an agent receives from the environment after taking an action to guide its learning. The reward indicates how well the agent performed relative to the objective. More importantly,

[2]The agent can be reset either in a fixed starting state (s_0) or within a distribution of possible states (ρ_0). Thus, $s_0 \sim \rho_0$ with a variable number of initial states.

[3]In the interactions of the two episodes presented in Figure 2.2 the optimal trajectories are considered.

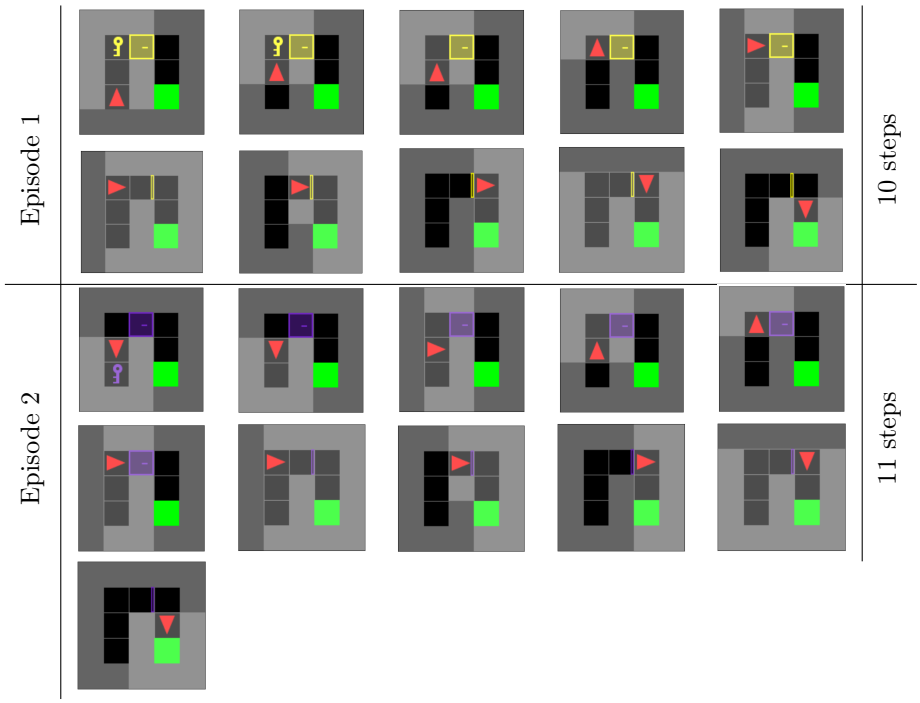


FIGURE 2.2: Example of two different episodes' interactions. The agent is the red arrow and the environment the maze and all the objects that surround it. The state is the visual perception of the environment, the actions are the set of permitted navigation movements and the set of object manipulation operations, and, the reward, is always zero except when arriving to the green square (the goal). The above two rows represent a single episode, while the remaining rows represent a different episode.

the agent's primary goal is to make decisions that maximize the rewards obtained from the environment, which is referred to as the return. Therefore, designing a reward function that provides adequate feedback signals is of utmost importance. In the following, extended definitions of rewards and returns are provide.

Rewards

The reward has to reinforce good decisions and discourage useless or wrong actions in order to make the agent achieve **what** we desire from it. This means that the agent´s success pivots on how well the feedback signals are coherent with the goal of the task. Some conceptualizations of reward functions, and subsequently, the rewards in each interaction, can be exemplified as follows:

Example 1, Robot. *Goal: make a robot run as fast as possible not falling. The reward could be inversely proportional to the required number of steps to arrive to a given destination without falling.*

***Example 2, Chess**. Goal: make an agent learn how to play chess. The intuitively rewards could be +1 for winning, -1 for losing and 0 for drawing.*

In such examples, the agent is guided to complete the task with sparse signals that evaluate the whole sequence of actions that leads to a given outcome. Nevertheless, a reward function's success is also subject to **how** the progress in reaching the objective is evaluated. For instance, sparsity can be circumvented by means of establishing easier subgoals or providing intermediate rewards (i.e., dense) that ease the credit assignment problem:

***Example 1, Robot**. The reward function can be designed to promote the forward motion at each step.*

***Example 2, Chess**. Intermediate rewards can be considered when taking opponent's pieces out.*

Nonetheless, this strategy could mislead the agent into a greedy search of subgoals achievement instead of focusing on the main goal.

***Example 2, Chess**. The agent could find difficulties to beat the opponent becoming greedy into taking the others pieces out rather than developing a winning strategy.*

It is important to remark that, even by designing a good reward function, the success and quality of the results might not be as expected due to other important aspects (e.g., model weights initialization, algorithmic limitations, bias-variance trade-off)[4]. Thus, opting for a naive and easy reward function (over a more complex one) is sometimes suggested.

For these reasons, its design is not trivial and sparse formulations are preferable at the expense of exploration challenges. We will refer later on this Chapter (Section 2.3) to methods to address the exploration-exploitation dilemma more efficiently although this is tangential to the main subject of this dissertation.

Return

Note that the main goal of the agent is to maximize the sum of rewards, which can be formalized with the *return*, G_t:

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T \tag{2.2}$$

where t and T stand for the current and final time steps in an episode, respectively. This calculation gives the same importance to all the decisions regardless of their temporal component. What is more, this formulation complicates the calculation of the return in continuous tasks, when there is no episodic boundaries and the return becomes a sum of infinite series. In light of this limitation, the *discount* concept was introduced by $\gamma \in [0, 1]$,

[4]This can be seen in humans clearly: for the same stimuli, environment, and objective, people require different time to converge to a solution. Moreover, multiple behaviors could lead to what is considered an optimal policy (even for the same reward function).

turning such operation in a finite calculation[5]. This discount factor allows also modulating the importance of immediate and distant rewards. This new return formulation is commonly referred to as *discounted return*:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.3}$$

This implies that a reward to be received after k steps in the future will be worth γ^{k-1} times less than one obtained immediately. Accordingly,

- $\gamma < 1$ is used to adjust the weights of future rewards.
- $\gamma = 0$ is known as "myopic-view" and only maximizes immediate rewards, $G_t = r_{t+1} + 0 \cdot r_{t+2} + 0 \cdot r_{t+3} + ... = r_{t+1}$.
- $\gamma = 1$ corresponds to the formal definition of return without discount, homogenizing the value of future and immediate rewards, $G_t = r_{t+1} + 1 \cdot r_{t+2} + 1 \cdot r_{t+3} + ... = r_{t+1} + r_{t+2} + r_{t+3}...$

In summary, the γ value regulates the effect of maximizing short-term or long-term behaviors, being $0.9 < \gamma < 1$ mostly selected to give credit to future actions and avoid the reward importance vanishing. As a consequence, a fifth (and seventh) element must be attached to the previously introduced MDP (POMDP) tuple: $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ ($\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{O}, \Omega\}$).

2.1.4 Policy and Value Function

Previously, it has been explained how the agent interacts with the environment through actions. A policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, is a function that maps the current state of an agent to an action to be taken, $a \sim \pi(s)$ and it can be either deterministic or stochastic. A deterministic policy maps each state to a single action, whereas a stochastic policy maps each state to a probability distribution over the possible actions that the agent can take.

The value function is a function that estimates the long-term reward that an agent can expect to receive in a given state or state-action pair, under a specific policy π. The **state value function**, $V_\pi(s)$, is responsible for estimating the expected return starting from a state s and following the policy π thereafter, i.e.,

$$V_\pi(s_t) = \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \tag{2.4}$$

where $\mathbb{E}[\cdot]$ denotes expected value. Similarly, the **action value function**, $Q_\pi(s, a)$, estimates the expected return starting from not only a state s,

[5] After a big number of steps, any future reward's effect can be considered insignificant. Furthermore, this only holds true as long as $\gamma \in [0, 1)$ because when $\gamma = 1$ all the rewards are considered equally important.

but also executing an action a, and following the policy π thereafter, i.e.,

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right]. \quad (2.5)$$

Interestingly, one property that applies over value functions is the recursive relationship involving the calculation of returns:

$$\begin{aligned} G_t &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + ...) \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned} \quad (2.6)$$

with the consequent reformulation of Equation (2.4):

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}_\pi[G_t|s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|s_t = s] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1}|s_t = s] \end{aligned} \quad (2.7)$$

being the rewards those that are obtained by following π actions in each of the encountered states from s onwards. Note that both V_π and Q_π are connected through the next equations:

$$V_\pi(s_t) = \mathbb{E}_\pi[Q_\pi(s_t, a_t)|s_t = s, a_t = a \sim \pi(s)] \quad (2.8)$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma V_\pi(s_{t+1})|s_t = s, a_t = a] \quad (2.9)$$

where the key difference lies in the fact that Q_π calculates the expected return assuming that the immediate action will be a_t, determining the next state $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and the associated reward $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$; whereas V_π does not presume any action in its return estimation, being this selection dependent on the current behavior of the policy π.

In addition to these two value estimators, a new function can be considered: **the advantage function,** $A_\pi(s, a)$. This function quantifies how much is a certain action a taken in state s a good or bad decision in relation to the expected value $V_\pi(s)$ in that state, i.e.,

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t|s_t = s, a_t = a) - V_\pi(s_t|s_t = s) \quad (2.10)$$

Last but not least, a policy π is considered to be better than another policy π' if the expected return is greater, that is, $\pi \geq \pi'$ iff (*if and only if*) $V_\pi(s) \geq V_{\pi'}(s)$. In this regard, there is always going to be a policy that is equal or better to the rest of policies, named the optimal policy π^*. Analogously, there will be optimal value functions representing the actual best returns that would be expected from each state s when following the optimal policy π^* thereafter, i.e.,

$$\begin{aligned} V^*(s_t) &= max_\pi V_\pi(s_t|s_t = s) \\ Q^*(s_t, a_t) &= max_\pi Q_\pi(s_t, a_t|s_t = s, a_t = a) \end{aligned} \quad (2.11)$$

2.1.5 On-policy VS Off-policy

In RL, a wide range of algorithms can be found. One of the criteria to opt for one schema is the strategy about how to use the data in the training, commonly categorised as **on-policy** or **off-policy** strategies.

On-policy techniques attempt to improve the policy that is being used to interact with the environment. Because of that, they can only use data that are representative of the current policy, π_t, which precludes the use of any data gathered with a different policy, including any previous policy state $\pi_{t-1}, \pi_{t-2}, ...$ Hence, they are prone to be less sample efficient yet more stable in the learning process. Within this group we can find SARSA(Rummery & Niranjan, 1994), REINFORCE and Trust Region Policy Optimization (TRPO) (Schulman, Levine, et al., 2017), among others.

On the other hand, **off-policy** methods learn a target policy with data generated by a different policy, known as behavior policy. In that case, the learning is said to be carried out from experiences "off" the target policy. Consequently, these algorithms exhibit better sample-efficiency, but are prone to overestimation and instabilities during training time. The most common off-policy algorithms are Q-learning (Watkins & Dayan, 1992) and its extended DL approach, DQN(Mnih et al., 2015); and other approaches that were built on top of DQN like Double DQN (van Hasselt et al., 2015), Dueling DQN(Z. Wang et al., 2016) and C51(Bellemare et al., 2017). Nevertheless, other popular and effective algorithms unrelated to DQN have also been proposed, such as Deterministic Policy Gradients (DPG)(Silver et al., 2014), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), Twin Delayed DDPG (TD3)(Fujimoto et al., 2018) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018).

2.1.6 Value-based VS Policy-based

Regarding the procedure to obtain the policy, RL algorithms can be divided into **value-based** or **policy-based** methods.

The first group, i.e. **value-based** methods, aims to learn a value function that evaluates the utility of each state (i.e., $V_\pi(s)$) and/or state-action pairs (i.e., $Q_\pi(s,a)$). For this purpose, the objective is to minimize the difference between the predicted return of each state ($V_\pi(s_t)$ or $Q_\pi(s_t,a_t)$) and the actual target return (G_t). Note that the actual return calculation is subject to the experiences gathered by the agent (e.g., $\tau = \{s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, ...\}$), which might well not represent the optimal return and will result in the learning of value functions according to these suboptimal target values. More importantly, the trajectories collected for this purpose will be very diverse due to the π's evolution dependence during training. Thus, the target return calculation will exhibit large variance and induce instabilities in the respective estimator function learning. To mitigate the possible variance (and bias)-related issues, any of the following proposed estimators can be adopted:

- **Monte Carlo**. All the rewards from the current state to the terminal state are included, $G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...$ It has no bias but exhibits variance problems.
- **Temporal Difference error (TD-error)**. Only the current reward is considered and then the rest is bootstrapped by using the value of the next state as an estimate of all the rewards to go, $G_t = r_{t+1} + \gamma V(s_{t+1})$. It copes well with the variance problem, but introduces a higher bias.
- ***n*-step**. It is the generalization of the TD-error ($n = 1$) for greater values of n. This means bootstrapping from a specific time step (n) to the terminal state: $G_{t:t+n} = r_{t+1} + \gamma r_{t+2} + .. + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$. The larger the n, the less bias and more variance; the lower the value of n, the higher bias but the less variance.
- **TD(λ)** can be explained as a way to average over the above mentioned n-step updates. Therefore, it requires the calculation of all the n-step returns to, afterwards, assign them more/less weight: $G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$. The TD-error is also known as TD(0) as it equals the case $\lambda = 0$ with just 1-step return.

For the sake of clarity, Figure 2.3 summarizes the strategies of n-step and TD(λ).

Once the value function has been obtained, ***value-based*** *methods distill their knowledge with some defined rules to build a policy.* One approach is to learn an action-value function $Q(s,a)$ that closely approximates, if not exactly, the optimal action-value function $Q^*(s,a)$. Then, the agent can greedily choose the action that maximizes the return in each state:

$$a_t = \arg\max_a Q^*(s_t, a) \tag{2.12}$$

This methodology is known as *greedy* and is used to exploit and evaluate the knowledge. However, using such strategy during the training (prior to obtaining $Q^*(s,a)$) could lead to policies with suboptimal behaviors due to insufficient exploration. This is the reason why other mechanisms that influence in the action selection are adopted (e.g., ϵ-greedy[6]). Here we can find algorithms like Q-learning (Watkins & Dayan, 1992), SARSA (Rummery & Niranjan, 1994) and DQN-family among others (Bellemare et al., 2017; Mnih et al., 2015; van Hasselt et al., 2015; Z. Wang et al., 2016).

On the opposite, **policy-based** *methods parameterize and optimize the policy directly* without the necessity of having a value function. Policies can be learnt by either derivative free methods such as genetic algorithms (Mirjalili, 2019) (recently compared with RL solutions (Martinez et al., 2021)) or policy gradient schemes. In all these methods, the objective is

[6]Refers to a strategy where the agent selects with probability $\epsilon \rightarrow [0,1]$ a random action and with $1-\epsilon$ the greedy action, balancing exploration-exploitation through ϵ parameter.

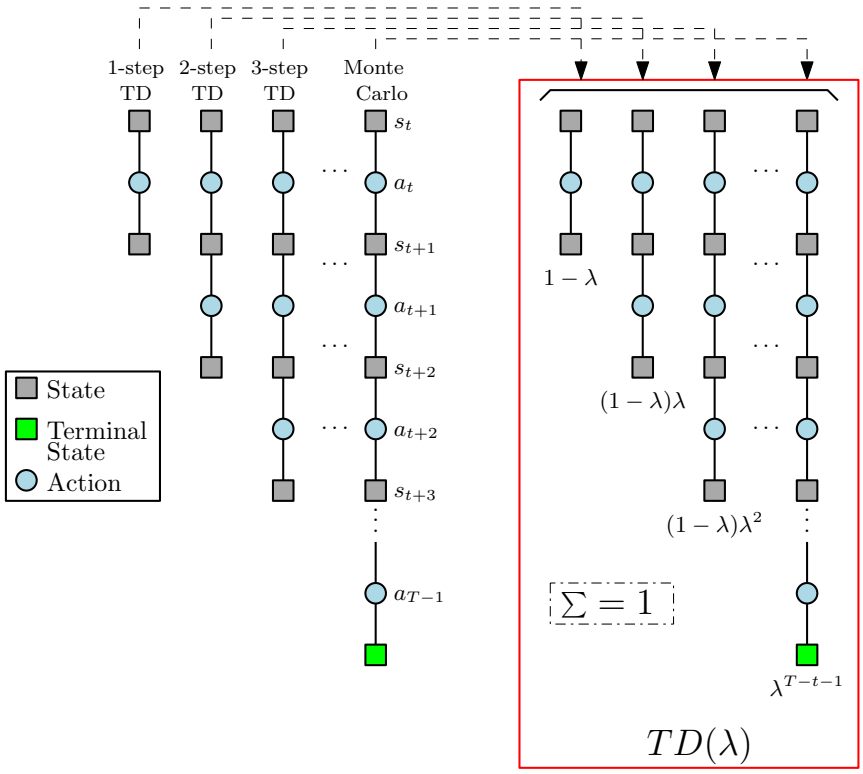


FIGURE 2.3: (Left) Spectrum of possible TD estimators from 1-step up to Monte Carlo (until termination of episode); in between, n-step calculation are placed. The return estimator is calculated with the real n rewards and then the estimated value of the nth next state. (Right) TD(λ) diagram used to weight the n-step returns (when being adopted). $\lambda = 0$ corresponds to just using the 1-step TD, whereas $\lambda = 1$ considers only the Monte Carlo update.

to maximize the performance via a fitness score (used for evaluation) or by maximizing directly the return, $J(\theta) = \mathbb{E}_\pi[G_t]$[7]. Additionally, policy gradient algorithms can handle both discrete and continuous actions spaces. Continuous actions can be more difficult to work with because it is not feasible to explicitly represent every possible action's value, as there are an infinite number of them. As a consequence, they are parameterized by either discretizing the range of possible action values in a discrete number of values, or using statistical distributions (e.g., Gaussian) from which the agent can sample specific values.

Overall, any **value-based** or **policy-based** method can result in deterministic or stochastic policies. Indeed, in **value-based** methods the agent learns the value of each action. Then, it usually selects the action with higher outcome leading to a deterministic policy. However, this can be bypassed by means of methods that perturb the action selection process (e.g., ϵ-greedy strategy) or by parameterizing the output values with

[7] θ is used to refer to the parameters that compose the policy π.

a soft-max function to generate a distribution, resulting in a stochastic policy[8]:

$$\pi(a|s) = \frac{\exp^{(s,a)}}{\sum_k \exp^{(s,k)}} \tag{2.13}$$

being k the total number of possible actions in $\mathcal{A}_k$ where the total sum of probabilities of selecting an action is equal to 1, $\sum_k \pi(a_k|s) = 1$. On the other hand, in **policy-based** methods the agent learns a probability distribution over the actions composing a discrete action space (or a distribution per action in continuous action spaces), and then samples from that distribution to select an action.

2.1.6.1 Policy Gradient methods

Policy gradient methods maximize the expected total reward by estimating the gradient, which can be obtained by differentiating the following objective:

$$L^{PG}(\theta) = \widehat{\mathbb{E}}_t [\psi_t \log \pi_\theta(a_t|s_t)] \tag{2.14}$$

that results in the popular formalization of the gradient as:

$$\widehat{g} = \widehat{\mathbb{E}}_t \left[\sum_{t=0}^{\infty} \psi_t \nabla_\theta log \pi_\theta(a_t|s_t) \right] \tag{2.15}$$

where ψ can be estimated in various ways (Schulman et al., 2015) –see Table 2.1– similar to the estimators previously mentioned for value-based methods.

TABLE 2.1: Different ψ estimators (Schulman et al., 2015) that can be used to compute the gradient in policy gradient methods as exposed in Equation (2.15).

ψ	Description
$\sum_{t=0}^{T} \gamma^t r_{t+1}$	Total reward of the trajectory from the initial state ($s_t|t=0$), Equation (2.3)
$\sum_{t=t_i}^{T} \gamma^t r_{t+1}$	The total reward from a time step (t_i) onward, "reward-to-go", Equation (2.3)
$\sum_{t=t_i}^{T} \gamma^t r_{t+1} - b(s_{t_i})$	A baseline (i.e. an average return over trajectories or a parallel V_π)
$Q_\pi(s_t, a_t)$	State-action value function, Equation (2.5)
$A_\pi(s_t, a_t)$	Advantage function, Equation (2.10)
$r_{t+1} + V_\pi(s_{t+1}) - V_\pi(s_t)$	TD-residual

At this point it is important to highlight that $\widehat{g}$ is calculated based on experiences belonging to a trajectory, whose probability depends not only on the initial state (s_0) and the transition probability function ($\mathcal{P}$), but

[8]Note that by the virtue of generating a distribution, an agent will sample different values even for the same state due to the randomness in the sampling distribution. Nonetheless, the outcome can be set to be deterministic by selecting the action with the highest selection probability (Sutton & Barto, 2018).

also on the current policy (π_t) and the subsequent action probabilities:

$$\begin{aligned} p(\tau|\pi_t) = & p(s_0) \cdot \pi_t(a_0|s_0) \\ & \cdot \mathcal{P}(s_1|s_0, a_0) \cdot \pi_t(a_1|s_1) \\ & \cdot \mathcal{P}(s_2|s_1, a_1) \cdot \pi_t(a_2|s_2) \\ & \dots \\ & \cdot \mathcal{P}(s_T|s_{T-1}, a_{T-1}) \cdot \pi_t(a_T|s_T) \end{aligned} \tag{2.16}$$

Therefore, once the policy is updated ($\pi_t \neq \pi_{t+1}$) the probability of sampling the same τ also changes, which leads to very different experiences, and consequently, to highly variant returns. In fact, some approaches (Espeholt et al., 2018; Horgan et al., 2018; Mnih et al., 2016; Stooke & Abbeel, 2019) use multiple parallel agents to calculate expectations on more diverse batches of experiences that end up stabilizing the variance over the gradient updates:

$$\widehat{g} = \widehat{\mathbb{E}}_t \left[\sum_{\tau \in \mathcal{D}_w} \sum_{t=0}^{\infty} \psi_t \nabla_\theta log \pi_\theta(a_t|s_t) \right] \tag{2.17}$$

being w the number of parallel agents and $\mathcal{D}_w$ the set of all the trajectories collected by all these agents.

The most basic approach is called REINFORCE (Williams, 1992) and resorts to $\psi = \sum_{t=0}^{T} \gamma^t r_{t+1}$ for the policy update. Posterior works, coined REINFORCE with baseline or Vanilla Policy Gradient (VPG), introduced $\psi = \sum_{t=t_i}^{T} \gamma^t r_{t+1} - b(s_{t_i})$, where a baseline $b_t(s_t) \approx V_\pi(s_t)$ was used in order to mitigate high variance gradient updates. Nevertheless, the most adopted ψ since its publication has been the Generalized Advantage Estimation (GAE) (Schulman et al., 2015), being also the one employed in this Thesis.

Generalized Advantage Estimation

Analogously to TD(λ), GAE is defined as an exponentially-weighted estimator of the **advantage function** (instead of the value function in TD(λ)). In that context, the TD-residual of the value-function is defined as $\delta_t^V = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$, which can be considered as an estimate of the advantage when executing an action a_t that provides a reward r_t and a new state s_{t+1}.

Similarly to the n-step target estimator, now we can calculate multiple advantage estimators by taking into account k-steps of the returns minus

the baseline $V(s_t)$ term:

$$\begin{aligned}
\widehat{A}_t^{(1)} &:= \delta_t^V && = -V(s_t) + r_{t+1} + \gamma V(s_{t+1}) \\
\widehat{A}_t^{(2)} &:= \delta_t^V + \gamma\delta_{t+1}^V && = -V(s_t) + r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) \\
\widehat{A}_t^{(3)} &:= \delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V && = -V(s_t) + r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V(s_{t+3}) \\
\widehat{A}_t^{(k)} &:= \delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V + \dots && = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V
\end{aligned}$$

being k the total number of experiences in the selected trajectory (and consequently, the number of advantages estimators to be calculated) and l the index of those advantages.

Once the k advantage estimators are obtained, GAE calculates the exponentially weighted average over all of those $\widehat{A}_t^{(k)}$, i.e.,

$$\begin{aligned}
\widehat{A}_t^{GAE(\gamma,\lambda)} &:= (1-\lambda)(\widehat{A}_t^{(1)} + \lambda\widehat{A}_t^{(2)} + \lambda^2\widehat{A}_t^{(3)} + ...) \\
\widehat{A}_t^{GAE(\gamma,\lambda)} &:= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned} \tag{2.18}$$

Consequently, when choosing $\psi \rightarrow \text{GAE}(\gamma,\lambda)$ the gradient will depend not only on the discount factor but also on $\lambda \rightarrow [0,1]$ (similar to when using TD(λ)). This new parameter is in charge of modulating the desired bias-variance trade-off, where:

- $\lambda = 1$ has high variance due to the sum off all terms being homogeneously weighted, $\widehat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V = \sum_{l=0}^{\infty} \gamma^l r_{t+1+l} - V(s_t)$. Notice that it is almost equivalent to the Monte Carlo return (where you would still have to subtract the baseline term, $V(s_t)$).

- $\lambda = 0$ exhibits low variance at the cost of a higher bias, $\widehat{A}_t := \delta_t^V = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$. It is equal to the TD-residual term, that is, the TD(0) minus the baseline.

In practice, λ tends to be enclosed between $0.9 < \lambda < 1$. For more details please refer to the original paper (Schulman et al., 2015).

Actor-Critic

Actor-critic refers to the methods that learn **both a policy and a value function**. This framework is set when using a decision-maker (*the actor*) that makes use of an additional module to estimate the return (*the critic*), and both functions are optimized jointly[9]. The mechanisms of this framework (illustrated in Figure 2.4) can be explained as follows:

The actor receives feedback from the critic in the form of a numeric value that represents how good/bad the current action was, and uses this

[9]For this reason, previously mentioned VPG can be considered within this framework when the baseline is calculated through a parameterized function (i.e. ANN).

feedback to adjust its decision-making strategy. The critic, on the other hand, learns from the actor's behavior by adjusting the return expectation, improving the agent's performance over time.

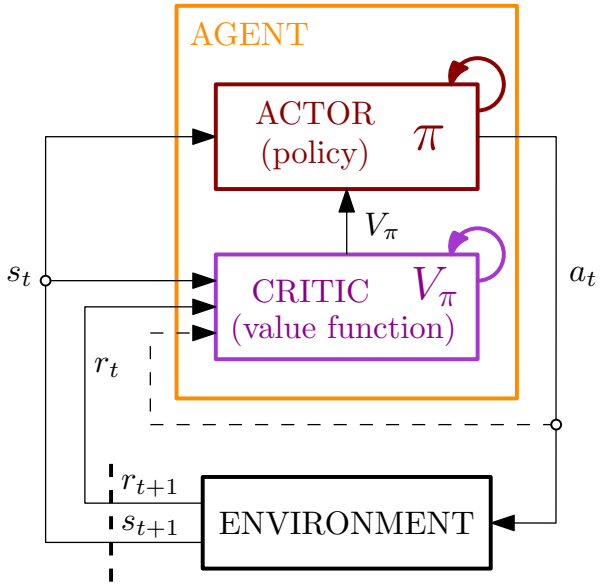


FIGURE 2.4: Actor Critic framework.

This framework can be seen as a combination of value-based and policy-based (i.e., policy gradients) methods, where the strengths of one method can be used to mitigate the weakness of the other. For instance, value-based methods manifest poor convergence properties because minor changes into the value estimate can have big consequences into the distilled policy, whereas policy gradient updates (generally on-policy) are more stable and result in a smoother training. On the contrary, policy gradients suffer from high variance and sample inefficiency while value-based methods exhibit less variance and are more sample-efficient. What is more, policy gradient methods can handle both discrete and continuous action spaces, being the latter not possible for value-based ones.

In summary, actor-critic RL can be seen as a hybrid approach that combines the strengths of value-based (critic) and policy gradient (actor) methods to achieve efficient and effective learning in complex environments. Within this group of methods are Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), Twin Delayed DDPG (TD3)(Fujimoto et al., 2018), Soft Actor-Critic (SAC) (Haarnoja et al., 2018), Advantage actor-critic (A3C and A2C) (Mnih et al., 2016), Importance Weighted Actor-Learner (IMPALA), Proximal Policy Optimization (PPO) (Schulman, Wolski, et al., 2017), etc. Next, PPO is explained in detail as it is the algorithm that is going to be used throughout this dissertation.

Proximal Policy Optimization

PPO is an actor-critic method that collects data on-policy and benefits from some concepts present in TRPO (Schulman, Levine, et al., 2017) yet being even simpler, more general and with a better sample-efficiency. Among its distinctions, PPO performs multiple epochs with the sampled data before discarding them (as opposed to other policy gradient methods – typically on-policy – that generate just one gradient update per collected data). It also uses a new formulation of its objective based on the TRPO's proposal:

$$L^{CPI}(\theta) = \widehat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \psi \right] = \widehat{\mathbb{E}}_t \left[\Gamma_t(\theta)\psi \right] \tag{2.19}$$

where CPI denotes conservative policy iteration, and $\Gamma_t(\theta) = \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ represents the ratio between the new/updated and old policies. Unfortunately, without any constraints, that loss function can cause excessively large updates between two consecutive policy instances (e.g., π_t, π_{t+1}). To fix this, PPO prevents the policy from being very different after each optimization step by characterizing a clipped surrogate function. The penalty that prevents such changes can be expressed in two ways:

- **Adaptive KL Penalty Coefficient**.

$$L^{KLPEN}(\theta) = \widehat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \psi - \beta KL \left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right] \tag{2.20}$$

  Akin to TRPO but, instead of using a fixed β parameter, it employs a dynamic value that depends on $d = \widehat{\mathbb{E}}_t[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]]$, so that:

  - if $d < (d_{targ}/1.5) \rightarrow \beta = \beta/2$
  - if $d > (d_{targ}x1.5) \rightarrow \beta = \beta x2$

- **Clipped Surrogate Objective**.

$$L^{CLIP}(\theta) = \mathbb{E}[\min\{\Gamma_t(\theta)\psi, \text{clip}(\Gamma_t(\theta), 1\text{-}\epsilon, 1\text{+}\epsilon)\psi\}] \tag{2.21}$$

Motivated by the results of the original paper (Schulman, Wolski, et al., 2017), in this Thesis the clipped option (Equation (2.21)) is set as the default PPO method.

In addition, the objective is usually augmented by adding an entropy term ($S[\pi_\theta]$) to ensure sufficient exploration into the policy (Mnih et al., 2016). Regarding the value function, it can be learned either as an independent module or by sharing the parameters with the policy too, which will require to take into account the V_θ's head objective into the update. Overall, the global objective of the algorithm is expressed through the maximization of the next loss function:

$$L^{CLIP+VF+S}(\theta) = \mathbb{E}\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right] \tag{2.22}$$

being c_1 and c_2 components that determine the weight of the value loss and the entropy respectively, $S[\pi_\theta]$ the entropy bonus and L_t^{VF} the squared-error loss for the critic $(V_\theta(s_t) - V_t^{targ})^2$.

Last but not least, PPO usually adopts the previously mentioned GAE as ψ estimator to calculate the advantage estimator for the policy, $L^{CLIP}(\theta)$, and also the return targets for the critic's loss, V_t^{targ}.

2.1.7 Deep Reinforcement Learning

In many tabular RL algorithms[10] (not covered in this dissertation) some assumption as having *"infinite data"* or being able to *"sample every state-action pair infinitely"* are reasonable due to the considered finite state and action spaces. Nevertheless, the **dimensionality** requirements can exponentially grow if the state space is either continuous (e.g., floats) or composed of high dimensional inputs (e.g., images). For instance, when having perception of the environment in the form of an image captured from a camera, the state space can be constituted by $(256^3)^{pixels} = (16777216)^{pixels}$ number of different states, which is intractable in a tabular RL setting. Same situation emerges when having large and/or continuous actions spaces. Consequently, tabular approaches are effective for small or simple environments, where the state and action spaces are small enough to be represented in a table.

In addition to the foregoing, one of the main limitations of tabular RL methods is that they are not well-suited for **generalization** (i.e., the ability of the agent to apply what it has learned in one context to new and/or similar contexts). Because they rely on storing the values for each individual state or state-action pair separately, they can not infer the outcome for similar but slightly different states due to the limitations of the tabular approach itself. The solution? Function approximations.

All the aforementioned policy and value functions can be approximated with a function parameterized by θ, where the goal consist of maximize an objective function. In fact, ANNs have been widely used in SL to automatically extract and preprocess the features of the input and bring the capability of generalizing to unseen data. In DRL the idea is to use ANNs as nonlinear function approximators to scale up the capabilities, capacity and use cases of tabular RL algorithms. One of the first successful works is (Mnih et al., 2015) where they extended Q-learning to be used with ANNs; more concretely, they used Convolutional Layers (O'Shea & Nash, 2015) to process image observations and make an agent play consistently in Atari games. Henceforth, DRL has been widely applied to address multiple problems.

[10] Approaches where the resulting solution can be represented as a look-up table. Each row or entry would correspond to a state, where the associated value for each action at that given state (or the optimal action) will be presented in columns; or vice versa (actions as rows and states as columns). The fact is that the final policy can be obtained by looking into a table of finite dimensions directly (e.g., Q-learning (Watkins & Dayan, 1992) and SARSA (Rummery & Niranjan, 1994)).

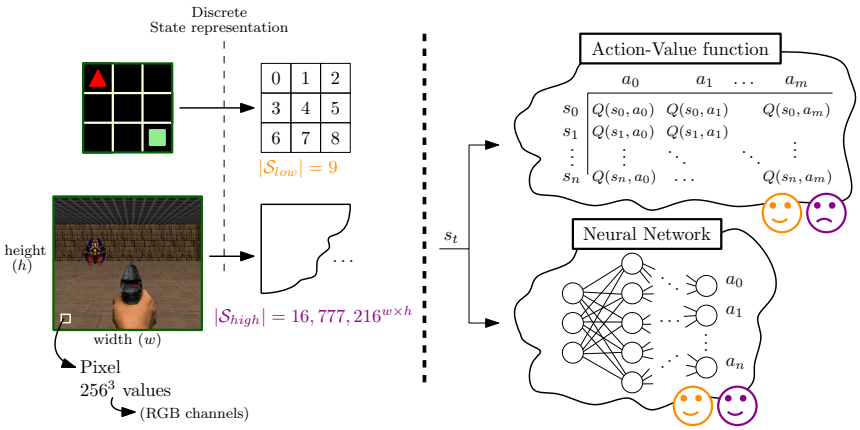


FIGURE 2.5: Tabular versus Deep Reinforcement Learning. (Left) Two problems with different state space dimensions: S_{low} where discretization is straightforward and S_{high} with a huge state space size. (Right) How the different states can be tackled from the perspective of Tabular RL (e.g., with value-based methods) or with Neural Networks; the first approach is suitable for S_{low} problems whereas the second option scales better, a property that makes it preferable when dealing with S_{high} problems.

2.2 Environments

As it has been explained before, a RL agent learns through the interaction with an environment. One of RL's main drawbacks is to build or have access to one environment that models the problem to be addressed; which is not always possible. In the meantime and with the purpose of making progress into the intrinsic challenges of RL methods themselves, multiple environments have been constructed in a *game-style* fashion and used as benchmarks to evaluate new technical approaches. In that context, each benchmark has different scopes:

- **Arcade Learning Environment** (Bellemare et al., 2013) for solving hundreds of Atari 2600 games with multiple challenges.
- **Unity Machine Learning Agents** (Juliani et al., 2020) for enriched virtual game environments.
- **MuJoCo** (Todorov et al., 2012) & **DeepMind Control Suite** (Tunyasuvunakool et al., 2020) for the development of robotics and biomechanics problems that use physics based simulations.
- **VizDooM** (Kempka et al., 2016) & **DeepMind Lab**(Beattie et al., 2016) for 3D navigation problems using first-person viewpoint observations.
- **Meta-world** (Yu et al., 2019) for multi-tasking and meta-RL to learn robotic manipulation tasks.

Although the diversity between environments is actually one of the strengths of those benchmarks, they do not target the generalization requirements inside the same environment, what is critical in real-world problems where the scenario is mutable. In fact, the large majority of benchmarks use singleton environments (the scenario instance does not change through training and/or evaluation) where the agent has been shown prone to overfitting (Nichol et al., 2018; C. Zhang et al., 2018). As a consequence, new benchmarks have been designed to test the generalization to new/unseen instances of the same environment, being the procedurally-generated the ones that have gained momentum.

2.2.1 Procedurally-Generated Environments

As the name suggests, Procedurally Content Generator (PCG) refers to environments (*generators*) designed to *construct* distinct set of instances for the same task in a *procedural* manner. These environments provide access to a large and diverse amount of episodes – also referred as levels – where their generation is subject to specified attributes (or seeds) that govern their characteristics and dynamics, see Figure 2.6.

FIGURE 2.6: Two images from the PCG CoinRun environment. (Up) Multiple levels with different colors and obstacles. (Down) Two different levels with notorious complexity differences. Both images legitimacy correspond to OpenAI and can be found in https://openai.com/blog/procgen-benchmark/ or (Cobbe et al., 2019).

Consequently, the agent is forced to learn relevant skills rather than memorize specific trajectories, promoting the desired **generalization** behavior. Nevertheless, in order to be able to get that knowledge, the agent needs to be provided with a large diversity of levels of the same task, being necessary in the particular case of the Procgen environments (Cobbe, Hesse, et al., 2020) around 500-1000 levels of training to be able to generalize to new/unseen levels. Recently, a bunch of PCG environments have

been released such as Sonic (Nichol et al., 2018), MiniGrid (Chevalier-Boisvert et al., 2018), Obstacle Tower Challenge(Juliani et al., 2019), NetHack (Küttler et al., 2020), Procgen(Cobbe, Hesse, et al., 2020) and XLand (Team et al., 2021) among others. Besides generalization, in the same way as singleton benchmarks, each PCG environment poses its own particular challenges too, such as sparse rewards to analyze the sample-efficiency mentioned in the previous chapter.

Throughout this Thesis some hard-exploration mazes from MiniGrid (Chevalier-Boisvert et al., 2018) are employed, where the agent has a partial egocentric view (POMDP) of the environment and its objective is to reach a given destination, being each level's configuration different despite the task is kept fixed. See some examples in Figure 2.7. The employed tasks in this Thesis are deemed sparse rewards problems because the agent only gets a non-zero reward when accomplishing the goal, i.e.,

$$\mathcal{R}(s_t, a_t, s_{t+1}) = \begin{cases} 1 - 0.9 \cdot \frac{t}{t_{max}}, & \text{if } t < t_{max} \text{ and } s_{t+1} \text{ is terminal} \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

being t_{max} the maximum number of steps per episode in each problem/task. Remark that the probability of achieving the goal by randomness is too small to learn a valid policy with any state-of-the-art (SOTA) RL-algorithm. Further details can be found later in this manuscript when those environments are employed as benchmark.

2.3 Exploration Strategies

When should the agents explore? It is a relevant question still unsolved and apparently highly problem dependant (Pîslar et al., 2022). The exploration-exploitation dilemma becomes fundamental in sparse reward formulations where the probability of getting a valuable feedback from the environment is close to zero in almost all the cases, $p(G_t = r_{t+1} + \gamma r_{t+2} + ... \neq 0) \approx 0$, which leads to a huge amount of uninformative interactions. In this context, acting greedily – exploiting the information that the agent already knows – is synonym of failure or very poor performance. Hence, the exploration becomes essential. Along the literature two main exploration strategies can be listed(Thrun, 1992): *Undirected exploration* and *Directed Exploration*.

