Vers la généralisation de l'apprentissage par renforcement

Quentin Gallouédec

NNT : 2024ECDL0013

THÈSE DE DOCTORAT

réalisée au sein de l'École Centrale de Lyon
délivrée par l'École Doctorale N°512

Spécialité : Informatique

Toward the Generalization of Reinforcement Learning

par
Quentin GALLOUÉDEC

le 26 Mars 2024

Jury

Président	David FILLIAT	Professeur des universités ENSTA Paris
Rapporteur	Guillaume ALLIBERT	Professeur des universités Université Côte d'Azur
Rapporteur	Paul HONEINE	Professeur des universités Université de Rouen Normandie
Examinatrice	Laëtitia MATIGNON	Maîtresse de conférence Université Claude Bernard Lyon 1
Examinateur	Olivier SIGAUD	Professeur des universités Sorbonne Université
Directeur de thèse	Emmanuel DELLANDRÉA	Maître de conférence HDR École Centrale de Lyon

“La simplicité est la sophistication suprême.”

— Léonard de Vinci

Abstract

Conventional Reinforcement Learning (RL) involves training a unimodal agent on a single, well-defined task, guided by a gradient-optimized reward signal. This framework does not allow us to envisage a learning agent adapted to real-world problems involving diverse modality streams, multiple tasks, often poorly defined, sometimes not defined at all. Hence, we advocate for transitioning towards a more general framework, aiming to create RL algorithms that more inherently versatile. To advance in this direction, we identify two primary areas of focus.

The first aspect involves improving exploration, enabling the agent to learn from the environment with reduced dependence on the reward signal. We present Latent Go-Explore (LGE), an extension of the Go-Explore algorithm. While Go-Explore achieved impressive results, it was constrained by domain-specific knowledge. LGE overcomes these limitations, offering wider applicability within a general framework. In various tested environments, LGE consistently outperforms the baselines, showcasing its enhanced effectiveness and versatility. The second focus is to design a general-purpose agent that can operate in a variety of environments, thus involving a multimodal structure and even transcending the conventional sequential framework of RL. We introduce Jack of All Trades (JAT), a multimodal Transformer-based architecture uniquely tailored to sequential decision tasks. Using a single set of weights, JAT demonstrates robustness and versatility, competing with its unique baseline on several RL benchmarks and even showing promising performance on vision and textual tasks. We believe that these two contributions are a valuable step towards a more general approach to RL. In addition, we present other methodological and technical advances that are closely related to our core research question. The first is the introduction of a set of sparsely rewarded simulated robotic environments designed to provide the community with the necessary tools for learning under conditions of low supervision. Notably, three years after its introduction, this contribution has been widely adopted by the community and continues to receive active maintenance and support. On the other hand, we present Open RL Benchmark, our pioneering initiative to provide a comprehensive set of and fully tracked RL experiments, going beyond typical data to include all algorithm-specific and system metrics. This benchmark aims to improve research efficiency by providing out-of-the-box RL data and facilitating accurate reproducibility of experiments. With its community-driven approach, it has quickly become an important resource, documenting over 25,000 runs.

These technical and methodological advances, along with the scientific contributions described above, are intended to promote a more general approach, we hope, represent a meaningful step toward the eventual development of a more versatile RL agent.

Résumé

L'apprentissage par renforcement conventionnel implique l'entraînement d'un agent unimodal sur une tâche unique et bien définie, guidé par un signal de récompense optimisé pour le gradient. Ce cadre ne nous permet pas d'envisager un agent d'apprentissage adapté aux problèmes du monde réel impliquant des flux de diverses modalités, des tâches multiples, souvent mal définies, voire pas définies du tout. C'est pourquoi nous préconisons une transition vers un cadre plus général, visant à créer des algorithmes d'apprentissage par renforcement plus adaptables et intrinsèquement polyvalents.

Pour progresser dans cette direction, nous identifions deux domaines d'intérêt principaux. Le premier est l'amélioration de l'exploration, qui permet à l'agent d'apprendre de l'environnement en dépendant le moins possible du signal de récompense. Nous présentons *Latent Go-Explore* (LGE), une généralisation de l'algorithme Go-Explore qui, malgré ses résultats impressionnants, était limité par une forte contrainte de connaissance du domaine. LGE atténue ces limitations et permet une application plus large dans un cadre plus général. LGE démontre son efficacité et sa polyvalence accrues en surpassant de manière significative les lignes de base dans tous les environnements testés. Le deuxième domaine d'intérêt est celui de la conception d'un agent polyvalent qui peut fonctionner dans une variété d'environnements, impliquant ainsi une structure multimodale et transcendant même le cadre séquentiel conventionnel de l'apprentissage par renforcement. Nous présentons *Jack of All Trades* (JAT), une architecture multimodale basée Transformers, spécialement conçue pour les tâches de décision séquentielle. En utilisant un seul ensemble de poids, JAT démontre sa robustesse et sa polyvalence, rivalisant avec son unique référence sur plusieurs benchmarks d'apprentissage par renforcement et montrant même des performances prometteuses sur des tâches de vision et textuelles. Nous pensons que ces deux contributions constituent une étape importante vers une approche plus générale de l'apprentissage par renforcement. En outre, nous présentons d'autres avancées méthodologiques et techniques qui sont étroitement liées à notre question de recherche initiale. La première est l'introduction d'un ensemble d'environnements robotiques simulés à récompense éparse, conçus pour fournir à la communauté les outils nécessaires à l'apprentissage dans des conditions de faible supervision. Trois ans après son introduction, cette contribution a été largement adoptée par la communauté et continue de faire l'objet d'une maintenance et d'un support actifs. D'autre part, nous présentons Open RL Benchmark, notre initiative pionnière visant à fournir un ensemble complet et entièrement enregistré d'expériences d'apprentissage par renforcement, allant au-delà des données typiques pour inclure toutes les métriques spécifiques à l'algorithme et au système. Ce benchmark vise à améliorer l'efficacité de la recherche en fournissant des données prêtes à l'emploi et en

facilitant la reproductibilité précise des expériences. Grâce à son approche communautaire, il est rapidement devenu une ressource importante, documentant plus de 25 000 exécutions.

Ces avancées techniques et méthodologiques, associées aux contributions scientifiques décrites ci-dessus, visent à promouvoir une approche plus générale de l'apprentissage par renforcement et, nous l'espérons, représentent une étape significative vers le développement à terme d'un agent plus polyvalent.

Remerciements

Tout d'abord, je tiens à exprimer ma sincère gratitude à Emmanuel Dellandréa, mon directeur de thèse, pour son soutien proactif et inébranlable tout au long de ces trois ans et demi de recherche. La confiance qu'il m'a accordée et sa disponibilité constante ont joué un rôle déterminant dans ce travail.

Ensuite, je souhaite remercier vivement les membres du jury et du comité de suivi de thèse, Guillaume Allibert, David Filliat, Paul Honeine, Laëtitia Matignon, Olivier Sigaud et Johanna Delanoy, pour avoir immédiatement accepté de consacrer leur temps précieux et pour leur investissement à l'évaluation de mon travail.

Ma gratitude va également à mes collègues et collaborateurs, Nicolas Cazin, Isabelle Dominique, Aqib Khan, Alexandre Chapin, Costa Huang, Antonin Raffin, Edward Beeching, Clément Romac, avec lesquels j'ai eu la chance de travailler au cours des trois dernières années. Nos discussions stimulantes, nos idées partagées et notre travail d'équipe harmonieux ont été d'une valeur inestimable.

Je souhaite également remercier mes professeurs, Françoise Le Guiner, Céline Lefebvre, Alberto Bosio, et un clin d'œil à Cédric Faure, avec qui j'ai écrit ma première ligne de code. Leur exigence, leur application et leur implication ont sans doute un rayonnement bien plus profond qu'ils ne l'imaginent.

Enfin, je voudrais remercier ceux que j'aime, Papa, Maman, Valentin, Clarisse, pour leur soutien indéfectible, ainsi que pour tant d'autres raisons que la brièveté de ce propos ne me permet pas d'évoquer.

Contents

List of Figures

List of Tables

List of Algorithms

N.B.: The algorithms in this document are deliberately simplified to foster readability and understanding, focusing on their core elements. While this enhances clarity, it is important to note that omitted minor aspects, such as environment reset and initialization, handling of non-bootstrapping cases, and buffer size considerations, are nonetheless important for practical implementation.

Acronyms

A2C Advantage Actor-Critic

ACKTR Actor-Critic with Kronecker-factored Trust Region

AI Artificial Intelligence

ALE Arcade Learning Environment

API Application Programming Interface

ARS Augmented Random Search

BPE Byte-Pair Encoding

CI Confidence Interval

CLI Command-Line Interface

CNN Convolutional Neural Network

COCO Common Objects in Context

CPU Central Processing Unit

CV Computer Vision

D4PG Distributed Distributional Deep Deterministic Policy Gradients

DDPG Deep Deterministic Policy Gradient

DIAYN Diversity Is All You Need

DP Dynamic Programming

DPG Deterministic Policy Gradient

DQN Deep Q-Network

DT Decision Transformer

FQF Fully Parameterized Quantile Function

GAE Generalized Advantage Estimation

GAIL Generative Adversarial Imitation Learning

GAN Generative Adversarial Network

GLIE Greedy in the Limit with Infinite Exploration

GPT Generative Pre-trained Transformer

GPU Graphics Processing Unit

HER Hindsight Experience Replay

HRL Hierarchical Reinforcement Learning

ICML International Conference on Machine Learning

IQM Interquartile Mean

IQN Implicit Quantile Network

IRL Inverse Reinforcement Learning

IS Importance Sampling

JAT Jack of All Trades

LfD Learning from Demonstration

LGE Latent Go-Explore

LLM Large Language Model

LSH Locality-Sensitive Hashing

LSTM Long Short-Term Memory

MAML Model-Agnostic Meta-Learning

MARL Multi-Agent Reinforcement Learning

MBPO Model-Based Policy Optimization

MC Monte Carlo

MCTS Monte Carlo Tree Search

MDP Markov Decision Process

ME-TRPO Maximum Entropy Trust Region Policy Optimization

MORL Multi-Objective Reinforcement Learning

MSE Mean Squared Error

MUSIC Mutual Information-based State Intrinsic Control

NGU Never Give Up

NLP Natural Language Processing

PER Prioritized Experience Replay

PGE Procedurally-Generated Environments

POMDP Partially Observable Markov Decision Process

PPO Proximal Policy Optimization

QR-DQN Quantile Regression Deep Q-Network

REDQ Randomized Ensembled Double Q-Learning

ReLU Rectified Linear Unit

RIDE Rewarding Impact-Driven Exploration

RL Reinforcement Learning

RLHF Reinforcement Learning with Human Feedback

RND Random Network Distillation

SAC Soft Actor-Critic

SARSA State-Action-Reward-State-Action

SB3 Stable-Baselines3

SGD Stochastic Gradient Descent

SIL Self-Imitation Learning

SoRB Search on Replay Buffer

TD Temporal Difference

TD3 Twin Delayed Deep Deterministic Policy Gradient

TQC Truncated Quantile Critics

TRPO Trust Region Policy Optimization

UCB Upper Confidence Bound

VIME Variational Information Maximizing Exploration

ViT Visual Transformer

VLM Visual Language Model

VPG Vanilla Policy Gradient

VQ-VAE Vector Quantized Variational Autoencoder

Notations

Symbol	Description	Definition
$\doteq$	Equality true by definition	
$\propto$	Proportional to	
$\odot$	Element-wise multiplication	
$\leftarrow$	Assignment in algorithm	
$\mathbb{N}$	Natural numbers	
$\mathbb{N}_k$	Natural numbers smaller than $k \in \mathbb{N}$	
$\mathbb{R}$	Real numbers	
Capital letter	Random variable	
$\cap_{X \in E} X$	Intersection of all events in E	
$\mathbb{1}$	Indicator function, $\mathbb{1}_A = 1$ if A is true, 0 otherwise	
$P(X = x)$	Probability that the random variable X takes on the value x	
$P(X = x \mid Y = y)$	Conditional probability of X given Y	
$X = x, Y = y$	Event $X = x$ and $Y = y$	
$\mathbb{E}[X]$	Expectation on X	
$\mathbb{E}[X \mid Y = y]$	Expectation of X conditional on $Y = y$	
$X \sim p$	X sampled from distribution $p(x) \doteq P(X = x)$	
$\mathcal{N}(\mu, \sigma^2), \mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ. σ^2 is the variance and Σ the covariance.	
$\mathcal{H}$	Entropy	
$D_{\mathrm{KL}}(p \mid\mid q)$	Kullback–Leibler divergence of p from q [139]	
$\nabla_x f(x)$	For a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$, gradient of f with respect to x, evaluated at the point x. This gradient is a vector in $\mathbb{R}^n$, with each component representing the partial derivative of f with respect to the corresponding component of x. For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, transposed Jacobian matrix of f with respect to x, evaluated at point x. Each column is the gradient of f in the dth dimension of $\mathbb{R}^m$.	
$\nabla_x f(x)\|_{x=y}$	Gradient of f with respect to x, but evaluated at a specific point y.	
$\mathcal{S}$, $\mathcal{S}^-$, $\mathcal{S}^+$	State space	2.1.1 and 2.1.6

Symbol	Description	Definition
$\mathcal{A}$, $\mathcal{A}(s)$	Action space	2.1.2
$\mathcal{R}$, $\mathcal{R}(s,a)$	Reward space	2.1.3
t	Discrete timestep ($t \in \mathbb{N}$)	
s, S_t	State (at timestep t)	
a, A_t	Action (at timestep t)	
r, R_t	Reward (at timestep t)	
$p(r, s' \mid s, a)$	Transition function	2.1.5
$p(s' \mid s, a)$	State transition function	2.1.7
$p(r \mid s, a)$	Reward distribution	2.1.8
T	Horizon	2.1.14
h, H_t	History (up to timestep t)	2.1.10
τ	Episode (or trajectory)	2.1.11
γ	Discount factor	2.1.15
g, G_t	Return	2.1.15
$G_{t:t+n}$	n-steps return	2.2.1
G_t^λ	λ return	2.2.2
π	Policy	2.1.16
Π	Set of policies	2.1.17
V^π	State-value function	2.1.18
Q^π	Action-value function	2.1.19
A^π	Advantage function	2.1.20
θ	Policy parameter ($\theta \in \mathbb{R}^n$)	
ϕ	Value function parameter ($\phi \in \mathbb{R}^n$)	
ρ	State distribution	2.3.3 to 2.3.8

Chapter 1

Introduction

It is a rather big ambition: to teach how to learn. This is the definition, or at least the intuition, that helps to explain Reinforcement Learning (RL). The ambition goes beyond that of supervised learning, which is *only* about teaching. The idea is thrilling, but it has taken a long time to put it into practice, until the great successes of the last decade. These include reaching human level in simple retro games [187, 188], surpassing the best human players in complex board games like chess and Go [250, 251], and even reaching Grandmaster status in StarCraft II [276]. These impressive and encouraging successes seem to be paving the way for the emergence of autonomous agents—entities perceiving and interacting independently with their environment—both in terms of their behavior and their learning. As we delve deeper into RL, we begin to realize that there are still major barriers to overcome. In this thesis, we identify some of these barriers and provide solutions to better understand and overcome them.