The *undirected exploration* strategies focus on injecting randomness into the action selection to promote the discovery of new states without taking into account the information of the environment. Typically, they tend to be simple and have good results in small state spaces and dense reward formulations, albeit struggle and inefficient in the opposite situations. In this category, algorithms random-walks (Anderson, 1986; Nguyen & Widrow, 1989), ϵ-greedy (Sutton, 1995; Watkins & Dayan,

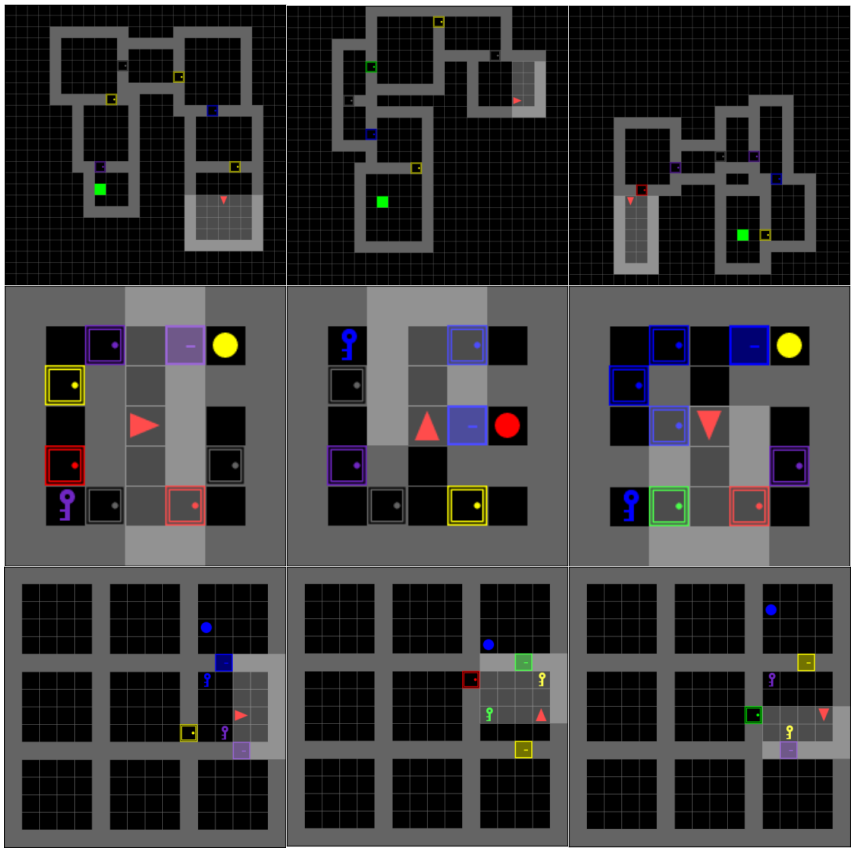


FIGURE 2.7: Rendering of PCG MiniGrid's MultiRoom-N7-S8 ≡ `MN7S8` (top row), KeyCorridor-S3-R3 ≡ `KS3R3` (middle row) and ObstructedMaze-2Dl ≡ `O2Dl` (bottom row) environments across three different levels. Each episode is generated with a different seed so that the configuration of objects and the initial spawn position (and orientation) of the agent are different. As a consequence, a huge number of diverse levels for the same tasks can be generated.

1992; Whitehead & Ballard, 1991) and Boltzmann distribution strategies (Cesa-Bianchi et al., 2017; L.-J. Lin, 1992; Sutton, 1990) are included[11].

Contrarily, *directed exploration* techniques memorize exploration specific knowledge to guide the future agent's behavior. The Upper Confidence Bound (UCB) (Auer et al., 2002) was one of the first approaches to implement this by estimating the expected return along with a measure of

[11] These methods always use some kind of parameter – ϵ (in ϵ-greedy) or τ (Boltzmann) to define the probability/frequency of selecting the greedy action or a random one. Just to clarify, the Boltzmann (or Gibs) distribution can be seen as a soft-max distribution (Equation (2.13)) over the possible $Q(s_t, \cdot)$-values/probabilities given by $\pi(\cdot|s_t)$ where the distribution is subject to a energy/temperature factor,τ: $\exp\frac{Q(s,a)}{\tau} / \sum_k \exp\frac{Q(s,k)}{\tau}$.

the uncertainty for each action:

$$a_t = \arg\max_a \left[Q(s_t, a_t) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right] \tag{2.24}$$

where the first term, $Q(s_t, a_t)$, stands for the expected return, whereas the second term, $\sqrt{\frac{ln(t)}{N_t(a)}}$, specifies the uncertainty of selecting an action (a) considering the number of times (N_t) that action was taken until that time step (t). That is, the first component aims to select the action that leads to the highest return (*exploitation*), whereas the second promotes the selection of actions inversely proportional to the number of times that they have been selected (*exploration*). Such exploration-exploitation trade-off is ultimately controlled by the hyperparameter $c \geq 0$. This idea fostered the proposal of Intrinsic Motivation (IM) methods, recently centered on generating intrinsic rewards to explore and discover new behaviors more efficiently, which is of utmost importance in sparse rewards settings to learn the optimal policy with the minimum amount of agent-environment interactions.

Below some of the most popular IM approaches that are going to be discussed in the following Chapters are detailed. Thereafter, Imitation Learning (IL) is also explained, and further discussed in Chapter 5, as an alternative approach when counting on expert demonstrations.

2.3.1 Intrinsic Motivation

By letting the agent explore the environment for its inherent satisfaction rather than for other exogenous stimuli, new behaviors emerge. In fact, this is related to psychology and how the babies can learn different skills in the early stages of their human life without additional feedback from the world (Grigorescu, 2020; Oudeyer et al., 2016; Ryan & Deci, 2000). IM methods, also referred to as curiosity or novelty, endow the agent with the ability of learning behaviors that are separate from their main task (Aubret et al., 2019) (task-agnostic exploration/behavior). This property becomes particularly interesting in the absence of explicit feedback from the primary task, as the agent is encouraged to learn a secondary goal (*intrinsic-goal*) that will eventually drive it to achieve the main objective (*extrinsic-goal*). This idea is formalized in an intrinsic reward (r_t^i) that is combined with the extrinsic reward provided by the environment (r_t^e) at each time step t through a weighting factor β:

$$r_t = r_t^e + \beta r_t^i. \tag{2.25}$$

In this context, several approaches can be found in the literature to generate the exploration bonuses.

Count-based methods

One mechanism to generate the aforementioned intrinsic rewards is by adopting a visitation count strategy, also known as *count-based* methods. Similar to UCB's exploration component (Equation (2.24)), the rationale is that the agent should be less curious in those states with less novelty. That is, the exploration bonus is inversely proportional to the number of times ($N(s_t)$) a given state (s) has been visited. The most common approach is to define $r^t_{counts} = 1/N(s_t)^{1/2} = 1/\sqrt{N(s_t)}$ (Strehl & Littman, 2008), although other alternatives without the square root (Kolter & Ng, 2009) or other exponential magnitudes to get the desired bonus decay (i.e., how smoothly the magnitude decreases, see Figure 2.8) can also be utilized.

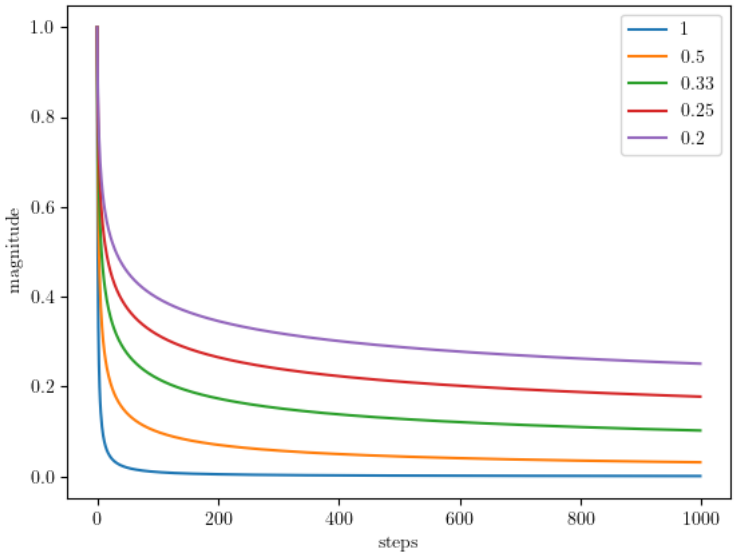


FIGURE 2.8: Visitation count bonus decay for different square values $\frac{\beta}{N(s_t)^{exp\_value}}$ for 1000 consecutive visits. The magnitude parameter is proportional to the selected numerator value, usually weighted with a parameter β. The particular case of $\beta = 1$ is illustrated.

This is a simple, yet effective, solution to quantify the degree to which a state is *unknown* for the agent. However, this is only possible when dealing with discrete state spaces. Contrarily, when having more complex domains with continuous state spaces other solutions are needed. One option is to discretize it by creating tiles/bins to embed multiple values at once. Other alternatives have been fruitfully: density models to measure the uncertainty and henceforth compute the bonus (Bellemare et al., 2016; Ostrovski et al., 2017), hashes to encode the states in a discrete manner

(Tang et al., 2017) or successor representations to leverage similarities for the exploration bonus generation (Machado et al., 2019).

Prediction-error methods

On the other hand, the intrinsic reward can be computed as the *prediction-error* when predicting the consequence of an agent's action in the environment; that is, measuring the predictability of the changes in the environment. The intuition in these methods is clear: the better the prediction, the more often might that situation has been encountered and the lower the novelty bonus should be.

Intrinsic Curiosity Module (ICM) (Pathak et al., 2017) was a game changer and distinct itself from other previous prediction approaches (Houthooft et al., 2017; Stadie et al., 2015) because it focuses on a smaller feature space to compute the expected changes that affect the prediction. Such a feature space is built to model the transitions between consecutive steps that were controlled by the agent or that directly affect it; while ignoring the rest. This was accomplished by using an inverse dynamics model in a self-supervised manner to predict the agent's action ($\widehat{a}_t$) given the current ($\phi(s_t)$) and next state ($\phi(s_{t+1})$) embeddings, so that only things affecting to the agent were modeled to obtain the desired feature space. At the same time, that embedding space ($\phi(s_t)$) altogether with the actual action (a_t) is used to train a forward dynamic model (Stadie et al., 2015) that predicts the feature representation in the next state ($\widehat{\phi}(s_{t+1})$), which in last instance is compared against the latent representation of the next state in the previously modeled feature space ($\phi(s_{t+1})$) to compute the intrinsic reward (r_t^i), see Figure 2.9.

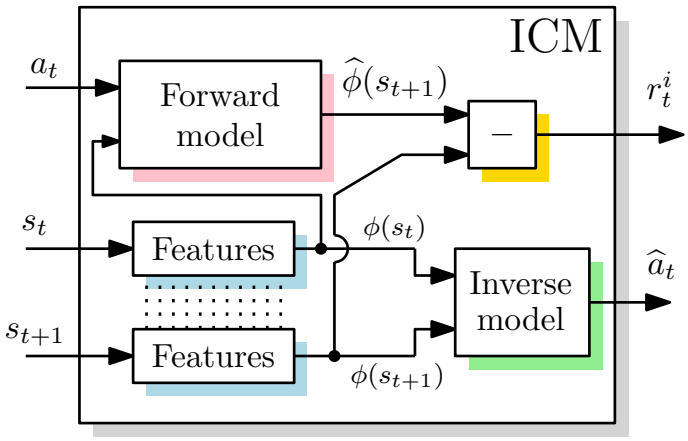


Figure 2.9: Intrinsic Curiosity Module (ICM)(Pathak et al., 2017), where the generation of the intrinsic reward r_t^i is illustrated. The intrinsic reward is computed as the prediction error in the feature space of the next state, that is, the difference between $\widehat{\phi}(s_{t+1})$ and $\phi(s_{t+1})$ given s_t, s_{t+1} and a_t.

Later on, (Burda, Edwards, Pathak, et al., 2018) conducted a large-scale study based on these prediction errors over 54 environments without any extrinsic reward –purely guided by intrinsic behaviors– in which they analyzed the efficacy of using various feature learning methods. In other words, they investigated the effect of using different feature spaces –$\phi(\cdot)$– such as relying on pixels, random features, variational audoencoders (Kingma & Welling, 2014) and the previously introduced inverse model (Pathak et al., 2017). One important remark is that they brought up the *noisy-TV* problem on this kind of algorithms: the agents tend to be attracted by stochastic dynamics of the environment which was clearly exemplified by introducing a TV into the environment that changed the channels randomly independently of the agent's actions. In order to solve this issue, (Pathak et al., 2019) proposed the use of an ensemble of forward dynamics models so that the reward was computed taking into account the variance with respect to their next state prediction; hence, they are not sensitive to agent's impact on the environment changes but to the parts of the environment that have been largely/shortly explored (the more a state has been visited, the less the disagreement between the outcome of all the forward models and the less variance even in a stochastic situation). Another idea is to use an episodic memory so that the distance/proximity –referred to as reachability in the paper– of past instances in reference to the current state can be measured (Savinov et al., 2019); in other words, how many steps away is the agent from experiencing those situations again. The episodic novelty module idea was extended and combined with a *life-long novelty module* so that curiosity across the episode and the whole training was modulated yielding new SOTA results in some benchmarks (Badia, Sprechmann, et al., 2020).

Special mention deserves Random Network Distillation (RND) (Burda, Edwards, Storkey, et al., 2018), which became popular due to its simplicity and good performance. Thi is the reason why it was picked over other *prediction-error* methods for this Thesis. In this strategy, two neural networks are required: a *target* $\phi(\cdot)$, and a *predictor* $\widehat{\phi}(\cdot)$. Both of them are initialized randomly and the *target*'s parameters are frozen thereafter. The *predictor*'s goal is to mimic the *target* network's output, so that the outcomes are as close as possible. Therefore, the intrinsic reward measures the closeness through: $||\widehat{\phi}(s_{t+1}) - \phi(s_{t+1})||$. As the *predictor* keeps learning to imitate the *target*, the intrinsic reward is supposed to be smaller and smaller as a reflection of the number of cumulative state visits, so that the curiosity concept about exploring novel states is satisfied. The authors identify three main factors to be relevant source of prediction errors:

- Factor 1. Prediction error is high when the predictor fails to generalize from previously seen data.
- Factor 2. Prediction error is high because the target is stochastic.
- Factor 3. Prediction error is high because necessary information for the predictor is not given (or the model capacity is too limited to accurately predict the target).

The last 2 factors can induce the aforementioned *noisy-TV* problem. Hence, RND was designed to overcome those undesired properties by fixing the prediction problem with a deterministic target and having two replicates of the same ANN architecture, so that the prediction error is not limited by the model capacity or architecture.

Last but not least, it is important to emphasize that when using intrinsic rewards the problem becomes bi-objective and the agent is accordingly going to optimize both goals[12]. Nevertheless, unexpected behaviors can arise in these settings due to an excessive exploration that hinders the exploitation of the main task (Badia, Sprechmann, et al., 2020; Rosser & Abed, 2021; Taïga et al., 2020). Most of the approaches neither control nor balance the importance of the extrinsic and intrinsic components during training. This is based on the following assumptions:

- The scale of both rewards is very different: very low intrinsic values in comparison to the extrinsic ones. As a result, possible goal-deviation occurs mainly in the absence of extrinsic rewards.
- Intrinsic rewards are non-stationary in nature. Their magnitude, regardless of β, decreases on average throughout the training as the state space is explored, resulting in an even larger difference between the two types of rewards/goals.

However, these assumptions sometimes are not enough and other solutions are required. Among those examples, there are meta-learning approaches where the functions that parameterize the intrinsic rewards are influenced by the direction of the extrinsic gradient(Dai et al., 2022; Du et al., 2019; Z. Zheng et al., 2018) (ensuring that the main extrinsic objective is aligned with the exploration component too), while other frameworks propose to directly decouple the two goals into different agents (E. Z. Liu et al., 2021; Schäfer et al., 2022).

2.3.2 Imitation Learning

Another solution to overcome exploration problems is the use of expert demonstrations, which is also known as *Imitation Learning (IL)* and/or *Learning from Demonstrations (LfD)(Hester et al., 2017; Vecerik et al., 2018)*. Within this framework, good (optimal or suboptimal) trajectories are assumed to be provided, $\tau^* = \{(s_0, a_0, r_0, s_1), (s_1, a_1, r_1, s_2), ...\}$, so that the agent can use those tuples to pre-train or even master a policy in an online fashion that prevents the agent from getting stuck in the early phases of the training (where no expertise is still developed). Nevertheless, key aspects such as different embodiment and observability between the expert and the learner make challenging its success application (Osa et al., 2018). Depending on how the demonstrations are used to distill the knowledge, two ways of learning can be found: *Behaviour Cloning (BC)* and *Inverse Reinforcement Learning (IRL)*.

[12] Recall that the agent maximizes the return (Equation (2.3)) in which the considered reward has now a new explorative component (Equation (2.25)).

On the one hand BC (Bain & Sammut, 2001; Pomerleau, 1988; Torabi et al., 2018) seeks to learn a policy through a mapping strategy where a given input is associated to an action; this is, it just requires state-action tuples, $\tau^* = \{(s_0, a_0), (s_1, a_1,), ...\}$. Standard supervised learning methods such as the log loss function (which can be embedded within a Cross Entropy loss (Gneiting & Raftery, 2007)) are used to map the probability of selecting an action to the specified input, which augments its future probability preference:

$$L_{BC} = -\frac{1}{|D|} \sum_{(s,a)\in D} \ln(\pi(a|s)) \tag{2.26}$$

where D refers to a pool of data where the demonstrations are contained and from the tuples are sampled. Nevertheless, these approaches suffer from compounding errors (Ross & Bagnell, 2010) derived from the fact that the policy to be updated exhibits different probabilities of collecting experiences with the assumed expert policy that provides samples. This is, a distribution shift exists in the sampling probability of trajectories (recall Equation (2.16)) between the policy that gathered the demonstrations and the policy that is being learned. Consequently, the future test data are influenced by the policy that is being learned, breaking the main assumption of most SL methods that assume the data to be independent and identically distributed (recall Chapter 1 when we explained the differences between RL and SL). Therefore, one of the most popular BC algorithms up to date – Dataset Aggregation (DAGGER) (Ross et al., 2011)– proposed to aggregate additional online data to the dataset used for training ($\mathcal{D}$), with the particularity that the visited states are subject to the learned policy distributions ($\pi(a_t|s_t) \rightarrow s_{t+1}$) but the stored action in each state is the expert's ($a^*_{t+1} \sim \pi^*(s_{t+1})$), so that $D \cup \{s_{t+1}, a^*_{t+1}\}$.

Alternatively, IRL(Finn et al., 2016) aims to learn the hidden reward function from the provided experiences under the assumption of being optimal (or very close to optimal) demonstrations. To do so, it uses that function to obtain rewards from which the agent's policy is learned, $\tau_0, \tau_1, ... \rightarrow \mathcal{R}_h \approx \mathcal{R}; \widehat{r}_t \sim \mathcal{R}_h(s, a)$[13]. These methods are highly sensitive to how good the reward function represents the desired (optimal) behavior. Within this taxonomy, adversarial IL methods can be taken into account too (Ho & Ermon, 2016; Ho et al., 2016), where the policy parameterizes a generative model that "creates" new experiences and the cost function (i.e., reward function) serves as an adversary.

In summary, the selection of one or another approach will depend on whether the BC's learned policy represents a valid mapping from states to actions or if IRL's distilled reward function is valid to learn a suitable policy for the desired behavior. Furthermore, the criteria is also subject to the availability of a *model* that makes possible the use of dynamics information of the environment (Osa et al., 2018).

[13]For simplicity, the calculated reward function is shown to be dependant on the state and action, although it can also be subject to the next state.

Chapter 3

Collaborative training between Heterogeneously skilled Agents in Environments with Sparse Rewards

Designing a reward function is one of the most challenging steps when formulating a problem that is meant to be solved with RL. As we have previously highlighted in Section 1.1, one way to overcome this cumbersome design is by using a single (sparse) reward signal that determines whether a RL task has been solved. In this context, the problem becomes more complex due to the lack of dense feedback signals that guide the learning process, ultimately hindering the correlation between successfully solving the task and the successive actions that lead to that outcome. To address this issue, a solution is to generate an exploration bonus (*intrinsic reward*) that promotes the novelty (*motivate the agent*) within the environment. This approach encourages diverse behaviors and enables the discovery of valid solutions through *exploration*, thereby fostering goal achievement. The family of algorithms that can generate these bonuses are known as Intrinsic Motivation (IM) techniques, which have been introduced previously in Section 2.3.1 of Chapter 2. Their utility can be better understood from the intuition gained from the following real-world example:

A bike rider wants to descend a given mountain across the shortest path and as fast as possible. However, the rider does not know the mountain, and the unique feedback signal will be received at the end of the route. Thus, the rider does not know whether the decision in a bifurcation is right, if they get stacked close to the final line, or even if they spend too much time when compared to other bikers. Due to so much uncertainty without feedback signals, the agent (bike rider) should drive their decisions based on their own motivation and curiosity.

A first question arises when examining this real-world example: what

happens when considering more than one agent? Multi-agent problems have been broadly addressed, assuming that the agents interact with the same environment[1]. Following the above example:

Instead of one bike rider, we consider two riders going downhill in the same mountain at the same time. Both of their actions could change the state of the mountain. In addition, knowledge gathered by any of them could improve the other rider's learning if shared (when to slow down, how to take a curve, which path to follow, etc).

Furthermore, agents can also be deployed at different instances of the same environment. For instance, this context is usually framed as a concurrent learning paradigm (Dimakopoulou et al., 2018). The example can be expanded accordingly to yield:

Both riders will learn independently from each other over the same mountain and paths (environment) and with the same opportunities, but they do not run during the same hours (namely, actions taken by one do not interfere into the decisions of the other)..

In this regard, research has been focused on improving the exploratory effort and minimize the required time by increasing the number of parallel (homogeneous) agents, and using posterior sampling (Silver et al., 2013), seed sampling (Dimakopoulou & Roy, 2018) and also sophisticated distributed approaches (Espeholt et al., 2018; Kapturowski et al., 2019; Mnih et al., 2016). In those cases the goal is to draw a larger amount of samples by deploying several experience collectors in parallel. However, another case emerges when the agents are heterogeneous in either their observation or their action space, which could make them have different optimal solutions and, in consequence, different action distributions:

Now both riders are assumed to have different bikes. One of such bikes allows making jumps over obstacles such as tree branches or rocks, whereas the other bike can not perform any jump. Consequently, some tracks of the mountain are only accessible for one agent. The optimal solution (the shortest path) is reachable only through a track blocked by obstacles.

As illustrated in the example, the problem can not be strictly classified as *single-agent* or *multi-agent* problem. In the targeted case, multiple agents with different skills interact with their own copy of the environment. Due to the agents having different action spaces, they can access specific subsets of the state space that may yield different optimal solutions, ultimately resulting in a diversification of the optimal solution space. Such problems are also present in real-life scenarios, e.g.:

- Exploration missions with robots of different skills (e.g., wheels versus legs introducing an increase in overall height).
- Robotic arms with different numbers of joints aiming to learn a task.

[1] We refer as multi-agent problems to those in which the involved agents interact between them within the same environment instance in a cooperative or competitive manner, so that the state transitions are the result of the joint action of all agents (Buşoniu et al., 2010).

- Autonomous vehicles with different steering wheel angles and/or reaction times.

These variations can cause the agents to perceive the environment differently, even if the overall state space is the same, resulting in the agents potentially converging to different optimal policies.

In this chapter we analyze this kind of problem, which cannot be addressed strictly from either the *single-agent* perspective (completely independent fashion) or the *multi-agent* perspective (under a fully collaborative framework). Instead, we aim to explore a hybrid strategy that is capable of deciding *what to share* and *when to do it*. We approach the problem from the perspective of a collaborative learning framework (Gokhale, 1995; Johnson & Others, 1994)[2]. Our goal is to study which modules can be shared to obtain a more efficient exploration strategy compared to sharing no information whatsoever. Accompanied by theoretical insights, the empirical results provide evidence that the collaborative strategy does not always yield benefits for the agent's learning process. However, if appropriately chosen, it may lead to performance improvements.

3.1 Related Work

The success of IM techniques has catalyzed their adoption in a wide range of RL problems (Charoenpitaks & Limpiyakorn, 2019; Taïga et al., 2020) beyond the single-agent domain. Indeed, these techniques have also been adopted in multi-agent RL (MARL) domains, which is an even more challenging setup as the actions executed by an agent could influence the observation/state space of other agents. In fact, *influence* was introduced in (Jaques et al., 2019) by proposing the generation of a causal influence reward through the KL divergence between policies influenced and not influenced by the actions selected by other agents. This causal influence is used to increase the reward signal obtained from the environment. Likewise, the influence can also be calculated as the impact on other agents' transition function and rewarding structure (T. Wang et al., 2019). In addition, the agents can be stimulated to take (jointly) actions whose effects are difficult to model through a composition of the predicted effects of each agent's actions taken independently (Chitnis et al., 2020) (i.e., by encouraging the selection of actions that have a greater impact when taken jointly rather than asynchronously within the environment state).

On the contrary, little research has been done on the application of curiosity in MARL assuming that the scenario does not change due to the agents' decisions (without measuring the influence). In this context, (Iqbal & Sha, 2019) analyzed the effect of using decentralized curiosity modules and then combining them with different methods. In that work an ablation

[2]For the sake of clarity, here we refer as Collaborative Learning to the challenge of being able to reuse, combine or even adapt knowledge transferred from one source to another.

study was conducted for different tasks and scenarios, and proposed a non-linear method that combines multiple novel ideas. In (Böhmer et al., 2019) an intrinsically rewarded centralized agent was proposed, which is linked to other decentralized agents through a shared replay buffer responsible for collecting training experiences. With that setup, although decentralized agents are not rewarded with intrinsic bonuses directly, they improve their exploration. Moreover, it was shown that if the decentralized agents were encouraged with IM rewards directly, their results got worse. Ultimately, the impact of individual and joint curiosity was examined in (Schafer, 2019), concluding that the latter has more stability with similar results.

All the prior examined works considered single and multiple agents (Espeholt et al., 2018; Kapturowski et al., 2019; Mnih et al., 2016) of the same nature. Nevertheless, very scarce works have considered heterogeneous agents. Oddly enough, the term *heterogeneous* can lead to confusion as it has been used in the literature with different meanings:

- In (H. Zheng et al., 2020), *heterogeneous* is used to refer to agents that were trained by means of different optimization algorithms (namely, TD3, SAC, EA), where the idea is to explore more efficiently by overcoming possible issues of each algorithmic solution that may yield a degradation of the learning convergence (e.g. the agent getting stuck in local optima).

- In (Kurek & Jaśkowski, 2016), it is used to differentiate between the type of team learning used to accomplish the task (i.e., learn independent policies for each agent).

- In (Wakilpoor et al., 2020), it denotes agents with different sensors that perceive the environment (i.e., observations) differently. Heterogeneity understood as such is addressed by using a new state encoding structure.

- In (Calvo & Dusparic, 2018), it implies agents with different capabilities in their action domains, which is a natural way to model junctions (or traffic lights) that lead to a variable number of possible paths. Similarly, (Zolna et al., 2019) used the *heterogeneous* term to refer to agents with modified actions, which potentially lead to different optimal solutions.

A potential solution to effectively realize information sharing (e.g., novelty) between agents is the adoption of knowledge distillation (Parisotto et al., 2016; Rusu et al., 2016), transfer (Rusu et al., 2022), and collaborative learning techniques (K. Lin et al., 2017). These methods permit the export of the knowledge of an already trained agent to a new one, so that the training can be accomplished more efficiently through knowledge re-usability. To this end, it is mandatory to have a connection between the two learning activities (Zhuang et al., 2020), where the success will strongly depend on selecting *what*, *when*, and *how* to manage the knowledge from a source to a target. The transferred knowledge does not always have the

desired impact, and may lead to a negative transfer, and subsequently, results in performance degradation (Z. Wang et al., 2019). More importantly, knowledge is governed under a *teacher-student* framework, wherein the teacher (*expert*) is an agent that has previously gained knowledge and gives instructions to the student (Da Silva et al., 2019). Thus, it is also required to have such a teacher, which can be an unrealistic assumption in real-world scenarios.

One way to circumvent the need for expert support is online knowledge transfer, by which various agents learn concurrently and transfer knowledge during training time. Feature representations (Zhu, Lin, Jain, et al., 2020) can be adopted to reuse feature extraction methods even if the reward function changes. Another option is to share parameters of the same ANN for better generalization and robustness against perturbations, while also leveraging an improved sample efficiency (Christianos et al., 2021; Song & Chai, 2018; Terry et al., 2022). This can be combined with attention mechanisms that enable a smarter feature selection (D. Chen et al., 2020). Ensembles of students that learn collaboratively and teach each other during the training process have also been proven to effectively model training experience in the form of posterior distributions (Y. Zhang et al., 2017). Furthermore, dual policy distillation has also been studied to prioritize distilling disadvantageous states from the peer policy, so that other aspects of the environment can be explored more efficiently (Lai et al., 2020). However, these studies were carried out by taking into account homogeneous students. Consequently, applying these advances to heterogeneous agents remains uncharted to date.

3.1.1 Contribution Beyond the State of the Art

One of the most distinctive characteristics of the work presented in this chapter with respect to the current literature is that we assume heterogeneous agents deployed in identical sparse reward scenarios with different action domains, where the dynamics of the environment change according to action effects that might not be available to all the agents. Concretely, we rely on IM to deal with exploration issues when having multiple agents. It differs from other studies because we focus on a concurrent paradigm rather than a MARL problem (Böhmer et al., 2019; Iqbal & Sha, 2019; Schafer, 2019). Regarding heterogeneity, herein it is used to refer to discrepancy in the action space between agents. In this sense, the most related works to handle such action heterogeneity are (Calvo & Dusparic, 2018) and (Zolna et al., 2019). However, none use IM to deal with exploration issues. The first, (Calvo & Dusparic, 2018), addresses the heterogeneity dilemma by training all agents independently of each other, whereas in the latter, (Zolna et al., 2019), the state space is not influenced by action space differences, and the advantage of using some actions or others hinges on the fact that, in nature, some actions available in one of the agents help achieve the goal faster, but do not imprint changes in the environment (e.g., an agent which can move two times faster forward than another agent).

3.2 Problem Statement

The exploration-exploitation dilemma is not new in RL. Using IM as a signal to encourage exploration unleashes new challenges:

- *How to generate the intrinsic reward* and,
- *How to correctly balance between extrinsic and intrinsic rewards*

As already seen in Section 2.3.1 and Section 3.1, there are many options to generate such exploration bonuses, each with their advantages and drawbacks. Moreover, establishing a proper balance between the intrinsic bonus and the extrinsic rewards returned by the environment is not easy to accomplish, being a central matter of research (Badia, Piot, et al., 2020; Badia, Sprechmann, et al., 2020). These problems are even more difficult to be tackled when multiple agents are considered, where the exploration of a given state can be subject to the behavior of other agents. Consequently, an important issue in these problems is *how* and when one agent's exploration information should be combined with that collected by other agents working in the same environment.

The problem tackled in this chapter is neither single-agent nor multi-agent, it is a concurrent paradigm: agents learn how to interact with the environment simultaneously, and share knowledge in an *online* fashion. More importantly, agents are heterogeneous because they have slightly different action spaces that make them diverge in their optimal solution space. Hence, how to share the knowledge is not straightforward, as this exchange can lead to absolutely no advantages, or at worst, to negative transfer. In summary, the problem can be formulated as follows:

How can information be shared between heterogeneous agents, avoiding negative transfer and achieving a better convergence in comparison to the case where no knowledge is shared?

In order to illustrate the issues arising from the problem under study, we pause at an example defined over a modified version of the ViZDooM's *My Way Home* scenario, which is depicted in Figure 3.1.a. In this scenario, a door is placed at a point in which, if open, the entrance to a corridor is enabled, making the path to the target point 20% shorter than going through elsewhere in the environment. We assume two different agents:

- *Skilled*: can open the door and access the corridor thanks to a certain action it can perform exclusively.
- *Non-skilled*: lacks the capacity to open the door and hence, it can not access and traverse the corridor.

Due to the different action spaces, their optimal path solutions diverge. However, a unique reward is only provided when reaching the final goal. Therefore, there is no extrinsic signal that provides the agents with information to ascertain whether they should share information, when to do it and how. By training those agents independently, they can achieve the

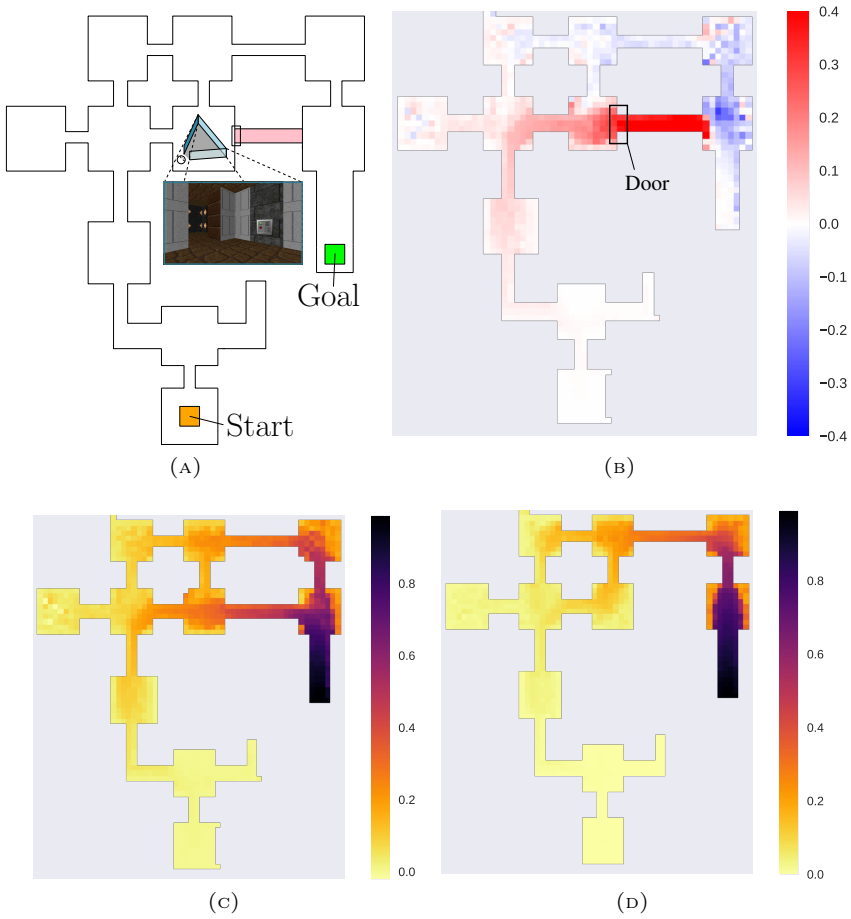


FIGURE 3.1: (a) Modified ViZDooM's *My Way Home* scenario where the goal is to arrive at a target point (highlighted in green) departing from an start point (marked in orange). At the bottom row, the state-value estimates for a skilled agent (c) that can open a door that gives way to a corridor and (d) for a non-skilled agent that is not capable of opening it are shown. The value estimate differences between such agents are plotted in a divergence map (b). Locations where the skilled agent has better value estimates are highlighted in red, whereas in the spots colored in blue the non-skilled agent has better estimates. In those locations where the policy distributions are supposed to be similar, the value estimates are almost equal (white), while differences increase when their optimal paths diverge.

goal through their respective optimal paths (see the heatmaps drawn in Figure 3.1.c and Figure 3.1.d from the respective trained agents). Nevertheless, when learning in a collaborative way, and as a consequence of those different optima and state spaces, the agent reaching the target will encourage the other to follow its steps. Both agents can achieve the goal by chance, although the non-skilled agent may get easier due its reduced

search space. This implies a two-sided competition where the non-skilled agent drives the skilled one into a longer path solution, whereas the skilled-agent pushes the former to take the shortcut that is not reproducible for it. Consequently, negative transfer problems may well arise. This situation can be observed from their value estimate difference which, as shown in Figure 3.1.b, differ remarkably from each other at critical points (near the corridor). These issues can be understood even clearer if the problem is represented as a MDP tree (Figure 3.2), in which the agents will have a share-view of the environment as long as they can reproduce the same trajectories. Nonetheless, some states will only be visited by one agent due to special capacities of its action space, generating an independent view of the problem for that particular agent.

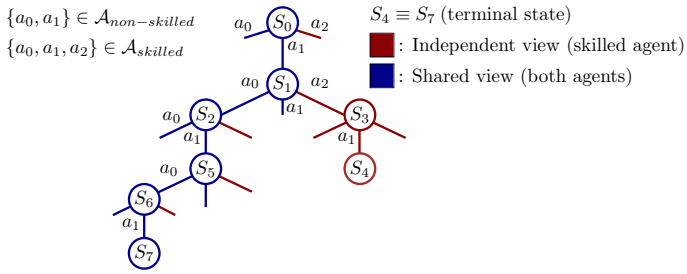


FIGURE 3.2: Example of a MDP as a tree where states are represented with nodes and the edges denote actions. Some states (e.g., S_3) can be reached by being in a specific state and executing a certain action (e.g., $S_1 \xrightarrow{a_2} S_3$). This results in parts accessible and shared between agents (shared-view) and others that are restricted to the capacities of the agents (independent-view).

These problems are not limited to the example shown in the above plot, but also to any scenario with heterogeneous agents. The contribution of this chapter is to expose this problem, and to sketch effective collaborative learning strategies under such circumstances.

3.3 Proposed Collaborative Framework

The design of the framework proposed in this chapter roots in the fact that there can be observations where the policy distributions of heterogeneous agents can be very similar to each other. In some cases, both agents can push each other towards the same direction, i.e., $\pi_{skilled} \equiv \pi_{nonskilled}$. However, in other cases those distributions can differ from each other because each agent pushes in a different direction based on their optimal solution learned at that time. In this situation, we aim to strengthen the shared knowledge between both of them, yet at the same time, to avoid negative transfer in places where the optimal solutions of each agents are in conflict. Consequently, the goal of the framework is to learn a shared-knowledge view while respecting those subspaces in the environment where the interest of the agents are not the same.

As already explained in previous sections, in problems characterized by sparse rewards the main issue to deal with is an efficient exploration of the environment. The application of IM and on-policy techniques does not permit to interfere in the action-sampling process directly, as the training experiences have to be representative of the current policy, i.e., $a \sim \pi(s)$. Hence, the use of past experiences or even samples collected by other policies is not tractable[3]. In this case, the policy is optimized as per Expression 2.14 where, aside from the inherent mechanism of the algorithm itself, the advantage estimator $\widehat{A}_t$ is the main factor that eases and pushes the learning process[4]. The latter advantage estimator can be estimated in different ways, but almost all of them are correlated to the reward r_{t+1} and the value function $V(s_t)$ through the TD-error:

$$\delta = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{3.1}$$

whose value changes iteratively as soon as $V(s_t)$ gets updated. This process can be said to converge when $V(s_t) = V^*(s_t)$.

The framework described in what follows aims at accelerating the learning process focusing on the exploration part, more concretely in how to generate better advantages. For that purpose, we propose a framework driven by two different design objectives (DO):

- DO1: How to generate more accurate and faster state value estimates $V(s)$.
- DO2: How to modify the intrinsic reward generation process to be tackled more efficiently when dealing with heterogeneous action spaces.

Next, multiple methods are proposed to address these objectives within a collaborative framework (see Figure 3.3), so that the ongoing ablation studies in Section 3.5 can inform about the best options among the postulated methods. For simplicity, hereinafter we consider only 2 heterogeneous agents, *skilled* and *non-skilled*, although the approaches could be extended to work with more agents.

3.3.1 Centralized Learning with Decentralized Execution

Our framework adopts an actor-critic policy gradient architecture with two separated networks:

- An actor whose policy (one for each agent) is fed just with its local observations.
- A critic with two output heads related to the extrinsic (V^e) and intrinsic (V^i) signals that is trained with the observations gathered by all the agents.

[3]Not tractable at least theoretically without any type of correction, such as importance sampling (Christianos et al., 2020; Schäfer et al., 2022).

[4]We assume $\psi = A_t$.

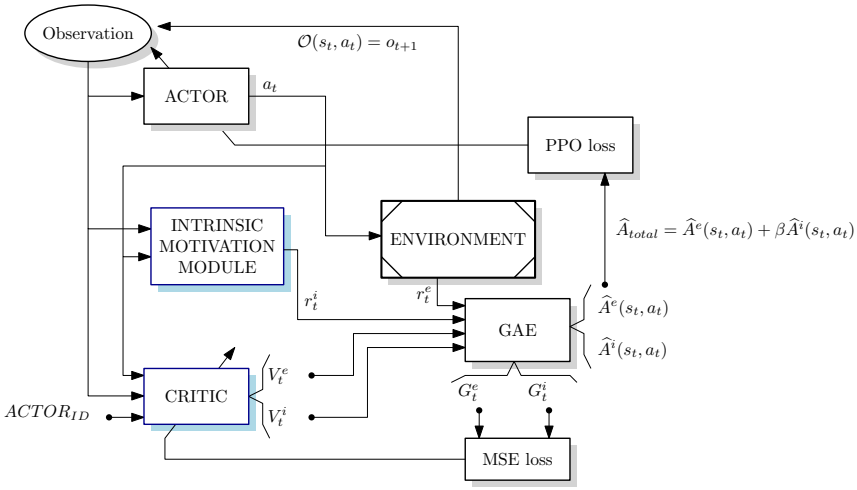


FIGURE 3.3: Flowchart of the collaborative framework, where we highlight in blue those modules that are usually performed independently for each agent, and that can be shared in our framework.

The core idea is to have a unique and centralized critic, so that its capabilities can be augmented with additional information corresponding to the different agents solely during the training phase. This strategy is also known in the literature as the *centralized learning with decentralized execution* (CLDE) paradigm (Foerster et al., 2017; Lowe et al., 2017). With this design, we aim to expedite the critic's learning process so as to generate more accurate and faster value estimates, contributing to DO1. Moreover, it gives rise to a scalable architecture which can easily take into account more agents with little additional complexity.

3.3.1.1 Decentralized Actors

In spite of using centralized learning strategy, the behavior of each agent can be very similar yet not equal. As a consequence, each agent is parameterized by an independent actor[5].

As above explained, the benefit of CLDE relies on learning faster and more accurate $V(s)$, which subsequently has a positive effect on $A(s,a)$, ultimately leading to an improved overall learning. However, the speed at which this is achieved depends on multiple factors. All this coupled with the fact of transient intrinsic rewards (r_t^i) and sparse extrinsic feedback (r_t^e), increased the importance of introducing Monte Carlo updates to latch on to these signals rapidly (Bellemare et al., 2016; Ostrovski et al., 2017). In our framework, this is instead circumvented by using GAE (Schulman et al., 2015) and calculating two independent advantages for the extrinsic and intrinsic streams, $A^e(s,a)$ and $A^i(s,a)$, which are then blended as

[5]For practical purposes, their learning works in the same way as when being done independently. That is, the actor is trained only with data captured by itself as it would do in a single agent scheme.

follows:

$$A(s,a) = A^e(s,a) + \beta A^i(s,a) \tag{3.2}$$

This implies having extrinsic (V^e) and intrinsic (V^i) streams with their respective independent returns, which allows for a higher flexibility to combine episodic and non-episodic returns. It also enables the use of different discount factors (i.e., γ^e and γ^i). Moreover, it is intuitively more suitable to separate both streams that are indeed stationary (V^e) and non-stationary (V^i) in nature. The extrinsic reward in a singleton environment has an associated V^{e*} because the extrinsic reward function does not change throughout the learning process[6]. On the contrary, V^{i*} will vary as the training evolves because the generated intrinsic rewards depend on a novelty measure that changes right after every interaction. Note that combining in this way the extrinsic and intrinsic streams is just another strategy (Burda, Edwards, Storkey, et al., 2018) that substitutes the naive idea of mixing both objectives in a weighted reward as in Equation (2.25).

3.3.1.2 Centralized Critic Module

When conceived within collaborative learning, a problem that requires attention is that the value function estimates, $V(s)$, can be different among agents for the same state, although it might be equal or very similar at many other states of the same scenario (recall Figure 3.1). Based on this intuition, the value of a state should depend not only on the state itself, but also on the possible actions of the agents. Henceforth, we propose to use a centralized action-value function, $Q(s,a)$ which, as shown in Figure 3.4, is fed with the observations of all agents, producing the value estimate of selecting an action a_t when being at state s_t. This is, instead of producing an estimation for the state value $V(s)$, the centralized module elicits all $Q(s,a)$ possible values for $a \in \mathcal{A}_{skilled} \cup \mathcal{A}_{non-skilled}$, regardless of the agent collecting the observation.

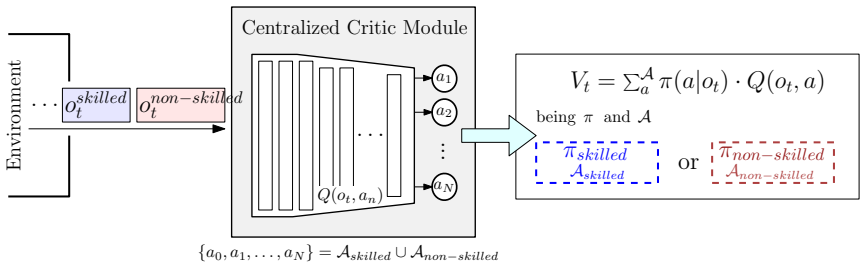


FIGURE 3.4: Centralized critic module based on $Q(s,a)$ (instead of $V(s)$) for 2 agents with different action spaces ($\mathcal{A}_{skilled}$, $\mathcal{A}_{non-skilled}$). In the image, how $V_t(s)$ is calculated for each case is shown.

This architectural change of the critic module implies several considerations. To begin with, $A(s,a)$, which is one of the key components for

[6] We are not considering environment with stochastic transitions.

the calculation of the the actor's loss, commonly requires a value estimate –$V(s)$ (not $Q(s,a)$)– to reduce its variance (Schulman et al., 2015). Therefore, we calculate different state values $V_x(s)$ for each agent by taking into account their action spaces, as follows:

$$V_x(s) = \sum_{a \in \mathcal{A}_x} \pi_x(a|s) \cdot Q(s,a) \tag{3.3}$$

where $x \in \{skilled, non-skilled\}$ and $\pi_x(a|s)$ denotes the probability of each agent x performing action $a \in \mathcal{A}_x$ in state s. Thus, an agent not capable of executing a given action will have a zero probability for that given option. This can be also regarded as a way of masking possible outcomes.

Additionally, the critic loss is slightly modified to accommodate the multiple action-wise outputs as opposed to the unique output neuron usually set when critic estimates directly the value of the state itself. Namely:

$$\mathcal{L}_{critic} = \frac{1}{T} \sum_{t=0}^{T} \left(Q(s_t, a_t) - \widehat{Q}_t \right)^2, \tag{3.4}$$

where a_t is the action taken by the agent at time step t, and $\widehat{Q}_t$ is a discounted return estimate of the T-length rollout over which the optimization step is performed.

Last but not least, the critic is updated with the tuples gathered by each agent individually, and executes an optimization step per collected batch of experiences:

$$\mathcal{B}_{skilled} = \{(s_t, a_t, r_t), (s_{t+1}, a_{t+1}, r_{t+1}) \ldots, (s_{T-1}, a_{T-1}, r_{T-1})\} \sim \pi_{skilled}$$
$$\mathcal{B}_{non-skilled} = \{(s_t, a_t, r_t), (s_{t+1}, a_{t+1}, r_{t+1}) \ldots, (s_{T\text{-}1}, a_{T\text{-}1}, r_{T\text{-}1})\} \sim \pi_{non\text{-}skilled}$$

As a consequence, the critic will take as many optimization steps in every training step as the number of agents at hand (in the considered case, 2 updates with $\mathcal{B}_{skilled}$ and $\mathcal{B}_{non-skilled}$).

Universal Value Function Approximator

An alternative to the previous proposed centralized critic is to adopt a so-called Universal Value Function Approximator (UVFA) design (Schaul et al., 2015), where the ANN will be conditioned to additional parameters (i.e., to a determined goal $V(s,g)$). Actually, in the proposed framework the value estimation is subject to the agent's capabilities:

$$V(s) \rightarrow V(s, actor_{id}) \tag{3.5}$$

Indeed, with the previously mentioned action-value architecture modification, it will be $Q(s, a, actor_{id})$ as shown in Figure 3.5. Analogously to the procedure followed for the other critic architecture, advantages will be

calculated with value estimates that will be obtained as in Expression 3.3.

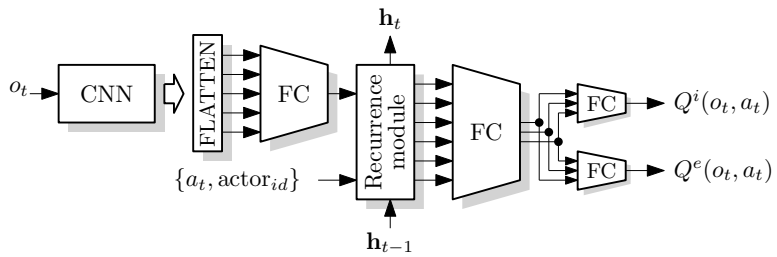


FIGURE 3.5: UVFA based centralized critic, where the convolutional (and the following FC) layers extract common features to both type of agents. The rest of the network is parameterized subject to the skills of each agent.

The design is inspired by the idea that the feature extraction of an observation can be linked to an agent but not to the additional information that can be inferred from a sequence. In this latter case, it could be inconsistent due to the agent's different capabilities to generate their own divergent trajectories that might well not be reproducible by other agents. In order to address this inconsistency during the training stage, and to aid the network in gaining insights about what knowledge must be shared and what must be preserved for individual use, information about the skills is provided to the network as an input ($actor_{id}$)[7]. In addition, the action in every time step a_t is also fed as an input, which can be useful to learn better temporal representations within the recurrent module. Other parameters such as the trade-off between intrinsic-extrinsic streams (i.e., β coefficient) or the collected rewards (i.e., r_t^e and r_t^i) could also be advantageous (Badia, Sprechmann, et al., 2020). Nevertheless, the study is limited to the aforementioned parameters in order to avoid over-parameterized critic architectures.

Overall, with the design of a centralized critic we aim to have a more robust and stable learning, where the shared-view value estimates of the environment should be easier to obtain, while not hindering the calculation and learning of the independent-view value estimates when the optimal solutions of the agents diverge. This closely aligns with the design objective DO1 established previously.

3.3.2 Centralized Intrinsic Curiosity Module

The most straightforward strategy to make the exploration of one agent depend on the exploration performed by others is to combine them by using a centralized module, which is directly related to the intrinsic reward generation (DO2). This idea relies on the principle of *divide and conquer*, where an observation should be discouraged to be visited if the other agent

[7]The information is encoded as a one-hot vector distinguishing between agents with different action domains, i.e., $actor_{id} \xrightarrow{skilled} [1,0]$ or $actor_{id} \xrightarrow{non-skilled} [0,1]$.

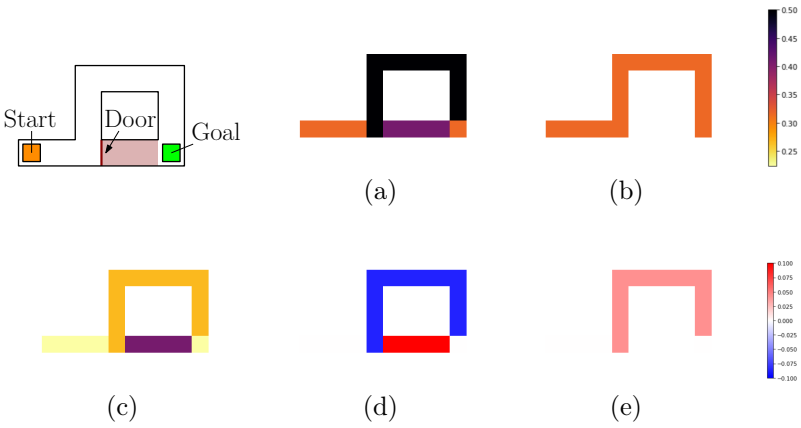


FIGURE 3.6: Evolution of the intrinsic rewards in a simplistic RL environment after 10 executions according to the number of visits (i.e. $r^i = 1/\sqrt{N(s)}$). The agent is initialized at the bottom-left corner and its goal is to arrive to the destination located at the bottom right. Going straight, in the middle is a door that obstructs the path, which can be only be opened by a skilled agent. (a) Intrinsic rewards heatmap of a skilled agent able to traverse the corridor through the door and go straight. (b) Intrinsic reward heatmap of a non-skilled agent not capable of opening the door, hence arriving at the target through the larger path. (c) Resulting intrinsic reward heatmap when combining both type of agents' visits for a total of 10 executions per agent (20 in total). (d) Relative difference of rewards using the centralized novelty (as in subfigure *(c)*) with respect to using two skilled agents (subfigure *a*) for the same amount of interactions. (e) Relative difference of rewards using the centralized novelty (subfigure *(c)*) with respect to using two non-skilled agents (subfigure *b*) for the same amount of interactions. In (a,b,c) darker colors mean higher reward; brighter the opposite. In (d,e) red means that the centralization with heterogeneous agents encourages visiting those locations more often with respect to using homogeneous agents, yielding higher intrinsic rewards in that location by virtue of having heterogeneous actions (blue the opposite).

has already been there, promoting the exploration of uncharted areas. The problem of this assumption is that if agents have different knowledge and/or capabilities, one agent may get discouraged to explore areas that are indeed crucial for finding its own optimal solution and enforced to visit unpromising areas instead.

In practice, by using a centralized curiosity approach with multiple heterogeneous agents, the experienced novelty is affected. Let's see the expected modifications following the example illustrated in Figure 3.6.

Firstly, the intrinsic bonuses for those states that can be reached by both agents will be smaller(Figure 3.6.c, yellow areas). By the same token, intrinsic returns should be higher along those trajectories in which the agent visits more novel states. This behavior is exacerbated in those states that are only accessible by one of the agents (i.e., skilled agent, Figure 3.6.a, corridor colored in purple), as they can only be visited by them

and its novelty decreases at a slower pace when compared to the rest of possible states (Figure 3.6.d, red). Therefore, the skilled agent will end up becoming more encouraged to visit restricted areas – namely, states that are only possible to be accessed by the use of the action that make them to be different – when compared to the behavior in the decentralized intrinsic module approach.

In regard to the non-skilled agent, using a centralized curiosity with an additional more skilled agent has little impact in its exploration procedure, as the novelty distribution will undergo no changes for it. Indeed, the parts that are critical for the skilled agent –the door and the corridor– do not influence the exploration of the non-skilled (Figure 3.6.e, corridor). The remaining state space will be similarly visited for both agents. However, if we assume that the skilled agent will be encouraged to visit more times those experiences leading the corridor, inversely the non-skilled agent will be discouraged to go over those same locations. Eventually, the non-skilled agent will be pushed towards exploring other alternatives. This can be observed in Figure 3.6.e, in which the non-skilled agent will be more encouraged to explore through the larger path (as told by the higher rewards colored in red) when combining its rewards with a skilled-agent with respect to doing it independently.

In conclusion, adopting a centralized curiosity module can be beneficial when heterogeneous agents are involved. On the one hand, actions yielding observations that can only be achieved by the one of the agents (i.e., open the door and access the corridor) will have larger intrinsic rewards, and hence, higher returns, fostering the exploration of that state space. At the same time, it discourages the agent who is not capable of executing such actions of exploring the state space that guides such non-reproducible situations (i.e., corridor), being advantageous to focus on exploring other promising zones.

3.3.2.1 Action-based Curiosity Module

Manifold means of calculating the novelty of a given state have been proposed in the literature. Mechanisms to deal with novelty are based on using either s_t (Bellemare et al., 2016), s_{t+1} (Burda, Edwards, Storkey, et al., 2018) or even the information related to the transition between successive states $\{s_t, s_{t+1}\}$ (Pathak et al., 2017)[8]. In this vein, when having multiple agents using this module in a centralized manner, they update it more frequently with the experiences sampled by their own independent action distributions, leading to different visitation strategies as those depicted in Figure 3.6. Notice that **the agent will be discouraged to visit states already inspected regardless the actions taken before**. This implies that the agent will have the same curiosity to visit a state and execute an action frequently selected (at that state) as selecting another action that has been barely chosen. Previous works have reported

[8]The intrinsic reward is generated just with s_{t+1}, but the update of the whole ICM framework requires a_t, s_t and s_{t+1}.

that no difference arises from considering the action (Tang et al., 2017), speculating that the policy itself was sufficiently random (i.e., had sufficient entropy) to entrust the exploration at each state. This hypothesis, however, was validated over RL environments with single agents whose individual exploration does not interfere with the interaction and learning of other agents. By contrast, when heterogeneous agents are involved, the action selection and its consequent exploration becomes more sensitive.

Therefore, we modify those intrinsic related approaches in order to account for the action as well, so that the generated intrinsic rewards become more informative for the critic (DO2). In fact, a strategy that takes into account both the action and the state when computing the novelty will encourage a more homogeneous action selection and a deeper exploration (Raileanu & Rocktäschel, 2020). This difference may not hinder convergence in single-agent RL problems, but can be problematic when having agents with different action spaces. In this latter case, actions that can only be executed by just one agent will become more affected, as shown previously in Figure 3.6.

3.3.2.2 Tree Filtering

Previous exploration strategies aim at sharing as much information as possible between the agents. Nevertheless, there might be states embedded in a trajectory that are not accessible by some agents where specific chunks of the trajectory might, in turn, be reproducible.

On the one hand, a trajectory can be thought to be shareable for both agents if the actions taken by the agent responsible for gathering the experiences belong to the mutual action space[9].

On the other hand, let us consider a trajectory gathered by the skilled agent that is not fully reproducible by the non-skilled agent. *Can that information be used in some way by the non-skilled agent (rather than being discarded)?* This is what **tree-filtering** is all about. In order to explain it and for the sake of clarity, consider the trajectory shown in Figure 3.7, where we can distinguish two main chunks of experiences:

- $\{(s_{49}, a_2), (s_{50}, a_3), \ldots\}$:

  From s_{49} onward, the whole trajectory is assumed to be reproducible by the non-skilled agent too. In spite of the non-skilled not being responsible of collecting such experiences, the curiosity of both agents at them is updated (i.e., decreased). As a consequence, future returns, and subsequently, their critic estimates, will reflect it[10].

[9] This also applies when selecting an action out of that mutual action space which has no effect on the environment, or which is interchangeable by one of the actions of the mutual action space.

[10] If the non-skilled agent is not capable of reproducing some of those states, the novelty update, from the perspective of that agent, will be insignificant, as it would never be able to explore that situation; on the contrary, it would assume that an agent with at least the same capabilities would have previously explored them (pretending that the non-skilled agent itself gathered them).

- $\{\dots(s_{45}, a_1), (s_{46}, a_1), (s_{47}, a_2), (s_{48}, a_4)\}$:

  At state s_{48}, the skilled agent executed an action that does not belong to the mutual action space, a_4, which is not reproducible by the other agent.

  Should we then decrease the novelty of the non-skilled agent for all those $\{s, a\}$ tuples?

  If so, that novelty reduction will be noticed when the non-skilled agent collects a trajectory containing any of those experiences and updates the critic. Let us examine the consequences:

  – Regarding (s_{48}, a_4), no impact will be caused, since this tuple is indeed impossible to be experienced in any trajectory performed by the non-skilled agent.

  – Nonetheless, for the rest of feasible tuples:

  $$\{\dots(s_{45}, a_1), (s_{46}, a_1), (s_{47}, a_2)\},$$

  the intrinsic reward signal will be lowered, discouraging the non-skilled agent from developing its own exploration strategy on account of an external update of the skilled-agent not playing the role of an equally skilled agent.

  In order to encourage the non-skilled agent to create its own personal experience, the novelty update of the tuples from s_{48} back to the initial state are not performed on the non-skilled agent, allowing it to keep on working on its independent individual view.

As a result of this filtering process, we propose to consider *novelty along sequences* rather than *novelty as attractiveness on isolated step-on states*[11]. This is, we aim to minimize the error between the globally generated novelty estimation of paths taking into account the intrinsic rewards generated at each experience and also their reproducibility, thus polishing the intrinsic reward recollection by allowing room for independent views on the environment (DO2). Ideally, the novelty through a path would be handled by a intrinsic curiosity module that takes into account sequences rather than single experiences. However, as we will further elaborate in Section 3.7, the design of such a novelty reward function is not trivial at all.

3.3.3 Summary of the Proposed Modules

To sum up, the proposed collaborative framework is composed of a centralized critic and modified intelligent exploration strategies, where:

- The use of a centralized critic enhances the learning process by ensuring more diverse experiences. At the same time, a robust knowledge

[11] In practice, the novelty of a sequence is calculated as the discounted intrinsic return for each the experiences belonging to that trajectory, which is a sum of independent intrinsic bonus as in Expression (2.3).

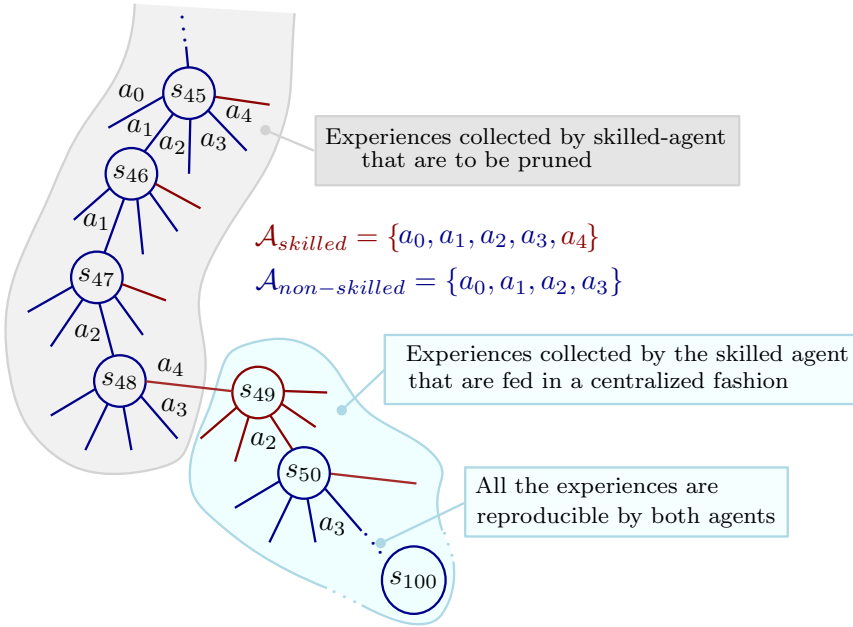


FIGURE 3.7: Tree-filtering process exampled with a trajectory collected by the skilled-agent (the nodes represent the states collected at time step t and the edges all the possible actions). From a state onward, the non-skilled is not able to reproduce the experience $\{(s_{48}, a_4)\}$. Because of that, all the experienced state-actions up to that point (grey shaded tree) are not taken into account for updating the non-skilled agent's curiosity.

view is constructed, capable of distinguishing between different types of agents and their respective action-solution domains. That is, it helps generate better and faster value estimates (DO1).

- The use of a centralized exploration strategy helps generate more suitable intrinsic rewards, taking into account the diversity between agents (DO2). When dealing with heterogeneous agents, the consideration of the action should be taken into account to generate the rewards so as to harness interesting albeit diverging behaviors throughout the experience of the agents with the environment.

We have defined multiple centralized critic and exploration strategies that can be combined in different ways to set the collaborative framework. Next, in Section 3.4 we specify two particular frameworks and analyze their results over various experimental setups.

3.4 Experimental Setup

In order to analyze the benefits of the previously introduced collaborative framework, we resort to the so-called ViZDooM's *My Way Home* environment (Kempka et al., 2016), recently adopted as the environment of choice

for studies related to sparse rewards problems and curiosity mechanisms (Iqbal & Sha, 2019; Pathak et al., 2017; Savinov et al., 2019). The goal is to learn how to reach a target position taking into account that the agent may spawn at different starting points. The modified versions of the original environment introduce corridors that modify the shortest path to the target goal yet require an action only available for one of the agents. Hence, we can gauge the impact of having heterogeneous agents with different action spaces and optimal solutions. We consider the following actions transversal to both agents:

- $a_0 \rightarrow$ `move forward`
- $a_1 \rightarrow$ `turn right`
- $a_2 \rightarrow$ `turn left`
- $a_3 \rightarrow$ `no-action`

These actions constitute the action space of the non-skilled agent, i.e., $|\mathcal{A}_{non-skilled}| = 4$. Note that none of these actions prepare the agent to go through the corridor.

Next, we present two study cases in which the environment modifications have no impact at all in the case of the non-skilled agent, but they do make a difference for the skilled one.

3.4.1 Case Study 1

Environment and $\mathcal{A}$ Modifications

In Figure 3.8 we present two setups (named as Setup 1 and Setup 2) where the corridor is displayed at different locations. The corridor exhibits a height constraint that limits the access only while the agent is crouched. To pass through the corridor, the skilled agent is endowed with the capability to perform an additional action:

- $a_4 \rightarrow$ `crouch and move forward`,

which as the name suggests, allows the agent to crouch while moving straight. Therefore, its action space expands to $\mathcal{A}_{skilled} = \mathcal{A}_{non-skilled} \cup \{a_4\}$, i.e., $|\mathcal{A}_{skilled}| = 5$. This ability has no advantage in any other parts of the environment, since the rest of rooms and corridors are high enough to be traversed without crouching. However, it can be still performed to move forward at the cost of modifying the observation of the agent when compared to moving forward without crouching (a_0).

Following the methodological procedure in (Pathak et al., 2017), we consider three different settings based on the location where the agents are initially deployed:

- *Dense* setting: agents are randomly located in any of the 16 available rooms/corridors.
- *Sparse* setting: agents always spawn at room 10.

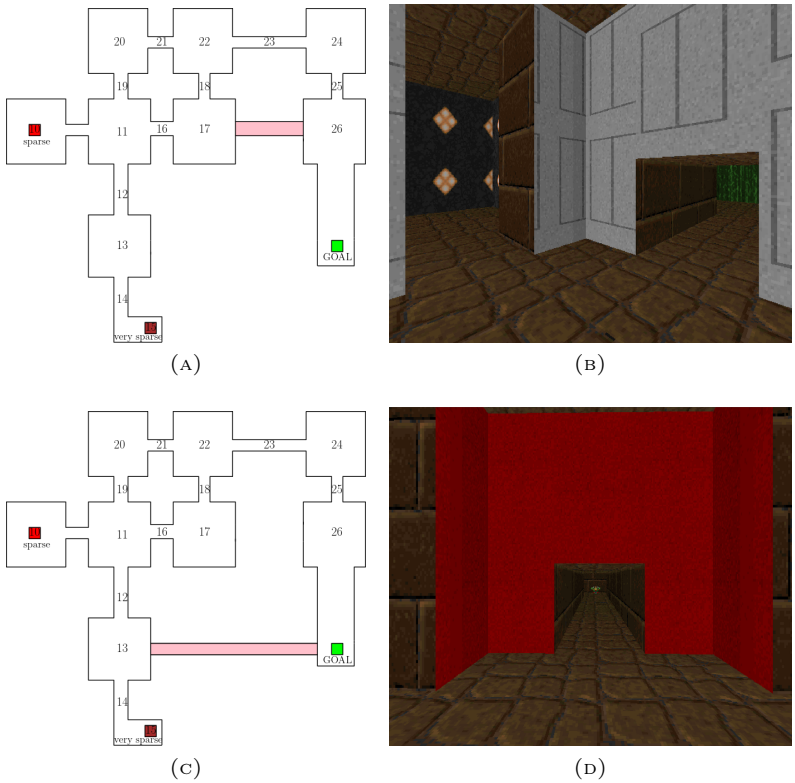


FIGURE 3.8: Modifications of the ViZDooM's *My Way Home* environment: a) Setup 1, in which a height-constrained corridor connects rooms 17 and 26; c) Setup 2, with a long similar corridor connects room 13 with the target goal. (b) and (d) views in Setup's 1 room 17 and Setup's 2 room 13, respectively. In both setups, the agent can be initially spawned in any room (*dense* setting), at room 10 (*sparse* setting, red square) or at room 15 (*very sparse* setting, brown square), being the goal to arrive to the target (green square).

- *Very sparse* setting: agents are always deployed at room 15.

None of these settings renders any type of feedback signal in any step but the final one. However, the probability of arriving at the final destination following a random policy varies depending on the proximity of the position where the agent is deployed. Accordingly, the dense setting can be claimed to be the easiest one to be solved, with approximately 23% of chances that a random agent reaches the destination. By contrast, the sparse and very sparse settings are more complex, with the aforementioned chances of success lowered down to 6% and 5%, respectively. These random-policy success rates vary depending on the specific setup and skills of the agent. They can be used as baselines for each simulated case.

Methodological Details

We adopt a simple centralized critic (Figure 3.4) in which RND is used to generate the intrinsic rewards (Figure 3.9). The action-value $Q(s, a)$ estimator updates its parameters in a sequential manner; this is, it executes an optimization step per agent[12]. On the contrary, the curiosity module, when being used jointly by both agents, is updated with a single optimization step by concatenating the observations collected by both agents.

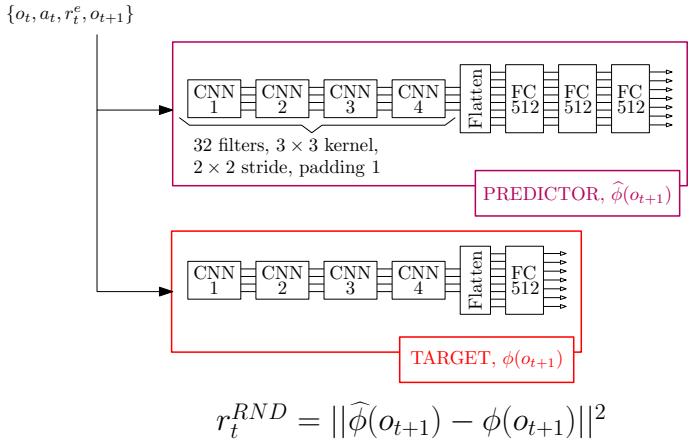


FIGURE 3.9: Random Network Distillation (RND) setup to generate intrinsic rewards.

In this first case study no multiple parallel environments are used, even though it might introduce some instability in the calculus of the gradients. The main reason is that when using multiple parallel environments, the agent has a higher probability of reaching the goal by mere chance, which is the only time the agent receives an *extrinsic* reward. Thus, the analysis of the benefits of a collaborative learning strategy might be severely influenced by rewards issued by sampling rather than by the developed strategy itself.

3.4.2 Case Study 2

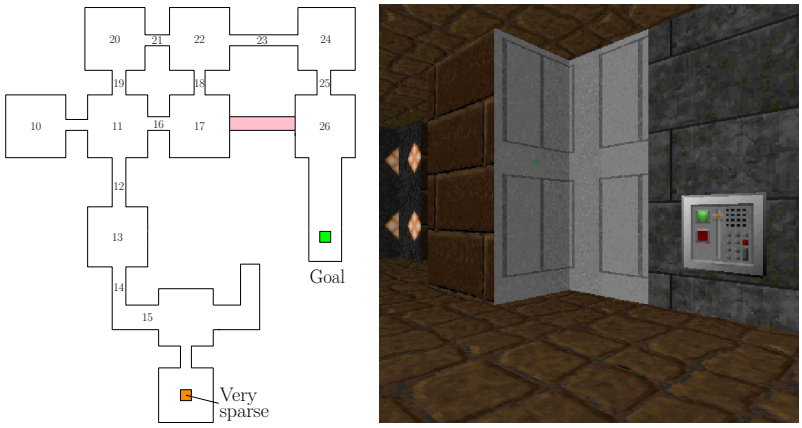
Environment and $\mathcal{A}$ Modifications

Akin to the Setup 1 shown in Figure 3.8, we set an alternative layout, namely Setup 3, in which the corridor can still be accessed from the same rooms. However, its access is constrained by an actionable door as shown in Figure 3.10. Therefore, the skilled agent is granted an additional action to be able to open this door:

[12]Every time an agent finishes the collection of a rollout, a backpropation update is carried out with the gradients corresponding to the experiences within the collected trajectory.

- $a_4 \rightarrow$ `open`

This ability is useless (i.e., `open` ≡ `no-action`) in any place except in front of the door. Note that **in this setup the agent does not posses `crouch and move forward` action**.



(A) (B)

FIGURE 3.10: Modification of the ViZDooM's *My Way Home* environment: a) corresponds to Setup 3, in which an actionable door hinders a corridor that connects rooms 17 and 26; (b) shows the modified view at room 17. In this setup only the *very sparse* spawn location (orange) is considered, which is farther than the one shown in Figure 3.8. We also note that now the view at room 17 differs from the one exemplified in Figure 3.8(b).

In Setup 3 we have increased the complexity of the most difficult known setting (*very sparse*), where the estimated number of steps to achieve the goal is around 350 by following an optimal policy (Pathak et al., 2017). In our modified environment, this number increases to around 430 steps for the extended *very sparse* setting. Moreover, the probability of a random agent to arrive at the goal is 4.6% for the skilled agent. Within this percentage of successful episodes, 60% of the time the agent manages to achieve the target through the corridor, this is, a random skilled agent is able to reach the goal through the corridor with a 2.8% of chance, and by the other alternative path with a 1.8%. In the case of not having the ability to open the door (non-skilled agent), its probability decreases slightly down to 3.6%.

Methodological Details

As opposed to the other case study, here an UVFA centralized critic design is adopted (Figure 3.5). Aside from the architecture itself, the UVFA critic also incorporates a recurrent neural network in order to assess its actual impact over the considered POMDP problem. Besides, a count-based strategy as in (Iqbal & Sha, 2019) is considered, so that the exploration bonus is made inversely proportional to the number of counts

for every state, i.e., $1/\sqrt{N(s)}$ (Bellemare et al., 2016). To realize this, we discretize the environment space in square bins, giving rise to a total of 30 and 26 bins along the horizontal and vertical axis, respectively. When using action-based approaches, the count is made subject also to the action (namely, $1/\sqrt{N(s,a)}$), hence increasing the total number of bins. It is important to note that these bins are only used to compute the intrinsic reward, but are not fed to the policy nor to the critic as an additional source of information.

Due to the additional complexity induced by Setup 3, in this case study we use parallel agents to augment the probability of sampling valid approaches. Nevertheless, we minimize the number of parallel agents in use to just 3, which we found out experimentally to be the minimum to yield consistently successful results (i.e., low variance in the task success rate).

3.4.3 Training Details

Data Collection and Processing

When turning the focus on the specifics of the training process, agents are trained using images as inputs. In the same way as other works adopting ViZDooM as benchmark (Pathak et al., 2017), we use normalized gray scale images resized to 42×42 pixels. Moreover, in order to deal with temporal dependencies, the observation representation is constructed by concatenating the current frame with the three previous frames (Mnih et al., 2015). In addition, we use an action repeat factor of 4 (i.e., actions are repeated 4 times between state transitions). Finally, we have empirically chosen a rollout size equal to 50 steps, aiming to attain a good balance between bias and variance. All the experiments are run for 6000 episodes, each with a maximum of either 2100 or 2600 steps (subject to the setup).

Algorithmic Details

When it comes to the PPO implementation, we employ GAE with $\lambda = 0.95$ to calculate the advantages of both extrinsic and intrinsic streams, which are balanced with a relation of 3 to 1, i.e., $\beta = 1/3$ in Expression (3.2). Moreover, we use a discount factor γ equal to 0.99 for both discounted returns, an epsilon value of 0.2 for clipping, an entropy coefficient of 0.01, a learning rate of 10^{-4}, and 4 epochs per training step.

Neural Network Architectures

Finally, network architectures of the actor and critic modules are summarized in Table 3.1.

Intrinsic Rewards Processing

With regard to intrinsic reward generation, we normalize the rewards by dividing them by a running estimate of the the standard deviations of the

TABLE 3.1: Conv2D(A1,A2,B,C,D,E): Convolutional layer with A1 input channels and A2 output channels, B kernel size B, stride C, padding D and activation function E (ELU: Exponential Linear Unit)

	Network Architecture	Training Parameters
Actor	Conv2D(4,32,3,2,1,ELU)+ Conv2D(32,32,3,2,1,ELU)+ Conv2D(32,32,3,2,1,ELU)+ Conv2D(32,32,3,2,1,ELU)+ Dense(256,ELU)+ Dense(# actions, softmax)	Orthogonal initialization Adam optimizer PPO loss
Critic	Conv2D(4,32,3,2,1,ELU)+ Conv2D(32,32,3,2,1,ELU)+ Conv2D(32,32,3,2,1,ELU)+ Conv2D(32,32,3,2,1,ELU)+ Dense(256,ELU)+LSTM(128)+ Dense(256,ELU)+...+ Dense(5) [extrinsic] & Dense(5) [intrinsic]	Orthogonal initialization Adam optimizer MSE loss in both critic heads

intrinsic returns in order to mitigate issues derive from the reward scale (Burda, Edwards, Storkey, et al., 2018), i.e., :

$$r_t^i = \frac{r_t^i}{\sigma(G_t^i(\tau))} \tag{3.6}$$

Moreover, a crucial matter when using ANN is normalizing the input to prevent several problems. Therefore, it also happens with IM methods that use ANN for the reward generation, but it becomes crucial when using RND[13]. Hence, the input to the RND modules is standardized and clipped within values between -5 and 5 as follows:

$$o_{clipped} = \max\left[-5, \min\left[\frac{o-\mu}{\sigma}, 5\right]\right] \tag{3.7}$$

Recall that the latter is only applied when using RND, i.e., only at Setups 1 and 2. More information regarding how RND performs in ViZDooM and why we decided not to use it at Setup 3 can be found at Appendix A).

3.4.4 Evaluation Metrics

In general, the main goal of knowledge reuse in RL is to accelerate the learning process. In order to analyze the benefits of using knowledge transfer, different metrics can be used (Taylor & Stone, 2009). However, a

[13]The target network has its parameters fixed (*frozen*) and cannot adjust its values according to the train data. Consequently, the obtained embeddings might not convey enough meaningful information and could result in high variance outcomes.

framework could report similar performance metrics to other possible options, but could still remain of interest due to other factors related to the training procedure, such as the number of required samples, the training time for a given computational power, and model complexity/size, among other factors. Consequently, discussions on the experimental results later held in this chapter consider two performance scores:

- *Average extrinsic result* (also referred to as *Success Rate*, SR), which is calculated as the average extrinsic score obtained through a window of the last 100 episodes.
- *Number of steps to achieve the goal*, measured from the starting point of the scenario until the agent reaches the target.

The reason for considering these two scores is that, by only inspecting the SR metric, the discussion only regards whether agents have reached the goal, disregarding the required number of steps (which represent the quality of the learned policy). Other works using this environment assume that no rewards are given except when arriving to the goal, when they actually give a small penalization referred to as *living reward*, equal to -0.0001 for each step. This small modification yields an optimal average extrinsic return of 0.97 approximately for 270 steps; this is, they have a reward function that parameterizes the optimality of the results subject to the number of steps. We instead fix a null living reward, and give a reward equal to 1 when achieving the goal (independent of the number of steps). In this way, we stand strict in regards to the sparse reward problem formulation.

Moreover, the environment itself is slightly different depending on the action space of each agent. Hence, in this case the skilled agent has different possibilities to achieve the target, being optimal the one that involves going through the corridor (labeled in what follows as `_OPT`). Therefore, we trace not only whether every agent reaches the target, but also if they navigate through their optimal paths.

Summary

On the one hand, Case Study 1 analyzes the impact of a standard centralized critic approach while using either an independent or a centralized RND-based curiosity module. Setup 1 and Setup 2 establish a corridor in different places (Figure 3.8) while allowing the agent to spawn at various locations based on the selected setting. More importantly, the agents' policies differ due to the presence of a `crouch and move forward` action in the policy of the skilled agent.

On the other hand, Case Study 2 examines a more sophisticated centralized critic design (with an UVFA architecture and LSTM layers). Instead of using RND, visitation counts are used to compute the curiosity and to assess the impact of making the latter independent, centralized and subject to the action space. In addition, it adopts a more challenging setup

(Setup 3, Figure 3.10), where agents differ due to the existence of an `open` action for the skilled agent to open a gate and access the corridor.

As a result of the above case studies, different algorithmic configurations are considered (summarized in Table 3.2):

- Full Independent PPO (`PPO`): the baseline PPO algorithm.
- Independent Curiosity (`IC_IC`): the PPO algorithm with independent curiosity (`IC`) and independent critics (`IC`).
  - Independent Curiosity (`IC_IC_3r`): Uses 3 parallel environments/runners to collect experiences.
  - Independent Curiosity (`IC_IC_6r`): Uses 6 parallel environments/runners to collect experiences.
- Independent Critic + Centralized Curiosity (`IC_CC`): both agents share a unique/centralized curiosity module yet they have independent critics.
- Centralized Critic + Independent Curiosity (`CC_IC`): both agents share a unique/centralized critic, but they remain independent in what refers to the generation of their intrinsic rewards.
- Centralized Critic + Centralized Curiosity (`CC_CC` ≡ `CC_CC_sh`): both agents share all parameters of both the critic and the curiosity modules to generate the intrinsic rewards. By default, solely the state is considered as input.
  - Centralized Critic + Centralized-Action-based Curiosity (`CC_CC_sh_ action`): In this case, the intrinsic bonus is made dependent on the state and the action, instead of just uniquely on the state.
  - Centralized Critic + Centralized-Action Curiosity + Tree Filtering (`CC_CC_sh_action_filter`): this scheme is equal to the previous one, but during the generation of the rewards it prunes those rollouts whose experiences are not reproducible by the non-skilled agent (see Section 3.3.2.2)[14].