1.1 Problems and Objectives

In the field of RL, the most widespread approach involves training an agent to maximize a reward signal in a given environment. While this methodology has undoubtedly produced some impressive achievements, it is rooted in a somewhat narrow perspective that is difficult to apply to the complexities of real-world scenarios. The conventional RL paradigm assumes that an agent is faced with a single, clearly defined task, supported by an available and sufficiently informative reward signal. However, this framework weakens when extended beyond the confines of well-defined tasks and controlled environments.

In this pursuit, we advocate for a shift in focus towards making RL more general and adaptable. Instead of resorting to specialized, intricate tricks that tailor RL algorithms to highly unique settings, our aim is to develop methodologies that enhance the general applicability of RL. This approach involves creating versatile and robust algorithms that can operate effectively across a diverse range of environments and tasks, without the need for context-specific tuning. Embracing this perspective is crucial for the evolution of RL into a tool that is not just theoretically sound but also practically relevant in a variety of real-world contexts.

The limitations of traditional RL become apparent when considering its

application to real-world problems, which are often multifaceted and not easily encapsulated by a single reward function. To bridge this gap between theoretical RL and its practical, real-world application, it is imperative to broaden our approach and relax the rigid assumptions underpinning current methodologies. Our work is thus motivated by the need to adapt and extend RL to more complex, realistic scenarios, moving beyond simplistic control problems or retro games.

To achieve this, we have identified several major objectives that form the cornerstone of our research. These objectives are critical in reshaping RL to be more flexible, adaptable, and applicable to a diverse array of real-world challenges. They include:

- **Improving autonomous exploration**: One of the main aims is to develop methods for autonomous exploration that do not rely on informative reward signals. The aim is to create general strategies that are robust to various possible dynamics, enabling an agent to navigate and learn from its environment independently, without the need for externally defined rewards.

- **Generalist agent development**: At the heart of our research is the creation of versatile agents capable of performing multiple tasks within a unified, multimodal framework. The goal is to develop a single-network agent that efficiently handles a diverse set of tasks and modalities. By integrating diverse data and experiences, such an agent would draw on a broad knowledge base, moving beyond task-specific learning to a more adaptive and general approach to RL. This is an evolution towards versatile agents capable of seamlessly navigating between different tasks, environments and data types.

Throughout our work, we have been confronted with a third crucial issue: the rationalization of effort. A major challenge we have observed is the difficulty of comparing and reproducing results from existing literature. In computer vision (CV)—although addressing only part of the issue—the establishment of datasets such as Pascal [75], COCO [166], and ImageNet [229], and their associated challenges and leaderboards, have played a key role in standardising performance evaluation and advancing the field. Despite commendable efforts by the community to promote open research and facilitate access to implementations, significant obstacles remain. One of the main reasons for this challenge is the lack of a universally accepted framework for testing, evaluating, reporting and sharing results, models and implementations.

Aware of the importance of filling this gap, we have devoted part of our research to developing and maintaining technical solutions to meet these needs. Our aim has been to propose tools and methodologies that not only facilitate but also standardize these processes in RL research. In doing so, we aim to simplify the replication and comparison of studies, thus contributing to a more cohesive and collaborative research environment.

1.2 Contributions and Publications

The array of challenges outlined in this thesis is multifaceted, spanning various dimensions of RL. Our contributions to addressing these challenges are captured

in a series of papers, some already published and others currently under review at leading conferences in the field. These contributions are both scientific and technical, reflecting our belief that advancing the field of RL hinges not just on theoretical breakthroughs but also on the ability of researchers to reproduce, apply, and effectively compare these findings.

Cell-Free Latent Go-Explore One of the main advances in our work concerns exploration methods in the context of RL. A fundamental foundation has been laid with Go-Explore [70, 71], an effective exploration method, albeit with significant constraints in terms of domain knowledge. This method requires a detailed understanding of the environment and its dynamics in order to divide it into a number of *cells*. On this basis, we developed Latent Go-Explore (LGE), an evolution of Go-Explore designed to relax these constraints. LGE abandons the traditional cell-based approach in favor of a learned latent representation of the environment. We have explored various methods of learning the representation, including inverse dynamics, forward dynamics and Vector-Quantized Variational AutoEncoder (VQ-VAE) [272], demonstrating that each can be a wise choice depending on the characteristics of the environment. The learned representation is used to estimate the latent visitation density, using a computationally light particle-based estimator. This estimated density is then used to sample goals, preferably in areas of low latent density, which are supposed to correspond to regions less frequented by the agent. Following a carefully selected list of intermediate subgoals, the agent is guided towards the final sampled goal. Once this goal has been reached, the agent continues its exploration from this point. LGE has proven its versatility in a multitude of contexts, whether visual or non-visual, low- or high-dimensional, and with full or partial observations. This innovation significantly extends the applicability of the Go-Explore method, enhancing its versatility and robustness. It has demonstrated superior performance in complex environments, excelling particularly in challenging scenarios such as Montezuma's Revenge. **LGE has been published at ICML 2023 [90]**.

Jack of All Trades, Expert of Some: a Multi-Purpose Transformer Agent Another important area of our research has been to address the limitations of RL applications, which are often confined to single environments or tasks. Inspired by the flexibility of Transformer models for Natural Language Processing (NLP), we set out to create a *Jack of All Trades network* (JAT), based on this model and designated to operate in a multimodal, multitasking and versatile way. JAT represents a paradigm shift, as it contains in a single network the ability to perform a wide range of different tasks, from sophisticated robot control to playing Atari games, and from engaging in conversations to describing images. The resulting model showed impressive performance, competing with its closest baseline Gato [224], despite Gato being significantly larger in size. It closely mirrors expert scores in the sequential decision-making domains on which it was trained. It also demonstrated promising capabilities in language processing and image comprehension tasks, further illustrating its effectiveness. To the best of our knowledge, this is the most advanced and efficient generalist agent for sequential decision-making. This pretrained agent is available as open source. We are also releasing the complete dataset used to train it, as well as all the expert agents used to create this dataset. At the time of writing, as far

as we know, this is the first dataset this diverse, and we hope it will continue to grow. **The work on JAT resulted in a paper [92], which is under review for ICML 2024.**

PandaGym[1]: Open-Source Goal-Conditioned Environments for Robotic Learning The RL community was in need of environments that were not only easy to customize but also freely accessible, open-source, and actively maintained, especially for evaluating complex robotic control tasks. Addressing this need, in 2020, we introduced PandaGym. This set of simulated robotic environments fulfills all these criteria and has significantly contributed to the field. The initial release of PandaGym included baseline results from leading algorithms, yet none were fully successful in solving all the tasks presented. **This work has been published at the 4th Workshop on Robotic Learning at NeurIPS 2021 [91].** Three years after its initial release, the community's enthusiastic adoption of PandaGym is evident. The environments have been downloaded tens of thousands of times, and the initial baselines have been surpassed multiple times. Numerous users have leveraged PandaGym to develop their own custom settings for research, adding new tasks and functionalities. The platform has not only evolved but also improved significantly, incorporating new tasks and enhancements, largely driven by the numerous contributions from the community. These contributions have led to the identification and rectification of some bugs, further refining the platform. Today, PandaGym's ecosystem is enriched with an abundance of pretrained models, tutorials, and courses that integrate it into their content. We believe that the journey of PandaGym underscores its significant contribution to the field, demonstrating how a well-maintained and community-driven platform can evolve over time, enriching research and broadening the scope of possibilities in RL.

Open RL Benchmark[2]: Comprehensive Tracked Experiments for Reinforcement Learning Proceeding from a concern similar to that of PandaGym, we believe that the RL literature often lacks a general framework, as well as a common method for reporting, comparing, and enabling the reproduction of previously obtained results. Firmly convinced that such a framework is essential for significant and solid progress, we propose a collaborative and open benchmark named Open RL Benchmark. This platform serves as a comprehensive collection of meticulously tracked and downloadable experiments in RL, going far beyond the traditional focus on the episodic return curve as a function of training steps. It offers an extensive range of training data, encompassing both static elements, like hyperparameters, and evolving values such as episodic return, loss metrics, episode length, or any algorithm-specific metrics. To ensure a complete picture of the training process, Open RL Benchmark also includes detailed information on software (e.g., libraries used) and hardware (e.g., number and models of GPUs), thereby providing a full understanding of the training environment. Emphasizing our focus on reproducibility, each experiment tracked within Open RL Benchmark is designed to include all the necessary information necessary for its exact replication. This includes pinned versions of dependencies and the exact commands used in the experiments. Such meticulous documentation

[1] `https://github.com/qgallouedec/panda-gym`
[2] `https://github.com/openrlbenchmark/openrlbenchmark`

ensures that researchers can not only analyze but also accurately replicate the studies, fostering an environment of transparent and verifiable scientific progress. Open RL Benchmark extends its capabilities far beyond simply facilitating standard comparisons. It enables researchers to conduct a wide range of analytical explorations. The platform enables conventional assessments, such as tracking episodic return against training steps. However, its true potential lies in opening up avenues for more innovative and less explored analysis. For example, researchers can compare the memory use of different algorithms, or assess the consistency of results obtained from different RL libraries. This range of analysis fosters a richer, more nuanced exploration of RL, paving the way for deeper understanding and advancing methodologies in the field. Finally, Open RL Benchmark is designed as an evolutionary tool, to be fed by the community. New runs are added every day, increasing the statistical robustness of the results and the spectrum of the benchmark. As of this writing, Open RL Benchmark includes over 25,000 tracked experiments from nearly fifty contributors. **Open RL Benchmark is the subject of a paper [118] to be submitted to the NeurIPS 2024 Datasets and Benchmarks track.**

1.3 Contents

In Chapter 2, we examine the essential concepts and theoretical frameworks of RL. This chapter serves as the cornerstone of the thesis, methodically outlining the fundamental principles of RL. We begin by examining the tabular case, providing a clear and comprehensive introduction to the fundamentals. This leads to a discussion of the integration of deep learning with RL, focusing on value-based deep RL approaches. The chapter then explores alternative approaches, first policy-based and then model-based. These sections aim to provide a balanced understanding of the theoretical aspects and state-of-the-art algorithms in this field. To conclude the chapter, we examine a series of emerging methodologies. With this introductory chapter, we lay a solid foundation for the extensive and multifaceted exploration of RL, paving the way for more advanced topics in subsequent chapters.

In Chapter 3, we focus on LGE, our contribution to exploration methods for RL. The chapter begins with a thorough review of existing methodologies, focusing particularly on Go-Explore [70, 71], its key contributions and its limitations. This review justifies the need for the advances we propose. At the heart of this chapter we detail the development of LGE. We describe the means used to make the approach more general and robust, in particular by using a latent representation of the environment rather than relying on predefined cells. The chapter then explores in detail the various aspects of LGE and examines how each of them contributes effectively to its success. Next, we engage in a detailed comparison of LGE's performance against various robust state-of-the-art methods in a wide range of environments. These include visual and non-visual contexts, as well as environments ranging from high- to low-dimensional spaces. We demonstrate the improved performance of LGE in complex situations, highlighting in particular its success in demanding environments such as Montezuma's Revenge. Through this analysis, we demonstrate that LGE offers a more general and efficient framework for exploration, outperforming current methods in the literature.

In Chapter 4 we focus on the development of JAT, a novel effort in our research to overcome the typical limitations of RL applications, which are often confined to single environments or tasks. The chapter begins with a detailed review of the emerging literature on multimodal general-purpose agents. It then focuses on the development of JAT, which has been inspired by the versatility of Transformer models [274] in NLP. We explain how JAT integrates a wide range of capabilities within a single network, explaining its multimodality, its ability to be used in sequential decision-making tasks, but also in text generation tasks. JAT demonstrates impressive performance, achieving scores comparable to expert benchmarks used in its training data. We also present its promising capabilities in NLP and image understanding. We then justify its architectural choices on the basis of several studies and examine the contribution of the different components to the functioning of JAT. The chapter concludes by charting new paths for RL research. These challenges include the development of strategies to effectively manage joint training on multiple tasks of varying complexity, each with its own unique dataset, whose volume and modalities may vary significantly. Finally, we motivate the idea that future work should also propose methods for integrating a wider range of data than that generated by experts. We justify that these developments are crucial to create adaptive and trainable models capable of handling the multiple facets of real-world RL scenarios.

In Chapter 5, we explore a range of works that represent key technical contributions to the RL community. Convinced that these contributions are crucial but often overlooked in facilitating research that effectively builds on (and not alongside) existing work, we present several tools designed to improve clarity, consistency, and reproducibility in this area. This chapter details our efforts in developing these tools, emphasizing their role in promoting a more integrated and coherent research. The chapter begins with an introduction to PandaGym, our open-source platform for simulated robotic learning. We discuss its development, key features, and the significant role it has played in the RL community. This section also includes a retrospective analysis of how PandaGym has been used and developed over the past three years, highlighting its lasting impact and relevance. We then introduce Open RL Benchmark, a comprehensive platform for tracking and sharing RL experiments. We explore its architecture, functionalities, and its central role in addressing the challenges of reproducibility and comparative analysis in RL research. The chapter also discusses the intended use of Open RL Benchmark, its current significance in the field, and our expectations for how the community will engage with and contribute to its development.