3.5 Results and Analysis

Results produced after the experiments held over the aforementioned setup are discussed in this section. For the sake of clarity in the discussion, results are commented based on the following research questions (RQ):

- RQ1: *Does a centralized critic provide any gain when compared to completely independent agents?*

[14] We assume an oracle that informs whether the action executed by the skilled-agent is is reproducible by the non-skilled agent.

TABLE 3.2: Summary of algorithmic configurations of critic and curiosity modules. Besides the setups, the case studies also differ in the use of a (1) standard or UVFA centralized critic and (2) a RND or visitation counts based curiosity module as explained in Sections 3.4.1 and 3.4.2. **: `sh` and `sh_action` are used to distinguish the input for the centralized curiosity module.*

Case Study	Configuration	Critic		Curiosity Module		
		Independent	Centralized	Independent (state)	Centralized (state)	Centralized (state-action)
1	PPO	✓				
	IC_IC	✓		✓		
	IC_CC	✓			✓	
	CC_CC		✓		✓	
2	IC_IC_3r	✓		✓		
	IC_IC_6r	✓		✓		
	CC_IC		✓	✓		
	CC_CC_sh*		✓		✓	
	CC_CC_sh_action*		✓			Naive
	CC_CC_sh_action_filter		✓			Filter

- RQ2: *Does a centralized curiosity yield better performance levels than maintaining the curiosity locally at every agent?*
- RQ3: *Should we compute curiosity incentives based on the (state,action) pair rather than only the state itself?*
- RQ4: *Should agents have their intrinsic rewards updated only by experiences that are reproducible as per their action spaces?*

We now analyze experimental results aiming to obtain informed responses to the above questions, using to this end the different configurations of the proposed collaborative framework that are represented in Table 3.2. Results are reported over 3 independent runs in order to account for their statistical variability. Unless otherwise stated, curves shown in the plots correspond to the average extrinsic return/success ratio (y-axis) obtained after a given number of train episodes (x-axis).

RQ1: *Does a centralized critic provide any gain when compared to completely independent agents?*

We begin our discussion by examining whether a centralized critic performs better than completely independent agents in the RL scenario under consideration. Responses to this question can be found in Figure 3.11, Figure 3.12 and Figure 3.13, which evince that a centralized critic (`CC_XC`) reaches better performance levels with respect to using independent critics (`IC_XC`).

With a centralized critic, both agents manage to solve the task consistently in all the considered setups and settings, while reaching the target through their optimal path in most of the attempts (as shown in the previously referred Figures with `_OPT`). By contrast, agents featuring individual critic modules (`IC_XC`) are more unstable and require a larger amount of episodes than those considered during training.

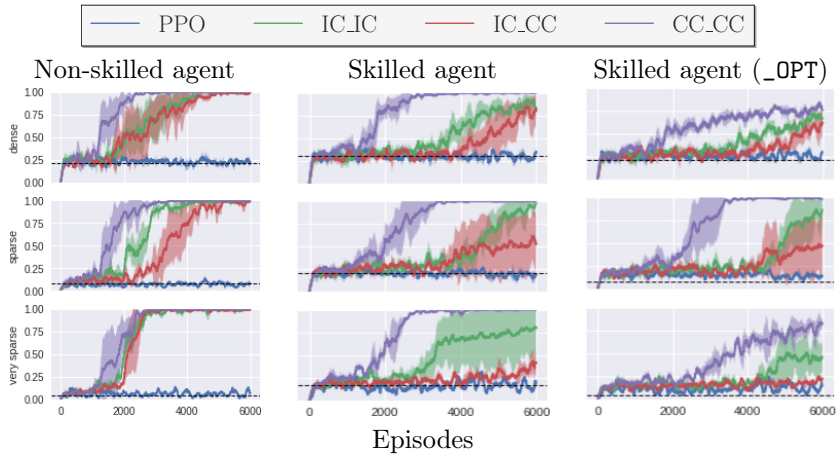


FIGURE 3.11: Average extrinsic return achieved in Setup 1 for different settings (i.e., agent's spawn initialization, each represented in a different row). The last column represents the score obtained by the skilled agent when going through its shortest path (i.e., corridor).

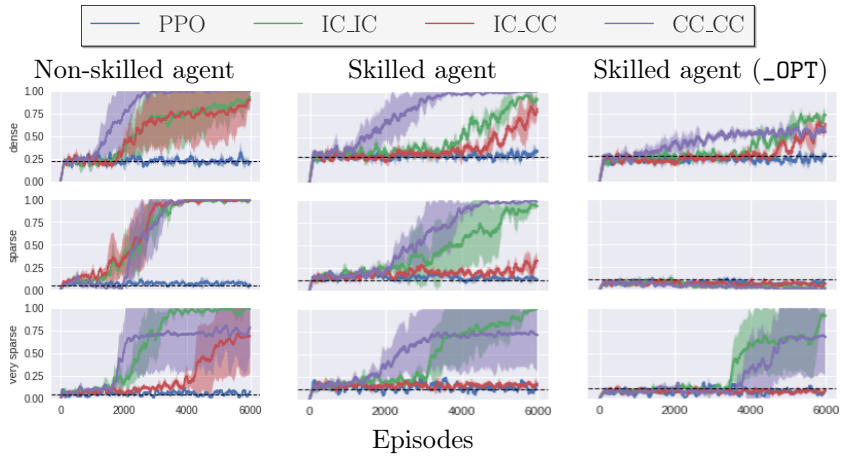


FIGURE 3.12: Same interpretation as in Figure 3.11, but for Setup 2.

Intuitively one can postulate that the advantage of using a centralized critic is that, for the same/unique ANN, more number of experiences are collected (and used). Thus, as we compute the gradients with larger amount of data (gathered by two agents instead of just one), benefits in terms of variance are expected. If this is the case, we can just increase the number of collected experiences by each worker by doubling the number of runners, which ensures each agent to have the same amount of experiences as they would have had when using a centralized critic. This hypothesis can be answered from Figure 3.13, where we observe that `IC_IC_6r` is not only unable to perform as `CC_IC`, but also performs worse than `IC_IC_3r`.

Additionally to less variance, another key difference relies on the fact that `CC_IC` is updated almost twice faster, as it executes an optimization step per trajectories collected by each worker. On the contrary, in `IC_IC_3r` and `IC_IC_6r` each worker has its own critic module, which is updated once for the experiences collected by their respective actor. Nevertheless, if the number of optimization steps was the key factor to perform better, then with twice as many number of episodes, any individual approach should achieve similar performance levels than those by a centralized critic. However, this is not the case either, thereby arriving at the conclusion that a centralized critic performs better than individual critic modules.

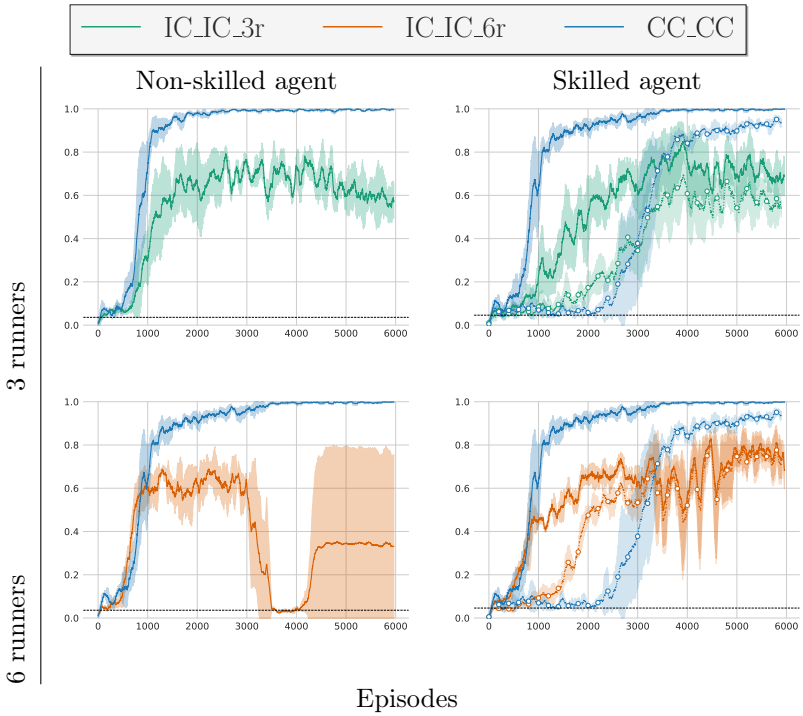


FIGURE 3.13: Average extrinsic return achieved in Setup 3 using independent curiosity for encouraging the exploration when using independent critics (`IC_CC`) and a single centralized critic for both agents (`CC_CC`). We show the curves when using either 3 (upper row) or 6 (bottom row) parallel agent runners for the independent critic case; whereas the centralized critic approach uses 3 parallel agents. Dashed lines with markers are used to plot skilled agent's `_OPT` curves.

RQ2: *Does a centralized curiosity yield better performance levels than maintaining the curiosity locally at every agent?*

Before delving into this second RQ, it is important to highlight that the addition of a curiosity module is undeniably necessary with respect to not using it, as PPO on its own is not able to outperform the behavior of a

random agent (included as a dashed horizontal line in each plot of Figures 3.11 and 3.12).

By using independent critics, results obtained by using either an individual (`IC_IC`) or a centralized (`IC_CC`) curiosity module elicit a better performance when using everything in an independent fashion. This statement is supported by the differences observed in Figures 3.11 and 3.12 for Setups 1 and 2, where `IC_IC` (green) exhibits higher success rates with a better sample efficiency. Besides, these differences are more notorious for the skilled agent, which undergoes more difficulties to go through the corridor when sharing the curiosity module, `CC_CC` (red), as seen in the `_OPT` curves.

On the other hand, when using a centralized critic, the adoption of a centralized curiosity strategy (`CC_CC_sh`) is slightly better with respect to the independent curiosity counterpart (`CC_IC`), which can be confirmed by the results obtained in Figure 3.14[15]. By zooming into these results, for the skilled agent the `CC_CC_sh` approach achieves a 90% of SR with 1309 episodes on average, whereas `CC_IC` requires 1522 (an improvement of 14%). This can be also observed when the skilled agent achieves the destination through the corridor over 80% of the total episodes. At this point of the learning process, the fully centralized approach requires 6% less episodes. In the case of a non-skilled agent, differences are visually negligible, but they represent an improvement of 8%. Furthermore, `CC_IC` finishes with a slightly better policy that requires less steps to achieve the goal.

Interestingly, the results obtained in Setups 1 and 2 with independent critics go against the intuition explained in Section 3.3.2 about centralizing the curiosity module (`IC_IC` > `IC_CC`), although the outcomes in Setups 1, 2 and 3 when using a centralized critic enforces this idea (`CC_CC` > `IC_IC`). We hypothesize that this occurs because the curiosity decreases for both agents when being shared, yet that knowledge is not persisted into their critic modules (when they have independent critics), estimating wrongly the intrinsic value of the state $V^i(s_t)$. This is effectively avoided when using a centralized critic. Therefore, results suggest that sharing the curiosity without sharing the critic as well is not actually beneficial. However, sharing both modules give rise to consistently better results.

RQ3: *Should we compute curiosity incentives based on the (state,action) pair rather than only the state itself?*

Previously, we have concluded that sharing curiosity information between agents yields advantages in terms of success rate and number of steps to reach the target as long as the critic is also shared.

[15]Indeed, the need for having a large number of episodes to actually see that the skilled agent is capable of traversing the corridor conceals any improvements that could arise from the experiments.

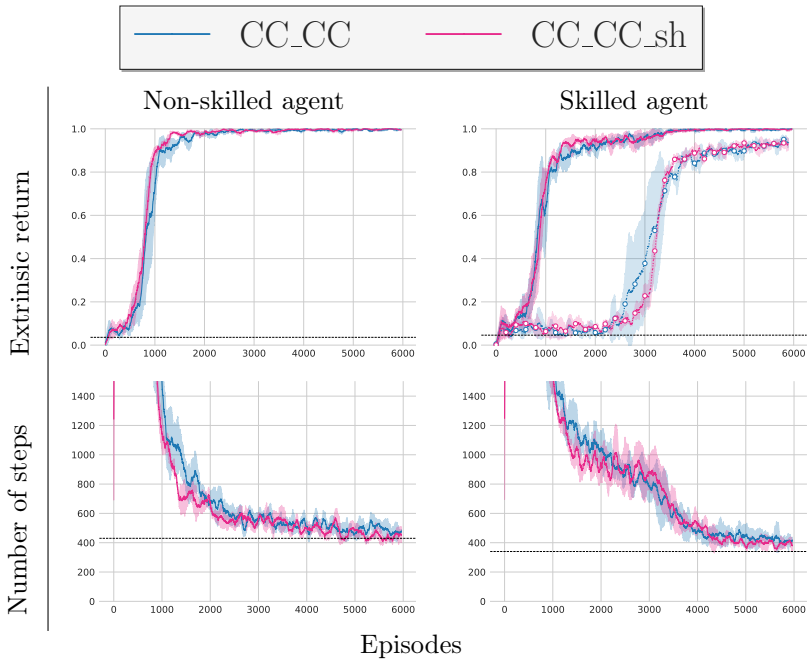


FIGURE 3.14: Average extrinsic return (top row) and number of steps (bottom rom) achieved in Setup 3 using a centralized critic while using either an independent curiosity(`CC_IC`) or a centralized approach (`CC_CC_sh`). Dashed lines with markers are used to plot skilled agent's `_OPT` curves.

Now we turn the focus on evaluating whether the intrinsic reward should be made dependent on both the state and action rather than just the state. In the past, the work in (Tang et al., 2017) showed no empirical differences between both approaches. However, in the cases under study they were not dealing with heterogeneous agents, where the novelty may be influenced by the actions available at each agent. Thus, as foretold in Section 3.3.2, our hypothesis is that by making the curiosity subject to the $\{s, a\}$ tuple, `CC_CC_sh_action`, different exploration behaviors can be induced into the agents, making it easier for the skilled agent to go through the corridor (as a consequence of inducing a larger curiosity for that special action).

In light of the results depicted in Figure 3.15, it is fair to claim that our hypothesis holds, where the skilled agent exhibits a convergence improvement of its success rate of almost 1000 episodes when considering success as traversing the corridor to reach the target. This enhancement can be attributed to a smoother exploration bonus, which is representative on how the required steps decay more abruptly after finding out that path. On the other side, once that the path is discovered, it gets stacked with a policy that is slightly worse than the two approaches analyzed previously. That is, it requires greater number of steps to achieve the goal. We hypothesize that the reason for this effect is the same that leads the agent to

find the path faster: the exploration component (*intrinsic reward*) is high when compared to the extrinsic bonuses, which makes the agent undergo noise in its learning process (higher entropy). The same behavior is also distilled into the policy learned by the non-skilled agent, whose scores are worse despite converging faster.

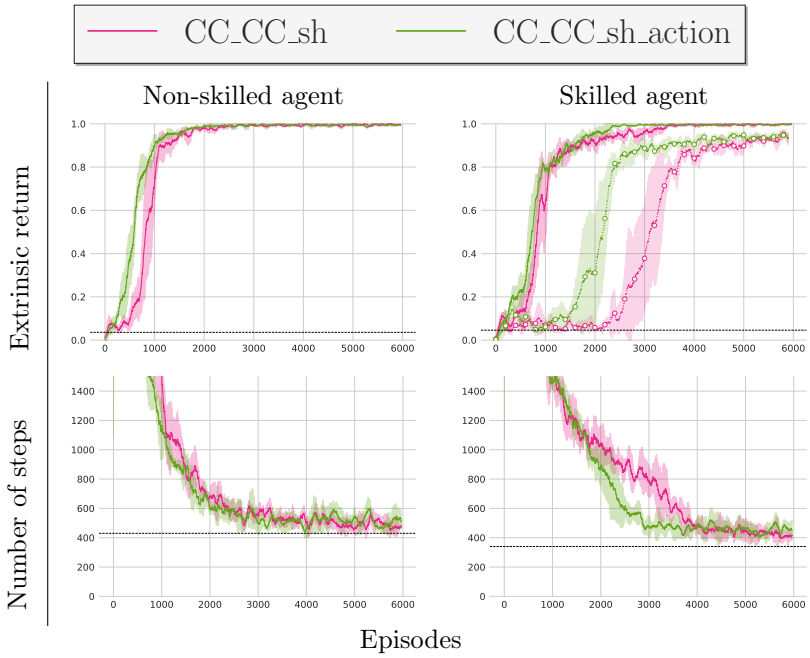


FIGURE 3.15: Average extrinsic return (top row) and number of steps (bottom rom) achieved in Setup 3 using a centralized and curiosity approach, yet making the curiosity to be subject to only the state (`CC_CC_sh`) or the state-action pair (`CC_CC_sh_action`). Dashed lines with markers are used to plot skilled agent's `_OPT` curves.

RQ4: *Should agents have their intrinsic rewards updated rewards only by experiences that are reproducible as per their action spaces?*

Finally, we evaluate the proposed collaborative framework configured with a centralized critic and a centralized action-based curiosity, but filtering according to the idea explained in Section 3.3.2.2, `CC_CC_sh_action_filter`. Differences should appear mainly for the non-skilled agent, so that its learning process changes by deleting those experiences that modify its curiosity inappropriately.

Plots nested in Figure 3.16 validate this hypothesis. A narrow performance gap arises between the two compared approaches `CC_CC_sh_action_filter` and `CC_CC_sh_action`. Both workers converge to a SR of 90% faster when compared to any of the previously analyzed configurations of

the framework, attaining an improvement of 7.7% (skilled agent) and 15% (non-skilled agent) in comparison to the second-best solution.

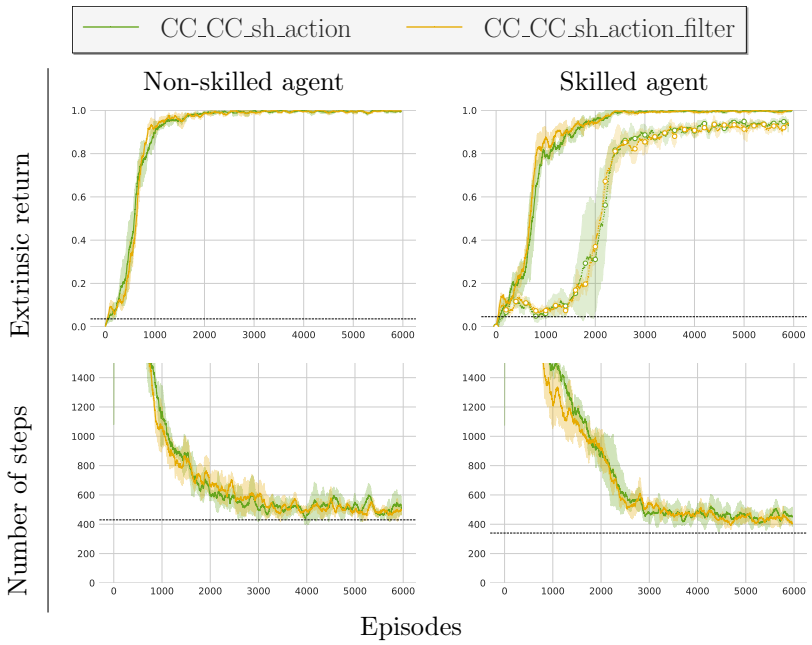


FIGURE 3.16: Average extrinsic return (top row) and number of steps (bottom rom) achieved in Setup 3 using a centralized critic and a centralized curiosity subject to both the state-action, and with (`CC_CC_sh_action_filter`) and without (`CC_CC_sh_action`) filtering the episodes in which the special action has been used (e.g., `open`). Dashed lines with markers are used to plot skilled agent's `_OPT` curves.

3.5.1 Exploration versus Exploitation: When?

One of the major issues arising from the analysis of the results is that the number of steps of the optimal policy is far from the number of steps taken by executing the minimum number of actions[16]. The reason is that, even at the final stages of the training process, the learned policy is too stochastic and still features significant variability. Depending on the problem, this might be a good result as it allows the agent to adapt to changes more easily (Haarnoja et al., 2017). However, if the aim is to learn to perform the task as efficiently as possible, the optimal policy should be the one that converges with the minimum required steps towards the target.

The challenge lies in the absence of a specific objective incorporated into the reward function that guides the problem-solving process with the fewest possible steps. In fact, the policy's enhancement relies on precise value estimates, denoted as $V(s)$, based on the discounted return.

[16]Experiments have considered a frame skip equal to 4, hence the optimal solution with 1 frame per step should require less interactions of the agent with the environment.

These value estimates are then utilized to calculate the advantages, $A(s, a)$, within the loss function of the actor. As a result, if these advantages are not appropriately balanced, meaning they fail to accurately represent the true values, the agent may explore when it should be exploiting information and vice versa. This issue becomes even more complex when considering the precision of the extrinsic-intrinsic streams outlined in Equation (3.2).

The aforementioned problem can be better understood by examining what occurs in Setup 3 with a skilled agent trained following `CC_CC_sh_action`. Figure 3.17 depicts a heatmap showing the balance between the extrinsic and intrinsic parts in the computation of the total advantage at different stages of the training process. We observe that, in Figures 3.17.a and 3.17.b, the agent rarely updates its policy by following the main extrinsic goal (red), although it becomes more influential as the training process evolves.

Interestingly, in Figure 3.17.c –which corresponds to what the agent experiences between episodes 1000 and 1500– it can be observed that the agent successfully navigates through the corridor. Nonetheless, according to Figure 3.15, it is not until 1500 to 2500 episode range when the skilled agent consistently learns to go through the corridor (i.e., `_OPT` curve). In fact, there is a noticeable disparity between the heatmaps in Figure 3.17.c and Figure 3.17.d, particularly in room 17 (the one that provides access to the corridor). In Figure 3.17.d, the heatmap appears more red, indicating that the agent is predominantly influenced by the extrinsic stream. Conversely, in states where white-blue advantages are more prevalent (e.g., in Figure 3.17.c in room 17), the agent's decisions are more influenced by the intrinsic stream, which may not necessarily align with the overall task goal. This suggests that the agent is still exploring and its value estimates are not accurate enough with respect to the extrinsic feedback. As the training process progresses, the agent's decisions undergo consistent updates to align with the recommendations of the extrinsic stream. This exploitation-driven phase, observed from approximately 2000 training episodes onward, focuses on enhancing the policy to achieve the goal of traversing the corridor with the fewest number of steps. The predominance of red-colored decisions in Figure 3.17.e reflects the agent's prioritization of following the extrinsic stream, demonstrating its dedication to efficiently navigating through the corridor. This improvement is further evident in Figure 3.15, where the number of required steps decreases when the agent takes the corridor.

One way to overcome this problem is by properly balancing the composition of the total advantage through β, as using a fixed value of this parameter seems not to be the best choice in view of the results. This unexpected behavior has already been noted in other works (Badia, Sprechmann, et al., 2020; Rosser & Abed, 2021) and still lacks a clear solution in the literature. To further delve into this matter, we compare the performance over one of the previously analyzed experiments, `IC_IC_3r`, for different β values, so as to ascertain whether different beta values yield any differences in the exploration and exploitation phases of the training

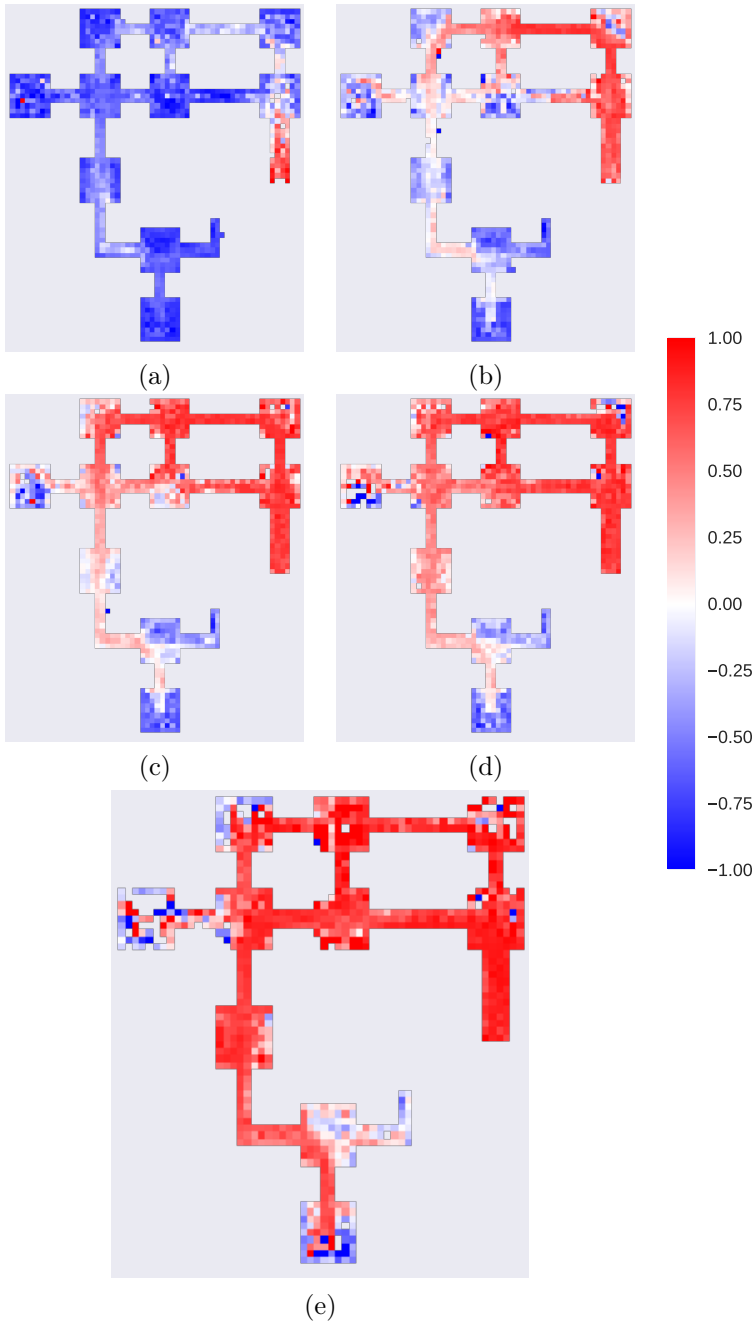


FIGURE 3.17: Divergence maps of the skilled agent in Setup 3, showing when the total advantage estimator is guided by either the extrinsic (red) or intrinsic (blue) advantage term, at different stages of the training process: (a) 0-500 episodes; (b) 500-1000 episodes; (c) 1000-1500 episodes; (d) 1500-2000 episodes; (e) 5500-6000 episodes. Results correspond to one of the `CC_CC_sh_action` simulations/seeds.

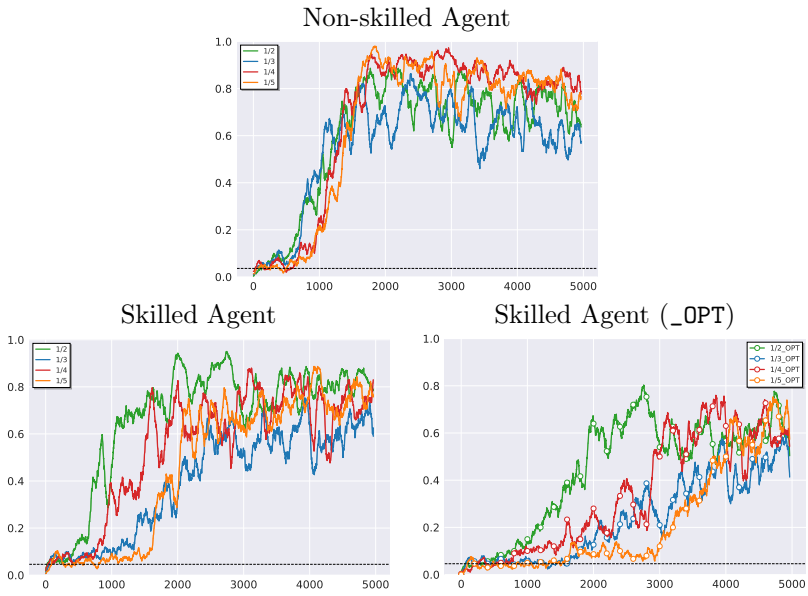


FIGURE 3.18: Performance of agents when using different β values to weight the intrinsic rewards importance in Setup 3.

process. Results of these side experiments are reported in Figure 3.18, where it can be noted that, under the basic β-weighted implementation of the total advantage, the instability of the learning process is not solved by considering different values of the β parameter.

Further along this line, we have carried out two additional experiments by switching off the curiosity for a given number of episodes (i.e., $\beta = 0$). With the purpose of just gauging its impact, we have manually tailored the learning process so that two simulations are run, one in which curiosity is deactivated after 1000 episodes, and another in which the deactivation is triggered after 3000 episodes. The values have been empirically selected as they have proved suitable to ensure the agents have chances to reach the destination through their optimal paths. We will refer to these tailored approaches as `CC_CC_sh_action_1000` and `CC_CC_sh_action_3000`.

As can be seen in Figure 3.19, one of the main consequences of an early curiosity stopping criterion (specifically, the one imposed in `CC_CC_sh_action_1000`, highlighted in brown) is that the skilled agent is unable to discover its optimal path, which is the corridor. Consequently, the policy's quality improves at a faster rate because the agent focuses on exploiting the known suboptimal path. However, the optimal policy requires traversing the corridor. Hence, the IM should be switched off when the agent has an intuition that it is the best option, after sufficiently exploring the environment, as demonstrated in `CC_CC_sh_action_3000`, represented in gray. This is reflected in Figure 3.19 where, despite prematurely turning off the

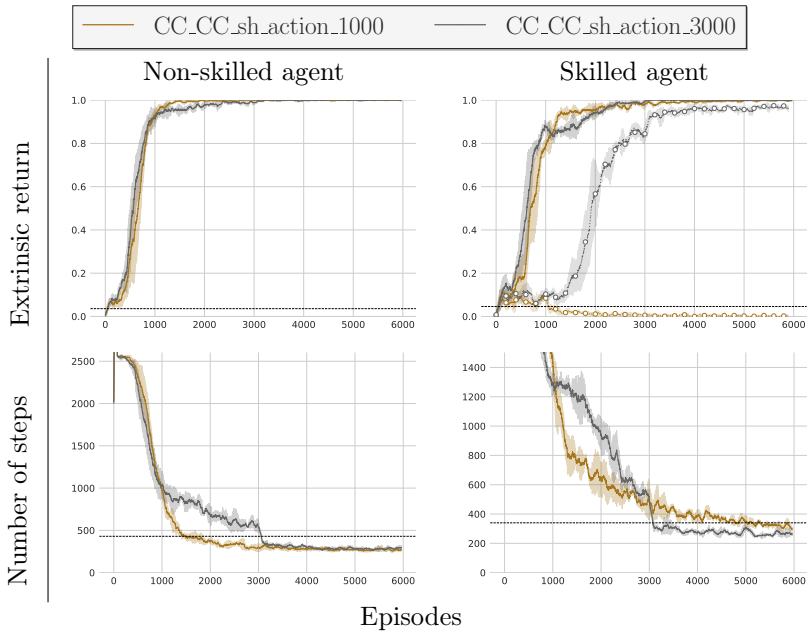


FIGURE 3.19: Performance comparison when switching off the intrinsic motivation stream at different stages of the training process (1000 and 3000 episodes) in Setup 3. Both the extrinsic return (top row) and the respective number of steps (bottom row) are shown. Dashed lines with markers are used to plot skilled agent's `_OPT` curves.

curiosity (i.e., `CC_CC_sh_action_1000`), resulting in a higher-quality policy at a faster rate, the policy obtained when going through the corridor (i.e., `CC_CC_sh_action_3000`) is ultimately better. As for the non-skilled agent, switching off the curiosity is less detrimental at any of those points because it has already discovered its optimal or near-optimal paths. Therefore, this agent simply needs to be encouraged to exploit the information and refine its policy to achieve slightly shorter routes.

3.5.2 Overall Comparison

To summarize the main outcomes from the aforementioned results, Table 3.3 provides an overall summary. For the sake of simplicity and clarity, it focus only on the results for Setup 3.

One intriguing observation is that the non-skilled agent consistently converges faster, which is expected due to its smaller action and solution space. Consequently, state value estimates – $V(s)$ – along the non-skilled agent's path receive more frequent updates and provide better estimates compared to states along the skilled agent's optimal path. Surprisingly, despite the low probability of successfully navigating the corridor once the skilled agent has discovered the optimal path (probability $\leq 10\%$ when SR $\approx 100\%$, as shown in Figure 3.14 and Figure 3.15), IM plays a significant

TABLE 3.3: Number of required episodes needed by each evaluated configuration in Setup 3 to achieve a SR equal to 90% (first column) and a SR equal to 80% when success is counted as such when the trajectory includes traversing the corridor (second column); quality of the trained policies in terms of the required number of steps to achieve the goal (statistics computed over the last 100 training episodes, removing those episodes in which the agent does not reach the target).

Algorithm	Number of required episodes				Number of steps to goal	
	90% SR		80% SR (corridor)			
	skilled	non-skilled	skilled	non-skilled	skilled	non-skilled
IC_IC_3r	≥ 6000	≥ 6000	≥ 6000	-	-	-
IC_IC_6r	≥ 6000	≥ 6000	≥ 6000	-	-	-
CC_IC	1522	1110	3668	-	410.72 ± 12.15	470.98 ± 19.15
CC_CC_sh	1309	1026	3445	-	403.07 ± 9.25	455.88 ± 5.05
CC_CC_sh_action	1412	1007	2389	-	456.81 ± 6.24	530.27 ± 8.47
CC_CC_sh_action_filter	1208	858	2378	-	418.47 ± 7.91	492.2 ± 21.06
CC_CC_sh_action_1000	1213	937	≥ 6000	-	314.45 ± 7.17	267 ± 3.17
CC_CC_sh_action_3000	1722	886	2472	-	267.9 ± 7.57	294.64 ± 3.44

role in maintaining exploration until the agent discovers an even better solution. This increases the chances of successfully navigating the corridor, highlighting the importance of maintaining a certain level of exploration to adequately explore all possible alternative paths. This finding is further supported by the analysis of `CC_CC_sh_action_1000` in Figure 3.19, where disabling the curiosity reward results in the skilled agent only learning the suboptimal path to the goal.

Analyzing the agents' ability to consistently reach the goal, the introduction of a centralized critic emerges as a significant differentiating factor. Additionally, employing an IM strategy that considers both state and action leads to better convergence along the optimal path through the corridor for the skilled agent. This results in a reduction of approximately 30% in the number of training episodes required to achieve a 80% SR trough the optimal path. In the same line, the performance of the non-skilled agent also improves by introducing centralized curiosity and further improves with action-based IM, reducing the total number of episodes needed for a 90% SR by almost 10%.

Once the agents are capable of reaching the destination consistently, they begin to improve their learned policies, resulting in a decreasing number of required steps. Notably, experiments maintaining curiosity throughout the entire training achieve a performance close to that of a human-driven agent, even with a frameskip of 4. However, the skilled agent generally requires fewer steps to reach the target compared to the non-skilled agent, as the skilled agent follows a shorter path. However, this is not the case for `CC_CC_sh_action_1000`, where the non-skilled agent achieves the goal faster due to the combined effect of achieving a 100% SR more quickly (thus having more time to refine its learned policy) and having a less diverse action space (i.e., lower entropy across the entire action space).

Overall, the results suggest the following key observations:

- The state, action and solution space strongly influence convergence

speed. Combining agents with different space sizes can bias their behavior, limiting their exploration.

- To mitigate the undesired impact of excessive exploration, the β parameter should be adjusted during training.
  - Curiosity enhances the exploration stage, which is crucial in the analyzed scenarios so that the skilled agent finds out and learns its optimal solution.
  - Deactivating curiosity promotes the exploitation stage, potentially leading to faster convergence.

3.6 Conclusions

In this chapter, we have analyzed different ways in which heterogeneous RL agents can share information with each other about the same environment, taking into account that the achievement of their results is unique and exclusive to each agent. The main goal for both agents is to learn faster than they would independently on their own, without any kind of knowledge sharing strategy. To address this problem and arrive at informed conclusions, we have proposed multiple strategies to build a collaborative learning framework and we have carried out an extensive experimentation over a modified first-person-view environment (i.e., ViZDooM, with observation in shape of images) in order to asses which configuration is the best in terms of *what* has to be shared between heterogeneous agents (critic, curiosity) and *when* it has to be done. Several main conclusions have been verified experimentally:

- First, a centralized critic yields better stability and quicker convergence to an optimal solution compared to the case when critics are not shared.
- Secondly, the exploration between agents can be affected depending on whether the novelty is shared: centralizing it entails some minor advantages (even negative if the critic is not also shared), but the differential impact was noted when the novelty is set dependent not only on the state, but also on the action. This interesting result had gone unnoticed in past literature dealing with single-agent environments (Tang et al., 2017), where it was reported that a dependence on the action had no effect in the convergence of the learning process. However, in environments with heterogeneous agents, our results show the opposite: an action-dependent novelty computation can have a significant impact when discovering the optimal solution, achieving a reduction of up to 30% in terms of the number of training episodes.
- Finally, the intrinsic bonus elicits a bi-objective problem that favors exploration yet after a certain moment it induces too much noise into the training, degrading the effectiveness of the exploitation phase and

preventing from getting an optimal policy (remains too stochastic). This aligns with other works where, once a certain degree of knowledge has been obtained and the exploration is already considered sufficient, the fact of continuing to use it results to be counterproductive for the learning process (Rosser & Abed, 2021; Taïga et al., 2020).

3.7 Lessons Learned & Future Work

Grounded on the insights extracted from the experiments and the analysis of the results, in this section we sketch learned lessons and interesting directions for future research. Some of the reflections offered in what follows relate to the heterogeneity between agents, whereas others relate to issues that lie at the conjunction of both RL and IM.

3.7.1 When to Explore? Exploration-Exploitation Dilemma with Heterogeneous Agents

A well-known challenge in RL is about deciding when to explore and when to exploit in single agent scenarios. Besides the strong dependence on the characteristics of the environment, there are different types of exploration strategies that can be followed with diverse results (Pîslar et al., 2022). Even in the simpler single-agent scenario, it is not clear how to make the agent explore efficiently. In other words, *when should a given agent explore?* This question, often regarded as the exploration-exploitation dilemma, is yet unsolved, as it is not straightforward to determine when the agent (or even a human) has explored enough when learning to solve a task. This problem is exacerbated in settings with sparse rewards, specially when the completion of the task can require long-term training horizons.

It has been seen in this chapter that one way to deal with exploration is to use IM techniques, with which the agent can explore the environment more smartly. However, this approach introduces a non-stationary novelty bonus, yielding a bi-objective problem with conflicting objectives: the main task's extrinsic goal and the exploration-related intrinsic goal. As consequence, a misalignment between these objectives can emerge, potentially leading to worse results that not using the aforementioned intrinsic streams whatsoever (Taïga et al., 2020).

In the considered concurrent learning problem the heterogeneous agents do not share anything (by default) as opposed to the assumptions made in multi-agent RL problems, where they share at least a team reward or the environment where they are deployed.

Should we impose a collaborative strategy when none of the actions executed by an agent influence in the other agents behavior?

It is complex to give an answer, and particularly if we do not know when an agent (independently of other agents) has explored enough on a given task,

as depicted in the previous paragraphs. Therefore, in the current chapter, we have assumed some kind of latent knowledge between agents and tasks[17] that have been formalized in terms of sharing the critic and curiosity module. We further assumed that both agents understand and perceive the environment in similar ways, which can be translated into developing congruent representations and exploration patterns, which, ultimately, can help bootstrapping the learning of the involved agents. Unfortunately, this might not be realistic in other RL scenarios.

3.7.2 Detachment-Derailment Problem

Solutions that rely on IM techniques exhibit the so-called *detachment-derailment* problem. This issue arises when an agent has explored the environment correctly, becoming close to discovering an interesting state space or to achieving the goal. At some point, however, the agent's learning gets stuck and the episode finishes. When the next episode is started, all decisions that the agent made to reach those spots are now regarded with less novelty (even being close to finding out promising locations). Consequently, the agent will be stimulated to examine other alternatives, even if it was in the right direction to discover novel states, degrading the effectiveness of the exploration. In this chapter, we realized that the *detachment-derailment* problem gets worse when the time horizon required to achieve any meaningful feedback signal increases.