We conclude this document with the Chapter 6 dedicated to its synthesis, which aims to provide an overall view of the proposed contributions. We place particular emphasis on the links between these contributions in order to provide a coherent vision of what this work has achieved. We also propose a series of reflections that have emerged from this work, which we hope will help to better guide future research, both in its technical and scientific practice. We extend this concluding chapter by taking a step back and proposing a discussion of the ethical issues surrounding artificial intelligence. As with good technical and scientific practice, we believe that it is crucial to develop a reflection on these issues, especially now that AI has emerged from the laboratory and is already beginning to reshape the world. At a more individual level, we'll take the opposite approach, explaining not the contribution of my work to AI, but the

contribution of AI to my work. Similarly, we strongly believe that this reflection needs to be promoted and discussed by researchers now to ensure transparent and trustworthy research.

Chapter 2

Reinforcement Learning: From Core Principles to Cutting-Edge Algorithms

The basic idea of RL is *learning what to do*. How an agent should interact with its environment in order to maximize a reward signal. To become familiar with this field, several notions are essential. This chapter is intended to lay the foundations for understanding the contributions presented later in this document. We introduce the fundamental concepts underlying RL, in particular the Markov decision process (MDP), and give some of its properties. We then discuss learning algorithms, starting with the tabular case and extending to the general case using function approximators. We present some of the most widely used algorithms from the literature and conclude with an overview of specialized RL frameworks that are also currently at the forefront of active research. This chapter draws upon components from [261], [283] and [1].

2.1 Introduction to Reinforcement Learning

2.1.1 Overall process

In this first section, we give the intuition of the process. A sequential decision-making process is composed of an environment and an agent, which interact with each other.

The interaction process is sequential, which means that the interactions occur only at certain discrete instants. At each timestep, the agent takes the action, according to the state of the environment. Then the environment changes its internal state according to this action. The state is the set of variables that describe the environment. The environment returns this new state to the agent. It also returns a reward, measuring *how well the agent does*. Considering this new state, and the reward, the agent takes a new action, and the process starts again. The interaction process ends when the agent reaches a so-called terminal state. The goal of the agent is to receive as much reward as possible. He must therefore learn to choose the actions that produce the greatest amount of reward.

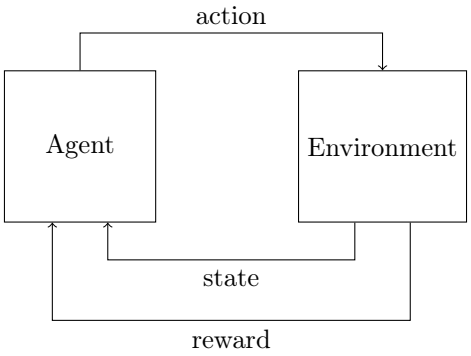


Figure 2.1: Schematic representation of the interaction between the agent and the environment.

2.1.2 Markov Decision Process

State, action, reward

Definition 2.1.1. The *state space*, denoted $\mathcal{S}$, is defined as an arbitrary set (which can be finite or infinite, continuous or not). An element of this set is called a *state*, and is used to describe the state of the environment.

Definition 2.1.2. Let $s \in \mathcal{S}$ be a state. The *action space in* s, denoted by $A(s)$, is defined as the set of all possible actions that can be taken in state s.

Remark 1. For simplicity, it is frequently assumed that the set of possible actions does not depend on the specific state, implying that the action space is constant across all states in S. Under this assumption, the action space is simply denoted as A rather than $A(s)$ for each state $s \in \mathcal{S}$. Henceforth, we shall refer to A as the *action space*, and this will be the standing assumption throughout this document.

Remark 2. If both the state and action spaces are finite, the problem is categorized as a *tabular problem*. This case is discussed extensively in Section 2.2.

Definition 2.1.3. Let $s \in \mathcal{S}$ be a state and $a \in \mathcal{A}$ an action. The *reward space* for this state-action pair, denoted by $\mathcal{R}(s, a)$, is defined as the subset of the real numbers $\mathbb{R}$ representing all possible rewards received after taking action a in state s.

Remark 3. For the sake of simplicity, it is often assumed that the reward does not depend on states or actions, implying that the reward function is constant over the product space $\mathcal{S} \times \mathcal{A}$. Therefore, we denote it simply as $\mathcal{R}$ and refer to it as the *reward space*. This will be the standing assumption in this document. Furthermore, since $\mathcal{R}$ is a subset of $\mathbb{R}$, we will often conflate it with $\mathbb{R}$.

Remark 4. In this document, we use capital letters to designate random variables. For instance, the action taken by the agent at time $t \in \mathbb{N}$ is denoted by A_t. Similarly, for the notions already introduced, we use S_t and R_t to represent the state and the reward, respectively, at time t.

Remark 5. The set of possible states $\mathcal{S}$, actions $\mathcal{A}$ and rewards $\mathcal{R}$ may be continuous. In a rigorous mathematical approach, it would be essential to consider the whole set of crossed cases, based on the topology of these spaces for the theorems and proofs in this document. Typically, the core differences in these cases would involve altering sums to integrals. However, for the sake of simplicity and practicality in this document, we will assume these spaces to be finite. It is important to note that in real-world applications, especially in computer science, it is usually feasible to approximate or discretize $\mathcal{R}$ into a finite set, albeit with potential for approximation errors.

Definition 2.1.4. The *initial state*, denoted by S_0, is defined as the state where the agent starts its interaction with the environment. This initial state is sampled according to a predefined distribution, referred to as the *initial state distribution*, denoted by ρ_0. The initial state distribution is formally defined as:

$$\rho_0 \doteq P(S_0 = \cdot) \tag{2.1}$$

Remark 6. In some cases, the initial state is always the same. For example, in the game of Go, the initial state is consistently an empty board. In these particular scenarios, we denote the constant initial state as s_0. The initial state distribution is thus expressed as:

$$\forall s \in \mathcal{S} \quad \rho_0(s) = \mathbb{1}_{s=s_0} \tag{2.2}$$

Transition function, terminal state, history, episode

Definition 2.1.5. The *transition function*, denoted by p, is defined as the probability function over the states and the rewards given a state and an action. formally, the transition function is defined as:

$$\begin{array}{rccl} p: & \mathcal{R} \times \mathcal{S} \times \mathcal{S} \times \mathcal{A} & \to & [0,1] \\ & (r, s' \mid s, a) & \mapsto & P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a) \end{array} \tag{2.3}$$

Remark 7. At this point, it might be necessary to make p depend on t. However, as we will explore in Definition 2.1.12, the Markovian property implies that the distribution remains constant across t. Therefore, for the sake of simplicity, we will omit the t dependence in our notation.

Definition 2.1.6. A state s is termed *terminal* if and only if:

$$\forall (a, r, s') \in \mathcal{A} \times \mathcal{R} \times \mathcal{S}, \quad p(r, s' \mid s, a) = \mathbb{1}_{s'=s} \cdot \mathbb{1}_{r=0} \tag{2.4}$$

This equation implies that in a terminal state s, regardless of the action taken, the system remains in state s with reward 0. The set of non-terminal states is denoted by $\mathcal{S}^+$, and the set of terminal states is denoted by $\mathcal{S}^-$. Thus, we have $\mathcal{S}^- \cup \mathcal{S}^+ = \mathcal{S}$ and $\mathcal{S}^- \cap \mathcal{S}^+ = \emptyset$.

Remark 8. A terminal state is defined as a state from which the agent cannot depart, and that invariably yields a reward of zero. Upon reaching a terminal state, the agent's interaction process is considered to be concluded.

For the following, it will be useful to have the reward and state distributions, which are obtained simply by marginalizing the joint distribution.

Definition 2.1.7. The *state transition function*, denoted by p, is defined as the probability distribution over states $s' \in \mathcal{S}$ given a current state $s \in \mathcal{S}$ and an action taken $a \in \mathcal{A}$. This function determines the likelihood of transitioning from one state to another state upon the execution of an action. Formally, the state transition function is defined as:

$$\begin{array}{rccl} p: & \mathcal{S} \times \mathcal{S} \times \mathcal{A} & \to & [0,1] \\ & (s' \mid s, a) & \mapsto & P(S_{t+1} = s' \mid S_t = s, A_t = a) \end{array} \tag{2.5}$$

Theorem 2.1.1.

$$\forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \quad p(s' \mid s, a) = \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \tag{2.6}$$

Proof. Immediate consequence of the law of total probability. □

Definition 2.1.8. The *reward distribution*, denoted by p, is defined as the probability distribution over rewards given a state and an action. Formally, the reward distribution is defined as:

$$\begin{array}{rccl} p: & \mathcal{R} \times \mathcal{S} \times \mathcal{A} & \to & [0,1] \\ & (r \mid s, a) & \mapsto & P(R_{t+1} = r \mid S_t = s, A_t = a) \end{array} \tag{2.7}$$

Theorem 2.1.2.

$$\forall (a, r, s') \in \mathcal{A} \times \mathcal{R} \times \mathcal{S} \quad p(r \mid s, a) = \sum_{s' \in \mathcal{S}} p(r, s' \mid s, a) \tag{2.8}$$

Proof. Immediate consequence of the law of total probability. □

At this point, let's introduce a few more elements of vocabulary, in particular the aptly named transitions, histories and episodes.

Definition 2.1.9. An element of $\mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S}$ is called a *transition*.

Definition 2.1.10. An *history*, denoted by h, is defined as a sequence of states, actions, and rewards up to timestep t. Formally, a history is defined as:

$$h = (s_0, a_0, r_1, s_1, \ldots, s_{t-1}, a_{t-1}, r_t, s_t) \tag{2.9}$$

where $s_i \in \mathcal{S}$, $a_i \in \mathcal{A}$, and $r_i \in \mathcal{R}$ for all $i \leqslant t$. The *length of the history* is defined as the integer t.

Remark 9. In a history, states range from s_0 (the initial state) to s_t (the last state, which is not necessarily a terminal state); actions range from a_0 to a_{t-1} (excluding the action taken in state s_t); and rewards range from r_1 to r_t (noting that there is no r_0, as the initial state does not yield a reward).

Remark 10. The set of possible histories is the union, for all $t \in \mathbb{N}$, of histories of length t.

Definition 2.1.11. An *episode* (or *trajectory*), denoted by τ, is defined as a history in which the last state is terminal. Formally, an episode is defined as:

$$\tau \doteq (s_0, a_0, r_1, s_1, \ldots, s_{t-1}, a_{t-1}, r_t, s_t) \quad \text{where } s_t \in \mathcal{S}^- \tag{2.10}$$

Markovian property

At this point, we introduce a central property of the processes known as the Markov property. All of the relationships used in RL are directly derived from the application of this property.

Definition 2.1.12. A stochastic process *verifies the Markov property* (or *is Markovian*) if the conditional probability distribution of future states and rewards depends only on the current state and action, and not on the past history. Formally, for any history $h = (s_0, a_0, r_1, \ldots, a_{t-1}, r_{t-1}, s_t)$, action a, reward r, and state s', the process satisfies:

$$\begin{aligned} &P\left(R_{t+1} = r, S_{t+1} = s' \mid H_t = h, A_t = a\right) \\ &= P\left(R_{t+1} = r, S_{t+1} = s' \mid S_t = s_t, A_t = a\right) \end{aligned}$$

This definition originates from the work of Russian mathematician Andrey Markov.

Remark 11. Intuitively, a process is Markovian when the future is independent of the past, given the present. Everything there is to know is contained in the current state.

Remark 12. In practice, and throughout this document unless otherwise stated, we assume that the process is Markovian. Theoretically, for a non-Markovian process, expanding the state space to include the entire history can transform it into a Markovian process.

Definition 2.1.13. A *Markov Decision Process (MDP)* is the tuple $\langle \mathcal{S}, \mathcal{A}, p \rangle$.

Remark 13. It this document, we assume that the length of the episodes is bounded from above. In other words, the interaction process always ends at some point. It is commonly referred to as a *finite-horizon MDP* or *episodic MDP*.

Definition 2.1.14. The *horizon*, denoted by T, is defined as the step beyond which the process is guaranteed to have reached a terminal state. It is formally defined as:

$$T \doteq \arg\min_{t \in \mathbb{N}} P(S_t \in \mathcal{S}^-) = 1 \tag{2.11}$$

2.1.3 Policy and value functions

In numerous scenarios, the reward resulting from a good action may not be immediate. Consider a chess game: a strategically sound move does not guarantee an immediate win. Consequently, the agent's objective is not to maximize the immediate reward, but the total reward along the way.

Return, policy

Definition 2.1.15. The *return*, denoted by G_t, is the sum of future discounted rewards. Formally, the return is defined as:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{i=0}^{T} \gamma^i R_{t+i+1} \tag{2.12}$$

where $\gamma \in [0, 1]$ is the *discount factor*.

Remark 14. The discount factor is used to make immediate rewards more valuable. When setting $\gamma = 0$, the agent only considers immediate rewards, disregarding future rewards. Conversely, $\gamma = 1$ implies that all future rewards are valued equally, regardless of their temporal distance. However, $\gamma = 1$ is permissible only if the length of the episodes is bounded from above (which is the case in this document, refer to Remark 13). Otherwise, the constraint $0 \leqslant \gamma < 1$ applies. In practice, γ is usually set around 0.99.

It is worth noting that the concept of return can be expressed recursively. To put it simply, what I'll get in the future can be decomposed into what I'll get tomorrow plus what I'll get in tomorrow's future.

Theorem 2.1.3.

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{2.13}$$

Proof.

$$\begin{aligned}
G_t &= \sum_{i=0}^{+\infty} \gamma^i R_{t+i+1} && ((2.12)) \\
&= R_{t+1} + \sum_{i=1}^{+\infty} \gamma^i R_{t+i+1} \\
&= R_{t+1} + \sum_{i=0}^{+\infty} \gamma^{i+1} R_{t+(i+1)+1} \\
&= R_{t+1} + \gamma \sum_{i=0}^{+\infty} \gamma^i R_{(t+1)+i+1} \\
&= R_{t+1} + \gamma G_{t+1} && ((2.13))
\end{aligned}$$

□

None of the concepts introduced so far, however, provide guidance on what actions to take in such a process. The ultimate goal, of course, is to choose the right actions to reap the maximum rewards. To achieve this, one must follow a plan, ideally an effective one. This plan is formally known as a *policy*.

Definition 2.1.16. The *policy*, denoted by π, is defined as a probability distribution over actions given states. It represents the probability that the agent takes a specific action in a given state. Formally, the policy is defined as:

$$\begin{array}{rccl} \pi: & \mathcal{A} \times \mathcal{S} & \to & [0,1] \\ & (a \mid s) & \mapsto & P(A_t = a \mid S_t = s) \end{array} \tag{2.14}$$

Remark 15. Sometimes, policies are deterministic, meaning each state corresponds to a unique action. This is known as a deterministic policy, as opposed to a stochastic policy. In the case of a deterministic policy, it is denoted by $a = \pi(s)$. In this document, policies are assumed to be stochastic by default.

Definition 2.1.17. The *set of all policies*, denoted Π, is defined as the set containing all possible policies. Formally, the set of all policies is defined as:

$$\Pi = \left\{\pi \in [0,1]^{\mathcal{A}\times\mathcal{S}} \mid \forall s \in \mathcal{S}, \pi(\cdot \mid s) \text{ is a probability distribution over } \mathcal{A}\right\} \tag{2.15}$$

State-value function, action-value function

Now that we have introduced the policy, the plan, we need to be able to evaluate the plan, to determine if it is a good plan. To do that, we're going to judge, for each possible state, how much reward we can expect from following it.

Definition 2.1.18. Let π be a policy. The *state-value function*, denoted by V^π, is defined as the expected return starting from state $s \in \mathcal{S}$ and thereafter following policy π. Formally, the state-value function is defined as:

$$\begin{array}{rccl} V^\pi : & \mathcal{S} & \to & \mathbb{R} \\ & s & \mapsto & \mathbb{E}\left[G_t \mid S_t = s, \left(\bigcap_{i=0}^{+\infty} A_{t+i} \sim \pi(\cdot \mid S_{t+i})\right)\right] \end{array} \tag{2.16}$$

Remark 16. To simplify the notation, $\mathbb{E}_\pi$ stands for the expectation under the condition that all actions are sampled according to the policy π (unless otherwise stated):

$$\mathbb{E}_\pi = \mathbb{E}\left[\cdot \mid \bigcap_{i=0}^{\infty} A_{t+i} \sim \pi(\cdot \mid S_{t+i})\right] \tag{2.17}$$

Consequently, we express the state-value function as:

$$V^\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \tag{2.18}$$

What happens if the policy isn't followed just once? This is a question that will be very useful to answer later when we show how to improve a policy. To answer this question, we introduce a new value function.

Definition 2.1.19. The *action-value function*, denoted by Q^π, is defined as the expected cumulative reward after taking an action in a state, and thereafter following the policy π. Formally, the action-value function is defined as:

$$\begin{array}{rccl} Q^\pi : & \mathcal{S}\times\mathcal{A} & \to & \mathbb{R} \\ & (s,a) & \mapsto & \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \end{array} \tag{2.19}$$

Remark 17. We can extend the term *state-value function* (respectively, *action-value function*) to any function mapping from $\mathcal{S}$ (respectively, $\mathcal{S}\times\mathcal{A}$) to $\mathbb{R}$. These functions are thus not necessarily associated with a specific policy. In such cases, we will denote them simply by $\mathbf{V}$ (respectively, $\mathbf{Q}$).

Was not following the policy the right thing to do? To find out, we need to compare what they got with what would have been gotten if the policy had been followed. This relative amount of reward is called the advantage. If it is positive, it was advantageous not to follow the policy.

Definition 2.1.20. The *advantage function*, denoted by A^π, is defined as the expected extra reward received for choosing a particular action in a state, over the expected reward of following the policy π from that state. Formally, the advantage function is defined as:

$$\begin{array}{rccc} A^\pi: & \mathcal{S}\times\mathcal{A} & \to & \mathbb{R} \\ & (s,a) & \mapsto & Q^\pi(s,a) - V^\pi(s) \end{array} \tag{2.20}$$

2.1.4 Bellman equations and optimal policies

Bellman expectation equations

The value functions are interconnected and can be expressed relative to each other. Returning to our previous question: what are the consequences when the agent deviates from the policy just once? This choice not only impacts the immediate reward but also leads the agent to a different next state than it would have reached by following the policy. This relationship is precisely formalized by the following theorem.

Theorem 2.1.4. For all $(s,a)\in\mathcal{S}\times\mathcal{A}$,

$$Q^\pi(s,a) = \sum_{(r,s')\in\mathcal{R}\times\mathcal{S}} p(r,s'\mid s,a)(r+\gamma V^\pi(s')) \tag{2.21}$$

Proof. Let $(s,a)\in\mathcal{S}\times\mathcal{A}$.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\left[G_t \mid S_t=s, A_t=a\right] && ((2.19)) \\ &= \mathbb{E}_\pi\left[R_{t+1}+\gamma G_{t+1} \mid S_t=s, A_t=a\right] && (\text{from } (2.13)) \\ &= \mathbb{E}_\pi\left[R_{t+1} \mid S_t=s, A_t=a\right] + \gamma\mathbb{E}_\pi\left[G_{t+1} \mid S_t=a, A_t=a\right] \end{aligned}$$

Using the Markov property,

$$\begin{aligned} \mathbb{E}_\pi\left[G_{t+1} \mid S_t=a, A_t=a\right] &= \sum_{s'\in\mathcal{S}} p(s'\mid s,a)\mathbb{E}_\pi\left[G_{t+1} \mid S_t=a, A_t=a, S_{t+1}=s'\right] \\ &= \sum_{s'\in\mathcal{S}} p(s'\mid s,a)\mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1}=s'\right] \\ &= \sum_{s'\in\mathcal{S}} p(s'\mid s,a)V^\pi(s') \end{aligned}$$

Therefore,

$$\begin{aligned} Q^\pi(s,a) &= \sum_{r\in\mathcal{R}} p(r\mid s,a)r + \gamma\sum_{s'\in\mathcal{S}} p(s'\mid s,a)V^\pi(s') \\ &= \sum_{r\in\mathcal{R}}\sum_{s'\in\mathcal{S}} p(r,s'\mid s,a)r + \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(r,s'\mid s,a)\gamma V^\pi(s') \\ &= \sum_{(r,s)\in\mathcal{R}\times\mathcal{S}} p(r,s'\mid s,a)(r+\gamma V^\pi(s')) && ((2.21)) \end{aligned}$$

□

Corollary 2.1.4.1. For all $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a \right] \tag{2.22}$$

Proof. Simple rewriting of Equation (2.21). □

Similarly, we can express the value of a state in terms of the action-state function.

Theorem 2.1.5. For all $s \in \mathcal{S}$,

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^\pi(s,a) \tag{2.23}$$

Proof. Let $s \in \mathcal{S}$.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left[G_t \mid S_t = s \right] && ((2.16)) \\ &= \sum_{a \in \mathcal{A}} P(A_t = a \mid S_t = s) \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\ &= \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^\pi(s,a) && ((2.23)) \end{aligned}$$

□

Corollary 2.1.5.1. For all $s \in \mathcal{S}$,

$$V^\pi(s) = \mathbb{E}_\pi \left[Q^\pi(s, A_t) \mid S_t = s \right] \tag{2.24}$$

Proof. Simple rewriting of Equation (2.23). □

We therefore introduce Bellman's expectation equations, which formalize the recursive relation on the two value functions. For each, the idea is to decompose the expectation of the future into the sum of the expectation of the immediate reward and the rewards that will come later.

Theorem 2.1.6 (Bellman expectation equations)**.** For all $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$V^\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s \right] \tag{2.25}$$
$$Q^\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] \tag{2.26}$$

Proof of Bellman expectation equation for V^π. Let $s \in \mathcal{S}$.

$$\begin{aligned} V^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^\pi(s,a) && ((2.23)) \\ &= \sum_{a \in \mathcal{A}} \pi(a \mid s) \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a \right] && \text{(from (2.22))} \\ &= \sum_{a \in \mathcal{A}} P(A_t = a \mid S_t = s) \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a \right] \\ & && \text{(from (2.14))} \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s \right] && ((2.25)) \end{aligned}$$

□

Proof of Bellman expectation equation for Q^π. Let $(s, a) \in \mathcal{S} \times \mathcal{A}$.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = a \right] && ((2.22)) \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma \mathbb{E}_\pi \left[Q^\pi(S_{t+1}, A_{t+1}) \right] \mid S_t = s, A_t = a \right] && \text{(from (2.23))} \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] && ((2.26)) \end{aligned}$$

□

Optimal policies

At this stage, we are able to evaluate the value of a policy. However, the goal is not just any policy; it is the optimal policy—the one that consistently makes the best decisions. In the following sections, we provide a formal definition of this optimality and introduce the properties that will guide us in finding this optimal policy.

Definition 2.1.21. Let π and π' be two policies. π' is said to be *better than* π, denoted

$$\pi \leqslant \pi'$$

if and only if

$$\forall s \in \mathcal{S}, \quad V^\pi(s) \leqslant V^{\pi'}(s) \tag{2.27}$$

Remark 18. Defined in this way, *is better than* constitutes a partial order on Π. It is important to note that this order relation is not necessarily a total order. That is, given any two policies, it is not always the case that one *is better than* the other.

Definition 2.1.22. The *optimal state-value function*, denoted by V^*, is the maximum of state-value functions over all policies. It is formally defined as:

$$\begin{array}{rccc} V^* : & \mathcal{S} & \to & \mathbb{R} \\ & s & \mapsto & \max\limits_{\pi \in \Pi} V^\pi(s) \end{array} \tag{2.28}$$

Remark 19. As a consequence of Remark 13, $(\pi \in \Pi, s \in \mathcal{S}) \mapsto V^\pi(s)$ is indeed bounded from above. Intuitively, this means that since the agent receives a finite reward, a finite number of times, the sum of the rewards is bounded from above. This implies that $\sup_{\pi \in \Pi} V^\pi(s)$ does exist. However, for stochastic policies, it cannot be guaranteed that this supremum is attained within Π. Nevertheless, we will proceed with the assumption that it is reached, though this is not formally justified.

Definition 2.1.23. The *optimal action-value function*, denoted by Q^*, is the maximum of the action-value functions over all policies. Formally, the optimal action-value function is defined as:

$$\begin{array}{rccc} Q^* : & \mathcal{S} \times \mathcal{A} & \to & \mathbb{R} \\ & (s, a) & \mapsto & \max\limits_{\pi \in \Pi} Q^\pi(s, a) \end{array} \tag{2.29}$$

Definition 2.1.24. A policy π is *optimal* if and only if its state-value function is the optimal state-value function. An optimal policy is usually denoted by π^*.

$$\pi \text{ is optimal } \iff V^\pi = V^* \tag{2.30}$$

Theorem 2.1.7.

$$\pi \text{ is optimal } \implies Q^\pi = Q^* \tag{2.31}$$

Proof. Let π be an optimal policy. Then $V^\pi = V^*$ (from Definition 2.1.24). Let s be a state and a an action.

$$\begin{aligned}
Q^\pi(s,a) &= \sum_{(r,s')\in\mathcal{R}\times\mathcal{S}} p(r,s' \mid s,a)(r+\gamma V^\pi(s')) && ((2.21)) \\
&= \sum_{(r,s')\in\mathcal{S}\times\mathcal{S}} p(r,s' \mid s,a)(r+\gamma V^*(s')) \\
&= \sum_{(r,s')\in\mathcal{S}\times\mathcal{S}} p(r,s' \mid s,a)(r+\gamma \max_{\tilde{\pi}\in\Pi} V^{\tilde{\pi}}(s')) && \text{(from (2.28))} \\
&= \max_{\tilde{\pi}\in\Pi} \sum_{(r,s')\in\mathcal{S}\times\mathcal{S}} p(r,s' \mid s,a)(r+\gamma V^{\tilde{\pi}}(s')) \\
&= \max_{\tilde{\pi}\in\Pi} Q^{\tilde{\pi}}(s,a) && \text{(from (2.21))} \\
&= Q^*(s,a) && \text{(from (2.29))}
\end{aligned}$$

□

Remark 20. The converse, however, is not true. This can be demonstrated by considering the nature of the action-value function $Q^\pi(s,a)$. This function essentially answers the question: *What return can I expect if I take action a and* ***then*** *follow policy* π*?*. As an example, consider a one-step MDP (i.e. the episode ends immediately after the first action), the policy followed thereafter becomes irrelevant. Consequently, the action-value function simply represents the expected return of taking action a. In such a scenario, all policies, whether optimal or not, share the same action-value function.

Remark 21. In finite MDP, such a policy does exist. However, the proof of its existence is non-trivial. In this document, we elect to use this result without providing its demonstration.

Greedy policy and policy improvement theorem

An effective strategy for improving a policy involves evaluating, at each step, what the outcome would have been if the recommended action had not been taken. If this evaluation shows that a better alternative action was available, then choosing it instead leads to policy improvement. This approach is known as *greedy*.

Definition 2.1.25. Let $\mathbf{Q}$ be an action-value function. The *greedy policy with respect to* $\mathbf{Q}$, denoted greedy($\mathbf{Q}$), is defined as the deterministic policy that, for each state s, chooses the action that maximizes $\mathbf{Q}(s, \cdot)$. Formally, the greedy policy with respect to $\mathbf{Q}$ is defined as:

$$\text{greedy}(\mathbf{Q}) \doteq s \mapsto \arg\max_{a \in \mathcal{A}} \mathbf{Q}(s, a) \tag{2.32}$$

Furthermore, for any policy $\pi \in \Pi$, the corresponding greedy policy is denoted by π_{greedy}, defined as $\pi_{\text{greedy}} \doteq \text{greedy}(Q^\pi)$.

Remark 22. In this discussion, we simplify our consideration by not focusing on cases where multiple actions yield the same maximum value, which would render the use of arg max ambiguous. To circumvent this issue, we adopt the convention that if multiple actions have the same value, the arg max function selects the action with the smallest index in the action set $\mathcal{A}$. This approach ensures a unique action selection for the greedy policy, even in scenarios where multiple actions have identical value.