Recently, it has been shown that an effective way to address this issue is by clustering representations, and by reinitializating the agent smartly in the environment (Ecoffet et al., 2021; Ugadiarov et al., 2021). However, these approaches require the environment to be *reset-free*[18]. In the scenario with heterogeneous agents tackled in this chapter, a similarity-based clustering of the state space might be suitable to identify promising states where the agent can be reset (Ecoffet et al., 2021; Ugadiarov et al., 2021). Unfortunately, it is difficult to make these techniques work in POMDPs with first-person-view observations due to (1) the dimensionality reduction of the state space, and (2) the generation of clusters and the determination on where (i.e., in which cluster of states) to reinitialize each agent considering that they might have different stimuli and optimal paths for the same goal.

In spite of the difficulty of implementing adequate mechanisms to deal with this phenomena is high, analysing and developing procedures to keep track of previous not-fully explored, albeit promising, routes, could complement IM techniques and make them efficient even in extraordinary complex circumstances

[17] Akin to the hypothesis behind Transfer Learning approaches.

[18] An environment in which the agent position and/or state perception can be manually selected without any constraints. This property grants flexibility to select new/desired start positions arbitrarily.

3.7.3 Potential of Recurrent Rewards

Another issue encountered during this research springs from the fact that intrinsic bonuses are generated from a given experience tuple rather than a sequence of tuples. This issue affects not only the scenario tackled in this chapter, but also other RL environments that generate intrinsic rewards based on single experiences. This mainly occurs when having a POMDP as changes in the environment cannot be directly reflected even if those changes have a clear impact in the environment. Next, we expose this problem by briefly discussing on two hypothetical environments.

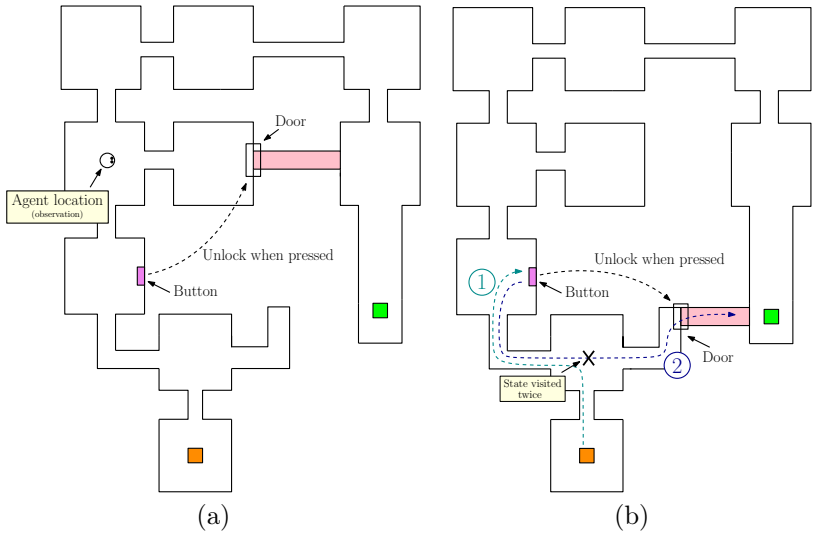


FIGURE 3.20: Hypothesized case studies to discuss on how to deal with long-term dependencies within sparse POMDP problems.

In the environments shown in Figure 3.20, the agent can unlock the colored passage by pushing the button that is located at a different location, relatively far from the entrance to the corridor. For this purpose, an action namely `open` is available by the agent but is useless anywhere else except in front of the door. In these environments a first-person-view observation hinders the agent from understanding the correlation between pushing the button and opening the door. What is more, the value of reaching the location where the button is located (and all the subsequent states to the destination) will differ depending whether:

- The button is pushed and the agent goes through the corridor.
- The button is pushed and the agent does not go through the corridor.
- The button is not pushed.

This issue, combined with long horizon returns and an agent that does not know how to interact and solve the problem correctly, leads to noisy updates and hampers the discovery of the correlation existing between the

button and the door. This is even more complex in scenarios as the one in Figure 3.20.b, where a given observation (e.g., the one marked with an X) must be visited twice: (1) when searching for the button that opens the passage, and another (2) to go through the passage itself[19].

Due to these inconsistencies, we believe that *novelty* needs to be redefined in one of the following two ways:

- As the intrinsic reward for a given experience tuple, aiming to quantify *how novel the experience is on its own.*
- As the discounted expected return within a given trajectory,considering the calculated intrinsic bonus or the experiences that make up that specific trajectory, answering *which degree of novelty this experience injects into future steps of the episode.*

The first definition relies solely on the experience itself to measure novelty. It is more practical and widely adopted in the research community. Nevertheless, this requires the temporal dependencies among the experiences to be modeled manually (e.g., stacking multiple instance frames, using memory mechanisms) or incorporating recurrent (and/or attention) modules at the actor, the critic or both (Hausknecht & Stone, 2015; Oh et al., 2016; Vaswani et al., 2017). In fact, in the architectures discussed in the experiments of this chapter, one of the algorithmic configurations adopted a LSTM-based neural architecture in the critic. However, there are no guarantees that this type of architecture retains the gathered knowledge at long-term horizons, nor is the novelty score used to compute the return stationary (it decreases over time). This instability in the expectation term over time ultimately hampers the long-term modeling capabilities of the recurrent/attention modules within ANN.

Alternatively, a solution could be to generate intrinsic rewards based not only on the current time step, but also on past experiences (i.e. a sequence of experiences, second definition). This is, designing a reward function that handles the temporal dependencies and provides a different reward value, so that an experience is determined to be novel taking into account a full episode or path with its inherent consequences. This problem has also been recently showcased in relation to goals in (Colas et al., 2022), opening a debate around how to address this problem in an online fashion with no previous knowledge about the environment. This discussion finds in the action heterogeneity of agents studied in this chapter another twist of its screw.

[19]Recall the agent is only provided by a first-person-view input; therefore, the same observation can receive different values estimates depending whether the button was previously pushed or not.

Chapter 4

An Evaluation Study of Intrinsic Motivation Techniques applied to Reinforcement Learning over Hard Exploration Environments

The claimed effectiveness of IM techniques in environments with sparse rewards has been proven in the previous chapter, when applied either collaboratively or independently in multi- and single- agent problems. Experiments performed in the previous chapter, which considered RND and count-based strategies to compute the intrinsic rewards, showcased the large amount of IM approaches that can be adopted to foster the exploration by combining the produced intrinsic signal with its extrinsic counterpart (e.g. as in Expression (2.25) or Expression (3.2)).

In this context, modern IM solutions (Badia, Sprechmann, et al., 2020; Raileanu & Rocktäschel, 2020; Seurin et al., 2021; T. Zhang et al., 2020) solutions propose not only their own method to calculate the exploration bonus, but also introduce other operations to weight and scale the magnitude of their generated intrinsic rewards. Table 4.1 lists several of such IM methods, building upon the early studies focused on the generation of curiosity information (Bellemare et al., 2016; Burda, Edwards, Storkey, et al., 2018; Pathak et al., 2017). Unfortunately, as per the current literature it remains unclear whether the research race towards superior IM methods is mainly driven by the proposed reward generation approach or instead, biased by other design choices, such as different base RL algorithms, decay of the exploration bonus, episodic scaling techniques adoption, neural network architectures and benchmarks for the evaluation of results.

Analogously to other studies in the field of RL (Andrychowicz et al., 2021a; Andrychowicz et al., 2021b; Henderson et al., 2019; Orsini et al.,

TABLE 4.1: Classification of various IM methods based on different design choices. We provide the parameters with which those approaches have been evaluated in the MiniGrid benchmark, except for NGU (Atari).

	Ref	RL-algorithm	Vary β_i	Scale r_i	ANN architecture
ICM	(Pathak et al., 2017)	IMPALA	✗	✗	Shared AC [3CNN,256LSTM,FC]
RND	(Burda, Edwards, Storkey, et al., 2018)	IMPALA	✗	✗	Shared AC, [3CNN,256LSTM,FC]
RIDE	(Raileanu & Rocktäschel, 2020)	IMPALA	✗	✓	Shared AC, [3CNN,256LSTM,FC]
BeBold	(T. Zhang et al., 2020)	IMPALA	✗	✓	Shared AC, [3CNN,256LSTM,FC]
DoWhaM	(Seurin et al., 2021)	IMPALA	✗	✓	Shared AC, [3CNN,1024LSTM,1024FC]
RAPID	(Zha, Ma, et al., 2021)	PPO	✗	✗	Independent AC, [2FC64]
AGAC	(Flet-Berliac et al., 2021)	PPO	✗	✓	Independent AC, [3CNN,512FC]
D&E	(Jing et al., 2021)	PPO	✓	✓	Independent AC, [3CNN,512FC]
NGU	(Badia, Sprechmann, et al., 2020)	R2D2	✓	✓	Single Q(s,a,β), [4CNN,512LSTM,512FC]

2021), a fundamental matter is to distinguish which design criteria are actually important and their impact on the performance of the agent. This is specially relevant in hard exploration environments, since it is known that under such circumstances, the proficiency of the agent is very sensitive w.r.t. the configuration of its compounding modules. For this reason, the goal of this chapter is to perform a fair evaluation of IM-based solutions present in the literature, aiming to decouple the contribution of the IM approach to the overall performance of the agent from the impact of additional design choices. As a result, insights will be given about which design choices matter when designing IM mechanisms, so that these approaches can be adapted and used in new RL problems thoughtfully.

4.1 Related Work

Before digging into the contribution of this chapter chapter, we first briefly review the concepts in which some IM solutions support their curiosity mechanisms.

Intrinsic Motivation

As we have already explained in Section 2.3.1 of Chapter 2, two main groups of IM algorithms can be found in the literature: *count-based* and *prediction-error* methods. The firsts calculate the reward inversely proportional to the number of times $N(s_t)$ a given state (s_t) has been visited:

$$r_t^{counts} = \frac{1}{\sqrt{N(s_t)}} \tag{4.1}$$

This idea can be also extended to other visitation count approaches that are suitable for high-dimensional state domains (Bellemare et al., 2016; Machado et al., 2019; Ostrovski et al., 2017; Tang et al., 2017).

On the other hand, *prediction-error* methods generate the exploration bonus taking into account the ability of the method to reliably predict changes in the environment. In order to accomplish it, ICM (Pathak et al., 2017) proposed a framework to calculate the difference between the actual next state (s_{t+1}) and a prediction of the next state taking into account the

current state and action, $\widehat{s}_{t+1} = f(s_t, a_t)$, being f the function that will learn the dynamics of the environment. Even more importantly, instead of calculating the error directly with the raw input state, in ICM a latent representation $\phi(\cdot)$ is learned to capture only the information that affects or is affected by the agent (preventing irrelevant features of the state space from biasing the prediction):

$$r_t^{ICM} = ||\widehat{\phi}(s_{t+1}) - \phi(s_{t+1})||_2 \tag{4.2}$$

where $||\cdot||_2$ stands for the L_2 (Euclidean) norm and $\widehat{\phi}(s_{t+1})$ represents the prediction of the s_{t+1} taking into account $\phi(s_t)$ and the actual action a_t as input; that is, $\widehat{\phi}(s_{t+1}) = f(\phi(s_t), a_t)$. Please refer to Figure 2.9 for better clarity.

Upon the idea of how state embeddings are learned, RIDE (Raileanu & Rocktäschel, 2020) proposed to calculate the exploration bonus by the difference between two consecutive states in their latent space:

$$r_t^{RIDE} = ||\phi(s_{t+1}) - \phi(s_t)||_2 \tag{4.3}$$

With this change, RIDE encourages the agent to perform actions that have an impact on the environment. The modification with respect to ICM can be seen in Figure 4.1[1].

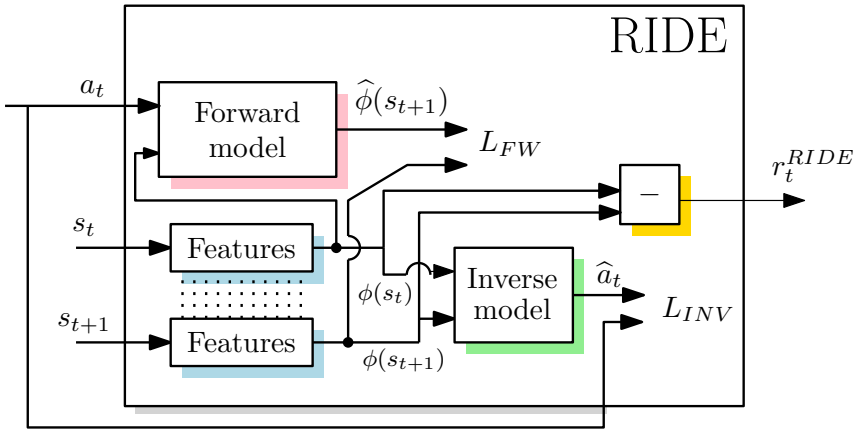


FIGURE 4.1: RIDE framework (Raileanu & Rocktäschel, 2020) to generate the intrinsic reward.

What is more, to ensure that the agent does not go back and forth between a sequence of states in order to get intrinsic rewards, the reward is discounted by the episodic state visitation counts:

$$r_t^{RIDE} = \frac{||\phi(s_{t+1}) - \phi(s_t)||_2}{\sqrt{N_{ep}(s_{t+1})}} \tag{4.4}$$

[1]Note that the forward model is now just used to build a better approximation of the feature space in the same way as the inverse model does.

so that the bonus now is calculated by combining *experiment-* and *episode-level* exploration (Pîslar et al., 2022; Stanton & Clune, 2018). Similar but more aggressively, in BeBold/NovelD (T. Zhang et al., 2020, 2022) the reward was restricted so that only the first time the agent visits a given state in an episode was valid:

$$r_t^{BeBold} = \max\left(\frac{1}{N(s_{t+1})} - \frac{1}{N(s_t)}, 0\right) \cdot \mathbb{I}[N_e(s_{t+1}) = 1] \tag{4.5}$$

where $N_e(\cdot)$ stands for the episodic state count that is reset every episode, and $\mathbb{I}[\cdot]$ is an indicator function taking value 1 if its argument is true (0 otherwise).

Following the idea of combining various degrees of exploration, NGU (Badia, Sprechmann, et al., 2020) calculated the intrinsic reward as the combination of two sub-rewards:

$$r_t^i = r_t^{episodic_i} \cdot \min\{\max\{r_t^{lifelong_i}, 1\}, 5\} \tag{4.6}$$

being $r_t^{episodic_i}$ calculated through an episodic memory (Pritzel et al., 2017) and $r_t^{lifelong_i}$ computed across the whole training. In addition, NGU adopted an UVFA (Schaul et al., 2015) framework so that the employed action-value function was subject to different β coefficients, $Q(s_t, a_t, \beta)$, which allows learning policies with different explorative behaviors using a single network. Last but not least, FaSo (Bougie & Ichise, 2021) combined local and global exploration by generating two different intrinsic rewards, depending on the quality of the reconstruction of two contexts built from the same state.

Aside from the method to calculate the exploration bonus itself, new IM solutions are shown to yield better results in their respective publications, yet using additional components which were not used when compared to the selected baselines. Thus, rather than proposing a new intrinsic generation module, in this chapter we carry out an evaluation study to gauge the impact of such modifications (Table 4.1) and to ascertain the contribution of the IM reward generation to the overall performance of the agent.

Reinforcement Learning Studies

Other benchmarks/studies have been done in recent times surrounding RL: to begin with, (Taïga et al., 2020) evaluates the performance of different exploration bonuses (pseudo-counts, ICM, RND and noisy networks) in the whole Atari 2600 suite with Rainbow (Hessel et al., 2017). By contrast, (Burda, Edwards, Pathak, et al., 2018) carried out a large-scale study based exclusively on prediction error bonuses over 54 environments, where they investigated the efficacy of using different feature learning methods with PPO (Schulman, Wolski, et al., 2017). This chapter also connects with (Andrychowicz et al., 2021a; Andrychowicz et al., 2021b; Henderson et al., 2019; Orsini et al., 2021), a series of evaluation studies aimed to understand what choices among high- and low-level algorithmic options affect

the learning process. As such, the studies in (Andrychowicz et al., 2021a; Andrychowicz et al., 2021b) focus on on-policy deep actor-critic methods (examining different policy losses, architectures and advantage estimators). On the other hand, (Orsini et al., 2021) addresses adversarial IM related decisions (multiple reward functions and observation normalization methods), whereas (Henderson et al., 2019) investigates reproducibility issues using different random seeds, activation functions, codebases, and reward scales, among other experimental choices.

Contribution

To the best of knowledge, there is no prior work that exhaustively evaluates different choices for the implementation of intrinsic motivation strategies. The study presented in this chapter of the Thesis takes a step further by analyzing different weight and scale strategies for the combination of intrinsic and extrinsic rewards, as well as the impact of adopting different neural networks architectures and its dimensions. The design choices here evaluated are applicable to any intrinsic curiosity generation module, so that conclusions about which ones are the most suitable given a task and an environment with sparse rewards can be drawn.

4.2 Methodology of the Study

After reviewing different solutions proposed in the literature to cope with hard exploration issues with IM techniques, we now proceed by describing the methodology adopted in this chapter to gauge the advantages and drawbacks of design choices that are present in some of them, giving an informed hint of their utility when extrapolated to the rest of IM solutions. The methodology is driven by the pursuit of responses to three research questions (RQ):

- RQ1: Does the use of a static, parametric or adaptive decaying intrinsic coefficient weight β affect the agent's training process?
- RQ2: Which is the impact of using episodic counts to scale the intrinsic bonus? Is it better to use episodic counts than to just consider the first time a given state is visited by the agent?
- RQ3: Is the choice of the neural network architecture crucial for the agent's performance and learning efficiency?

Departing from these questions, the following methodology has been devised:

4.2.1 RQ1: Varying the Weight of the Intrinsic Reward Coefficient β

In general, it is not advisable to combine raw extrinsic and intrinsic reward signals directly due to their potentially diverging value scales. Moreover,

even if taking values from comparable ranges, the agent could need to grant more importance to exploration than to exploitation at specific periods. In fact, in sparse rewards settings, the explorer role of the agent must be strengthen and enlarged in comparison to the exploitative behaviour to guide the agent by an artificial bonus in the absence of knowledge about the target task. This balance between exploration and exploitation is usually controlled by the intrinsic reward coefficient β, whose value is often tuned manually depending on the environment and task to be accomplished. A priory, this value might be fixed and kept unaltered, or dynamically updated, as is further explained in what follows:

`Static` $\boldsymbol{\beta}$`:` commonly, the β coefficient is stationary along the whole training. In such cases, we refer to this fixed and default value as β_s (*static*). On this basis, diverse fixed intrinsic coefficient values can be used to learn a family of policies with different exploration-exploitation balances, so as to concentrate on maximizing the extrinsic reward (a policy with $\beta = 0$) while maintaining a degree of exploration (rest of policies with $\beta > 0$) (Badia, Sprechmann, et al., 2020). Contrarily to the rest of approaches, when using multiple (fixed) intrinsic coefficients, training more than one agent is required.

`Dynamic` $\boldsymbol{\beta}$`:` to focus on the extrinsic signals provided by the environment, it is interesting to modulate the weight given to the intrinsic rewards generated by the agent in a dynamic fashion. Without loss of generality, we herein consider two options: parametric decay and adaptive decay. For the *parametric* decay, the value of β decreases by following a modified sigmoid function that controls the smoothness of the decay:

$$\beta_t = A + \frac{K - A}{\left(1 + \exp\left(-16B\left(1 - \frac{t}{F}\right)\right)\right)^{20}} \tag{4.7}$$

where K is a value proven to deliver a good performance and well-balanced trade-off between exploration and exploitation (e.g., the fixed value β_s that one could select under a fixed β strategy); A is the final value of β, which can be defined from K (e.g. $A = K/100$) to reflect that at the end of the learning process, the agent should receive hardly any intrinsic signal bonus; and F denotes the number of frames (equivalently samples or steps) we expect the whole train to take. Moreover, B permits to control the *smoothness* of the progression of β throughout the training process (Figure 4.2). We note that this parametric decay can also be used to sample different β values for each policy learned by means of the approach with multiple static intrinsic coefficients β, by defining F as the number of agents.

In turn, we can vary the intrinsic coefficient by adopting an `adaptive decay strategy`. Motivated by (Jing et al., 2021) for concurrent environments, we propose to calculate a decay factor d_τ based on the ratio between the agent's intrinsic return at the current rollout, G^i_τ, and the

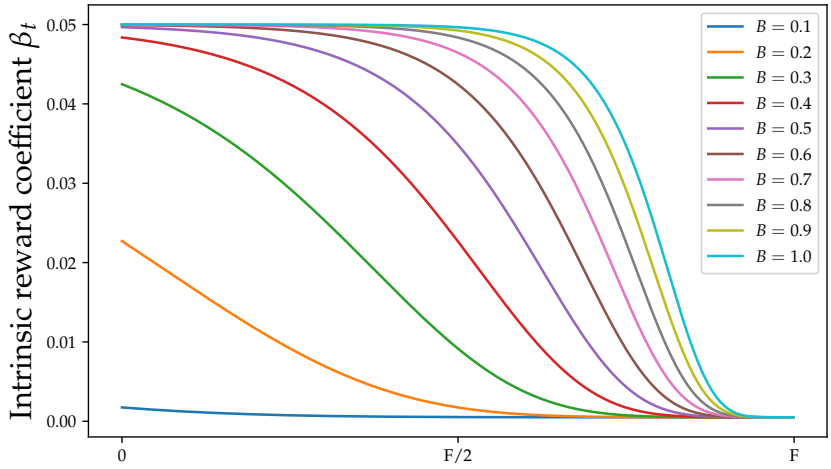


FIGURE 4.2: Example of the parametric decay evolution of β_t for multiple values of the smoothness control parameter B, with $K = 0.05, A = 0.0005$ and $F = 2e7$.

averaged historical intrinsic return values in past rollouts H_J[2]:

$$\beta^i_\tau = \beta_s d_\tau = \beta_s \min\left[\frac{G^i_\tau}{H_J}, 1\right] = \beta_s \min\left[\frac{G^i_\tau}{\frac{1}{J}\sum_{j=0}^{J} G^i_{\tau_j}}, 1\right] \tag{4.8}$$

Consequently, under this rationale the agent is discouraged from exploring those trajectories that are more familiar than the average and means less novelty.

In fact, the intrinsic return during the training may vary due to the non-stationary nature of the intrinsic reward generation process. Thereby, to stabilize the training, instead of leveraging the whole historical data, we also propose the use of a moving average with a sliding window, H^ω_J, which strictly considers just the latest returns (ω) and avoids the case of discouraging the exploration due to previous intrinsic returns that may well bias the decay factor calculation:

$$\beta^i_\tau = \beta_s d_\tau = \beta_s \min\left[\frac{G^i_\tau}{H^\omega_J}, 1\right] = \beta_s \min\left[\frac{G^i_\tau}{\frac{1}{\omega}\sum_{j=J-\omega}^{J} G^i_{\tau_j}}, 1\right] \tag{4.9}$$

[2]By default, τ is used to refer to the current rollout. Akin to t that is used to refer to time steps, here j is adopted for the accountability of rollouts (i.e., an ensemble of consecutive time steps, recall Section 2.1.2 in Chapter 2) gathered at different points of the training. As such, J is used to refer to the number of total rollouts gathered during training so far, and τ_j to those experiences collected at a specific checkpoint.

4.2.2 RQ2: Scaling the Intrinsic Rewards Episodically

As defined in (Pîslar et al., 2022), there are different periods in which the exploration mode can be carried out: *step-level*, *experiment-level*, *episode-level* and *intra-episodic*.

Over the years the use of *step-level* exploration (i.e. ϵ-greedy) has proven to yield good results in a diversity of simple RL environments. However, advances in learning algorithms have paved the way towards RL problems of higher complexity, requiring more sophisticated strategies such as the IM mechanisms under target in this chapter. The intrinsic rewards generated with IM methods are prone to suffer a quick vanishing of their value/magnitude over the course of the training, reducing the attractiveness as the training evolves. This condition is exacerbated when facing long-time horizon problems (Bougie & Ichise, 2021). Actually, by analyzing the rewards obtained during a concrete episode, few differences in terms of novelty are appreciated between similar/close states, even if one has been already visited and the other remains unexplored. This is due to the persistence of curiosity-related information from past episodes (*experiment-level*), which is propagated forward during the agent's training, leaving little novelty difference between similar (even identical) states inside the scope of the same episode. Additionally, in environments where state transitions are reversible, using intrinsic rewards to guide the exploration can lead into an agent bouncing back and forth between sequences of states that are more novel than others in the same episode (Raileanu & Rocktäschel, 2020; T. Zhang et al., 2020).

As a solution to this issue, recent studies (Badia, Sprechmann, et al., 2020; Bougie & Ichise, 2021; Raileanu & Rocktäschel, 2020; Seurin et al., 2021; T. Zhang et al., 2020) combine two degrees of novelty rather than just one: local (*episode-level*) and global (*experiment-level*). More concretely, (Raileanu & Rocktäschel, 2020) introduced an episodic visitation count term to encourage the agent to visit as many different states as possible within an episode. Similarly, (T. Zhang et al., 2020) incorporated a more aggressive variation that rewards the agent only when it visits a given state for the first time within an episode. However, approaches at the forefront of the state of the art (e.g. ICM, RND) do not implement these ideas to scale their rewards. In this context, it is unclear whether new proposed IM modules outperform previous approaches due to episodic state-count regularization or to conceptually new algorithmic schemes. If such regularization contributed to improve the performance, already proposed IM schemes that do not implement it (and also future IM methods) could adopt this strategy to meliorate their designs.

4.2.3 RQ3: Sensitiveness of Neural Network Architectures

In the literature surrounding RL, plenty of network architecture proposals have been used to solve any given problem. As an example, the work in (Zha, Ma, et al., 2021) simplified the architectures previously proposed

in (Raileanu & Rocktäschel, 2020) yet achieving similar results[3] in the same environment and task. Moreover, those approaches rely on different base RL algorithms – PPO (Schulman, Wolski, et al., 2017) and IMPALA (Espeholt et al., 2018) – thereby hindering a fair comparison, a proper interpretability and attribution of the reported performance results.

In order to minimize possible misunderstandings, our specific experimentation evaluates the effect of the network architecture on the performance of the RL agent by considering a fixed RL algorithm and IM module, and by assessing various network configurations. By reporting the dimensions and characteristics of different neural network architectures and the performance of RL agents using them, we can gain intuition about the performance improvement (degradation) incurred when increasing (decreasing) the complexity of the neural architectures in use. Our experiments also measure the required amount of time when using those architectures, so that latency implications can be examined. This third research question is also aligned with practical concerns arising when deciding on which implementation is more suitable for a real-world deployment, specially in resource-constrained scenarios (e.g. embedded robotic devices).

4.3 Experimental Setup

We give answer to the research questions over procedurally generated RL tasks from the Minimalistic Gridworld Environment (MiniGrid (Chevalier-Boisvert et al., 2018)). As detailed in Section 2.2.1 of Chapter 2, this framework allows creating tasks of varied levels of difficulty, allows the use of either image or compact representations while selecting any MDP or POMDP setup. Most importantly, it runs fast, thereby easing the implementation of massive RL benchmarks. Last but not least, we recall that MiniGrid is a PCG benchmark, hence the agent will face, for the same task, a different level in each episode as depicted in Figure 2.7. This requires an agent capable of generalizing to newly produced unseen environment realizations.

4.3.1 Environments

To design a representative benchmark for the study, among all the possible RL environments that can be selected/generated in MiniGrid, we consider 1) those labeled as `MultiRoomNXSY` (shortened as `MNXSY`, with `X` denoting the number of rooms and `Y` their size), 2) `KeyCorridorS3R3` (`KS3R3`); and 3) `ObstructedMaze2Dlh` (`O2Dlh`). These scenarios belong to hard exploration tasks (i.e., rewards are sparse), in which the agent fails to complete the task without the help of any IM mechanism. We refer to Figure 4.3 for further information about each scenario and its associated goal.

By default, we adopt a POMDP setup where the observations are essentially egocentric and partial views of the environment. A 7×7 tile set

[3]We note that the choice of the neural network architecture is not just for the actor-critic modules, but also for IM approaches that hinge on neural computation.

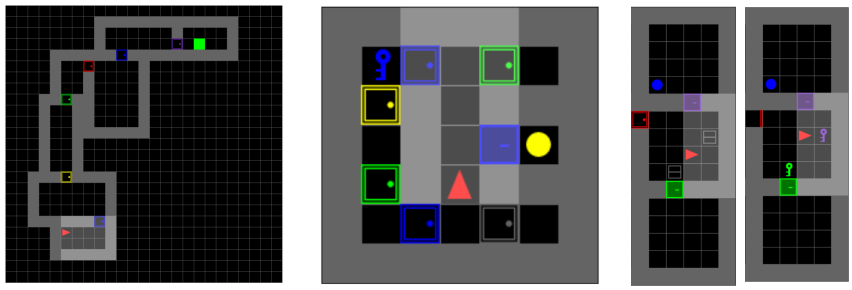


FIGURE 4.3: Examples of MiniGrid scenarios. `MN7S8`: the agent has to open multiple doors to reach the distant goal (green square); `KS3R3`: the agent has to first collect the blue key in order to open the blocked door of the room leading to the yellow ball that must be picked `O2Dlh`: the agent has to discover keys hidden below some boxes, take the proper key and open the door to the blue ball (target).

in the direction that the agent is facing composes the observation. Concretely, an observation is featured by a $7\times7\times3$ matrix, being the 3 features of the last dimension information of interest such as type, colour and status of the object (e.g., doors, keys, balls, or walls) placed in the specific tile. The agent is incapable to see through walls or doors. Seven basic actions are available to solve all scenarios: `turn left`, `turn right`, `move forward`, `pick up` (an object, for instance keys or balls), `drop` the object (if carried), `toggle` (open doors, interact with objects) and `done`. Nevertheless, some of these actions are only useful at specific locations, whereas others become useless for certain tasks (for instance, `pick`/`drop` and `done` in `MNXSY` environments).

Not all the environments require the same amount of steps to be solved. For instance, in `MNXSY` environments, a maximum number of $20 \cdot \texttt{X}$ steps is set before resetting the episode to make it dependent on the number of rooms `X`. Hence, the three considered environments that fall within this set (`MN7S4`, `MN7S8` and `MN10S4`) are assumed to take at most 140, 140 and 200 steps, respectively. For `KS3R3` and `O2Dlh`, 270 and 576 steps are set as maximum.

As formulated in Expression (2.23), the expected rewards depend on the above maximum number of steps, resulting in a different return for each task. In fact, since the environments are PCG, even within the same task, each level's optimal return can differ from each other, as their configuration does not guarantee that all levels can be solved at an equal number of steps. Hence, we set the following optimal average extrinsic returns by taking the obtained median score when evaluating an optimal policy[4]: 0.77 (`MN7S4`), 0.76 (`MN10S4`), 0.65 (`MN7S8`), 0.9 (`KS3R3`), and 0.95 (`O2Dlh`). Moreover, we also refer as suboptimal behavior to those policies that managed to reach at least 95% of the optimal score corresponding to each task.

[4]Equal to other previously reported results (T. Zhang et al., 2020, 2022).

Finally, in terms of complexity, `MN7S4` and `MN10S4` are assumed to be the easiest ones to solve, followed by `MN7S8` and `KS3R3`, which are harder, and `O2Dlh`, which is the most difficult task in the present study.

4.3.2 Baselines

All the experiments employ PPO (Schulman, Wolski, et al., 2017) as the RL algorithm in use. On top of it, state-of-the-art IM techniques are used in order to obtain intrinsic rewards that augment the exploration efficiency of the agent, considering that a naive PPO solution has been shown to be insufficient for learning the task (Zha, Ma, et al., 2021). Hence, we analyze COUNTS[5], RND (Burda, Edwards, Storkey, et al., 2018), ICM (Pathak et al., 2017), and RIDE (Raileanu & Rocktäschel, 2020).

Hyperparameters

Regarding PPO, we use a clipping factor $\epsilon = 0.2$, 4 epochs per train step, a discount factor γ equal to 0.99 and $\lambda = 0.95$ for GAE (as per Expression (2.18)). We use 16 parallel environments to gather rollouts of size 128. Hence, we set a total horizon of 2048 steps between updates. Moreover, a batch size equal to 256 is considered. Unless otherwise specified, the following values – selected from an offline grid search procedure over `MN7S4` – will be used to configure the intrinsic coefficient and entropy: $\beta = 0.05$ and $c_2 = 0.0005$ for RND, ICM and RIDE; $\beta = 0.005$ and $c_2 = 0.0005$ for COUNTS. In what refers to the dynamic update of the intrinsic coefficient β, we select $\boldsymbol{B} = 0.5$ in Expression (4.7) as it represents a balanced trade-off for the agent to explore in the early stages of the training process, evolving towards a behavior mainly driven by extrinsic signals.

Network Architectures

Finally, experiments devised to answer RQ3 are performed with two different neural network architectural designs, which differ in terms of the type of neural layers, their composition and the number of trainable parameters.

Following Figure 4.4, on one hand a *lightweight* neural architecture as in RAPID (Zha, Ma, et al., 2021) is considered, in which both the actor and the critic (independent) are made of 2 fully-connected layers (FC) with 64 neurons each. This dual FC-64 architecture also applies to the embedding networks required for RND, ICM and RIDE.

Additionally, we include a more sophisticated neural design based on what is proposed in RIDE (Raileanu & Rocktäschel, 2020), where both the actor and critic are combined into a two-headed (one for the policy, the other for the critic) shared network with 3 convolutional neural layers (32

[5]In this case, we take advantage of the 2D grid (discrete state space) and map each state directly to a dictionary when using COUNTS. Nevertheless, when facing more complex state spaces pseudo-counts (Bellemare et al., 2016) can be applied as an alternative as in (Taïga et al., 2020).

filters with kernel 3×3, stride equal to 2, and padding 1) and a FC-256 layer. Originally in (Raileanu & Rocktäschel, 2020) they used an LSTM of 256 units instead of a FC-256. We analyze the results with no recurrence despite being in an POMDP setting, which will also allow the comparison whether if it is actually necessary the use of recurrence modules in these environments. What is more, even if (Raileanu & Rocktäschel, 2020) defined the previously mentioned architecture design, in their GitHub implementation they seem to use larger networks (https://github.com/facebookresearch/impact-driven-exploration). This is the reason why in Table 4.1 we do not specify the FC units. This last architecture will be labeled as the *default* architecture to endow the agent with more learning capabilities and to ensure that it is not limited by a restricted network.

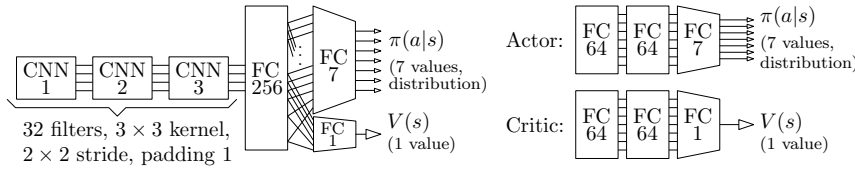


FIGURE 4.4: (a) Sophisticated/default and (b) lightweight network architectures.

4.4 Results and Analysis

In this section experimental results are presented and discussed towards answering the research questions posed in Section 4.2. Scripts and results have been made available in a public GitHub repository (https://github.com/aklein1995/intrinsic_motivation_techniques_study) to foster reproducibility and stimulate follow-up studies. For all the experiments described in this section we provide the mean and standard deviation of the average return computed over the past 100 episodes, performing 3 different runs (each with a different seed) to account for the statistical variability of the results.

4.4.1 RQ1: Does the use of a static, parametric or adaptive decaying intrinsic coefficient weight β affect the agent's training process?

Our first set of results compares the multiple weighting strategies introduced in Section 4.2.1, which differently tune the importance granted to the intrinsic rewards with respect to extrinsic signals coming from the environment.

The results are shown in Table 4.2. It is straightforward to note that RIDE outperforms COUNTS and RND. At this point we remind that

TABLE 4.2: Results of different IM strategies over several MiniGrid scenarios with static (*_s*), multiple static (*_ngu*) (as in NGU Badia, Sprechmann, et al., 2020), a parametric (*_pd*) or adaptive decay (*_ad*) weight β to modulate the importance of the intrinsic bonus in the computation of the reward. Cell values denote the training steps/frames ($1e6$ scale) at which the optimal average extrinsic return is achieved; between parentheses, steps at which 95% of the optimal average extrinsic return is reached. The best results for every (IM strategy, scenario) combination are highlighted in bold.

	MN7S4	MN10S4	MN7S8	KS3R3	O2Dlh
COUNTS_*s*	**0.93** (0.86)	1.87 (1.78)	> 30	> 30	> 50
COUNTS_*ngu*	1.17 (1.11)	2.67 (2.35)	> 30	> 30	> 50
COUNTS_*pd*	0.96 **(0.83)**	2.27 (1.67)	> 30	**22.91 (22.49)**	> 50
COUNTS_*ad*	1.03 (0.92)	**1.81 (1.65)**	24.23 (24.10)	> 30	> 50
COUNTS_*ad*1000	1.03 (0.92)	**1.81 (1.65)**	23.63 (23.56)	> 30	> 50
RND_*s*	3.83 (3.78)	7.84 (7.79)	> 30	10.83 (9.72)	> 50
RND_*ngu*	2.69 (2.62)	5.78 (5.75)	> 30	8.12 (7.50)	> 50
RND_*pd*	4.04 (3.94)	6.02 (5.99)	> 30	9.24 (8.07)	> 50
RND_*ad*	**2.02 (1.39)**	**3.21 (2.65)**	> 30	**6.02 (5.43)**	> 50
RND_*ad*1000	3.62 (1.42)	3.59 (3.50)	> 30	7.47 (6.66)	> 50
RIDE_*s*	**2.49** (1.82)	2.27 (2.14)	4.00 (3.68)	6.63 (4.39)	30.88 (25.87)
RIDE_*ngu*	3.85 (2.40)	2.59 (1.26)	> 30	7.18 (3.91)	36.07 (29.96)
RIDE_*pd*	5.20 (2.14)	5.01 (1.96)	**3.73 (3.49)**	6.42 (3.87)	29.27 **(20.84)**
RIDE_*ad*	2.89 **(0.91)**	**1.60 (0.99)**	> 30	5.93 (2.99)	**27.65** (20.91)
RIDE_*ad*1000	2.54 **(0.91)**	**1.60 (0.99)**	3.88 (3.70)	**4.70 (3.00)**	28.00 (23.01)

RIDE is configured with episodic count scaling, in accordance with the final solution proposed in (Raileanu & Rocktäschel, 2020). Count-based generated rewards seem to be the best solution when facing easy exploration scenarios (MN7S4 and MN10S4), but its performance degrades when facing scenarios that require more sophisticated exploration strategies. A similar pattern can be observed when analyzing the results of RND, which is unable to solve MN7S8 and O2Dlh with any kind of weighting strategy. Contrarily, RIDE manages to solve all the tasks by its naïve implementation, although it achieves better results when using more sophisticated weighting exploration strategies.

We now focus the discussion on gaps arising from the use of different weighting strategies. The static (default) weighting strategy (indicated with a suffix *_s* appended to each approach) is surpassed by any of the other proposed weighting approaches in the majority of the cases. When using multiple static values (*_ngu*), the only approach that takes advantage of such a strategy is RND, yielding worse results for both COUNTS and RIDE in all the cases. This might happen due to the slow pace at which the intrinsic rewards values decay in RND in reference to the other strategies[6]. On the other hand, the use of parametric decay (*_pd*), which

[6]The error output by RND has higher amplitude values than those of RIDE, thereby RND is a better candidate to get benefit of applying the *_ngu* strategy by the use of agents with smaller intrinsic coefficient weights (avoiding over-exploration issues in the case of RND and oppositely having under-exploration issues with RIDE).

decreases the weight of the intrinsic reward as the training evolves to favor exploration, provides significant gains in almost all simulated scenarios. This approach is similar to *_ngu*. However, instead of using multiple agents with different static intrinsic coefficients, the parametric decay strategy modulates a single value during the course of training. When employing the *_pd* strategy, COUNTS is able to get a valid solution in `KS3R3`, RND improves all its scores and RIDE improves its behavior in the most challenging scenarios `MN7S8`, `KS3R3` and `O2Dlh`. Nevertheless, *_ngu* and *_pd* highly depend on the intrinsic coefficients given to each agent and the evolution of a single intrinsic coefficient during training, respectively. This strongly impacts on the agent's performance for a given scenario and dictates when those approaches might be better. Indeed, it can be seen as a tuning parameter like ϵ in ϵ-greedy strategies.

Finally, the use of adaptive decay (*_ad*) produces better results in COUNTS and RND when compared to the static case (*_s*). For RIDE, however, this statement does not strictly hold true, as its performance degrades in `MN7S4` and `MN7S8` (the agent does not even solve the task in the latter case). We hypothesize that this is due to the fact that the initial intrinsic returns are too high. Hence, calculating the historical average intrinsic returns biases the computation of the decay factor. As outlined in Section 4.2.1, a workaround to overcome this issue is to calculate returns with a moving average over a window of ω steps/rollouts. We hence include in the benchmark an adaptive decay with a window size of $\omega = 1000$ rollouts (*_ad*1000). With this modification, RIDE improves its behavior in all the complex scenarios. Nevertheless, *_ad*1000 performs slightly worse than *_ad* in RND, but never worse than its static counterpart *_s*. In general, *_ad*1000 promotes higher intrinsic coefficient values than *_ad*, as the calculated average return is a better fit to the actual return values. This leads to a lower decay value and a higher intrinsic coefficient, forcing the agent to explore more intensely than with *_ad* (but less than with *_s*).

4.4.2 RQ2: Which is the impact of using episodic counts to scale the intrinsic bonus? Is it better to use episodic counts than to just consider the first time a given state is visited by the agent?

Answers to this second question can be drawn from the results of Table 4.3. A first glance at this table reveals that the use of episodic counts or first-time visitation strategies for scaling the generated intrinsic rewards leads to better results. In the most challenging environments (`MNS78`, `KS3R3` and `O2Dlh`), these differences are even wider, as they require a more intense and efficient exploration by the agent. In fact, when the training stage is extended to cope with a more complex task, intrinsic rewards also decrease, inducing a lower explorative behaviour in the agent the longer the training period is extended. Hence, the agent does not seek as much novelty as it should, what might explain why the baseline

implementation of intrinsic motivation (*_noep*) fails in those scenarios as opposed to when using the scaling strategies (e.g., COUNTS and RND in `O2Dlh`). By contrast, in environments requiring less exploration (`MN7S4` and `MN10S4`), differences are narrower when using *episode-level* exploration and may be even counterproductive in some cases (i.e. COUNTS at `MN10S4` with *_1st*).

TABLE 4.3: Comparison of different IM strategies when using no scaling (*_noep*), episodic (*_ep*) or first-time visit (*_1st*) to scale the generated intrinsic reward and combine two types of exploration degrees. Interpretation as in Table 4.2.

	`MN7S4`	`MN10S4`	`MN7S8`	`KS3R3`	`O2Dlh`
COUNTS_*noep*	0.93 (0.86)	1.87 (1.78)	> 30	> 30	> 50
COUNTS_*ep*	**0.76** (0.56)	**1.55 (1.47)**	2.77 (2.56)	3.99 (2.00)	**33.17 (29.79)**
COUNTS_1*st*	0.85 **(0.48)**	> 20	**1.64 (1.42)**	**1.97 (1.19)**	45.26 (37.29)
RND_*noep*	3.83 (3.78)	7.84 (7.79)	> 30	10.83 (9.72)	> 50
RND_*ep*	1.41 (0.96)	1.72 (1.34)	3.60 (3.30)	**4.31** (2.63)	**18.54** (14.07)
RND_1*st*	**1.18 (0.59)**	**1.36 (0.78)**	**1.97 (1.72)**	4.78 **(2.29)**	21.19 **(9.88)**
RIDE_*noep*	4.71 (4.54)	5.29 (5.20)	> 30	11.44 (9.63)	39.68 (35.15)
RIDE_*ep*	**2.49** (1.82)	**2.27 (2.14)**	4.00 (3.68)	6.63 (4.39)	**30.88 (25.87)**
RIDE_1*st*	3.17 **(1.34)**	3.27 (2.33)	**1.95 (1.83)**	**5.13 (2.26)**	32.14 (28.03)
ICM_*noep*	2.67 (2.55)	> 20	> 30	8.02 (6.75)	34.04 (26.78)
ICM_*ep*	3.25 (1.26)	**1.68** (1.59)	> 30	5.32 (3.14)	**19.05** (13.87)
ICM_1*st*	**1.56 (0.87)**	1.90 **(1.07)**	**2.11 (1.77)**	**4.72 (4.23)**	20.74 **(10.09)**

To better understand the superiority of RIDE over ICM as shown in (Raileanu & Rocktäschel, 2020), we also evaluate the performance of both approaches under equal conditions, with (*_ep*, *_1st*) and without (*_noep*) scaling strategies. In this way, we can examine the actual improvement between the two types of exploration bonus strategies. Surprisingly, ICM gives better results in almost all the cases for the analyzed scenarios, yet exhibiting a larger variance in several environments that lead to failure (`MN10S4` and `MN7S8`). The reason might reside in how RIDE encourages the agent to perform actions that affect the environment, forcing the agent to assess all possible actions, so that the entropy in the policy distribution decays slowly. This hypothesis is buttressed by the results obtained in `MN7S4` and `MN10S4`: we recall that there are 3 useless actions in these scenarios (`pick up`, `drop` and `done`), and RIDE performs clearly worse (except for the *_ep* case in `MN7S4`). In more complex scenarios, when those actions are relevant for the task, performance gaps between RIDE and ICM become narrower.

For the sake of completeness of the results discussed for RQ1 and RQ2, Figure 4.5 shows the training convergence plots of COUNTS, RND and RIDE for different weighting and scaling strategies.

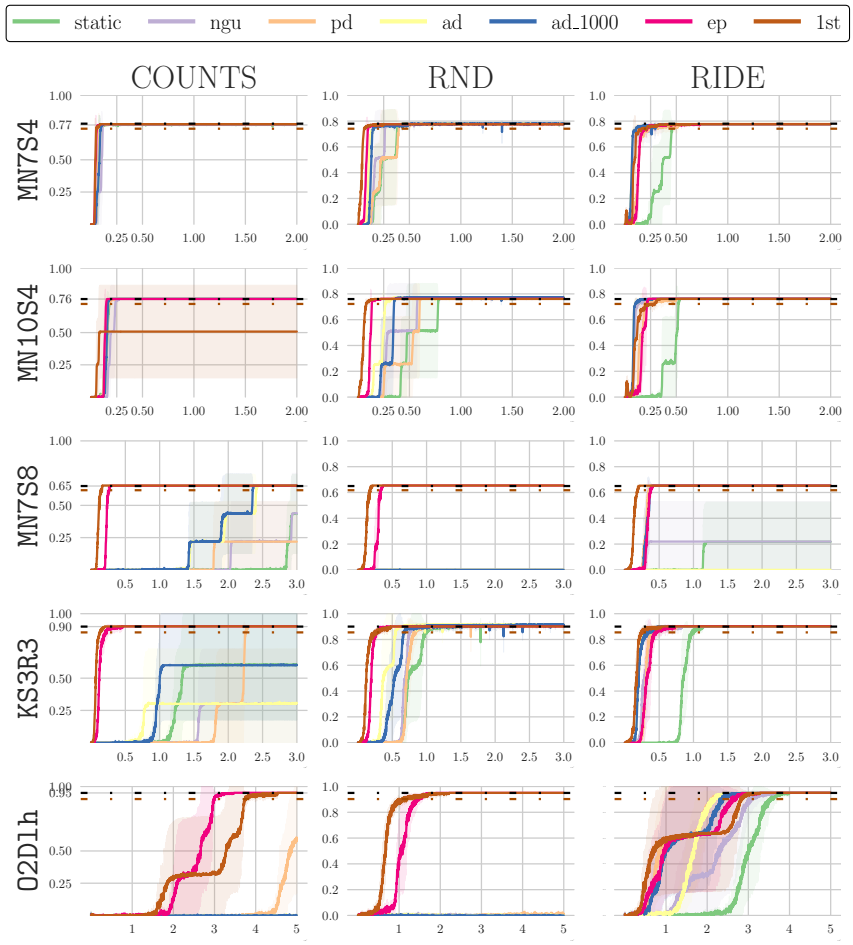


FIGURE 4.5: Convergence plots of the schemes reported in Tables 4.2 and 4.3. Each column represents a Intrinsic Motivation type (COUNTS, RND and RIDE from left to right); each row represents the different scenarios (`MN7S4`, `MN10S4`, `MN7S8`, `KS3R3` and `O2Dlh`, from top to bottom). All figures depict the average extrinsic return as a function of the number of training steps/frames (in a scale of $1e7$). For each scenario, optimal and suboptimal scores are highlighted with horizontal black and brown lines, respectively.

4.4.3 RQ3: Is the choice of the neural network architecture crucial for the agent's performance and learning efficiency?

One of the most tedious parts when implementing an algorithm is to determine which network architectures to use. First of all, when using an actor-critic RL framework it is necessary to establish whether a single but two-headed network or two different (and independent) networks will be adopted for the actor and the critic modules. In addition, some IM approaches are based on neural networks to generate the intrinsic rewards.

Herein we evaluate two of those solutions: RND and RIDE, evaluating the contribution of different neural network architectures to the overall performance of the agent. We use similar architectures to the ones used in RIDE and RAPID[7]: (a) a two-headed shared actor-critic network built upon convolutional and dense layers and (b) two independent MLP networks for the actor and the critic, respectively (Figure 4.4). Moreover, we fix the RL algorithm (PPO) and detail the number of parameters and time taken for the forward and backward passes in each network for an informed comparison.

TABLE 4.4: Comparison of number of parameters and required forward and backward passes between the ANN architectures described in Section 4.2.3 when being used with different IM modules.

	Lightweight (*lw*)		Default	
	Parameters	Time (ms)	Parameters	Time (ms)
Actor	14,087	-	-	
Critic	13,697	-	-	
Actor+Critic	27,784	-	29,896	-
Dictionary	-	83.66	-	95.11
Total COUNTS	27,784	724.25	29,896	937.37
Embedding	13,632	-	19,392	-
RND	27,264	336.39	38,784	721.64
Total RND	55,048	986.13	68,937	1,408.42
Inverse	12,871	-	18,439	-
Forward	12,928	-	18,464	-
Embedding	13,632	-	19,392	-
RIDE	39,431	388.84	56,295	844.43
Total RIDE	67,215	1,177.75	86,191	1,791.70

First of all, Table 4.4 informs about these details of the neural architectures in use for COUNTS, RND and RIDE. It reports the differences in terms of the number of parameters of each network, and the latency taken by the sum of both forward and backward passes through those IM modules (we note that COUNTS uses a dictionary and not a neural network for the reward generation). In addition, we summarize the total number of parameters depending on the implemented IM module, together with the actor-critic parameters. Referred to the total elapsed time, we report the total amount of time required for a rollout collection. This elapsed time takes into account both the forward and backward passes in the IM modules, and just the forward pass across the actor-critic, among other operations executed when collecting samples. Times are calculated when executing the experiments over an Intel(R) Xeon(R) CPU E3-1505M v6 processor running at 3.00GHz.

[7]Even with different neural architectures and base RL algorithms, they successfully solve the same tasks in MiniGrid with different sample-efficiency.

On the other hand, Table 4.5 shows the performance of the agent when configured with such different network configurations. It can be seen that when reducing the number of parameters in both the actor-critic and the IM modules (*_lw_tot*), the agent's behavior degrades critically. This occurs even with COUNTS, where the modification should have had less impact as the generation of intrinsic rewards does not depend on a neural network, but on a dictionary. When inspecting the performance of RIDE, its performance gets worse in all cases except for `MN7S4`, where the exploration requirements are the lowest among all the analyzed scenarios. As for RND, the full lightweight configuration of the networks makes the tasks not solvable by the agent.

TABLE 4.5: Performance obtained with COUNTS, RND and RIDE when 1) using the default network configurations, 2) a lightweight architecture for the IM modules and keeping actor-critic with a default configuration (*_lw_im*), and 3) when both the IM and the actor-critic modules are implemented with the lightweight networks (*_lw_tot*). Values in the cells represent the training steps/frames (in a scale of $1e6$) when the optimal average extrinsic return is achieved. Within brackets, the training steps when a suboptimal behavior is accomplished.

	`MN7S4`	`MN10S4`	`MN7S8`	`KS3R3`	`O2Dlh`
COUNTS	0.93 (0.86)	1.87 (1.78)	> 30	> 30	> 50
COUNTS_*lw_im*	0.93 (0.86)	1.87 (1.78)	> 30	> 30	> 50
COUNTS_*lw_tot*	1.64 (1.48)	2.52 (2.36)	> 30 (29.96)	> 30	> 50
RND	3.86 (3.79)	7.84 (7.79)	> 30	10.84 (9.72)	> 50
RND_*lw_im*	5.66 (5.44)	6.68 (6.61)	> 30	10.97 (9.45)	> 50
RND_*lw_tot*	> 20	> 20	> 30	> 30	> 50
RIDE	2.49 (1.82)	2.27 (2.14)	4.01 (3.38)	6.63 (4.39)	30.88 (25.87)
RIDE_*lw_im*	1.63 (1.31)	1.75 (1.53)	> 30	9.44 (5.08)	> 50
RIDE_*lw_tot*	1.42 (1.05)	> 20	> 30	8.00 (5.69)	> 50

Going back again to Table 4.4, it can be seen that the number of parameters to be learned is mostly dependent on the IM networks under consideration, whereas joining the actor and the critic into a single two-headed network barely increases the dimensionality requirements[8]. Nevertheless, the time required to perform a forward pass increases in approximately 25% when an unique actor-critic network is employed. Moreover, by using a single network, part of the parameters of the network are shared between the actor and the critic, which can induce more instabilities but also a faster learning since the model may share features between the actor and the critic and require less samples to learn a given task. With this in mind, we carry out an additional ablation study considering only the reduction of parameters at IM modules, and maintaining the actor-critic as a single two-head network.

Such results are provided in the second row of every group of results in Table 4.5 (*_lw_im*). These outcomes evince that when using

[8]We note that the number of parameters is slightly increased, but they also differ in the type of layers that are used in each network (the two-headed network uses CNNs while the independent actor-critic only uses dense layers.

RND_*lw*_*im*, slightly worse results are achieved with respect to RND with the default network setup. However, its performance does not degrade dramatically down to failure as with RND_*lw*_*tot*. Hence, using parameter sharing in a single actor-critic network yields a faster learning process and positively contributes for this case, inferring also that the dimensionality reduction in IM modules is not that critical in RND. Regarding RIDE_*lw*_*im*, in some cases (`MN7S4` and `MN10S4`) it attains better results, whereas in `MN7S8` and `KS3R3` it suffers from a notorious performance degradation (`MN7S8` is not solved). It can also be observed that the use of the single actor-critic network might be beneficial when reducing the complexity of the IM network (_*lw*_*im*), as it mitigates the performance degradation in 3 out of 5 scenarios (still, `MN7S8` and `O2Dlh` are not solved). This clashes with the results for separated actor-critic networks (_*lw*_*tot*), which fail to solve `MN7S8`, `O2Dlh` and `MN10S4`).

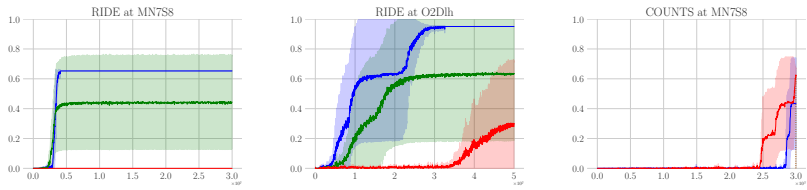


FIGURE 4.6: Convergence plots of COUNTS and RIDE for some scenarios when using the default network (blue), _*lw*_*im*(green) and _*lw*_*tot*(red). All the figures depict the average extrinsic return as a function of the number of training frames.

Finally, we include Figure 4.6 in order to help the reader extract further conclusions and gain insight about the behavior of the learning process. This figure reveals that, in the two cases in which RIDE_*lw*_*im* failed (namely, `MN7S8` and `O2Dlh`), the agent learned to solve the task in two out of the three experiments that were run (*seeds*). This underscores the impact of using different actor-critic architectures. Moreover, with the default actor-critic architecture and using the COUNTS approach, the agent is also capable of solving the `MN7S8` task in 2 out of the 3 runs. When using COUNTS_*lw*_*tot*, the agent reaches suboptimal performance and almost the optimal one within the frame budget.

4.5 Conclusions

In this chapter we have studied the actual impact of different design choices when implementing RL agents augmented with IM mechanisms. More concretely, we have evaluated multiple weighting strategies to grant different importance when combining the intrinsic and extrinsic rewards (i.e., the β coefficient). Moreover, we have analyzed the effect of applying distinct degrees of exploration (to scale generated intrinsic rewards, r_i) along with the influence of the complexity of the network architectures on the performance of both actor-critic and IM modules. To conduct the study we have

utilized environments belonging to the MiniGrid benchmark, so as to test the quality of the considered schemes in a variety of tasks characterized by a hard to very-hard demand of an exploratory behavior of the agent.

On one hand, we have shown that using a static intrinsic coefficient might not be the best strategy if focusing on sample efficiency. Adaptive decay strategies have proven to be promising, although they require a good parameterization of the sliding window. The parameter decay approach, in turn, has performed competently. However, the parameter values of the decay function are more dependent on the task at hand than the previous scheme, making this strategy more sensitive to the environment and the task. This resounds what occurs with ϵ-greedy strategies in some value-based algorithms. The use of multiple agents (as in NGU), each featuring a different exploration-exploitation balance, also suffers from the need for a good parameterization, but it reports worse results.

On the other hand, the use of *episode-level* exploration along with *experiment-level* strategies seem to be preferable when having environments with hard exploration requirements. It is not a clear winner nor a preference between episodic counts and first visitation strategies, as their performance is subject to the environment and the selected IM strategy. However, both achieve significant performance gains. The adoption of any of these strategies can be advised in future IM-related studies.

We have also analyzed the impact of the neural network architecture on both the actor-critic and IM modules. Results have shown that reducing the number of parameters in the IM modules deteriorates the performance of the agent, making it fail in some challenging scenarios which are feasible for the complex neural configuration. What is more, when reducing the dimensions of the IM network, it is preferable to use a shared two-headed actor-critic as it provides better results, although it is not clear whether those results are due to the use of a single neural network (and the underlying parameter sharing and common feature space for the actor and the critic), or instead to the adoption of different neural processing architectures (e.g. CNNs). Further research is necessary in this direction.

All in all, the evaluation study presented in this chapter can serve as a reference for the community in the implementation of intrinsic motivation strategies to address (1) tasks with sparse rewards; or (2) hard exploration scenarios where classic exploration techniques do not suffice.

Chapter 5

Towards Improving Exploration in Self-Imitation Learning using Intrinsic Motivation

The previous chapter has analyzed the impact of using different design factors over rewards generated with IM techniques. We have evaluated those algorithms not in singleton but in procedurally generated environments, where the generalization capabilities of the agent are essential for it to exhibit an overall good performance. Continuing with the idea of improving the sample efficiency over hard exploration PCG environments, in this chapter we further examine the use of Imitation Learning (IL) for this purpose.

Over years the use of IL and Transfer Learning has been widely adopted to accelerate the learning process and to reduce the amount of required training data (Hua et al., 2021; Nair et al., 2021; Wu et al., 2022). The strategy of using expert demonstrations has been also adopted to tackle exploration issues in hard exploration scenarios with sparse rewards, by either initializing a buffer with good behavior trajectories (Hester et al., 2017; Vecerik et al., 2018) or by generating a curriculum-style learning and re-initializing the agent smartly (Aytar et al., 2018; Salimans & Chen, 2018).

Unfortunately, such expert demonstrations are not always available in practice. This motivated the idea of storing trajectories – self-collected by the agent– featuring good exploration properties for a later replay, forging what is now known as *self-Imitation Learning* (*self-IL*[1]). Despite its effectiveness to alleviate the need for expert demonstrations, *self-IL* methods are highly sensitive to the early discovery of sufficiently good trajectories, which can be challenging in hard exploration scenarios.

[1]There is an approach named directly as SIL. Thus, for the sake of clarity, in this chapter we refer as *self-IL* to the family of algorithm in which the agent collects the experiences by itself for augmenting its sample efficiency, whereas SIL will denote the specific approach presented in (Oh et al., 2018).

In contrast, the Thesis has so far proven that exploration problems can be addressed with IM, so that the agent is encouraged to interact with the environment based on its inner curiosity. Therefore, we propose to use both methods together: IM to foster the exploration directly and ease the probability of finding out good trajectories, and *self-IL* to effectively replay and prioritize previously seen experiences. Our hypothesis is that, by combining both approaches, we can boost generalizable knowledge by reinforcing those trajectories that are attractive in terms of the main objective (i.e., task completion), as well as those experiences that induce novelty and that potentially lead to optimal behaviors.

Although the idea of combining both methods is not new (Ning et al., 2021; Oh et al., 2018; Sovrano, 2019), this chapter shows, for the first time, how to use such methods effectively in PCG environments where the need for generalization may not be easy to achieve by existing off-the-shelf *self-IL* methods.

5.1 Related Work

Imitation Learning (IL)

Experience replay buffers have been employed to stabilize the learning in RL when using ANN (Mnih et al., 2015). Instead of just focusing on the whole history of experiences, different works have proposed to prioritize those samples based on the TD-error (Schaul et al., 2016), taking into account trajectories (as a whole) with large extrinsic returns (Zha, Lai, et al., 2021), and even learning an additional policy to optimize the data to be sampled (Zha et al., 2019). The intuition behind these approaches is to sample the most promising experiences more frequently, potentially enhancing and expediting the overall knowledge learned by the agent.

Similarly, IL methods have resorted to expert demonstrations to accelerate the learning stage by forcing the agent to learn the inherent decision making embedded in those examples (Hester et al., 2017; Vecerik et al., 2018). However, collecting expert demonstrations is not easy to achieve, and most of the time the agent is fed with suboptimal demonstrations that hinder the learning of an optimal policy. Several workarounds have been explored to circumvent these weaknesses, such as resetting the states and estimating the advantage (Nair et al., 2018; Ning & Huang, 2020), encouraging more exploration that subsequently updates the buffer content (Zhu, Lin, Dai, et al., 2020), or adopting maximum entropy RL concepts (Gao et al., 2019; Reddy et al., 2019) from soft-Q-learning (Haarnoja et al., 2017).

When lacking an expert capable of providing examples, *self-IL* was proposed so that an agent, without any kind of prior knowledge, is responsible for collecting and replay again those that can make the agent learn faster by the virtue of exhibiting good behaviors (Abolafia et al., 2018; Z. Chen & Lin, 2020; Guo et al., 2018; Oh et al., 2018; Zha, Lai, et

al., 2021). Interestingly, *self-IL* approaches aim to deal with exploration-exploitation dilemma by encouraging the agent to exploit the information that has not been previously learned, so as to achieve a better exploration strategy and ultimately, a near-optimal performance. Despite their proven efficiency at hard exploration environments (Ecoffet et al., 2021; Guo et al., 2021; Oh et al., 2018), these methods tend to struggle in tasks characterized by very sparse rewards due to their reduced capability to find good trajectories in the firsts stages of the training by naive exploration (Pshikhachev et al., 2021). This can be the reason why *self-IL* solutions have so far been evaluated mainly over non-procedurally-generated environments, where the generalization capabilities of the agent are not tested since the state/observation space does not change from episode to episode (Ecoffet et al., 2021; Guo et al., 2021; Oh et al., 2018).

Alternatively, new approaches like Ranking the Episodes (RAPID) (Zha, Ma, et al., 2021) have emerged to deal with procedurally-generated environments by ranking the episodes not only in terms of their extrinsic rewards, but also by considering exploration scores related to the level/episode. As a result of this multi-criteria ranking, the agent can effectively mimic the suitable experiences and overcome the exploration needs of very sparse rewards tasks. Other *self-IL* methods take into account only the extrinsic scores for their prioritization (Gangwani et al., 2019; Guo et al., 2018; Oh et al., 2018), which might lead to insufficient exploration in sparse reward scenarios.

Intrinsic Motivation (IM)

As we have broadly discussed so far in the dissertation, IM is one of the mechanisms that can be used to enhance the exploration and thereby achieve a good overall performance when facing hard exploration scenarios. Some of the IM methods were not originally designed to tackle procedurally-generated environments, such as COUNTS (Bellemare et al., 2016), RND (Burda, Edwards, Storkey, et al., 2018) or ICM (Pathak et al., 2017). More recently, new approaches have emerged with the aim to learn more robust and generalized knowledge: this is the case of BeBold/NovelD (T. Zhang et al., 2020, 2022), Exploration via Maximizing Deviation from Explored Regions (MADE) (T. Zhang et al., 2021), RIDE (Raileanu & Rocktäschel, 2020) or Adversarially Guided Actor-Critic (AGAC) (Flet-Berliac et al., 2021). One of the common aspects to these methods is that the produced rewards are commonly used with on-policy algorithms (i.e A2C/A3C, PPO, IMPALA), which discard the collected experiences after their respective optimization updates. This aspect reduces the potential benefits of IM methods in what refers to sample-efficiency, even more so when considering the disentanglement/derailment problem (Ecoffet et al., 2021) explained in Section 3.7.2 and catastrophic forgetting due to the decay of the intrinsic reward values (Huang & Tsai, 2022).

Combined IL+IM approaches

The idea of combining IL and IM is not new. Previously some works have analyzed if they are complimentary to each other. However, most previous studies have considered singleton environments where the generalization capabilities are not assessed.

In this vein, SIL (Oh et al., 2018) used count-based intrinsic rewards to augment the exploration capabilities in singleton Apple-Gold mazes. Similarly, Diverse Trajectory-conditioned Self-Imitation Learning (DTSIL) (Guo et al., 2021) employed the same environment and evaluated its generalization ability by training over 12 random mazes and testing the trained agent over 6 unseen instances where it requires a hierarchical policy (no IM technique is considered). Nevertheless, they combined DTSIL and IM to analyze the exploration improvements over some Atari's (non-procedurally-generated) environments. However, they did not analyze the generalization concerns targeted in this chapter. The approach presented in (Sovrano, 2019) was also tested in Atari – more concretely in Montezuma, Solaris and Venture environments – in which different prioritizing strategies were studied with Prioritized Experience Replay (PER) (Schaul et al., 2016) and IM methods. More recently, (Ning et al., 2021) evaluated its solution with the aforementioned SIL and BeBold approaches in relatively "easy" sparse MiniGrid environments (so does it RAPID (Zha, Ma, et al., 2021) too). Remarkably, such works were not tested in more challenging – *hard* – environments as in other related IM works (Flet-Berliac et al., 2021; T. Zhang et al., 2020). At this point, we note that *hard* exploration refers to tasks that are more complicated to be solved than others within the same benchmark. For instance, in MiniGrid (Chevalier-Boisvert et al., 2018) hard exploration can refer to the use of more rooms in MultiRoom environments; a larger room size in KeyCorridor levels; or more complex ObstructedMaze scenarios with increasing complexity. In these tasks the generalization and learning requirements are of utmost importance, and call for better exploration strategies during the agent's learning process.

Contribution

This chapter formally proposes the use of a *self-IL* strategy together with IM, showing that this combination succeeds at solving hard exploration environments. The proposal herein presented builds upon RAPID (Zha, Ma, et al., 2021), in which its ranking strategy for past stored episodes is combined with intrinsic motivation mechanisms. We advance over the state of the art by showing that previous *self-IL* approaches did not consider (Ning et al., 2021) or failed to solve (Zha, Ma, et al., 2021) procedurally-generated environments with hard exploration requirements. Our experiments in PCG scenarios with different learning challenges validate this claim, and prove that the combination of RAPID and IM methods can meet the exploration requirements needed to solve hard exploration tasks where generalization is mandatory for their successful completion.

5.2 Synergistic Exploration with Self-Imitation Learning and Intrinsic Motivation through Ranked Episodes

Self-IL methods allow the agent to reproduce self-collected past experiences towards inducing a better explorative behavior in its learning process (Oh et al., 2018). The core idea is to replay experiences that potentially improve the performance of the agent, even though that information was not persisted in the agent because experiences were not exploited enough. This makes the agent explore more effectively. This behavior is emphasized when dealing with on-policy algorithms, where samples are discarded after their optimization step. However, *self-IL* methods are highly sensitive to the discovery of good samples, and depend on the capability of the agent to find such experiences on their own (e.g. by using stochastic policies). This might not be sufficient in settings with very sparse rewards, where the probability of achieving a non-zero rewarded episode can be substantially low. This is the reason why IM, combined with *self-IL*, can imprint the explorative behavior needed for collecting good episodes and learning therefrom.

In what follows we describe a *self-IL* and a IM approach, namely, RAPID (Section 5.2.1) and BeBold (Section 5.2.2) respectively, which lie at the heart of the framework presented in this chapter.

5.2.1 Ranking the Episodes

RAPID (Zha, Ma, et al., 2021) was proposed to endow an agent with a general criterion to detect good exploration behaviors and reproduce them with more frequency. To this end, RAPID treats episode's experiences as a whole, and assigns episodic scores – extrinsic reward (S_{ext}) and local score (S_{local}) – to each of those experiences. These scores are combined with long-term views – global score (S_{global}) – for their posterior ranking. Figure 5.1 depicts schematically this ranking process.

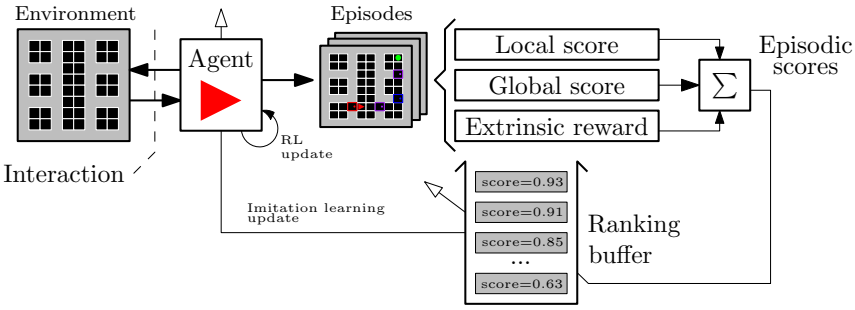


FIGURE 5.1: RAPID (Zha, Ma, et al., 2021) overview.

The intuition behind this design arises from the way how humans judge an agent's performance: instead of focusing on how good/bad an agent performed in a specific time step (state-level), we as humans tend to analyze how it did in the whole problem/task (episode-level). With this purpose, each episode has an overall score calculated as a weighted sum of three distinct scores:

$$S = w_0 \cdot S_{ext} + w_1 \cdot S_{local} + w_2 \cdot S_{global} \tag{5.1}$$

where S_{ext} is the total extrinsic reward of the episode; S_{local} encourages the exploration inside the episode by maximizing the diversity across visited states; and S_{global} models a long-term exploration view by using the average curiosity of the states inside the episode. The latter is computed in practice by means of a visitation count strategy that considers not only the current episode, but the whole training process. Based on these scores, the most promising episodes are kept in the replay buffer, so that highly ranked episodes are replayed and ultimately imitated by the agent to enhance its exploration.

5.2.2 Beyond the Boundary of Explored Regions

BeBold (Zha, Ma, et al., 2021) is an IM method that circumvents the *short-sightedness* and *detachment* problems by issuing a reward signal (r_t^i) whenever the novelty of the next state (s_{t+1}) is higher than that of the current one (s_t). As previously stated in Chapter 4, more concretely in Section 4.1, it can be mathematically formalized according to:

$$r_t^i = \max\left(\frac{1}{N(s_{t+1})} - \frac{1}{N(s_t)}, 0\right) \tag{5.2}$$

where $N(s_t)$ denotes the number of times the agent has visited state s_t during the training phase. An interesting property of BeBold is that it minimizes the undesired behavior of the agent going back and forth by imposing the reception of a (intrinsic) reward only the first time a state is encountered in an episode.

5.2.3 Proposed Framework

As stated in (Levine, 2021), IM-based exploration methods provide an auxiliary objective to collect more diverse data rather than learning to utilize it. Our proposed framework aims to exploit efficiently the diverse data collected during the agent's interaction (used in the on-policy RL updates). In doing so, it goes one step further by bolstering its knowledge with an off-policy/supervised loss, which replays and prioritizes the most promising episodes from which to learn.

Bearing this in mind, our framework first generates an intrinsic reward (r_t^i) at each step as in Expression (4.5). This intrinsic reward is weighted by an intrinsic coefficient (β) before being added to the extrinsic reward (r_t^e) given by the environment, yielding an overall reward $r_t = r_t^e + \beta r_t^i$.

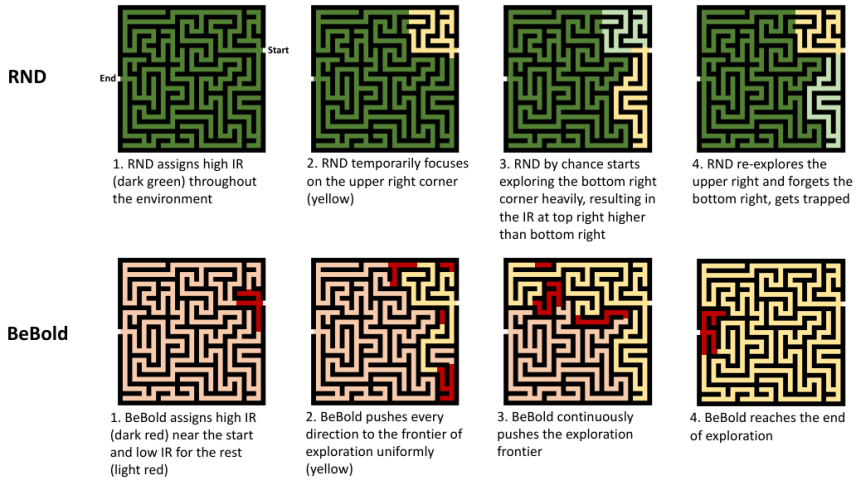


FIGURE 5.2: Hypothetical RL environment demonstrating how exploration would be carried out when using RND (or any other IM method) when compared to BeBold. As shown in this plot, BeBold continuously encourages pushing towards the frontier of exploration. Image belonging to (T. Zhang et al., 2020) paper.

This overall reward is then used by the RL algorithm to maximize the discounted return given in Expression (2.3). Then, the most promising experience tuples –$\{s, a\}$– are stored in a buffer of limited size. The criterion to retain episodes is driven by (1) an extrinsic component (non-biased Monte Carlo extrinsic return, $G_t = \sum_{t=0}^{T} r_t^e$) and also by (2) scores that foster the exploration behavior, i.e., the local and global scores defined in RAPID as per Expression (5.1). After a given number of steps, a batch of experiences is sampled from the buffer and the policy is enforced to match previously executed actions by *Behavioral Cloning* (BC) as in Expression (2.26). We note that other IL techniques could be used instead.

With this proposal, the RL update will be fed with a intrinsic reward that promotes exploration and augments the probability of sampling good episodes for the ranking buffer. At the same time, the buffer stores previous highly-ranked episodes to keep improving the agent even when the on-policy updates are not enough and when its intrinsic exploration bonus vanishes. Likewise, both losses will foster exploration, while steadily maintaining an exploitative learning focus on the given task.

5.3 Experimental Setup

This section describes the considered environments and the baselines for comparison. The selected hyperparameter values and neural network architectures are also described.

As in previous chapters, we report the mean and standard deviation of the average return computed over the past 100 episodes for each experiment, performing 3 different runs (with different seeds) to account for the statistical variability of the results. For transparency and reproducibility of the experiments later discussed, the code is available in a public GitHub repository: https://github.com/aklein1995/exploration_sil_im.

5.3.1 Environments

We evaluate our proposed approach over MiniGrid (Chevalier-Boisvert et al., 2018), as explained in Chapter 4 (Section 4.3.1). Specifically, we evaluate the framework over the following scenarios (for further information about the environments and their tasks, please refer to Chevalier-Boisvert et al., 2018): MultiRoom (`MN7S8` and `MN12S10`), KeyCorridor (`KS4R3`) and ObstructedMaze (`O2Dlh`). The criterion to select these environments relies on their difficulty as verified in (Zha, Ma, et al., 2021), where `MN12S10` and `KS4R3` were identified as the most difficult scenarios under analysis: the first was solved by RAPID and RIDE, while the latter remained unsolved for the given train steps by any of the baselines under consideration. In the case of (Ning et al., 2021), where the performance of SIL+BeBold was analyzed in MiniGrid, the most difficult environments were `KS3R3` and `MN6S`, which are more easily solvable than `KS4R3` and `MN12S10` (they use smaller rooms and less number of rooms respectively). Additionally, we include another very hard exploration scenario, not considered in the aforementioned works, which possesses different characteristics and requirements than the previous environments: `O2Dlh`.

5.3.2 Baselines and Hyperparameters

We select RAPID (Zha, Ma, et al., 2021) and SIL (Oh et al., 2018) as *self-IL* baseline methods, and BeBold (T. Zhang et al., 2020) as the IM. All strategies use PPO as their core RL algorithm, which uses a number of steps equal to 128 and 4 minibatches of size 32 for training (one unique agent). Each train step comprises 4 epochs, where optimization updates are carried out with a learning rate of 10^{-4}, a clipping factor of $\epsilon = 0.2$, $\gamma = 0.99$ and $\lambda = 0.95$ for the advantages calculation with GAE as per Expression (2.18). Furthermore, the loss function (recall Expression (2.22)) is weighted by a entropy coefficient of $c_2 = 0.01$ and a value coefficient of $c_1 = 0.5$. Moreover, we employ 2 independent fully-connected layers for the actor and the critic – each with 64 neurons – for all the experiments and baselines.

Specific parameters of RAPID are configured as in its original implementation reported in the paper where it was first presented: a buffer size of $\mathcal{D} = 10^4$ experiences, batch size of 256 and 5 off-policy updates after every episode completion. Moreover, the weights to rank the replay buffer episodes – Expression (5.1) – are set to $w_0 = 1$, $w_1 = 0.1$ and $w_2 = 0.001$ according to the sensitivity analysis shown in the original approach (Zha, Ma, et al., 2021).

In the case of SIL, for the sake of fairness with respect to RAPID the same replay buffer size ($\mathcal{D} = 10^4$) and the same off-policy update ratio (5) are used. Moreover, a SIL loss weight of 0.1 and a SIL value loss weight of $\beta_{sil} = 0.01$ are set. Regarding PER (Schaul et al., 2016), we select a prioritization exponent $\alpha_{PER} = 0.6$ and a bias correction factor $\beta_{PER} = 0.1$. All these parameter values were chosen according to the supplementary material provided in (Oh et al., 2018)[2], and taking into account that we aim to solve hard exploration environments. On the other hand, the intrinsic reward when using BeBold is computed as described in Section 5.2.2, calculating the novelty with visitations counts (taking advantage of the discrete state space) and using an intrinsic coefficient of $\beta = 0.005$. The value of this coefficient (together with that of the entropy coefficient, c_2) was tailored based on the results of a grid search carried out over scenario `MN7S8` – whose results are shown in Figure 5.3 – while keeping the values for other parameters fixed (e.g. the RAPID weight values above referred, namely, w_0, w_1 and w_2).

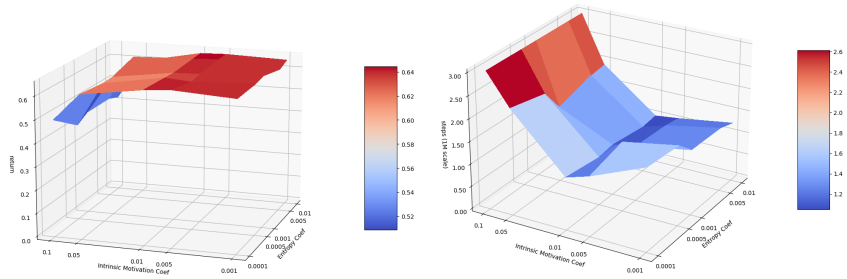


FIGURE 5.3: Results of a grid search over the `MN7S8` scenario to determine β (intrinsic motivation coefficient) and c_2 (entropy coefficient). (Left) Returns obtained after $3{\cdot}10^6$ training steps; (Right) Number of steps (in scale of millions, 10^6) required for the agent to achieve an optimal average return (≈ 0.65) for the first time.

5.4 Results and Analysis

This section presents the results of the proposed approach in PCG environments, examining them in depth from different angles:

5.4.1 Performance of self-IL and IM Techniques: Independent versus Combined

To begin with, Figure 5.4 analyzes the actual impact on the performance of the agent when using IM and *self-IL* techniques, either independently or jointly. We observe that BeBold (light blue curve) shows a good behavior only in 2 out of the 4 environments under consideration (namely, `MN7S8` and `KS4R3`). However, it completely fails when dealing with the challenging

[2] http://proceedings.mlr.press/v80/oh18b/oh18b-supp.pdf

scenarios of MultiRoom and ObstructedMaze series (i.e., `MN12S10` and `O2Dlh`). When using just SIL (green curve), it performs poorly in all scenarios. We here recall what we stated at the beginning of this chapter: other works (e.g., (Ning et al., 2021)) have analyzed the complementarity of SIL and IM, but over problems with sparse rewards that are not so complex as the ones considered in this chapter.