Theorem 2.1.8 (Policy improvement theorem)**.** Consider a policy π. The greedy policy with respect to the action-value function Q^π is better than π. Formally, this is expressed as:

$$\forall \pi \in \Pi, \quad \pi \leqslant \pi_{\text{greedy}} \tag{2.33}$$

Lemma 2.1.8.1. Let $\pi \in \Pi$.

$$\forall s \in \mathcal{S} \quad V^\pi(s) \leqslant Q^\pi(s, \pi_{\text{greedy}}(s)) \tag{2.34}$$

Proof of the lemma. Let $\pi \in \Pi$. The greedy policy π_{greedy} is deterministic. Consequently, for any action a and state s, $\pi_{\text{greedy}}(a \mid s)$ is either 0 or 1. In line with the notation introduced in Remark 15, we use $\pi_{\text{greedy}}(s)$ to denote the action consistently chosen by π_{greedy} in state $s \in \mathcal{S}$. Formally:

$$\forall s \in \mathcal{S}, \quad \pi_{\text{greedy}}(s) = \arg\max_{a \in \mathcal{A}} Q^\pi(s, a) \tag{2.35}$$

$$\begin{aligned} V^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^\pi(s, a) && ((2.23)) \\ &\leqslant \underbrace{\sum_{a \in \mathcal{A}} \pi(a \mid s)}_{=1} \max_{\tilde{a} \in \mathcal{A}} Q^\pi(s, \tilde{a}) \\ &= \max_{\tilde{a} \in \mathcal{A}} Q^\pi(s, \tilde{a}) \\ &= Q^\pi(s, \pi_{\text{greedy}}(s)) && ((2.34)) \end{aligned}$$

□

Proof of the theorem. For all $k \in \mathbb{N}$, we define the notation $(A_{t:t+k} = \pi_{\text{greedy}}(S_{t:t+k}))$ to indicate that the current and the next k actions are chosen according to π_{greedy}. Formally, this is expressed as:

$$(A_{t:t+k} = \pi_{\text{greedy}}(S_{t:t+k})) \doteq \bigcap_{i=0}^{k} (A_{t+i} = \pi_{\text{greedy}}(S_{t+i})) \tag{2.36}$$

We apply the following relations recursively until reaching the final state S_T, which, by convention, has a value of 0:

$$V^\pi(s) \leqslant Q^\pi(s, \pi_{\text{greedy}}(s)) \tag{(2.34)}$$

and

$$Q^\pi(s, \pi_{\text{greedy}}(s)) = \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s, A_t = \pi_{\text{greedy}}(S_t) \right] \tag{(2.22) applied to π_{greedy}}$$

For readability, we omit in subsequent expressions that the expectations are conditioned on $S_t = s$.

$$\begin{aligned}
V^\pi(s) &\leqslant Q^\pi(s, \pi_{\text{greedy}}(s)) \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid A_t = \pi_{\text{greedy}}(S_t) \right] \\
&\leqslant \mathbb{E}_\pi \left[R_{t+1} + \gamma Q^\pi(S_{t+1}, \pi_{\text{greedy}}(S_{t+1})) \mid A_t = \pi_{\text{greedy}}(S_t) \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 V^\pi(S_{t+2}) \mid A_{t:t+1} = \pi_{\text{greedy}}(S_{t:t+1}) \right] \\
&\leqslant \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 Q^\pi(S_{t+2}, \pi_{\text{greedy}}(S_{t+2})) \mid A_{t:t+1} = \pi_{\text{greedy}}(S_{t:t+1}) \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V^\pi(S_{t+2}) \mid A_{t:t+2} = \pi_{\text{greedy}}(S_{t:t+2}) \right] \\
&\dots \\
&\leqslant \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} \underbrace{V^\pi(S_T)}_{=0} \mid A_{t:T} = \pi_{\text{greedy}}(S_{t:T}) \right] \\
&= \mathbb{E}_{\pi_{\text{greedy}}} \left[G_t \mid S_t = s \right] \\
&= V^{\pi_{\text{greedy}}}(s)
\end{aligned}$$

□

Corollary 2.1.8.1. Let $\pi \in \Pi$.

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A} \quad Q^\pi(s, a) \leqslant Q^{\pi_{\text{greedy}}}(s, a) \tag{2.37}$$

Proof. Let $\pi \in \Pi$. For all $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned}
Q^\pi(s, a) &= \sum_{(r, s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma V^\pi(s')) && \text{((2.21))} \\
&= \sum_{r \in \mathcal{R}} p(r \mid s, a) r + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^\pi(s') \\
&\leqslant \sum_{r \in \mathcal{R}} p(r \mid s, a) r + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^{\pi'}(s') && \text{((2.33))} \\
&= \sum_{(r, s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma V^{\pi_{\text{greedy}}}(s')) \\
&= Q^{\pi_{\text{greedy}}}(s, a) && (2.38)
\end{aligned}$$

□

Now, let's suppose that we have access to the optimal value function (this does not imply that we have access to an optimal policy). Then, in this case, an

immediate way of finding an optimal policy is to systematically take the action which, according to this value function, leads to the greatest return. In other words, take the greedy policy with respect to this optimal value function.

Theorem 2.1.9.

$$\text{greedy}(Q^*) \text{ is optimal.} \tag{2.39}$$

Proof. Let π^* be an optimal policy. Then, by the definition of an optimal policy and the optimal action-value function, we have:

$$V^{\pi^*} = V^* \qquad ((2.31))$$

$$Q^{\pi^*} = Q^* \qquad ((2.30))$$

Consequently,

$$\text{greedy}(Q^*) = \text{greedy}(Q^{\pi^*}) = \pi_{\text{greedy}} \tag{2.40}$$

From Theorem 2.1.8, it follows that π_{greedy} is better than π^*:

$$\pi^* \leqslant \pi_{\text{greedy}} \qquad ((2.33))$$

Then, we deduce that:

$$\begin{aligned} \forall s \in \mathcal{S}, \quad V^{\pi^*}(s) &\leqslant V^{\pi_{\text{greedy}}}(s) && \text{(from (2.1.21))} \\ V^{\pi^*}(s) &\leqslant V^{\pi_{\text{greedy}}}(s) \leqslant V^*(s) && \text{(from Definition (2.1.22))} \\ V^*(s) &\leqslant V^{\pi_{\text{greedy}}}(s) \leqslant V^*(s) && \text{(since } \pi^* \text{ is optimal)} \end{aligned}$$

Therefore, $V^* = V^{\pi_{\text{greedy}}}$, and hence $\pi_{\text{greedy}} = \text{greedy}(Q^*)$ is also optimal. □

Definition 2.1.26. Let $\mathbf{Q}$ be a action-value function. The *ε-greedy policy with respect to* $\mathbf{Q}$, denoted $\varepsilon-\text{greedy}(\mathbf{Q})$, is defined as the policy that chooses a random action (from uniform distribution on $\mathcal{A}$) with a probability ε, and the best action according to $\mathbf{Q}$ with a probability $1-\varepsilon$.

$$A_t \sim \varepsilon\text{-greedy}(\mathbf{Q})(\cdot \mid s) \iff \begin{cases} A_t = \text{greedy}(\mathbf{Q})(s) & \text{with a proba. } 1-\varepsilon \\ A_t \sim \mathcal{U}(\mathcal{A}) & \text{with a proba. } \varepsilon \end{cases} \tag{2.41}$$

Remark 23. The definition of the ε-greedy policy, while not highly formal, is advantageous for straightforward algorithmic implementation.

Bellman optimality equations

We continue to focus on optimal policies. Like all policies, they satisfy the Bellman equations. However, they verify additional relations that are useful for approximating them. These relations are called Bellman optimality equations.

Theorem 2.1.10 (Bellman optimality equations)**.** For all $(s,a) \in \mathcal{S} \times \mathcal{A}$

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{(r,s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma V^*(s')) \tag{2.42}$$

$$Q^*(s,a) = \sum_{(r,s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a')) \tag{2.43}$$

Proof. Let $\pi = \text{greedy}(Q^*)$. Then, for all $s \in \mathcal{S}$, we have:

$$\begin{aligned} V^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^\pi(s,a) && ((2.23)) \\ &= Q^\pi \left(s, \arg\max_{a \in \mathcal{A}} Q^*(s,a) \right) \\ V^*(s) &= Q^* \left(s, \arg\max_{a \in \mathcal{A}} Q^*(s,a) \right) && \text{(since } \pi \text{ is optimal)} \\ &= \max_{a \in \mathcal{A}} Q^*(s,a) && (2.44) \end{aligned}$$

On the other hand,

$$Q^\pi(s,a) = \sum_{(r,s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma V^\pi(s')) \qquad ((2.21))$$

Since π is optimal,

$$Q^*(s,a) = \sum_{(r,s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma V^*(s')) \qquad (2.45)$$

By applying Equation (2.44) to Equation (2.45), we deduce the Bellman optimality equation for Q^*.

$$Q^*(s,a) = \sum_{(r,s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma \max_{a' \in \mathcal{A}} Q^*(s', a')) \qquad ((2.43))$$

Similarly, by applying Equation (2.45) to Equation (2.44), we deduce the Bellman optimality equation for V^*.

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{(r,s') \in \mathcal{R} \times \mathcal{S}} p(r, s' \mid s, a)(r + \gamma V^*(s')) \qquad ((2.42))$$

□

At this stage, we have established the basic concepts and are ready to introduce algorithms that aim to approximate an optimal policy.

2.1.5 Algorithm types

A variety of strategies have been developed to find an optimal policy. The subsequent section explores the most prevalent methods documented in the literature. However, before delving into these approaches, this section introduce a categorization framework to organize these methods.

Model-based, value-based, policy-based

Various methods exist for identifying an optimal policy in RL, and these can be broadly categorized, though it should be noted that these categories are not mutually exclusive. The first category, *model-based methods*, involves approximating a *model of the environment*. This model represents the agent's understanding of the environment's dynamics in response to its actions. Specifically, it aims

Table 2.1: RL algorithm classes.

	Model	Value function	Policy
Model-free	✗	-	-
Model-based	✓	-	-
Value-based	-	✓	✗
Policy-based	-	✗	✓
Actor-critic	-	✓	✓

to estimate the probability of transitioning from one state to another, given an action, along with the expected reward, denoted as $p(\cdot, \cdot \mid \cdot, \cdot)$, (see Definition 2.1.5). When a learning method utilizes this model for decision-making, as opposed to interacting directly with the environment, it is engaging in *planning*. The second category, *value-based methods*, focuses on constructing a value function. This function evaluates the worth of states and actions, and the policy for interacting with the environment is derived from it. An example is the use of an ε-greedy policy based on this value function. The third category, *policy-based methods*, directly establishes a relationship between states and actions. Unlike the previous categories, these methods do not attempt to estimate the value of states and actions but rather develop the policy itself. Finally, the fourth category discussed in this document is *actor-critic methods*. These methods employ a dual approach, utilizing both a policy (the actor) and a value function evaluating the policy (the critic), hence the nomenclature.

Off-policy and on-policy learning

The policy that an agent uses to select actions is referred to as the *behavioral policy*, denoted by β. This policy is distinct from the target policy, denoted by π, which the agent aims to optimize or evaluate. When β is equal to π, the agent's learning process is concurrent with its interaction with the environment. This scenario is termed *on-policy learning*, signifying that the agent is *learning on the job*.

Conversely, if β differs from π, the agent is engaged in *off-policy learning*. In this approach, the agent observes and learns from the behavior dictated by a different policy. This method can be likened to *looking over someone else's shoulder*, as the agent utilizes experiences generated from a policy other than the one it seeks to evaluate or optimize. A significant benefit of off-policy learning is its ability to incorporate a wide range of experiences, including past experiences and those from other agents, into the learning process.

2.1.6 Partially observable MDPs and interpreter

RL problems are fundamentally grounded in the framework of MDPs. However, Some RL problems often diverge from standard MDPs in two key aspects.

Partially Observable MDP

The first divergence is related to the assumption of complete state observability in classical MDPs. In many real-world situations, an agent may not have full access to the environment's state. Instead, the agent perceives the state through an observation function. A Partially Observable Markov Decision Process (POMDP) is a generalization of a MDP in which the agent does not have complete and accurate information about the current state of the environment.

Definition 2.1.27. Formally, a POMDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, p, \Omega, O \rangle$ where:

- Ω is a finite set of observations.
- O is the observation function: $O : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$, which defines the probability of observing a particular observation given a state and an action.

In a POMDP, the agent receives observations that depend on the underlying state of the environment and the actions taken but may not uniquely identify the state. As a result, the agent must make decisions under uncertainty about the actual state.

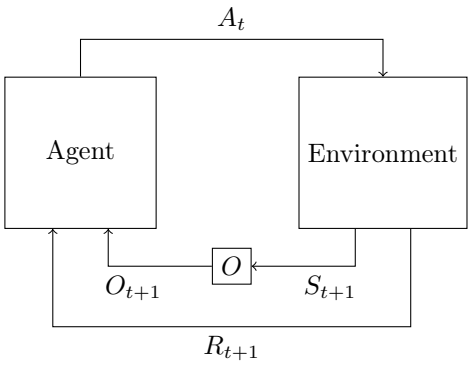


Figure 2.2: Schematic illustration of POMDP.

Throughout this document, the assumption is that the MDP is fully observable, except where specified otherwise. Our focus will be on algorithms tailored to Markovian processes. Adapting these algorithms for use in POMDPs presents significant challenges. In the context of a POMDP, the agent effectively deals with a process that is not strictly Markovian (from its perspective). To mitigate this and approximate Markovian characteristics, one approach involves utilizing the history of observations. However, this method substantially increases the size of the observation space, adding to the complexity.

Interpreter

The second notable difference in RL involves the origination of rewards. Contrary to common assumptions, the *environment* in an RL scenario often does not directly generate rewards. Rather, the determination of rewards is typically

handled by a separate component known as the *interpreter*, which computes the reward based on the current state. This relationship can be formally expressed as:

$$R_t \doteq \mathcal{I}(S_t) \tag{2.46}$$

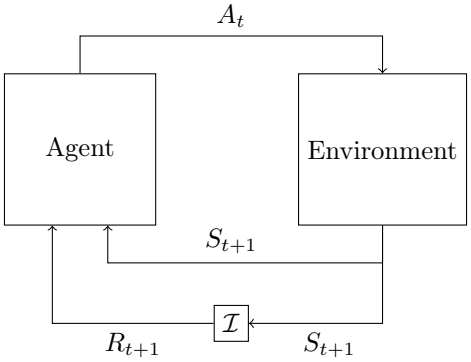


Figure 2.3: Schematic illustration of the interpreter.

Determining the appropriate reward structure to guide an agent towards desired behaviors can be a complex task. There exists a specialized literature focused on this topic, known as reward shaping. This field studies methods and strategies for effectively designing reward functions that facilitate learning and goal achievement [100].

For the sake of simplicity in our discussions, we will proceed under the assumption that the environment directly provides the reward, thus omitting the detailed role of the interpreter. This assumption effectively integrates the interpreter's function into the environment's mechanism.

2.2 Tabular Case: Value-Based Approaches

2.2.1 Dynamic programming

In the first section, we introduced all the concepts necessary to understand a RL problem. We also introduced the term *planning* (see Section 2.1.5), which consists in solving a RL problem without interacting with the environment, using only a model of it. Dynamic Programming (DP) describes the set of algorithms to compute the optimal policy based on the knowledge of a perfect model of the environment. In the following, we assume that $p(\cdot, \cdot \mid \cdot, \cdot)$ perfectly reflects the dynamics of the world, and we'll present various methods for solving an RL problem based on this model.

Policy evaluation

Consider an arbitrary π policy. The goal here is to approximate its value function Q^π. To do this, we use the *iterative policy evaluation* algorithm, which consists of iteratively computing a sequence (Q_k) that tends toward an increasingly

accurate estimate of Q^π. To do this, let's first note that we can write Q^π as follows.

Theorem 2.2.1. For all $(s,a) \in \mathcal{S} \times \mathcal{A}$

$$Q^\pi(s,a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r + \gamma \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q_\pi(s', a') \right) \tag{2.47}$$

Proof.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma Q^\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] && ((2.26)) \\ &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r' + \gamma Q^\pi(s', A_{t+1}) \right) \\ &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r' + \gamma \sum_{a' \in \mathcal{A}} \pi(a \mid s) Q^\pi(s', a') \right) && ((2.47)) \end{aligned}$$

□

The value iteration algorithm initiates with an arbitrary function Q_0 and uses the following update rule for any $k \in \mathbb{N}$:

$$Q_{k+1}(s,a) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r + \gamma \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q_k(s', a') \right) \tag{2.48}$$

Over iterations, the sequence (Q_k) converges towards its fixed point Q^π.

It is important to note that Equation (2.48) relies on Q_k for its updates, which is an approximation of Q^π. This method of refining an estimate based on a previous estimate, known as *bootstrapping*, introduces an inherent bias as it effectively *learns a guess from a guess*.

Algorithm 1: Iterative policy evaluation.

Input: the policy π to be evaluated and a model p of the environment
initialize $Q_0(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$; $k = 0$
repeat forever
  for $s \in \mathcal{S}$ *and* $a \in \mathcal{A}$ **do**

$$Q_{k+1}(s,a) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r + \gamma \sum_{a' \in \mathcal{A}} \pi(a' \mid s') Q_k(s', a') \right)$$

  $k \leftarrow k + 1$

Policy iteration

Policy iteration operates on the principle of successively estimating and enhancing a policy π to obtain an improved policy π'. Given the action-value function of a policy, the most straightforward approach to derive an improved policy is to adopt a greedy strategy with respect to this action-value function (see Theorem 2.1.8). Iteratively applying this process a sufficient number of times leads to the optimal policy π_* (although we do not provide a formal demonstration here).

Algorithm 2: Q-policy iteration (or policy iteration).

Input: a model of the environment p
arbitrarily initialize $\pi(s)$ for all $s \in \mathcal{S}$
while *policy is improving* **do**
  estimate state-value function $Q \approx Q^\pi$ with iterative policy evaluation
  $\pi \leftarrow \text{greedy}(Q)$

Value iteration

A more straightforward method involves directly using Bellman's optimality equation (see Equation (2.43)) to estimate the optimal value function. The resulting algorithm is named *value iteration* the update rule used in this algorithm is as follows:

$$Q_{k+1}(s,a) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r + \gamma \max_{a' \in \mathcal{A}} Q_k(s', a') \right) \tag{2.49}$$

Similarly, iteratively applying this process a sufficient number of times leads to the optimal value function Q^*.

Algorithm 3: Q-value iteration (or value iteration).

Input: a model of the environment p
arbitrarily initialize $Q_0(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$
repeat forever
  for $s \in \mathcal{S}$ *and* $a \in \mathcal{A}$ **do**
    $$Q_{k+1}(s,a) \leftarrow \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \left(r + \gamma \max_{a' \in \mathcal{A}} Q_k(s', a') \right)$$
  $Q_k \leftarrow Q_{k+1}$

2.2.2 Monte Carlo methods

Dynamic programming is based on the strong assumption that a reliable model of the environment is available. However, in practical scenarios, such a comprehensive model is often unavailable or merely approximate. In contrast, Monte Carlo (MC) methods present a model-free approach, bypassing the need for an exact model of the environment. These methods rely solely on experiential data. This section is dedicated to exploring MC methods as an initial approach to solving RL problems.

The core idea behind MC methods is to use the return G_t, as an estimator for the state-action value function Q^π (refer to Definition 2.1.18). Consider a scenario where the environment has been sampled N times, resulting in a collection of state, action, and corresponding return. This data is stored in a replay buffer, structured as a series of tuples:

$$\mathcal{D} = \{\langle S_i, A_i, G_i \rangle \quad \text{for } i = 1, 2, \ldots, N\} \tag{2.50}$$

We can then approximate the value function of the state action as follows:

$$Q^\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \approx \frac{\sum_{i=1}^N \mathbb{1}_{S_i=s}\mathbb{1}_{A_i=a} G_i}{\sum_{i=1}^N \mathbb{1}_{S_i=s}\mathbb{1}_{A_i=a}} \tag{2.51}$$

Given an approximate value of the state-action function, we can construct a better policy by taking the greedy policy with respect to Q. This new policy is demonstrably better, as demonstrated in Theorem 2.1.8. However, an inherent limitation arises from the initial value of Q: certain states may remain unexplored due to their underestimated value. To overcome this, an ε-greedy policy (defined in 2.1.26) is used. This approach balances exploration and exploitation by gradually reducing the value of epsilon toward zero during the learning process. The resulting algorithm is called Greedy in the Limit with Infinite Exploration (GLIE) [253].

Algorithm 4: Greedy in the Limit with Infinite Exploration (GLIE).

arbitrarily initialize $Q(s,a)$ for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}(s)$
$\varepsilon \leftarrow 1$
$\pi \leftarrow \varepsilon$-greedy$(Q)$
repeat forever
  collect N interactions with π and store them in $\mathcal{D}$
  $Q \leftarrow$ approximation of Q^π using Eq. (2.51)
  $\pi \leftarrow \varepsilon$-greedy$(Q)$
  decrease ε

Intuitively, the learning process reveals that some states are *understood* more rapidly than others, for example due to the frequency at which they are encountered. The ε-greedy policy, however, does not differentiate between states in terms of exploration needs. A efficient alternative involves giving a bonus when the policy explores state it does not know much. This bonus is the *upper confidence* $U_t(s,a)$ of a state-action pair. This algorithm known as the *Upper Confidence Bound* (UCB) has been originally proposed by Auer et al. [15]. UCB uses the following confidence score:

$$\begin{array}{rccc} U_t: & \mathcal{S}\times\mathcal{A} & \to & \mathbb{R} \\ & (s,a) & \mapsto & \sqrt{\frac{-\log p}{2N_t(s,a)}} \end{array} \tag{2.52}$$

where $p \in (0,1)$ is reduced over time to ensure that the continues to be exploratory, and $N_t(s,a)$ is the number of times the state-action pair (s,a) has been encountered.

2.2.3 Temporal-Difference

After exploring MC methods, which rely on complete episodes to estimate value functions, we now turn our attention to Temporal Difference (TD) learning. TD methods represent a different approach, blending some aspects of MC methods with ideas from DP.

Unlike MC methods, which require the final outcome of an episode for updates, TD learning can update value estimates based on incomplete episodes, using

Algorithm 5: Upper Confidence Bounds (UCB).

initialize the counter $N(s,a) = 0$ for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$
arbitrarily initialize $Q(s,a)$ for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$
initialize $p \in (0,1)$
initialize $U(s,a) = +\infty$ for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$
repeat forever
  sample episode $S_0, A_0, R_1, \ldots, A_{T-1}, R_T, S_T$ by acting greedily w.r.t $Q + U$
  for *each timestep t of the episode* **do**
    $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
    $G_t \leftarrow R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$
    $U(S_t, A_t) \leftarrow \sqrt{\frac{-\log p}{2N_t(S_t, A_t)}}$
    decrease p

current estimates to predict future outcomes. This allows TD methods to learn from experience at each step of the episode, rather than waiting for the episode's conclusion. As such, TD learning is particularly advantageous in continuous or very long tasks where episode ends are infrequent.

TD learning is grounded in the principle of bootstrapping, where current estimates are updated based on new experiences and existing estimates (Equation (2.25)). A key aspect of TD methods is their ability to update policies after each timestep, using the difference (or *temporal difference*) between successive predictions to guide learning. This incremental update process often leads to faster convergence compared to MC methods.

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) \mid S_t = s] \tag{(2.25)}$$

The objective is to find a V that minimizes the *TD error*, δ_t, which is defined as follows:

$$\delta_t \doteq \underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{TD target}} - V(S_t) \tag{2.53}$$

As previously discussed, to compute the TD error, only the immediate reward R_{t+1} and the subsequent state S_{t+1} are required. This facilitates the approximation of $V^\pi(S_t)$. Consequently, the agent can learn *on the fly*, without the necessity of awaiting the conclusion of the episode.

Remark 24. For clarity and simplicity, the algorithms described herein do not account for scenarios where S_{t+1} is a terminal state. In practical applications, bootstrapping is not feasible when S_{t+1} is terminal, necessitating the omission of the term $\gamma V(S_{t+1})$ in such cases. The appropriate formulation, considering terminal states, is given by:

$$\delta_t \doteq R_{t+1} + \gamma(V(S_{t+1}) - V(S_t))\mathbb{1}_{\mathcal{S}^-}(S_{t+1}), \tag{2.54}$$

where $\mathbb{1}_{\mathcal{S}^-}$ denotes the termination indicator.

Approximate the value function using TD

TD(0) is an algorithm that focus on iteratively approximating the value function V^π for a given policy π. This approximation is grounded in the iterative update of the value function based on the TD error computed at each step. In each iteration, the value of the current state $V(S_t)$ is adjusted in the direction that reduces the TD error. This process, referred to as TD update, is governed by the learning rate $\alpha \in \mathbb{R}_+$, which determines the extent to which new information overrides old information. The algorithmic representation of TD(0) is presented in Algorithm 6.

Algorithm 6: Temporal-Difference (TD(0)).

Input: a policy π to be evaluated.
arbitrarily initialize $V(s)$ for each state $s \in \mathcal{S}$
repeat forever
  initialize the environment and get the initial state S_0
  repeat
    choose $A_t \sim \pi(\cdot \mid S_t)$
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    $\delta_t \leftarrow R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
    $V(S_t) \leftarrow V(S_t) + \alpha\delta_t$
  until *the end of the episode*;

In summary, TD learning uses a more stable estimator that facilitates learning on the fly. However, it is important to acknowledge that this estimator is inherently biased due to the use of the approximated value function. Conversely, MC methods uses an unbiased estimator, which, while possessing high variance, requires a full episode for its computation. These differences are summarized in Table 2.2.

Table 2.2: Comparison of MC and TD methods.

	LEARN FROM INCOMPLETE EPISODE	LOW VARIANCE	UNBIASED
MC	no	no	yes
TD	yes	yes	no

To balance between the two extremes, an intermediate method is to use a combination of these two estimators.

Definition 2.2.1. The n-steps return, denoted $G_{t:t+n}$ is defined as

$$G_{t:t+n} \doteq \sum_{i=0}^{n-1} \gamma^i R_{t+i+1} + \gamma^n V(S_{t+n}) \tag{2.55}$$

Examples:

$$
\begin{aligned}
G_{t:t+1} &= R_{t+1} + \gamma V(S_{t+1}) && \text{1-step return} \\
G_{t:t+2} &= R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) && \text{2-steps return} \\
G_{t:t+3} &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3}) && \text{3-steps return} \\
&\dots
\end{aligned}
$$

As the value of n increases, the bias decreases while the variance increases. To strike a balance between these two factors, we introduce the the λ return.

Definition 2.2.2. Let $\lambda \in \mathbb{R}$. The λ return, denoted G_t^λ is defined as

$$G_t^\lambda \doteq (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t \tag{2.56}$$

Remark 25. $G_t^0 = R_{t+1} + \gamma V(S_{t+1})$ is the TD(0) estimator and $G_t^1 = G_t$ is the MC estimator.

In practice, to continue to allow an update at each interaction, we use an eligibility trace E to spread the reward over the visited states. It is equivalent to use λ-return.

Algorithm 7: Temporal-Difference with λ-return (TD(λ)).

Input: a policy π to be evaluated.
arbitrarily initialize $V(s)$ for each $s \in \mathcal{S}$
repeat forever
  initialize eligibility $E(s) = 0$ for each $s \in \mathcal{S}$
  initialize the environment and get the initial state S_0
  repeat
    choose $A_t \sim \pi(\cdot \mid S_t)$
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
    for *each state* $s \in \mathcal{S}$ **do**
      $E(s) \leftarrow \lambda\gamma E(s) + \mathbb{1}_{S_t = s}$
      $V(s) \leftarrow V(s) + \alpha \delta_t E(s)$
  until *the end of the episode*;

Remark 26. For $\lambda = 0$, the Algorithm 7 is actually equivalent to the TD(0) introduced previously. Hence the name.