When it comes to RAPID, it is capable of solving `MultiRoom` environments, but struggles over `KS4R3` and `O2Dlh` (as expected). This latter environments are assumed to have larger state spaces and an increasing difficulty from the perspective of exploration. On top of the *self-IL* approaches, BeBold fosters the exploration and, consequently, renders some actionable learning when using SIL (pink curve). However, results are worse than those obtained when using BeBold in isolation (light blue). This suggests that the SIL prioritization mechanisms are not working properly. Contrarily, results are outstanding when combined with RAPID (light green curve), reducing drastically the number of samples to achieve the same performance level, and attaining a better overall learning when compared to using RAPID in its naive version (blue plot). Besides these improvements, it is interesting to notice that the benefits of using IM remain even when the latter is not enough to learn in isolation: BeBold does not capture any knowledge over `MN12S10` and `O2Dlh`, but it augments the capabilities of RAPID when used in those scenarios.

5.4.2 Evaluation of RAPID with Various IM Strategies

A key aspect to study empirically is the capacity of IM to enhance the agent's exploration while learning. Therefore, it is of utmost importance to assess the sensitivity of the proposed *self-IL*+IM combination with respect to the selection of the IM approach. With that in mind, and considering that the current implementation is based on BeBold's tabular version (see Section 5.2.2), we now evaluate the agent's performance with other two visitation counts strategies: *counts* (i.e. $r_t^i = 1/\sqrt{N(s_{t+1})}$) and *counts1st*, which is the same as *counts* but with episodic restriction. This second set of experiments allows comparing very similar IM strategies that have proven to yield different results due to their intrinsic reward generation scheme (Andres et al., 2022; T. Zhang et al., 2020).

The results provided in Figure 5.5 suggest that there is a high relationship between what the agent can learn with IM (without *self-IL*) and what it actually does by combining them altogether. This can be regarded as a measure of the effectiveness of IM methods when implemented in isolation, where their base functionality of exploring is not wide-spread with the *self-IL* counterpart. At this point, by just inspecting the results reported in (Andres et al., 2022; T. Zhang et al., 2020), it is clear that *counts* is the worst method, followed by *counts1st* and BeBold, $counts < counts1st < BeBold$. Differences between *counts1st* and BeBold are unclear: most of the contribution seems to be related to the

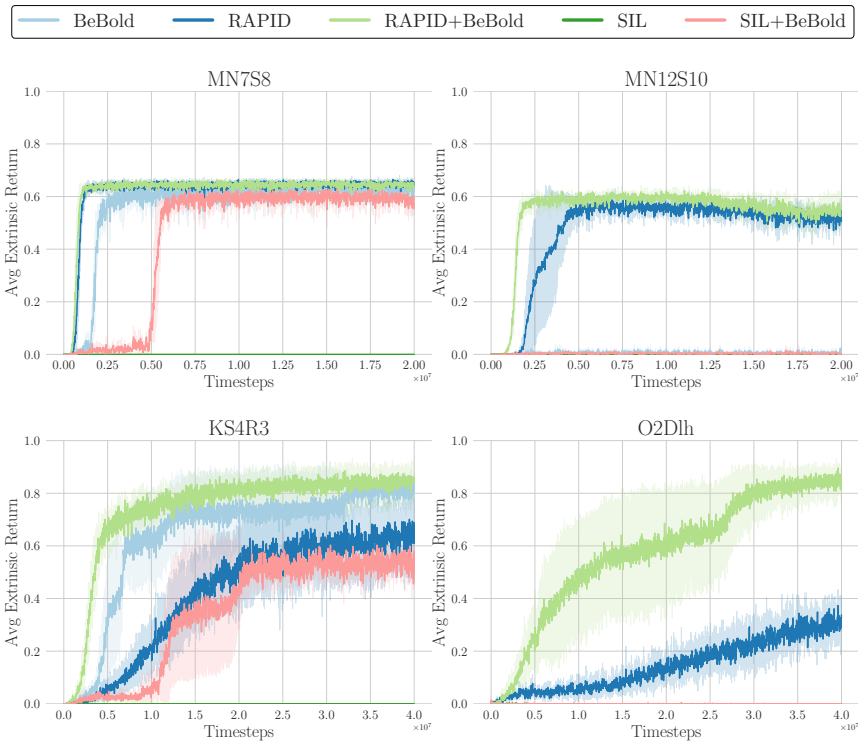


FIGURE 5.4: Results over multiple procedurally generated hard exploration environments in MiniGrid. Both RAPID and SIL always achieve better results when combined with BeBold.

episodic restriction part. However, going beyond the boundaries of already explored regions seems to be promising as well, as it yields better results when compared to RND with episodic restriction (T. Zhang et al., 2020).

The same comparative performance between IM methods holds when combining them with the ranking replay strategy, where RAPID+*counts* (red curve) performs slightly better or equal to RAPID in isolation (blue plot), yet being the worst out of the three IM options. Moreover, the choice of one IM strategy over another can actually deteriorate the performance of the agent, as observed in `KS4R3`. In this particular case, the aforementioned RAPID+*counts*(red curve) is worse than using RAPID without IM (blue curve). Nevertheless, when selecting demonstrably good IM strategies, the agent combining *self-IL*+IM – both RAPID+*counts1st* (yellow curve) and RAPID+*BeBold* (light green curve) – improves its performance even when it was not able to do it with just the IM strategy.

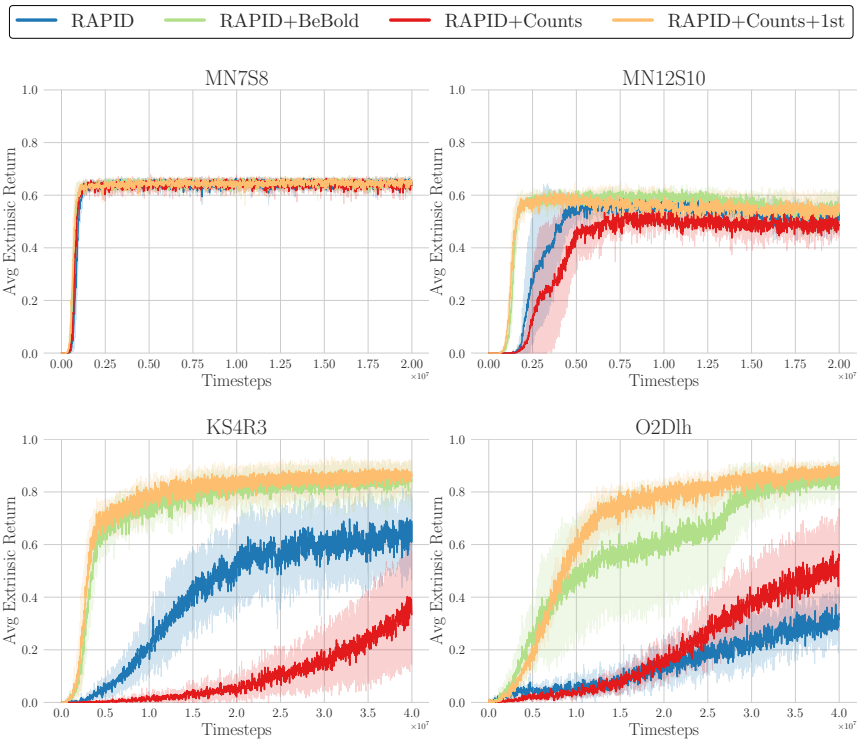


FIGURE 5.5: Performance comparison of RAPID when combined with different IM methods, namely, *counts*, *counts1st* and BeBold.

5.4.3 Exploration-exploitation Parameters Evolution in self-IL+IM

By introducing IM into the on-policy loss, the agent has to deal with multiple objectives (exploration-exploitation) in various stages: 1) on-policy, by balancing the extrinsic and intrinsic rewards; and 2) off-policy, by keeping in the buffer the most promising experiences parameterized by the extrinsic, local and global scores.

In this regard, Figure 5.6 depicts the evolution of some representative values concerning how the exploration is carried out during an experiment. Initially $G^i > G^e$(i.e., the episodic discounted intrinsic and extrinsic returns calculated as described in Expression 2.3), which evinces that the agent learning process is guided by IM in the absence of extrinsic signals from the environment. Eventually, extrinsic feedback is obtained and gains more importance for the agent's ability to complete the task. Similarly, the impact of the extrinsic score in Expression (5.1) – $w_0 \cdot S_{ext}$, which promotes the exploitation of highly extrinsic rewarded episodes – quickly increases, so that those potential trajectories are more often replayed. However, the selection criterion is also subject to the local score – $w_1 \cdot S_{local}$, which aims to maximize the diversity of observations inside the episode – that also increases until reaching its maximum value of 0.1

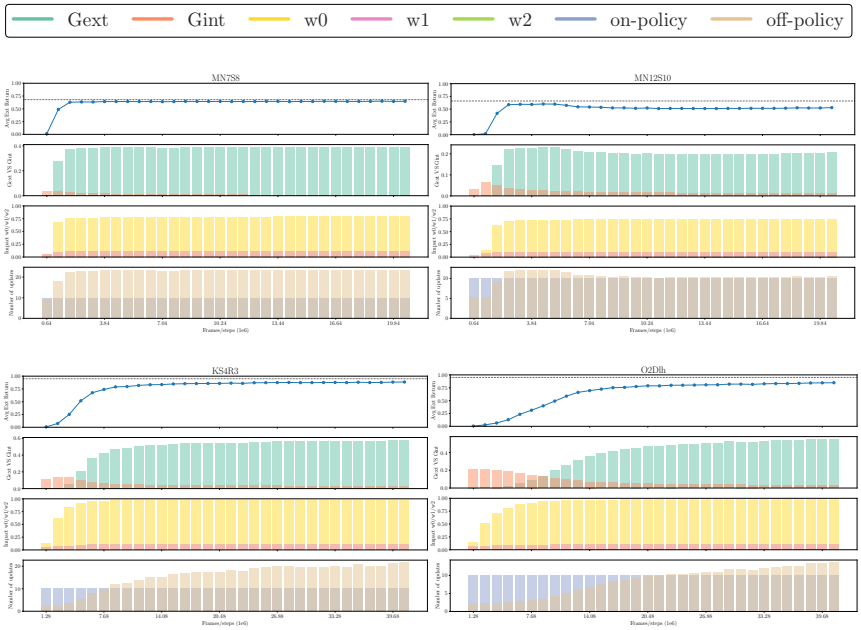


FIGURE 5.6: Summary of the evolution of different critical values that impact the learning for a given seed in all the scenarios, using RAPID+BeBold. Plots in the first row denote the average extrinsic reward. Plots in the second row depict the difference between the discounted extrinsic ($Gext \equiv G^e$) and intrinsic ($Gint \equiv G^i$) returns used in the on-policy update (RL-loss). Figures in the third row show the influence of each component/score of the ranking buffer (w_0, w_1 and w_2) when sampling from its collected experiences. Finally, plots in the last row indicate the average number of off-policy updates per 10 on-policy updates (ratio of updates, ξ). All depicted data correspond to the average value in the given time slots.

(which is subject to $max(S_{local}) = 1$ and $w_1 = 0.1$). To a lower extent, the global score ($w_2 \cdot S_{global}$) also plays its role in the selection criterion, which can be helpful during the initial learning stages, when there are no success episodes to complete the task, and also to untie when two episodes require the same amount of steps for the completion of the task. However, its relative importance is lower in comparison to the other scores due to the selected value of the w_2 parameter (0.001)[3].

Frequency of Updates

We now proceed by exposing how the ratio ξ between the number of on-policy and off-policy updates changes over the curse of training. In what follows ξ is represented as on-policy:off-policy ratio: a ξ value of 1:2 will thus imply that the off-policy updates are executed 2 times more frequently than the on-policy ones.

[3]Recall that the criteria to select such weight values (w_0, w_1, w_2) is due to reported results in (Zha, Ma, et al., 2021).

As was explained in Section 2.1.2, an episode can be larger or shorter than a trajectory. On-policy optimization steps are executed once a trajectory[4] has been finished, and it remains fixed during the whole training. By contrast, off-policy updates are applied once an episode finishes, which varies depending on the maximum steps per episode configured for each environment, and also on the optimality of the agent's policy at that moment. The decision to execute off-policy updates at the end of the episode was taken from the original paper where RAPID was proposed (Zha, Ma, et al., 2021).

Such ratio ξ can change from 1:1 to 1:3 in `MultiRoom` environments, and more dramatically in other scenarios like `KS4R3`, which initially implies a ratio of 4:1 and can evolve up to a 4:13 relation. In words, the off-policy loss can undergo a modification in its schedule that makes it update more than 10× at its initial frequency (Table 5.1). Such a balance has a critical importance in the agent's learning process, as it would turn to optimize what is stored in the buffer rather than what is actually experiencing (or vice versa). This generates in turn a big difference between both methods. In fact, in IL this ratio is usually balanced by either using a weight when combining both losses or by carefully tailoring the frequency update (Hester et al., 2017; Sovrano, 2019).

TABLE 5.1: On-policy versus off-policy ratios that can be achieved in each scenario when the supervised loss is backpropagated to when the episode finishes. Each scenario has a different maximum number of steps (row 2) and also different expected number of optimal steps (row 3) (we include an estimation of the optimal steps as it differs from seed to seed). We show the expected initial ratios (ξ) when the agent cannot solve the task (rows 4 & 6) and when it accomplishes the task via an estimated optimal policy (rows 5 & 7). We also report those values when the rollout size is $T = 128$ (rows 4-5) and $T = 2048$ (rows 6-7).

		`MN7S8`	`MN12S10`	`KS4R3`	`O2Dlh`
Max steps per episode		140	240	480	576
Expected optimum steps		50	105	37	32
$T = 128$	Initial	1:1	2:1	4:1	5:1
	Final	1:3	2:2	4:13	5:18
$T = 2048$	Initial	1:14	2:17	4:17	5:18
	Final	1:40	2:40	4:216	5:320

5.4.4 Scheduling self-IL Updates

To shed further light on the importance of the aforementioned ratio ξ, we now fix the off-policy loss to be constant and subject directly to the on-policy updates. We then analyze how the performance varies under several

[4]Here we refer as a trajectory to the experiences collected on-policy with a **fixed** amount of interactions, whereas an episode's length might vary depending the environment and the learned policy.

values for this ratio.

Figure 5.7 summarizes the results obtained for this study. In the family of `MultiRoom` scenarios, the agent is very sensitive to a reduction of the frequency of the off-policy updates, which can eventually make the agent fail when increasing their complexity (e.g. 10:1 in `MN12S10`). Contrarily, in `KS4R3` the original adopted schema (blue curve) with a ratio of 4:1 performs much better than a more frequent update (green plot) of the off-policy part (1:1). This fact is also observed when using a more conservative ratio of 10:1 (red result), suggesting that, although a higher off-policy update frequency can be beneficial at initial stages to bootstrap the learning process in hard exploration tasks, it can eventually degrade the learned knowledge in the long term. These conclusions can also be inferred when using BeBold, but with a better sample-efficiency and optimal solutions. Similar conclusions hold when analyzing `O2Dlh`.

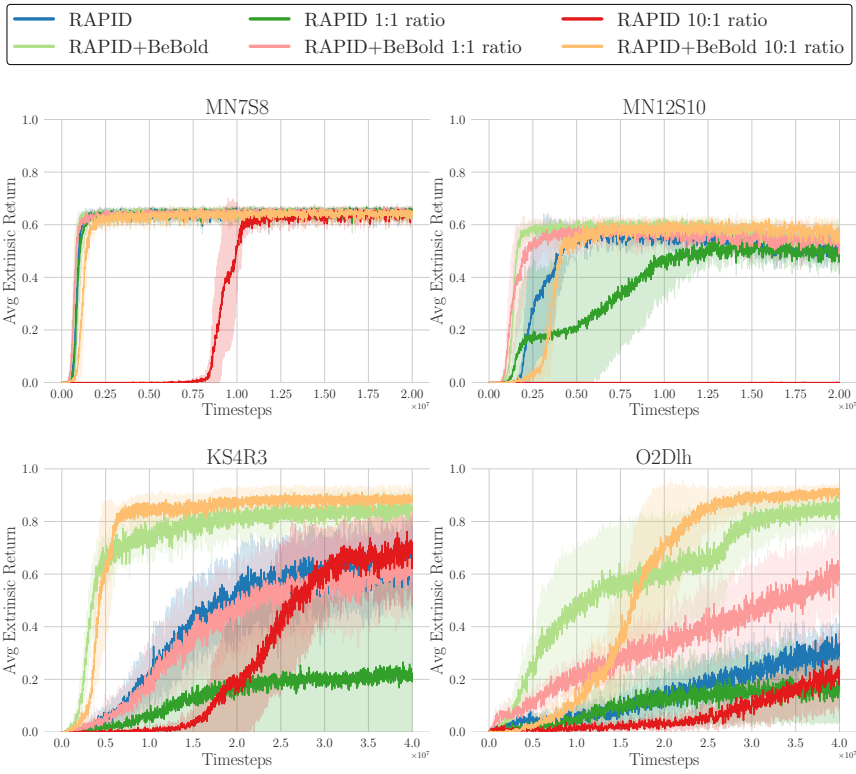


FIGURE 5.7: Results over multiple procedurally generated MiniGrid hard exploration environments using different ratios ξ between on-policy (PPO) and off-policy (RAPID) updates. The default RAPID approach has a dynamic update ratio, by which it executes an optimization step every time an episode finishes (see Table 5.1).

5.4.5 Addressing Inter-episode Variance

So far, the selected value of the ratio ξ seems to be decisive for the success and sample efficiency of the training process. However, the obtained outcomes are very noisy and barely close to optimal results.

We hypothesize that this can be due to one of the two losses being unstable. While the seminal work presenting RAPID used PPO with a rollout size[5] of $T = 128$, other similar works considering the same environment use a larger time horizon equal to $T = 2048$, with better and more stable results (Andres et al., 2022; Flet-Berliac et al., 2021). In PCG environments each level is configured differently depending on the selected seed. Consequently, by training the agent with less episodes in a single update, it might get biased to learn specific features present in that subset of episodes, rather than getting the required high-level skills to solve the desired task in the whole possible episode/level distribution. Hence, the increase of the rollout size implies that the agent will be trained – in the on-policy update – with a larger set of episodes (see Table 5.1 to check episode lengths). This forces the algorithm to extract generalizable knowledge in this wider set of slightly different environments, avoiding a *by-heart* learning. Furthermore, this also reduces the variance of the on-policy updates through the ANN, as the minibatch size will be larger. However, the agent will perform less optimization steps during the training process for the same amount of steps/frames. On this basis, the following question arises:

How does the use of larger rollout size impact on the on-policy update regarding the performance and the stabilization of the learned knowledge?

The answer can be found by analyzing Figure 5.8. The on-policy update is substantially improved, as can be told from the performance of BeBold (light blue) without being corrupted by off-policy updates. Indeed, this IM approach is able to solve all the environments with the expected optimal steps, obtaining the best result in both `KS4R3` and `O2Dlh`. On the contrary, RAPID (blue) performs worse, and its contribution when combined with BeBold (light green) is also not as good as it has been observed in the previous analysis. The reason for these bad results also connects to what we have previously highlighted: the ratio ξ.

By increasing the rollout size (T) and by making the off-policy updates be subject to the episode completion, the relevance of the off-policy loss in the agent's learning process grows up to be 14×, 8×, 4× and 4× more frequent than the on-policy counterpart in `MN7S8`, `MN12S10`, `KS4R3` and `O2Dlh`, respectively, just at the start of the training process (Table 5.1). As we have already observed in Figure 5.7, these ratios do not necessarily guarantee a better learning process. Thus, when adjusting the

[5] The rollout size is directly related with the number and minibatch size. The increase of the first implies that the minibatch size is also augmented (for the same number of minibatches). For instance, using $T = 1024$ and 4 minibatches means to have 256-sized minibatches, whereas with $T = 128$ and using the same number of minibatches this size decreases to 32 units.

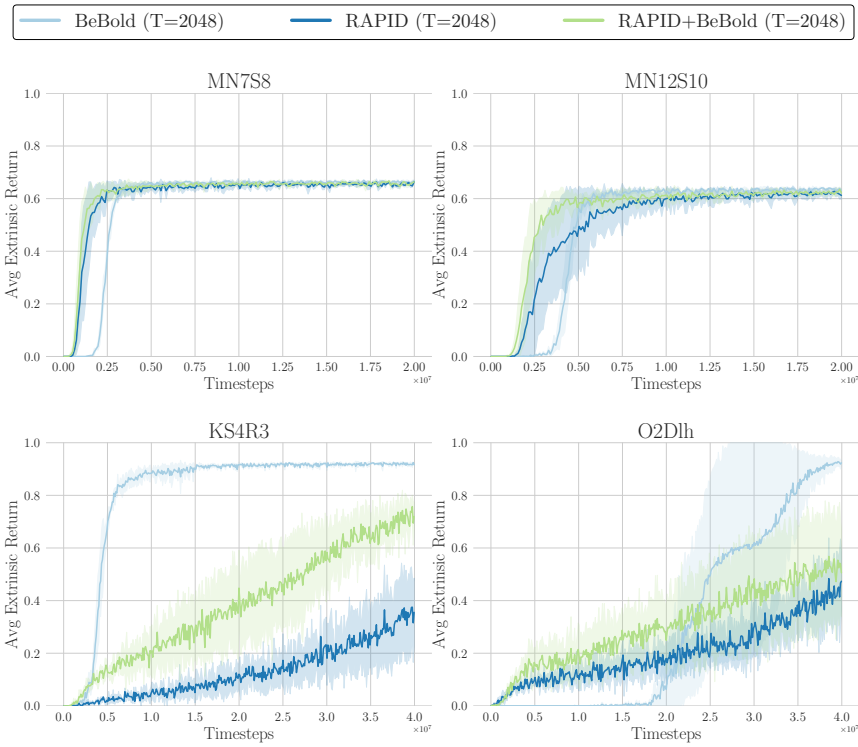


FIGURE 5.8: Results on multiple hard exploration procedurally-generated environments in MiniGrid when increasing the time horizon up to 2048 in on-policy (RL-loss) updates. Off-policy (supervised/imitation) updates remain with fixed batch size of 256.

ratio again with the new rollout size, the performance of both RAPID and RAPID+BeBold drastically changes, as informed in Figure 5.9. A better sample-efficiency can be noted when using a more conservative ratio (1:1, green and pink curves) in both `KS4R3` and `O2Dlh` with respect to the default episode termination setting (blue and light green results). This also occurs when decreasing the off-policy updates down to a 10:1 ratio (red and yellow curves). In this case, the convergence speed can be affected, although it manages to achieve the optimal policy in less steps (the 1:1 ratio struggles more to finally achieve it). In contrast, when applying those updates at the end of the episode, which corresponds with approximately a 1:4 ratio initially in `KS4R3` and `O2Dlh` (Table 5.1), results get worse, just surpassed by the BeBold approach. Concerning `MultiRoom` environments, increasing the number of off-policy updates seems to be a good strategy, which is difficult to be outperformed by any state-of-the-art solution. In fact, decreasing the frequency of the replayed experiences has a negative impact that can make the agent not learn in the absence of intrinsic rewards.

The above discussed behaviors strengthen the claim posed in this chapter:

the off-policy loss can help improve the learning process, although using it in excess can be counter-productive. This is related to what is actually aiming to replay and if it is worth the value that update for the agent at that moment.

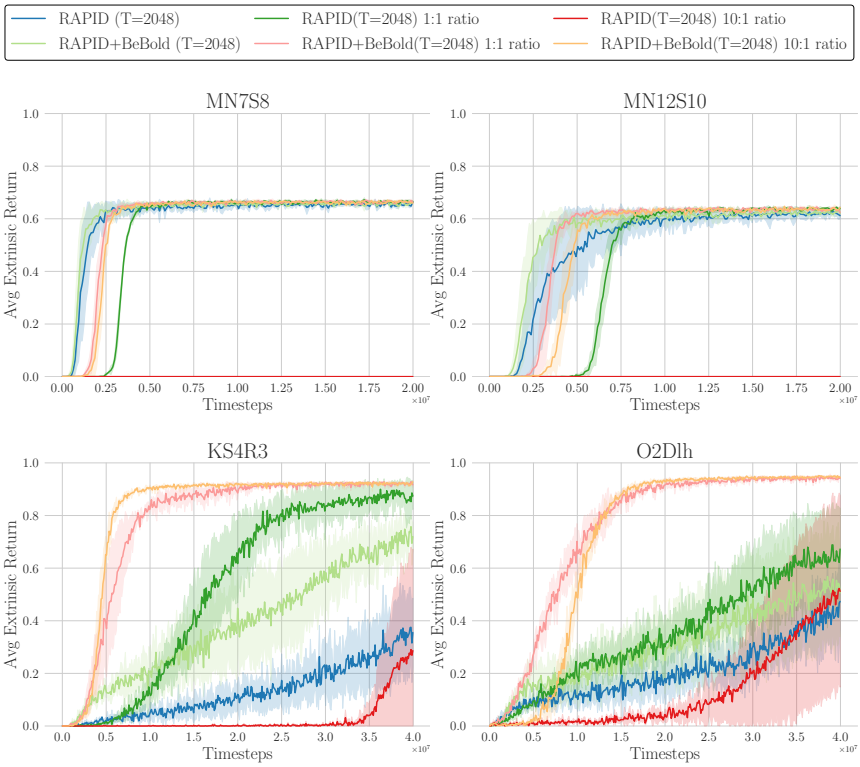


FIGURE 5.9: Ratios as in Fig 5.7, but when the rollout size is increased to 2048. Recall the default ratio is dynamically adjusted based on when the episode finishes (see Table 5.1).

5.5 Discussion and Limitations

The agent will have an easier time learning certain levels due to their difficulty, the way in which they are explored or even by how many times some patterns are reproduced in subsequent episodes. What is more, the agent knowledge changes over the curse of the training, modifying the optimal strategy replay correspondingly. Therefore, *what to replay and when to do it* is not straightforward. This section discusses three critical aspects that directly affect the presented framework.

5.5.1 Environment Requirements

We have noticed outstanding results in either (Zha, Ma, et al., 2021) and this chapter for `MultiRoom`. However, such benefits are not so clear when

tackling other scenarios. We hypothesize that this is due to the latent knowledge needed to interact with the environment and to accomplish the task. In other words: *what the agent must actually learn in each environment to accomplish the task.*

On the one hand, in `MultiRoom`'s environments the agent, independently of the destination, must always move forward until it discovers a door of any color. At this moment, the agent opens the door and again proceeds alike until reaching the goal. Hence, the agent must learn that the way to solve the task is to find the next door as soon as possible, and opens it with the consequent action. This behavior can be inferred and faster learnt when exploiting past episodes, as all this information does not change from one level to another. On the other hand, in `KS4R3` and `O2Dlh` the agent must learn how to interact with multiple objects (the door can be closed or locked), the relationship between objects and their utility (a key is used to open lock doors), and the information embedded in the colors of such objects (the key of a given color only unlocks the door of that same color). Furthermore, the location of doors, keys and the destination change from one level to another. Therefore, exploiting past good episodes does not necessarily mean to be the best strategy, because the stored episodes might be biased to certain patterns that can influence on the agent's learned knowledge (i.e. blue doors represent the best strategy to achieve the goal), hindering what it actually has to learn.

5.5.2 Intra-inter Level Diversity

A degree of diversity in the buffer is desired when the environment evolves during training, as it can maximize the chances of having useful information in the buffer for the agent's learning at the moment. In this context, RAPID's hyper-parameterization selection is decisive for the buffer configuration[6], prioritizing the emulation of past levels with high **intra-episodic** diversity of states through local bonus. On the other hand, a long-term exploration is encouraged via the global bonus, which can be regarded as a way to address the diversity across levels (**inter-level**). Nevertheless, ensuring it may not be necessarily effective, as the observations of certain levels can also be present in other levels due to similarities of the observation space. A representative example of this bonus not being enough (either by conceptualization or hyper-parametrization) is depicted in Figure 5.10, where the inter-level diversity is nil after some point during the training process.

These statically weighted, fixed scores, with no prior insights about what the agent actually manages to solve at each moment, make it impossible to ensure that the buffer contents will be advantageous to learn conveniently generalizable knowledge. As a consequence, the agent can be forced to imitate levels that do not hold any guarantees of usefulness when exploiting generalization.

[6]For the sake of comparison, in this chapter parameters are set to the same values as in the original paper (Zha, Ma, et al., 2021).

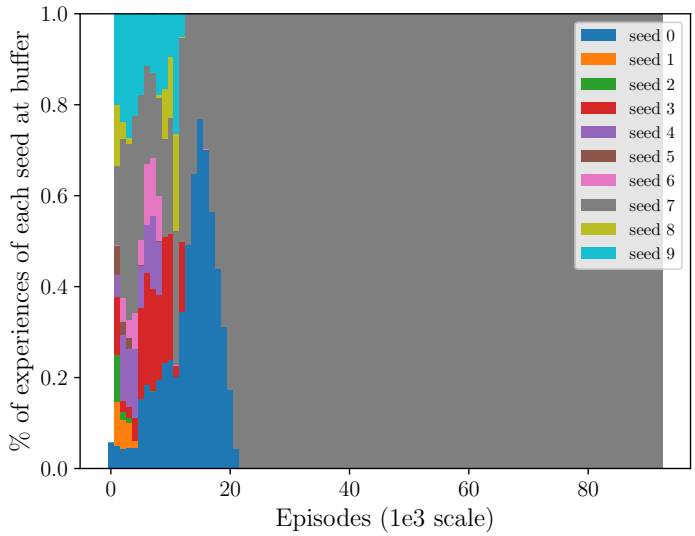


FIGURE 5.10: Experiences stored in the ranking buffer through the training process in `O2Dlh` when fixing the number of training levels to 10. The y-axis represents the percentage of experiences corresponding to each level stored in the buffer. Across those levels we have an average optimal number of steps of 18, being the level with seed 7 the one with a configuration that requires the minimum steps to solve the environment with just 13 steps. After 20,000 training episodes, the agent just stores and imitates greedily the level corresponding to seed 7.

5.5.3 Suboptimal Demonstration Replay

Another concerning issue with IL techniques is the replay of non-optimal demonstrations, which can potentially make the agent learn a suboptimal policy. In *self-IL* methods this is even more exacerbated, as the agent has to deal with a large amount of suboptimal tuples throughout the training process.

In fact *the way the sparse extrinsic reward is calculated in PCG environments* is of utmost relevance, as it is used as a prioritizing measure. The reward function is commonly designed taking into account the number of steps until the goal is achieved. In MiniGrid, the reward is calculated as in Expression (2.23). Nevertheless, the levels have a minimum number of optimal steps based on the sorted configuration. Hence, *the same reward in different levels would not necessarily represent the same degree of optimality.*

The aforementioned issue also occurs in our framework (and other *self-IL* solutions (Gangwani et al., 2019; Guo et al., 2018; Oh et al., 2018)) when using the extrinsic reward for prioritizing some experiences over others. Consequently, *the agent will be more encouraged to learn from suboptimal demonstrations of those* easy *configured levels* just because solving them in an suboptimal fashion requires less steps – higher return – than doing it

optimally in other levels. Furthermore, the agent can behave greedily and bias its learning to those *easy* levels that do not necessarily represent the whole level distribution of the given task, and fail to provide actionable knowledge for generalization. What is more, even in the case that the stored episodes are optimal, there exists a risk that the agent just focuses on those episodes that might not yield the required inter-level diversity as explained above (see Figure 5.10). A valid solution to deal with the extrinsic return can be to normalize each level's score with respect to their optimal number of steps. However, this will require knowing the optimal number of steps for each level in advance, which can be far from being a realistic assumption in practical settings.

In summary, the hyper-specialization of the buffer may degrade the effectiveness of the off-policy learning process, no matter the amount of levels processed. In this sense, off-policy updates could be dynamically scheduled to maximize the contribution during the agent's training phase. Ideally, those updates should be selected from a buffer which fairly matches the optimal trajectories across levels, while ensuring the diversity needs for a better generalization.

5.6 Conclusions

This chapter has hypothesized that the use of Intrinsic Motivation can improve the sample efficiency of *self-IL* approaches in PCG sparse reward scenarios where the exploration needs hinder the collection of good episodes to be replayed.

Departing from this research hypothesis, a framework that combines RAPID and BeBold has been proposed. This framework has been experimentally proven to expose an equal or better sample efficiency when compared to RAPID, BeBold or SIL approaches on their own, when solving challenging tasks formulated over MiniGrid's procedurally-generated environments. We have shown that this advantage holds as long as the selected IM method is efficient enough. Furthermore, we highlight the necessity of scheduling correctly the on-policy and off-policy updates, as well as the use of a rollout size that spans multiple episodes in the same optimization step to reduce the variance and to achieve an optimal policy in the context of PCG problems.

Finally, we have discussed the reasons why off-policy updates do not necessarily contribute to the generalization of the agent's knowledge due to 1) the environment complexity and learning requirements; 2) the diversity between stored episodes in the buffer; and 3) the way in which prioritization is applied, as the reward function may not distinguish between optimal solutions across levels due to the different steps required to solve them to optimality.

Chapter 6

Concluding Remarks

The main motivation behind this Thesis was to analyze and make valuable contributions in the overcoming of some of the limitations currently hindering the application of RL solutions in real-world problems. As highlighted in Chapter 2, RL algorithms face a lot of challenges, ranging from the *conceptualization* (i.e., how to model the problem itself), *learning* (i.e., inherent algorithmic challenges), to their *operationalization* in a system (i.e., noisy/intermittent signals, memory and real-time inference constraints). Indeed, the large majority of real-world problems present the following characteristics:

- (1) difficulties to model problems as Markov Decision Problems; being Partially Observable Markov Decision Processes the most natural substitute,
- (2) objectives are not straightforward translatable to RL environments, especially the reward function definition, i.e., guidance signals, whose design must endow the algorithm with appropriate feedbacks for learning,
- and last but not least, (3) the implementation itself may alter the original problem. In fact, embedding a real use case into an artificial environment entails restricting it to the capabilities of the simulator. In turn, the deployment of the knowledge into the nature causes principally adaptability and safety concerns.

In this context, we have focused our research on POMDPs in sparse reward settings with no prior assumption or information with a view to reflect the most realistic scenario. More importantly, we aimed at decreasing the required number of interactions with the environment in order to converge faster to the optimal policy.

The Thesis contributions and conclusions are below outlined, along with future work suggestions that, in our humble opinion, are more likely to lead RL to an unprecedented performance:

• **Chapter 3**. *How to set a collaborative framework between heterogeneous agents with the aim of learning optimal policies faster (with less samples).* For this purpose, we proposed centralizing the critic, the curiosity module, or both of them. The performance of these schemes were thoroughly and

systematically evaluated by means of an ablation study. The conclusion drawn from this work can be summarised as follows:

- A centralized critic has greater stability and also leads to faster convergence of optimal policy. As long as the critic is centralized, centralizing also the curiosity module brings advantages that are most noticeable when considering the action to generate the exploration bonus.
- The use of IM converts the problem into a bi-objective function in which the explorative side may induce noise into the attainment of the main task objective, ultimately slowing down the learning.

One way to address these issues might be through decoupling the exploration and the exploitation behaviours by two different agents (Schäfer et al., 2022) or transforming the problem into a multi-objective approach (Hayes et al., 2021). An interesting avenue would be also reformulate our heterogeneous agent proposal into off-policy strategies (e.g., DQN) where the agents could share their replay buffers and benefit directly from episodes representing how others undertook the same task from different perspectives (Christianos et al., 2020). Additionally, tailoring techniques to leverage expert demonstrations so as to cope with the heterogeneity of the action spaces would be interesting to analyze (e.g., using IL techniques that only rely on observations and do not strictly depend on the actions (Torabi et al., 2018)).

• **Chapter 4**. *Analysing fairly the contribution to performance of the state-of-the-art IM algorithms.* IM techniques have been shown to be effective for promoting the exploration in RL. Nevertheless, it is not always clear if the proposals are superior due to the presence of novel reward-related procedures or to peripheral or additional design choices. On this ground, we conducted a study to try to detach both components and the conclusions were as follows:

- Using an adaptive intrinsic coefficient β based on the return of previous rollouts outperforms strategies relying on a fixed parameter.
- The inclusion of *episode-level* (e.g., episodic visitation counts) for the generation of intrinsic rewards are beneficial in comparison with disregarding *episode-level* information.
- Adopting different neural network architectures is critical to guarantee the success. Indeed, when reducing the number of parameters of the IM modules the performance is deteriorated, which gets even worse if the actor-critic parameters are also decreased.

In future extensions, the study of more environments (e.g., Procgen, with high-dimensional observations (Cobbe, Hesse, et al., 2020)) and more IM algorithms to solve efficiently hard exploration environments would be of great interest.

• **Chapter 5**. *How to collect good trajectories to improve self-IM algorithms performance* Attracted by the idea of replaying not only good

trajectories in terms of performance but also novel trajectories, we proposed the use of IM to promote exploration and discover episodes with interesting properties for the agent's learning. We evinced that:

- As long as the selected IM approach and fitting is appropriate, the benefits are clear.
- The method is sensitive to the diversity of the replayed trajectories and the rollout size, i.e. when to execute the updates of the agent's policy. These are decisive to make the agent generalize well to the whole level distribution of the task.

We firmly believe that the results can be improved even more if the diversity of the trajectories is guaranteed; this is, if the demonstrations are not biased and represent the whole level distribution. In addition, more effective ways to manage the scheduling of losses (or even the combination of them in a single loss function (Rajeswaran et al., 2018)) should be studied as well.

6.1 List of Publications

As a result of the research conducted during the development of this PhD Thesis, several contributions were published in conferences and journals related to the areas of reinforcement learning and neural networks:

- **Journal publications:**
  - **Alain Andres**, Esther Villar-Rodriguez and Javier Del Ser, "Collaborative training of heterogeneous reinforcement learning agents in environments with sparse rewards: what and when to share?" Neural Computing & Applications, published on-line, 2022. https://doi.org/10.1007/s00521-022-07774-5 (IF: 5.102, Q2, 45/145 ARTIFICIAL INTELLIGENCE).
- **Conference contributions:**
  - **Alain Andres**, Esther Villar-Rodriguez, Aritz D. Martinez and Javier Del Ser, "Collaborative Exploration and Reinforcement Learning between Heterogeneously Skilled Agents in Environments with Sparse Rewards," 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, pp. 1-10, 2021. https://doi.org/10.1109/IJCNN52387.2021.9534146.
  - **Alain Andres**, Esther Villar-Rodriguez and Javier Del Ser, "An Evaluation Study of Intrinsic Motivation Techniques Applied to Reinforcement Learning over Hard Exploration Environments," in: A. Holzinger, P. Kieseberg, A. M. Tjoa, E. Weippl (eds). Machine Learning and Knowledge Extraction (CD-MAKE 2022), Lecture Notes in Computer Science, vol 13480, Springer, 2022. https://doi.org/10.1007/978-3-031-14463-9_13

– **Alain Andres**, Esther Villar-Rodriguez and Javier Del Ser, "Towards Improving Exploration in Self-Imitation Learning using Intrinsic Motivation," IEEE Symposium Series on Computational Intelligence (SSCI), Singapore, pp. 890-899, 2022. https://doi.org/10.1109/SSCI51031.2022.10022199
– **Alain Andres**, Lukas Schäfer, Esther Villar-Rodriguez, Stefano V. Albrecht and Javier Del Ser, "Using Offline Data to Speed-up Reinforcement Learning in Procedurally Generated Environments," Adaptive and Learning Agents (ALA) Workshop at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), accepted, London, UK, 2023.

6.2 Future Research Lines

This Thesis concludes by outlining future research lines that have been identified as interesting directions during the PhD Thesis:

As we have highlighted during this document, sample-efficiency is crucial in RL because despite simulators provide unlimited number of interactions with a good throughput rate, in real-world the systems are actually slow, fragile and expensive to operate, preventing the adoption of RL solutions. This is translated in having a high cost in terms of agent-environment interactions.