SARSA

As in previous discussions, the Bellman optimality equation is employed to update the approximate optimal action-value function Q. The resulting algorithm is termed SARSA, an acronym derived from the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ used in its process.

Similarly, eligibility traces can be integrated to use the λ-return. This resulting algorithm SARSA(λ).

Algorithm 8: State-Action-Reward-State-Action (SARSA(0)).

arbitrarily initialize $Q(s, a)$ for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}(s)$
$\varepsilon \leftarrow 1$
repeat forever
  initialize the environment and get the initial state S_0
  choose the first action A_0 by acting ε-greedily w.r.t. Q
  repeat
    choose the next action A_{t+1} by acting ε-greedily w.r.t. Q
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    $\delta_t \leftarrow R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta_t$
    decrease ε
  until *the end of the episode*;

Q-learning and double Q-learning

Instead of iteratively evaluating, improving, etc., Watkins [279] suggested using the Bellman optimality equation (2.43) to approximate Q^*. The new update rule is

$$Q_{k+1}(S_t, A_t) \leftarrow Q_k(S_t, A_t) + \alpha(R_{t+1}) + \gamma \max_{a \in \mathcal{A}} Q_k(S_{t+1}, a) - Q_k(S_t, A_t)) \quad (2.57)$$

This method is referred to as SARSA-max, or Q-learning.

Algorithm 9: Q-learning.

arbitrarily initialize Q
$\varepsilon \leftarrow 1$
repeat forever
  initialize the environment and observe the initial state S_0
  repeat
    choose the action $A_t \sim \varepsilon$-greedy(Q)
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    $\delta_t \leftarrow R_{t+1} + \gamma \max_{a \in \mathcal{A}}(Q(S_{t+1}, a)) - Q(S_t, A_t)$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta_t$
    decrease ε
  until *the end of the episode*;

While Q-learning represents a significant advancement in RL, it is not without its shortcomings, notably in the tendency to overestimate value estimates. This overestimation issue primarily arises from the policy improvement step based on the same samples used for policy evaluation. A notable solution to this problem is the *double Q-learning*, as introduced by van Hasselt et al. [273]. Double Q-learning employs two distinct value functions, Q^A and Q^B. In this approach, one value function is randomly selected for sampling actions, while the other is used for updating value estimates, thereby mitigating the risk of overestimation inherent in standard Q-learning.

Algorithm 10: Double Q-learning.

arbitrarily initialize Q^A and Q^B
$\varepsilon \leftarrow 1$
repeat forever
  initialize the environment and observe the initial state S_0
  repeat
    choose the action $A_t \sim \varepsilon$-greedy $\big(\text{average}(Q^A, Q^B)\big)$
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    randomly between A and B
    if A **then**
      $a^* \leftarrow \arg\max_{a \in \mathcal{A}}(Q^A(S_t, a))$
      $\delta_t \leftarrow R_{t+1} + \gamma Q^B(S_{t+1}, a^*) - Q^A(S_t, A_t)$
      $Q^A(S_t, A_t) \leftarrow Q^A(S_t, A_t) + \alpha\delta_t$
    if B **then**
      $a^* \leftarrow \arg\max_{a \in \mathcal{A}}(Q^B(S_t, a))$
      $\delta_t \leftarrow R_{t+1} + \gamma Q^A(S_{t+1}, a^*) - Q^B(S_t, A_t)$
      $Q^B(S_t, A_t) \leftarrow Q^B(S_t, A_t) + \alpha\delta_t$
    decrease ε
  until *the end of the episode*;

2.2.4 Importance sampling

The algorithms discussed so far are on-policy, which implies that their learning is limited by their reliance on the current policy. This reliance can lead to slower convergence and less effective exploration of the state-action space. In contrast, off-policy learning is advantageous because it facilitates evaluation and improvement of a target policy while operating under a different, possibly exploratory or sub-optimal, behavioral policy (denoted β). This approach allows more efficient learning from a wider range of experience. To convert an on-policy algorithm into an off-policy algorithm, *importance sampling* (IS) can be used. Importance sampling adjusts the value function estimate by multiplying it by the ratio of the probabilities of the actions under the target and behavioral policies, respectively.

Table 2.3: Approximator of $V^\pi(S_t)$ according to the method and policy used for sampling.

	Sampling policy	
	π	β
MC	G_t	$\left(\prod_{\tau=t}^{T} \frac{\pi(A_\tau \mid S_\tau)}{\beta(A_\tau \mid S_\tau)}\right) G_t$
TD	$R_{t+1} + \gamma V(S_{t+1})$	$\frac{\pi(A_t \mid S_t)}{\beta(A_t \mid S_t)} (R_{t+1} + \gamma V(S_{t+1}))$

In practice, IS is not used with MC estimation because it tends to increase the already high variance associated with MC estimates. However, its use

in TD learning is more feasible and common. The Algorithm 11 provides an illustrative example of how importance sampling can be integrated into a TD-based algorithm.

Algorithm 11: Importance Sampling for off-policy TD(0).

Input: a policy π to be evaluated.
Input: a policy β to be followed.
arbitrarily initialize $V(s)$ for each state $s \in \mathcal{S}$
repeat forever
  initialize the environment and get the initial state S_0
  repeat
    choose $A_t \sim \beta(\cdot \mid S_t)$
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    $\delta_t \leftarrow \frac{\pi(A_t|S_t)}{\beta(A_t|S_t)} (R_{t+1} + \gamma V(S_{t+1})) - V(S_t)$
    $V(S_t) \leftarrow V(S_t) + \alpha\delta_t$
  until *the end of the episode*;

2.3 Value-Based Deep RL

The algorithms discussed thus far are efficient in environments with a relatively small number of states. However, as the number of states increases, they encounter what Richard Bellman has termed *the curse of dimensionality*, where the computational demands grow exponentially with the number of states. This challenge becomes particularly pertinent in many RL applications where the state space is vast, such as scenarios involving data from cameras or numerous sensors. In these cases, it becomes intractable for an agent to explore every possible combination of inputs, like pixels in images. In this section, we'll study how to overcome the curse of dimensionality using deep neural networks. Deep neural networks are a type of function approximator that exhibit excellent capabilities, particularly for RL problems. RL involving deep neural networks is commonly referred to as Deep RL.

2.3.1 Some additional definitions and tools

Value functions

Definition 2.3.1. A *parametrized state-value* (resp. *action-value*) *estimation* is a state-value (resp. action-value) function whose values depend on a parameter vector in $\mathbb{R}^d$, where d is the number of parameters. This parameter vector is typically denoted by ϕ, and the corresponding parametrized state-value (resp. action-value) function is represented as V_ϕ (resp. Q_ϕ). Alternative notations used are $V(\cdot, \phi)$ (resp. $Q(\cdot, \cdot, \phi)$).

Definition 2.3.2. A *differentiable state-value* (resp. *action-value*) *estimation* is a parametrized state-value (resp. action-value) estimation that can be differentiated with respect to its parameter vector. When this condition is met, the expressions $\nabla_\phi V_\phi$ (resp. $\nabla_\phi Q_\phi$) is well-defined.

Visitation distributions

Definition 2.3.3. Let $k \in \mathbb{N}$. The *state visitation distribution after k timesteps starting from s under policy* π, denoted by $\rho_k^\pi(\cdot \to \cdot)$, is defined as the function that assigns to each pair of states the probability of transitioning from one to the other after k timesteps while following policy π. It satisfies:

$$\forall (s, s') \in \mathcal{S}, \quad \rho_k^\pi(s \to s') \doteq P\left(S_{t+k} = s' \mid S_t = s, \bigcap_{i=0}^{k-1} A_{t+i} \sim \pi(\cdot \mid S_{t+i})\right) \tag{2.58}$$

Remark 27. Although t is used in this definition, its actual value does not influence the distribution, a consequence of the Markov property 2.1.12. This notation is retained for clarity, with the hope that it does not lead to confusion.

Theorem 2.3.1. The state visitation distribution after k timesteps starting from s satisfies the following recurrence formula.

$$\forall (s, s'') \in \mathcal{S}^2 \quad \rho_{k+1}^\pi(s \to s'') = \sum_{s' \in \mathcal{S}} \rho_k^\pi(s \to s') \rho_1^\pi(s' \to s'') \tag{2.59}$$

Proof. Let $(s, s'') \in \mathcal{S}^2$. For simplicity, we omit the explicit mention that all actions are sampled according to the policy π. Consequently, the probability expression $P(S_{t+k} = s'' \mid S_t = s\ , A_{t:t+k-1} \sim \pi)$ is simplified to $P(S_{t+k} = s'' \mid S_t = s)$.

$$\begin{aligned}
\rho_{k+1}^\pi(s \to s'') &= P(S_{t+k+1} = s'' \mid S_t = s) \\
&= \sum_{s' \in \mathcal{S}} P(S_{t+k} = s', S_{t+k+1} = s'' \mid S_t = s) \\
&= \sum_{s' \in \mathcal{S}} P(S_{t+k} = s' \mid S_t = s) P(S_{t+k+1} = s'' \mid S_t = s, S_{t+k} = s') \\
&\overset{(i)}{=} \sum_{s' \in \mathcal{S}} P(S_{t+k} = s' \mid S_t = s) P(S_{t+i+1} = s'' \mid S_{t+k} = s') \\
&= \sum_{s' \in \mathcal{S}} P(S_{t+k} = s' \mid S_t = s) P(S_{t+1} = s'' \mid S_t = s') \\
&= \sum_{s' \in \mathcal{S}} \rho_k^\pi(s \to s') \rho_1^\pi(s' \to s'')
\end{aligned}$$

(i) is the consequence of the Markov assumption: the future is independent of the past given the present. □

Definition 2.3.4. The *state visitation distribution after k timesteps under policy* π, denoted by ρ_k^π, is defined as the function that assigns to each state the probability of visitation after k timesteps, starting from an initial state sampled by the environment. It satisfies:

$$\forall s' \in \mathcal{S} \quad \rho_k^\pi(s') \doteq \sum_{s \in \mathcal{S}} \rho_0(s) \rho_k^\pi(s \to s') \tag{2.60}$$

Remark 28. ρ_k^π is indeed a distribution over states:

$$\forall k \in [0, T-1] \quad \sum_{s \in \mathcal{S}} \rho_k^\pi(s) = 1 \tag{2.61}$$

Notably, ρ_0^π is the initial state distribution:

$$\rho_0^\pi = \rho_0 \tag{2.62}$$

Theorem 2.3.2. The state visitation distribution after k timesteps under π satisfies the following recurrence formula.

$$\forall s' \in \mathcal{S} \quad \rho_{k+1}^\pi(s') = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \rho_k^\pi(s) \pi(a \mid s) p(s' \mid s, a) \tag{2.63}$$

Proof. For consistency in notation, we assume that the states are visited sequentially in the order $s \to s' \to s''$.

$$\begin{aligned}
\rho_{k+1}^\pi(s'') &= \sum_{s \in \mathcal{S}} P(S_0 = s) \rho_{k+1}^\pi(s \to s'') && \text{(from (2.60))} \\
&= \sum_{s \in \mathcal{S}} P(S_0 = s) \sum_{s' \in \mathcal{S}} \rho_k^\pi(s \to s') \rho_1^\pi(s' \to s'') && \text{(from (2.59))} \\
&= \sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} P(S_0 = s) \rho_k^\pi(s \to s') \rho_1^\pi(s' \to s'') \\
&= \sum_{s' \in \mathcal{S}} \rho_k^\pi(s') \rho_1^\pi(s' \to s'') && \text{(from (2.60))} \\
&= \sum_{s' \in \mathcal{S}} \rho_k^\pi(s') P(S_{t+1} = s'' \mid S_t = s', A_t \sim \pi) && \text{(from (2.58))} \\
&= \sum_{s' \in \mathcal{S}} \rho_k^\pi(s') \sum_{a \in \mathcal{A}} \pi(a \mid s') p(s'' \mid s', a) \\
&= \sum_{(s',a) \in \mathcal{S} \times \mathcal{A}} \rho_k^\pi(s') \pi(a \mid s') p(s'' \mid s', a)
\end{aligned} \tag{2.64}$$

□

Definition 2.3.5. The *state visitation distribution under policy* π, is defined as the function that assigns to each state the probability of being visited under policy π. It is denoted ρ^π and satisfies:

$$\forall s \in \mathcal{S} \quad \rho^\pi(s) \doteq \frac{1}{T} \sum_{t=0}^{T-1} \rho_t^\pi(s) \tag{2.65}$$

Remark 29. Intuitively, $\rho^\pi(s)$ is the fraction of the total time that the agent spends in state s when following the policy π.

Definition 2.3.6. The *state transition distribution under policy* π, denoted by ρ^π, is defined as the function that assigns to each state the probability of being visited under policy π. It satisfies:

$$\rho^\pi(s \to s') \doteq \frac{1}{T} \sum_{t=0}^{T-1} \rho_t^\pi(s \to s') \tag{2.66}$$

Remark 30. Intuitively, $\rho^\pi(s \to s')$ is the fraction of the total time that the agent spends in state s' after the state s when following the policy π.

Definition 2.3.7. The *action visitation distribution under* π, denoted by $\rho^\pi(\cdot, \cdot)$, is defined as the function that assigns to each state and action the probability of being visited under policy π. It satisfies:

$$\forall (s, a) \in \mathcal{S} \times \mathcal{A} \quad \rho^\pi(s, a) \doteq \rho^\pi(s)\pi(a \mid s) \tag{2.67}$$

Remark 31. Intuitively, $\rho^\pi(s, a)$ is the fraction of the total time that the agent spends in state s and takes the action a when following the policy π.

Definition 2.3.8. The *return visitation distribution under* π, denoted by $\rho^\pi(s, a, g)$, is defined as the function that assigns to each state, action and return, the probability of being visited under policy π. It satisfies:

$$\rho^\pi(s, a, g) \doteq \rho^\pi(s, a) P(G_t = g \mid S_t = s, A_t = a) \tag{2.68}$$

Remark 32. Intuitively, $\rho^\pi(s, a, g)$ is the fraction of the total time that the agent spends in state s, takes the action a and get the return g when following the policy π.

Remark 33. In practice, the previous definitions are not commonly used, but they are valuable for the subsequent demonstrations. They serve to abstract away the inherent sequential nature of MDPs, thereby simplifying the analysis.