One way to overcome it is using offline data to speed up the learning. Imitation Learning approaches have shown an incredible potential as long as demonstrations are available, although their success is usually highly dependant to the quality, quantity and also the diversity of the trajectories. Indeed, we analyzed this issue in PCG environments in a paper that is currently under review –"Using Offline Data to Speed-up Reinforcement Learning in Procedurally Generated Environments"– where IL could overfit the model towards the provided examples. As explained in Section 2.3.2, the most broadly used IL technique is BC due its simplicity and good results. However, better results can be expected when using more advanced techniques such as adversarial IL (Ho & Ermon, 2016; Orsini et al., 2021), curriculum strategies that prioritize demonstrations over others (Bajaj et al., 2022) and even using approaches that take into account temporal dependencies (Paine et al., 2019). Akin to Imitation Learning, Offline RL focus on how to learn in the absence of online interactions. This subfield of RL has shown promising results when having data that do not resemble a demonstration but random data or when being trained with suboptimal and noisy data (Kumar et al., 2022). However, this kind of algorithms exhibit challenges regarding the distribution shift between the offline data and the actual problem distribution, reason why some approaches constrain the policy to not deviate too far from the behavior policy (Kostrikov et al., 2021; Kumar et al., 2020); whereas others focus on prioritizing the usage of experiences to maximize the data coverage or the discovery of skills (H. Liu & Abbeel, 2021a, 2021b), ultimately learning a good representation and a versatile policy (Yang & Nachum, 2021). In

view of the necessities and potential of these techniques, using offline data envisages an exciting path.

Another fascinating branch is the one related to Representation Learning and few-shot learning, which are closely related when generalization is pursued. The ability to understand and discover automatically the key features that govern a task is indeed a game-changer, as it brings the policy with the capacity to quickly adapt when changes in the environment are made (e.g., goal modification, state domain variation), minimizing the total number of online interactions with the environment within the RL domain (X. Chen et al., 2021). Nevertheless, how learn a valid representation is not trivial, requiring sometimes to have different representations between the actor's policy and the critic (Cobbe, Hilton, et al., 2020; Raileanu & Fergus, 2021). In fact, value-based methods might have some issues when it becomes to generalization capabilities (Ehrenberg et al., 2022; Lyle et al., 2022), which can explain why the large majority of off-policy solutions (that tend to be more sample-efficient than their on-policy counterparts) struggle in PCG environments (Ehrenberg et al., 2022; Mohanty et al., 2021).

Last but not least, we feature world models (Ha & Schmidhuber, 2018; Wu et al., 2022) and unsupervised environment design (Dennis et al., 2020; Parker-Holder et al., 2022) as a proxy to avoid the large costs of real-world environment interactions by the virtue of using techniques (e.g. generative models) to generate new instances of the problem without the necessity of explicitly having access to the environment itself.

Appendix A

Random Network Distillation - Limitations

One of the critical aspects when using any prediction error method is how the scale of rewards can vary, not only between environments, but also at points in time in the same scene, making it difficult the selection of hyper-parameters. Additionally, if such IM approach uses DL, the normalization of inputs is important for an appropriate prediction. Nonetheless, the latter, is crucial when using RND, as the target network's parameters are frozen and hence can not adjust the scale of the upcoming observations.

According to the recommendations (Burda, Edwards, Storkey, et al., 2018), we normalized the observations as in Expression (3.7). Unexpectedly, we find out that the reward scale was biased towards the features of each room in ViZDooM environment. In order to account for that issue, we proceed as follows:

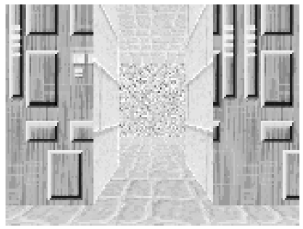
- First, we select observations gather by the agent at different points of the Setup 3 shown in Figure 3.10, which results in the visualizations shown in Figure A.1.
- Afterwards, we train the predictor network, $\hat{\phi}(\cdot)$, during 100 consecutive randomly sampled episodes, and we store both the frozen –$\phi(\cdot)$– and trained predictor networks parameters.
- Finally, we evaluate which would have been the the obtained intrinsic reward at the selected checkpoints after each episode's updates.

The evolution of the intrinsic rewards considering different changes are shown in Figure A.2. Overall, it can be seen that there is a trend in all the checkpoints to decrease the intrinsic reward over time. However, it is not consistent with the novelty we are pursuing, as the points that rarely might have been visited –the ones that are far from the start position and are very difficult to be experienced without knowledge (e.g., 49 and 50) – have lower bonus respect to others that are closer to the spawn location and that are more often observed (e.g., 0 or 14). In fact, the largest values are given always for observations at rooms 22 and 24. We also experiment if the issue was related to how the input was processed by either providing higher dimensions and using RGB images instead of the default grayscale

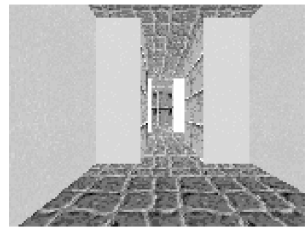
configuration (Figure A.2, middle), or by the adopted ANN architecture (Figure A.2, bottom). Nevertheless, there was no significant changes except the amplitude of the novelty signal.

Therefore, we conclude that RND presents unforeseen difficulty to capture the actual curiosity and should be taken into account when being used in ViZDooM.

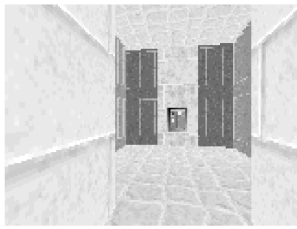
(0) Initial spawn position looking forward



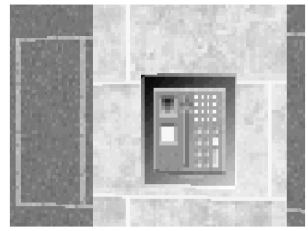
(14) At room 13 looking forward



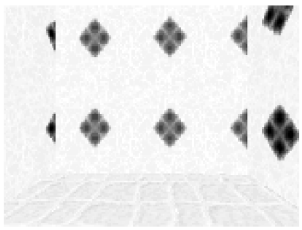
(22) At corridor 16 oriented to the door



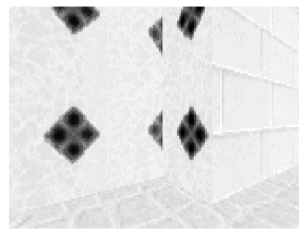
(23) At room 17 in front of the door



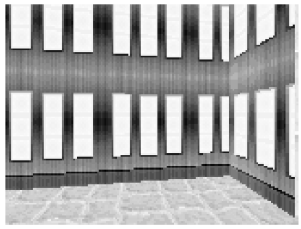
(40) At room 22 oriented to the wall



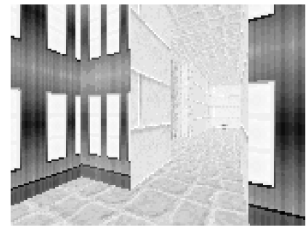
(41) At room 22 partially oriented to the next corridor



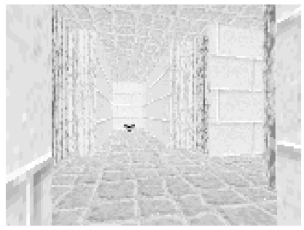
(46) At room 24 oriented to the wall



(47) At room 24 partially oriented to the next corridor



(49) At corridor 25 oriented to goal/vest



(50) At room 26 oriented to the goal/vest

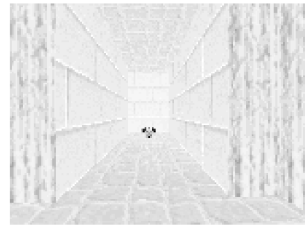


FIGURE A.1: Observations (grayscale,120x160) at 10 different checkpoints of VizDoom's *My way home* environment.

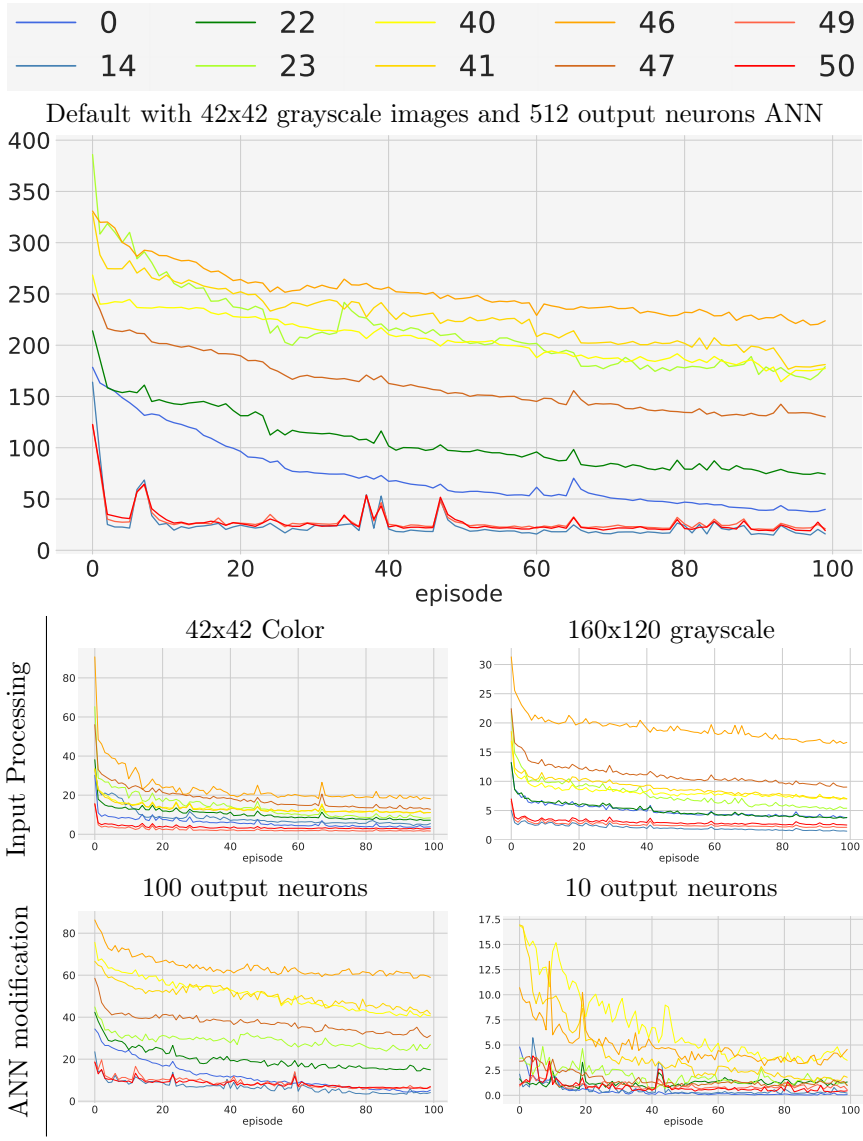


FIGURE A.2: Intrinsic rewards evolution throughout 100 randomly sampled episodes at different checkpoints explained in Figure A.1. Cold colors represent locations that are close to the spawn position and farther from the goal/vest. At the top row the default performance with 42x42 grayscale images and the adopted ANN architecture is shown; the middle row shows the impact when varying the input image by either using 42x42 colored images (left) or 160x120 images; the bottom row results illustrate how changes in the ANN architecture affect when using 100 output neurons (left) or just 10 output neurons (right).

Bibliography

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *21st International Conference on Machine Learning (ICML)*, 1.

Abolafia, D. A., Norouzi, M., Shen, J., Zhao, R., & Le, Q. V. (2018). Neural Program Synthesis with Priority Queue Training [arXiv:1801.03526].

Anderson, C. W. (1986). Learning and Problem-Solving with Multilayer Connectionist Systems (Adaptive Strategy Leraning, Neural Networks, Reinforcement Learning). *Doctoral Dissertation*, 1–260.

Andres, A., Villar-Rodriguez, E., & Del Ser, J. (2022). An Evaluation Study of Intrinsic Motivation Techniques Applied to Reinforcement Learning over Hard Exploration Environments. In A. Holzinger, P. Kieseberg, A. M. Tjoa, & E. Weippl (Eds.), *Machine Learning and Knowledge Extraction* (pp. 201–220). Springer International Publishing.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., & Bachem, O. (2021a). What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study [arXiv:2006.05990].

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., & Bachem, O. (2021b). What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study. *9th International Conference on Learning Representations (ICLR)*.

Aubret, A., Matignon, L., & Hassas, S. (2019). A survey on intrinsic motivation in reinforcement learning [arXiv:1908.06976].

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, *47*(2), 235–256.

Aytar, Y., Pfaff, T., Budden, D., Paine, T., & Wang, Z. (2018). Playing hard exploration games by watching YouTube. *Advances in Neural Information Processing Systems (NeurIPS)*, 12.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., & Blundell, C. (2020). Agent57: Outperforming the Atari Human Benchmark. *37th International Conference on Machine Learning (ICML)*, *119*.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., & Blundell, C. (2020). Never Give Up: Learning Directed Exploration Strategies. *International Conference on Learning Representations (ICLR)*.

Bain, M., & Sammut, C. (2001). A Framework for Behavioural Cloning, 37.

Bajaj, V., Sharon, G., & Stone, P. (2022). Task Phasing: Automated Curriculum Learning from Demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., … Petersen, S. (2016). DeepMind Lab.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A Distributional Perspective on Reinforcement Learning [arXiv:1707.06887]. *34th International Conference on Machine Learning*.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents [arXiv:1207.4708]. *Journal of Artifical Intelligence Research*, *47*(1), 253–279.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying Count-Based Exploration and Intrinsic Motivation. *Advances in Neural Information Processing Systems (NeurIPS)*.

Böhmer, W., Rashid, T., & Whiteson, S. (2019). Exploration with Unreliable Intrinsic Reward in Multi-Agent Reinforcement Learning [arXiv:1906.02138]. *2nd Exploration in Reinforcement Learning Workshop at the International Conference on Machine Learning (ICML)*.

Bougie, N., & Ichise, R. (2021). Fast and slow curiosity for high-level exploration in reinforcement learning. *Applied Intelligence*, *51*(2), 1086–1107.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., & Efros, A. A. (2018). Large-Scale Study of Curiosity-Driven Learning [arXiv:1808.04355]. *International Conference on Learning Representations (ICLR)*.

Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). Exploration by Random Network Distillation [arXiv:1810.12894].

Buşoniu, L., Babuška, R., & De Schutter, B. (2010). Multi-agent Reinforcement Learning: An Overview. In J. Kacprzyk, D. Srinivasan, & L. C. Jain (Eds.), *Innovations in Multi-Agent Systems and Applications - 1* (pp. 183–221). Springer Berlin Heidelberg.

Calvo, J. A., & Dusparic, I. (2018). Heterogeneous Multi-Agent Deep Reinforcement Learning for Traffic Lights Control. *26th Irish Conference on Artificial Intelligence and Cognitive Science*, 2–13.

Cesa-Bianchi, N., Gentile, C., Lugosi, G., & Neu, G. (2017). Boltzmann Exploration Done Right. *Advances in Neural Information Processing Systems (NeurIPS)*, 10.

Charoenpitaks, K., & Limpiyakorn, Y. (2019). Curiosity-Driven Exploration Effectiveness on Various Environments. *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 1–6.

Chen, D., Mei, J.-P., Wang, C., Feng, Y., & Chen, C. (2020). Online Knowledge Distillation with Diverse Peers. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chen, X., Toyer, S., Wild, C., Emmons, S., Fischer, I., Lee, K.-H., Alex, N., Wang, S. H., Luo, P., Russell, S., Abbeel, P., & Shah, R. (2021). An Empirical Investigation of Representation Learning for Imitation [Datasets and Benchmarks Track]. *Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, Z., & Lin, M. (2020). Self-Imitation Learning in Sparse Reward Settings [https://arxiv.org/abs/2010.06962].

Chevalier-Boisvert, M., Willems, L., & Pal, S. (2018). Minimalistic Gridworld Environment for Gymnasium.

Chitnis, R., Tulsiani, S., Gupta, S., & Gupta, A. (2020). Intrinsic Motivation for Encouraging Synergistic Behavior. *International Conference on Learning Representations (ICLR)*.

Christianos, F., Papoudakis, G., Rahman, A., & Albrecht, S. V. (2021). Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, *139*, 1989–1998.

Christianos, F., Schäfer, L., & Albrecht, S. V. (2020). Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*, 10707–10717.

Cobbe, K., Hesse, C., Hilton, J., & Schulman, J. (2020). Leveraging Procedural Generation to Benchmark Reinforcement Learning. *37th International Conference on Machine Learning (ICML)*.

Cobbe, K., Hilton, J., Klimov, O., & Schulman, J. (2020). Phasic Policy Gradient. *38th International Conference on Machine Learning (ICML)*.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., & Schulman, J. (2019). Quantifying Generalization in Reinforcement Learning. *36th International Conference on Machine Learning*.

Colas, C., Karch, T., Sigaud, O., & Oudeyer, P.-Y. (2022). Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: A Short Survey. *Journal of Artificial Intelligence Research*, *74*.

Da Silva, F. L., Warnell, G., Costa, A. H. R., & Stone, P. (2019). Agents teaching agents: A survey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems*, *34*(1), 9.

Dai, T., Du, Y., Fang, M., & Bharath, A. A. (2022). Diversity-augmented intrinsic motivation for deep reinforcement learning. *Neurocomputing*, *468*, 396–406.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., & Levine, S. (2020). Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. *Advances in Neural Information Processing Systems (NeurIPS)*, 13.

Dimakopoulou, M., Osband, I., & Roy, B. V. (2018). Scalable Coordinated Exploration in Concurrent Reinforcement Learning.

Dimakopoulou, M., & Roy, B. V. (2018). Coordinated Exploration in Concurrent Reinforcement Learning. *Advances in Neural Information Processing Systems*.

Du, Y., Han, L., Fang, M., Liu, J., Dai, T., & Tao, D. (2019). LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, *32*.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, *110*(9), 2419–2468.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., & Clune, J. (2021). First return, then explore. *Nature*, *590*(7847), 580–586.

Ehrenberg, A., Kirk, R., Jiang, M., Grefenstette, E., & Rocktäschel, T. (2022). A Study of Off-Policy Learning in Environments with Procedural Content Generation. *Workshop on Generalizable Policy Learning in Physical World (ICLR)*.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *35th International Conference on Machine Learning (ICML)*.

Filos, A. (2019). Reinforcement Learning for Portfolio Management [Masther Thesis at Imperial College London].

Finn, C., Levine, S., & Abbeel, P. (2016). Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *International Conference on Machine Learning (ICML)*.

Flet-Berliac, Y., Ferret, J., Pietquin, O., Preux, P., & Geist, M. (2021). Adversarially Guided Actor-Critic [arXiv:2102.04376 [cs, stat]]. *International Conference on Learning Representations (ICLR)*.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2017). Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Fu, Q., Han, Z., Chen, J., Lu, Y., Wu, H., & Wang, Y. (2022). Applications of reinforcement learning for building energy efficiency control: A review. *Journal of Building Engineering*, *50*, 104165.

Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *35th International Conference on Machine Learning (ICML)*.

Gangwani, T., Liu, Q., & Peng, J. (2019). Learning Self-Imitating Diverse Policies. *International Conference on Learning Representations (ICLR)*.

Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., & Darrell, T. (2019). Reinforcement Learning from Imperfect Demonstrations.

Gneiting, T., & Raftery, A. E. (2007). Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, *102*(477), 359–378.

Gokhale, A. A. (1995). Collaborative Learning Enhances Critical Thinking. *Journal of Technology Education*, *7*(1).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Gottesman, O., Johansson, F., Komorowski, M., Faisal, A., Sontag, D., Doshi-Velez, F., & Celi, L. A. (2019). Guidelines for reinforcement learning in healthcare. *Nature Medicine*, *25*(1), 16–18.

Grigorescu, D. (2020). Curiosity, intrinsic motivation and the pleasure of knowledge [Publisher: Petroleum - Gas University of Ploiesti]. *Journal of Educational Sciences & Psychology*, *10*(1), 16–23.

Guo, Y., Choi, J., Moczulski, M., Feng, S., Bengio, S., Norouzi, M., & Lee, H. (2021). Memory Based Trajectory-conditioned Policies for Learning from Sparse Rewards. *Advances in Neural Information Processing Systems (NeurIPS)*.

Guo, Y., Oh, J., Singh, S., & Lee, H. (2018). Generative Adversarial Self-Imitation Learning [arXiv:1812.00950].

Ha, D., & Schmidhuber, J. (2018). Recurrent World Models Facilitate Policy Evolution. *Advances in Neural Information Processing Systems (NeurIPS)*.

Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement Learning with Deep Energy-Based Policies. *34th International Conference on Machine Learning (ICML)*.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *35th International Conference on Machine Learning (ICML)*.

Hausknecht, M., & Stone, P. (2015). Deep Recurrent Q-Learning for Partially Observable MDPs. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., Howley, E., Irissappane, A. A., Mannion, P., Nowé, A., Ramos, G., Restelli, M., Vamplew, P., & Roijers, D. M. (2021). A Practical Guide to Multi-Objective Reinforcement Learning and Planning. *Autonomous Agents and Multi-Agent Systems*, *36*(1), 26.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2019). Deep Reinforcement Learning that Matters. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2017). Rainbow: Combining Improvements in Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z., & Gruslys, A. (2017). Deep Q-learning from Demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ho, J., & Ermon, S. (2016). Generative Adversarial Imitation Learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Ho, J., Gupta, J. K., & Ermon, S. (2016). Model-Free Imitation Learning with Policy Optimization. *33rd International Conference on Machine Learning (ICML)*.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Hasselt, H. v., & Silver, D. (2018). Distributed Prioritized Experience Replay. *International Conference on Learning Representations (ICLR)*.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2017). VIME: Variational Information Maximizing Exploration. *Advances in Neural Information Processing Systems (NeurIPS)*.

Hua, J., Zeng, L., Li, G., & Ju, Z. (2021). Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning. *Sensors*, *21*(4), 1278.

Huang, B.-Y., & Tsai, S.-C. (2022). Solving hard-exploration problems with counting and replay approach. *Engineering Applications of Artificial Intelligence*, *110*, 104701.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., & Levine, S. (2021). How to Train Your Robot with Deep Reinforcement Learning; Lessons We've Learned. *The International Journal of Robotics Research*, *40*(4-5), 698–721.

Iqbal, S., & Sha, F. (2019). Coordinated Exploration via Intrinsic Rewards for Multi-Agent Reinforcement Learning [arXiv:1905.12127].

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P. A., Strouse, D. J., Leibo, J. Z., & de Freitas, N. (2019). Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. *36th International Conference on Machine Learning (ICML)*.

Jing, X., Zhu, Z., Li, H., Pei, X., Bengio, Y., Che, T., & Song, H. (2021). Divide and Explore: Multi-Agent Separate Exploration with Shared Intrinsic Motivations.

Johnson, D. W., & Others, A. (1994). *Cooperative Learning in the Classroom*. Association for Supervision; Curriculum Development, 1250 N.

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2020). Unity: A General Platform for Intelligent Agents [arXiv:1809.02627].

Juliani, A., Khalifa, A., Berges, V.-P., Harper, J., Teng, E., Henry, H., Crespi, A., Togelius, J., & Lange, D. (2019). Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. *28Th International Joint Conference on Artificial Intelligence (IJCAI)*.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., & Dabney, W. (2019). Recurrent Experience Replay in Distributed Reinforcement Learning. *International Conference on Learning Representations (ICLR)*, 19.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaśkowski, W. (2016). ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning [The 2022 IEEE Transactions on Games Outstanding Paper Award]. *IEEE Transactions on Games*, *11*(3), 248–259.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes [arXiv:1312.6114].

Kolter, J. Z., & Ng, A. Y. (2009). Near-Bayesian exploration in polynomial time. *26th International Conference on Machine Learning (ICML)*, 1–8.

Kostrikov, I., Nair, A., & Levine, S. (2021). Offline Reinforcement Learning with Implicit Q-Learning [arXiv:2110.06169].

Kumar, A., Hong, J., Singh, A., & Levine, S. (2022). When Should We Prefer Offline Reinforcement Learning Over Behavioral Cloning? *International Conference on Learning Representations (ICLR)*.

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative Q-Learning for Offline Reinforcement Learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Kurek, M., & Jaśkowski, W. (2016). Heterogeneous team deep q-learning in low-dimensional multi-agent environments. *IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Küttler, H., Nardelli, N., Miller, A., Raileanu, R., Selvatici, M., Grefenstette, E., & Rocktäschel, T. (2020). The NetHack Learning Environment. *Advances in Neural Information Processing Systems*, *33*, 7671–7684.

Lai, K.-H., Zha, D., Li, Y., & Hu, X. (2020). Dual Policy Distillation. *29th International Joint Conference on Artificial Intelligence (IJCAI)*.

Levine, S. (2021). Understanding the World Through Action. *Conference on Robot Learning (CoRL), Blue Sky Track*.

Li, Y. (2019). Reinforcement Learning Applications [arXiv:1908.06973].

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*.

Lin, K., Wang, S., & Zhou, J. (2017). Collaborative Deep Reinforcement Learning. *ACM Woodstock conference*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*(3), 293–321.

Liu, E. Z., Raghunathan, A., Liang, P., & Finn, C. (2021). Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices. *38th International Conference on Machine Learning (ICML)*.

Liu, H., & Abbeel, P. (2021a). APS: Active Pretraining with Successor Features. *38th International Conference on Machine Learning (ICML)*.

Liu, H., & Abbeel, P. (2021b). Behavior From the Void: Unsupervised Active Pre-Training. *Advances in Neural Information Processing Systems (NeurIPS)*.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Advances in Neural Information Processing Systems (NeurIPS)*.

Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., & Kim, D. I. (2019). Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Communications Surveys & Tutorials*, *21*(4), 3133–3174.

Lyle, C., Rowland, M., Dabney, W., Kwiatkowska, M., & Gal, Y. (2022). Learning Dynamics and Generalization in Reinforcement Learning. *39th International Conference on Machine Learning (ICML)*.

Machado, M. C., Bellemare, M. G., & Bowling, M. (2019). Count-Based Exploration with the Successor Representation. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Martinez, A. D., Del Ser, J., Osaba, E., & Herrera, F. (2021). Adaptive Multifactorial Evolutionary Optimization for Multitask Reinforcement Learning. *IEEE Transactions on Evolutionary Computation*, *26*(2), 233–247.

Mirjalili, S. (2019). Genetic Algorithm. In S. Mirjalili (Ed.), *Evolutionary Algorithms and Neural Networks: Theory and Applications* (pp. 43–55). Springer International Publishing.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *33rd International Conference on Machine Learning (ICML)*.

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., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Mohanty, S., Poonganam, J., Gaidon, A., Kolobov, A., Wulfe, B., Chakraborty, D., Šemetulskis, G., Schapke, J., Kubilius, J., Pašukonis, J., Klimas, L., Hausknecht, M., MacAlpine, P., Tran, Q. N., Tumiel, T., Tang, X., Chen, X., Hesse, C., Hilton, J., … Cobbe, K. (2021). Measuring Sample Efficiency and Generalization in Reinforcement Learning Benchmarks: NeurIPS 2020 Procgen Benchmark. *Machine Learning Research (PMLR)*, *133*, 361–395.

Nair, A., Gupta, A., Dalal, M., & Levine, S. (2021). AWAC: Accelerating Online Reinforcement Learning with Offline Datasets.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Overcoming Exploration in Reinforcement Learning with

Demonstrations. *International Conference on Robotics and Automation (ICRA)*.

Nguyen & Widrow. (1989). The truck backer-upper: An example of self-learning in neural networks. *Joint Conference on Neural Networks*, 357–363 vol.2.

Nian, R., Liu, J., & Huang, B. (2020). A review On reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, *139*, 106886.

Nichol, A., Pfau, V., Hesse, C., Klimov, O., & Schulman, J. (2018). Gotta Learn Fast: A New Benchmark for Generalization in RL.

Ning, K.-P., & Huang, S.-J. (2020). Reinforcement Learning with Supervision from Noisy Demonstrations [arXiv:2006.07808]. *ACM Symposium on Neural Gaze Detection*.

Ning, K.-P., Xu, H., Zhu, K., & Huang, S.-J. (2021). Co-Imitation Learning without Expert Demonstration [arXiv:2103.14823]. *Workshop on Reincarnating Reinforcement Learning (ICLR)*.

Oh, J., Chockalingam, V., Singh, S., & Lee, H. (2016). Control of Memory, Active Perception, and Action in Minecraft. *33rd International Conference on Machine Learning (ICML)*.

Oh, J., Guo, Y., Singh, S., & Lee, H. (2018). Self-Imitation Learning. *35th International Conference on Machine Learning*.

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., Salimans, T., … Zhang, S. (2019). Dota 2 with Large Scale Deep Reinforcement Learning.

Orsini, M., Raichuk, A., Hussenot, L., Vincent, D., Dadashi, R., Girgin, S., Geist, M., Bachem, O., Pietquin, O., & Andrychowicz, M. (2021). What Matters for Adversarial Imitation Learning? *Advances in Neural Information Processing Systems (NeurIPS)*.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., & Peters, J. (2018). An Algorithmic Perspective on Imitation Learning. *Foundations and Trends in Robotics*, *7*(1-2), 1–179.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Van Roy, B., Sutton, R., Silver, D., & Van Hasselt, H. (2020). Behaviour Suite for Reinforcement Learning. *International Conference on Learning Representations (ICLR)*.

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks [arXiv:1511.08458].

Ostrovski, G., Bellemare, M. G., Oord, A. v. d., & Munos, R. (2017). Count-Based Exploration with Neural Density Models. *34th International Conference on Machine Learning (ICML)*.

Oudeyer, P.-Y., Gottlieb, J., & Lopes, M. (2016). Intrinsic motivation, curiosity, and learning. In *Progress in Brain Research* (pp. 257–284). Elsevier.

Paine, T. L., Gulcehre, C., Shahriari, B., Denil, M., Hoffman, M., Soyer, H., Tanburn, R., Kapturowski, S., Rabinowitz, N., Williams, D.,

Barth-Maron, G., Wang, Z., de Freitas, N., & Team, W. (2019). Making Efficient Use of Demonstrations to Solve Hard Exploration Problems [deepmind paper arXiv:1909.01387].

Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2016). Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. *International Conference on Learning Representations (ICLR)*.

Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., & Rocktäschel, T. (2022). Evolving Curricula with Regret-Based Environment Design. *39th International Conference on Machine Learning (ICML)*.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven Exploration by Self-supervised Prediction. *34th International Conference on Machine Learning (ICML)*.

Pathak, D., Gandhi, D., & Gupta, A. (2019). Self-Supervised Exploration via Disagreement. *36th International Conference on Machine Learning (ICML)*.

Pîslar, M., Szepesvari, D., Ostrovski, G., Borsa, D., & Schaul, T. (2022). When should agents explore? *International Conference on Learning Representations (ICLR)*.

Pomerleau, D. A. (1988). ALVINN: An Autonomous Land Vehicle in a Neural Network. *Advances in Neural Information Processing Systems*, *1*.

Pritzel, A., Uria, B., Srinivasan, S., Puigdomènech, A., Vinyals, O., Hassabis, D., Wierstra, D., & Blundell, C. (2017). Neural Episodic Control. *34th International Conference on Machine Learning (ICML)*.

Pshikhachev, G., Egorov, V., Ivanov, D., & Shpilman, A. (2021). Self-Imitation Learning from Demonstrations. *Deep RL Workshop (NeurIPS)*, 12.

Raileanu, R., & Fergus, R. (2021). Decoupling Value and Policy for Generalization in Reinforcement Learning. *38th International Conference on Machine Learning (ICML)*.

Raileanu, R., & Rocktäschel, T. (2020). RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. *International Conference on Learning Representations (ICLR)*.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., & Levine, S. (2018). Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations [arXiv:1709.10087]. *Robotics: Science and Systems (RSS)*.

Reddy, S., Dragan, A. D., & Levine, S. (2019). SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards. *International Conference on Learning Representations (ICLR)*.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., & de Freitas, N. (2022). A Generalist Agent [arXiv:2205.06175]. *Transactions on Machine Learning Research*, 42.

Ross, S., & Bagnell, D. (2010). Efficient Reductions for Imitation Learning. *13th International Conference on Artificial Intelligence and Statistics*, 661–668.

Ross, S., Gordon, G. J., & Bagnell, J. A. (2011). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning [arXiv:1011.0686]. *14th International Conference on Artificial Intelligence and Statistics*.

Rosser, C., & Abed, K. (2021). Curiosity-Driven Reinforced Learning of Undesired Actions in Autonomous Intelligent Agents. *IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 39–42.

Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using Connectionist Systems, 21.

Russell, S., & Norvig, P. (2022). *Artificial Intelligence: A Modern Approach, 4th US ed.*

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., & Hadsell, R. (2016). Policy Distillation. *International Conference on Learning Representations (ICLR)*.

Rusu, A. A., Flennerhag, S., Rao, D., Pascanu, R., & Hadsell, R. (2022). Probing Transfer in Deep Reinforcement Learning without Task Engineering. *1st Conference on Lifelong Learning Agents*.

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, *25*(1), 54–67.

Salimans, T., & Chen, R. (2018). Learning Montezuma's Revenge from a Single Demonstration. *Advances in Neural Information Processing Systems (NeurIPS)*.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., & Gelly, S. (2019). Episodic Curiosity through Reachability. *International Conference on Learning Representations (ICLR)*.

Schafer, L. (2019). Curiosity in Multi-Agent Reinforcement Learning [Masther Thesis, University of Edinburgh].

Schäfer, L., Christianos, F., Hanna, J. P., & Albrecht, S. V. (2022). Decoupled Reinforcement Learning to Stabilise Intrinsically-Motivated Exploration. *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal Value Function Approximators. *32nd International Conference on Machine Learning (ICML)*, 1312–1320.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized Experience Replay. *International Conference on Learning Representations (ICLR)*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., & Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604–609.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2017). Trust Region Policy Optimization. *32nd International Conference on Machine Learning (ICML)*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-Dimensional Continuous Control Using Generalized Advantage Estimation. *International Conference on Learning Representations (ICLR)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms [arXiv:1707.06347].

Seurin, M., Strub, F., Preux, P., & Pietquin, O. (2021). Don't Do What Doesn't Matter: Intrinsic Motivation with Action Usefulness. *30th International Joint Conference on Artificial Intelligence (IJCAI)*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. *31st International Conference on Machine Learning (ICML)*, 9.

Silver, D., Newnham, L., Barker, D., Weller, S., & McFall, J. (2013). Concurrent Reinforcement Learning from Customer Interactions. *30th International Conference on Machine Learning (ICML)*, 924–932.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359.

Song, G., & Chai, W. (2018). Collaborative Learning for Deep Neural Networks. *Advances in Neural Information Processing Systems*.

Sovrano, F. (2019). Combining Experience Replay with Exploration by Random Network Distillation. *IEEE Conference on Games (CoG)*, 1–8.

Stadie, B. C., Levine, S., & Abbeel, P. (2015). Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models. *International Conference on Learning Representations (ICLR)*.

Stanton, C., & Clune, J. (2018). Deep Curiosity Search: Intra-Life Exploration Can Improve Performance on Challenging Deep Reinforcement Learning Problems [arXiv:1806.00553]. *Deep RL Workshop (NeurIPS)*.

Stooke, A., & Abbeel, P. (2019). Accelerated Methods for Deep Reinforcement Learning [arXiv:1803.02811].

Strehl, A. L., & Littman, M. L. (2008). An analysis of model-based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, *74*(8), 1309–1331.

Sutton, R. S. (1990). Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In B.

Porter & R. Mooney (Eds.), *Machine Learning Proceedings 1990* (pp. 216–224). Morgan Kaufmann.

Sutton, R. S. (1995). Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. *Advances in Neural Information Processing Systems*, *8*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd Edition). The MIT Press.

Taïga, A. A., Fedus, W., Machado, M. C., Courville, A., & Bellemare, M. G. (2020). On Bonus-Based Exploration Methods in the Arcade Learning Environment. *International Conference on Learning Representations (ICLR)*.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2017). #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Taylor, M. E., & Stone, P. (2009). Transfer Learning for Reinforcement Learning Domains: A Survey. *The Journal of Machine Learning Research*, *10*.

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., McAleese, N., Bradley-Schmieg, N., Wong, N., Porcel, N., Raileanu, R., Hughes-Fitt, S., Dalibard, V., & Czarnecki, W. M. (2021). Open-Ended Learning Leads to Generally Capable Agents [deepmind paper arXiv:2107.12808].

Terry, J. K., Grammel, N., Son, S., & Black, B. (2022). Parameter Sharing For Heterogeneous Agents in Multi-Agent Reinforcement Learning [arXiv:2005.13625].

Thrun, S. B. (1992). *Efficient Exploration In Reinforcement Learning* (Technical Report). Carnegie Mellon University. USA.

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral Cloning from Observation. *27th International Joint Conference on Artificial Intelligence (IJCAI)*.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., & Tassa, Y. (2020). Dm_control: Software and tasks for continuous control. *Software Impacts*, *6*, 100022.

Ugadiarov, L., Skrynnik, A., & Panov, A. I. (2021). Long-Term Exploration in Persistent MDPs [arXiv:2109.10173]. *Mexican International Conference on Artificial Intelligence (MICAI)*, *13067*.

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*.

Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., & Riedmiller, M. (2018). Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards [arXiv:1707.08817].

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Wakilpoor, C., Martin, P. J., Rebhuhn, C., & Vu, A. (2020). Heterogeneous Multi-Agent Reinforcement Learning for Unknown Environment Mapping [arXiv:2010.02663]. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wang, T., Wang, J., Wu, Y., & Zhang, C. (2019). Influence-Based Multi-Agent Exploration. *International Conference on Learning Representations (ICLR)*.

Wang, Z., Dai, Z., Póczos, B., & Carbonell, J. (2019). Characterizing and Avoiding Negative Transfer. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. *33rd International Conference on Machine Learning (ICML)*.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Whitehead, S. D., & Ballard, D. H. (1991). Learning to Perceive and Act by Trial and Error. *Machine Learning*, *7*(1), 45–83.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3), 229–256.

Wu, P., Escontrela, A., Hafner, D., Goldberg, K., & Abbeel, P. (2022). DayDreamer: World Models for Physical Robot Learning. *Conference on Robot Learning (CoRL)*.

Yang, M., & Nachum, O. (2021). Representation Matters: Offline Pretraining for Sequential Decision Making. *38th International Conference on Machine Learning (ICML)*.

Yu, T., Quillen, D., He, Z., Julian, R., Narayan, A., Shively, H., Bellathur, A., Hausman, K., Finn, C., & Levine, S. (2019). Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. *Conference on Robot Learning (CoRL)*.

Zha, D., Lai, K.-H., Zhou, K., & Hu, X. (2019). Experience Replay Optimization. *28th International Joint Conference on Artificial Intelligence (IJCAI)*.

Zha, D., Lai, K.-H., Zhou, K., & Hu, X. (2021). Simplifying Deep Reinforcement Learning via Self-Supervision [arXiv:2106.05526].

Zha, D., Ma, W., Yuan, L., Hu, X., & Liu, J. (2021). Rank the Episodes: A Simple Approach for Exploration in Procedurally-Generated Environments. *International Conference on Learning Representations (ICLR)*.

Zhang, C., Vinyals, O., Munos, R., & Bengio, S. (2018). A Study on Overfitting in Deep Reinforcement Learning [arXiv:1804.06893].

Zhang, T., Rashidinejad, P., Jiao, J., Yuandong, T., Gonzalez, J. E., & Russell, S. (2021). MADE: Exploration via Maximizing Deviation from Explored Regions. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., & Tian, Y. (2020). BeBold: Exploration Beyond the Boundary of Explored Regions [arXiv:2012.08621].

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., & Tian, Y. (2022). NovelD: A Simple yet Effective Exploration Criterion. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, Y., Xiang, T., Hospedales, T. M., & Lu, H. (2017). Deep Mutual Learning. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zheng, H., Wei, P., Jiang, J., Long, G., Lu, Q., & Zhang, C. (2020). Cooperative Heterogeneous Deep Reinforcement Learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zheng, Z., Oh, J., & Singh, S. (2018). On Learning Intrinsic Rewards for Policy Gradient Methods. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhu, Z., Lin, K., Dai, B., & Zhou, J. (2020). Learning Sparse Rewarded Tasks from Sub-Optimal Demonstrations [arXiv:2004.00530].

Zhu, Z., Lin, K., Jain, A. K., & Zhou, J. (2020). Transfer Learning in Deep Reinforcement Learning: A Survey [arXiv:2009.07888].

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A Comprehensive Survey on Transfer Learning [arXiv:1911.02685].

Zolna, K., Rostamzadeh, N., Bengio, Y., Ahn, S., & Pinheiro, P. O. (2019). Reinforced Imitation in Heterogeneous Action Space [arXiv:1904.03438]. *Imitation Learning and its Challenges in Robotics Workshop (NeurIPS)*.