Cost function and approximator of its gradient

In this section, we explore the definition of the cost function, which quantifies the imprecision of our policy evaluation. By applying an optimization algorithm to this cost function, we can iteratively improve the approximation of the value function. The optimization algorithms that are commonly used in this context, such as stochastic gradient descent (SGD) [226] or Adam [134], rely on an approximation of the gradient of the cost function. We will also discuss methods for approximating this gradient.

Definition 2.3.9. Let $||\cdot||$ be a norm on $\mathbb{R}$, $f : \mathcal{R} \to \mathbb{R}$ an arbitrary function on states and $\rho : \mathcal{S} \to \mathbb{R}$ a state visitation distribution. The *weighted norm with respect to the state visitation* ρ, denoted $||\cdot||_\rho$, is defined as follows.

$$||f||_\rho \doteq \sum_{s \in \mathcal{S}} \rho(s)||f(s)|| \tag{2.69}$$

Similarly, we can define the weighted norm on the visitation of the action-states:

Definition 2.3.10. Let $||\cdot||$ be a norm on $\mathbb{R}$, $f : \mathcal{S} \to \mathbb{R}$ an arbitrary function on states, and $\rho : \mathcal{S} \to \mathbb{R}_+$ a state visitation distribution. The *weighted seminorm with respect to the state visitation* ρ, denoted $||\cdot||_\rho$ is defined as

$$||f||_\rho \doteq \sum_{s \in \mathcal{S}} \rho(s)||f(s)|| \tag{2.70}$$

Remark 34. Although we won't exhibit the proof in this document, $||f||_\rho$ is indeed a seminorm on $\mathbb{R}^{\mathcal{S}}$.

Remark 35. Intuitively, we want to give more weight to the states that are more frequently visited. This definition is useful because it allows to naturally give more weight to the states that are more frequently visited. Moreover, we will see in the following that it also allows to simplify the algorithmic implementation, by sparing the estimation of a normalization with respect to the number of visits.

Suppose we aim to estimate the value function V^π of a policy π using a function approximator, denoted as V_ϕ. The primary goal is to ensure that V_ϕ closely approximates V^π. In other words, the objective is to minimize the distance $||V^\pi - V_\phi||_\rho^2$, representing the squared norm of the difference between the value function and its estimate.

Definition 2.3.11. The *cost function*, denoted $L : \mathbb{R}^d \to \mathbb{R}$, is defined as the function that maps parameter $\phi \in \mathbb{R}^d$ to the corresponding weighted mean distance between the state-value function V^π and its estimation V_ϕ.

$$L(\phi) \doteq ||V_\phi - V^\pi||_\rho^2 = \sum_{s \in \mathcal{S}} \rho(s)(V_\phi(s) - V^\pi(s))^2 \tag{2.71}$$

Remark 36. In this document, we use the terms *loss function*, *objective function* and *cost function* interchangeably. While there is a subtle semantic distinction between the three, many authors treat them as synonymous. For the sake of simplicity and consistency with prevailing literature, we adopt this same assumption.

Remark 37. The distribution ρ can be any probability distribution; it is not necessary for it to be identical to ρ^π. In other words, the data used to approximate V^π does not necessarily need to be collected under the policy π itself. As a consequence of Remark 16, when $\rho = \rho^\pi$, the notation is simplified. For example, we can write $\mathbb{E}_\pi \doteq \mathbb{E}_{S_t, A_t, G_t \sim \rho^\pi}$.

Theorem 2.3.3. The cost function is differentiable and

$$\nabla_\phi L(\phi) = 2\mathbb{E}_{S_t \sim \rho}\left[(V_\phi(S_t) - V^\pi(S_t))\nabla_\phi V_\phi(S_t)\right] \tag{2.72}$$

Proof.

$$\begin{aligned}
\nabla_\phi L(\phi) &= \nabla_\phi ||V_\phi - V^\pi||_\rho^2 && ((2.71)) \\
&= \nabla_\phi \sum_{s \in \mathcal{S}} \rho(s)(V_\phi(s) - V^\pi(s))^2 \\
&= \sum_{s \in \mathcal{S}} \rho(s)\nabla_\phi (V_\phi(s) - V^\pi(s))^2 \\
&= 2\sum_{s \in \mathcal{S}} \rho(s)(V_\phi(s) - V_\pi(s))\nabla_\phi V_\phi(s) \\
&= 2\mathbb{E}_{S_t \sim \rho}\left[(V_\phi(S_t) - V^\pi(S_t))\nabla_\phi V_\phi(S_t)\right] && ((2.72))
\end{aligned}$$

□

Remark 38. The previous result is also valid for the action-value function:

$$\nabla_\phi L(\phi) = 2\mathbb{E}_{S_t \sim \rho}\left[(Q_\phi(S_t, A_t) - Q^\pi(S_t, A_t))\nabla_\phi Q_\phi(S_t, A_t)\right] \tag{2.73}$$

The gradient of the cost function is expressed as an expectation, allowing for its estimation through sampling. To estimate V^π, we can employ one of the methods discussed in Section 2.2, such as MC, TD(0), or even TD(λ).

To illustrate, consider that we have collected interactions under policy π (implying that $\rho = \rho^\pi$). If we use MC to approximate $V^\pi(S_t)$, defined as $V^\pi(s) = \mathbb{E}_{A_t, G_t \sim \rho(s,\cdot,\cdot)}[G_t]$, the gradient of the cost function can be approximated as follows:

$$\nabla_\phi L(\phi) = 2\mathbb{E}_\pi \left[(V_\phi(S_t) - \mathbb{E}_\pi[G_t]) \nabla_\phi V_\phi(S_t) \right] \tag{2.74}$$

$$= 2\mathbb{E}_\pi \left[(V_\phi(S_t) - G_t) \nabla_\phi V_\phi(S_t) \right] \tag{2.75}$$

In the case of using SGD as the optimization algorithm, the update rule is as follows[1]:

$$\phi \leftarrow \phi - \alpha (V_\phi(S_t) - G_t) \nabla_\phi V_\phi(S_t) \tag{2.76}$$

where α is the *learning rate*.

2.3.2 From online deep Q-learning to Deep Q-Networks

The first tabular algorithm that can be adapted with function approximators is the Q-learning algorithm (see Algorithm 9). To achieve this, we replace the tabular form of Q with a parameterized version Q_ϕ and employ an optimization algorithm to minimize the cost function, defined as follows:

$$L(\phi) = \mathbb{E}_\pi \left[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} (Q_\phi(S_{t+1}, a')) - Q_\phi(S_t, A_t) \right] \tag{2.77}$$

Remark 39. In practice, while computing the gradient of $L(\phi)$ in Equation (2.77), a *stop gradient* operation is applied around the target value. This is crucial to avoid violating the causality principle, which states that the future is independent of the past given the present. Therefore, the value function of the next state should not depend on rewards received prior to the current state. Consequently, in Algorithm 12, we denote the target value as y, rather than y_ϕ, even though y is dependent on ϕ.

Contrary to Q-learning, Online Deep Q-learning exhibits instability due to the bootstrapping. The next state value in the target $y = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_\phi(S_{t+1}, a')$ undergoes continual changes with each iteration. To illustrate this, we have represented on Figure 2.4 the evolution of the loss several trainings.

To solve the instability problem of Online Deep Q-learning, Mnih et al. [188] introduced two new ingredients which are a target network and experience replay.

- The target network, denoted by parameters ϕ_{targ}, is an old version of the online network with parameters ϕ. Every τ timesteps, the online parameters are copied into the target parameters ($\phi_{\text{targ}} \leftarrow \phi$).
- Experience replay, initially introduced by Lin [165], involves storing observed transitions for some time and then uniformly sampling from this buffer to update the network.

The resulting algorithm is called Deep Q-Networks (DQN). The results are shown in Figure 2.5.

[1] The factor 2 is omitted as it can be integrated into the learning rate α.

Algorithm 12: Online Deep Q-learning.

arbitrarily initialize Q_ϕ
$\varepsilon \leftarrow 1$
repeat forever
  initialize the environment and observe the initial state S_0
  repeat
    choose the action $A_t \sim \varepsilon$-greedy(Q_ϕ)
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    compute target value

$$y = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} (Q_\phi(S_{t+1}, a'))$$

    update ϕ by one step of gradient descent in the direction of

$$\nabla_\phi (y - Q_\phi(S_t, A_t))^2$$

    decrease ε
  until *the end of the episode*;

The advent of DQN marked a pivotal moment in the evolution of RL, setting a new standard for what could be achieved in this domain. One of the most notable milestones achieved through the application of DQN was its remarkable success in mastering a variety of Atari 2600 video games [21]. This breakthrough, first demonstrated by Mnih et al. [188] in their seminal 2015 paper, showcased DQN's ability to outperform existing RL methods and even human experts in complex gaming environments. The significance of this achievement lay not only in its demonstration of superior gaming performance but also in the validation of DQN as a powerful tool for handling high-dimensional sensory inputs—something that traditional Q-learning methods struggled with.

2.3.3 From of DQN to Rainbow

Limitations and challenges of DQN

.

Despite the significant progress made by DQN, several inherent limitations and challenges have been identified that have motivated further development and adaptation. Some of them are

- **The exploration vs. exploitation dilemma:** A fundamental challenge in DQN, as in all RL algorithms, is the balance between exploration and exploitation. While DQN uses ϵ-greedy to address this, finding the optimal balance remains complex, often leading to either sub-optimal policy learning or inefficient exploration.
- **Overestimation of Q-values**: DQN tends to overestimate Q-values due to the max operator used in Q-learning. This overestimation occurs because the same network is used to both select and evaluate an action, leading to biased high value estimates and consequently sub-optimal policy decisions.

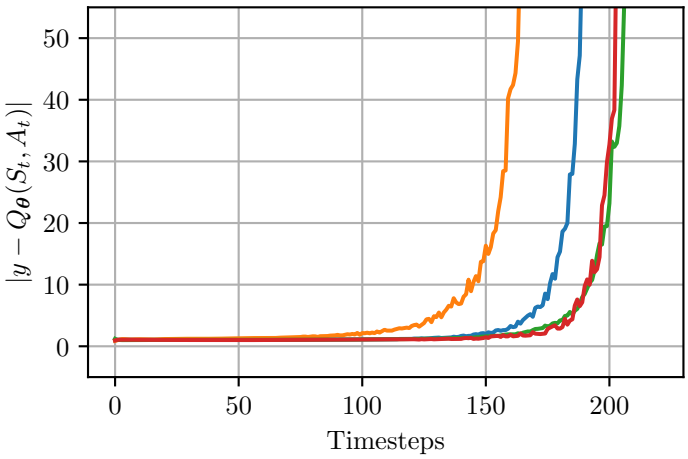


Figure 2.4: Evolution of the absolute loss $|y - Q_\phi(S_t, A_t)|$ over timesteps in online Deep Q-learning on CartPole-v1. The figure depicts four separate experiments, illustrating the point of divergence in the loss function as a consequence of the instability in the target value.

- **Stability:** One of the critical challenges in DQN is maintaining stability during the training process. The repeated updates to the Q-network, using correlations present in the sequence of observations, can lead to significant fluctuations in the policy and convergence difficulties.

These challenges have been the incentive for the development of several extensions to the original DQN algorithm that aim to address each of these specific issues while retaining the strengths of the original algorithm. In the following, we present some of these enhancements, which eventually led to an algorithm that combines them all, called Rainbow [110].

Some DQN variants

Double Q-learning Among the notable enhancements to the DQN framework is Double Q-Learning, which addresses the overestimation of Q-values inherent in the standard Q-learning approach. This enhancement has already been introduced in Section 2.2.3, and we invite the reader to refer to it.

Prioritized Experience Replay DQN utilizes a uniform sampling method from the experience replay buffer, treating all experiences as equally important for learning. However, not all experiences contribute equally to the learning process. Prioritized Experience Replay (PER) [232] addresses this by assigning a priority to each experience in the replay buffer, based on the magnitude of the TD error. This error reflects the difference between the currently estimated Q-values and the target Q-values, indicating how much learning potential an experience holds. The sample priority for transition j in buffer is computed as

$$P(j) = \frac{{p_i}^\alpha}{\sum_i {p_i}^\alpha} \tag{2.78}$$

Algorithm 13: Deep Q-Networks (DQN).

Input: a differentiable action-value function Q_θ parameterized with θ
randomly initialize θ and $\theta_{\text{targ}} \leftarrow \theta$
$\varepsilon \leftarrow 1$
initialize a empty replay buffer $\mathcal{D}$
repeat forever
  initialize the environment and get the initial state S_0
  repeat
    choose the action $A_t \sim \varepsilon-\text{greedy}(Q_\theta(S_t, \cdot))$
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    store the transition $(S_t, A_t, R_{t+1}, S_{t+1})$ into $\mathcal{D}$
    uniformly sample a batch of transitions $B = \{(s, a, r, s')\}$ from $\mathcal{D}$
    compute target value

$$y(r, s') = r + \gamma \max_{a' \in \mathcal{A}} (Q_{\theta_{\text{targ}}}(s', a'))$$

    update θ by one step of gradient descent in the direction of

$$\nabla_\theta \frac{1}{|B|} \sum_{(s,a,r,s') \in B} (y(r, s') - Q_\theta(s, a))^2$$

    every τ timesteps, $\theta_{\text{targ}} \leftarrow \theta$
    decrease ε
  until *the end of the episode*

where p_i is the TD-error when it was used the last time as a sample.

It is important to note that prioritized sampling alters the original distribution of experiences, resulting in a discrepancy where ρ in Equation (2.73) no longer equals ρ_π. To address this problem, we use importance sampling (refer to Section 2.2.4), which introduces a corrective weight into the gradient approximation, ensuring that the learning process remains unbiased despite prioritization. This weight, denoted as w, is computed as

$$w_j \leftarrow \frac{(NP(j))^{-\beta}}{\max_i w_i} \tag{2.79}$$

Overall, this approach ensures that experiences from which the agent can learn the most are replayed more often, leading to more efficient and effective learning.

Dueling Networks A proposed method for enhancing the architecture of the standard DQN network is Dueling DQN [278]. This approach, while not directly aimed at mitigating overestimation, significantly improves the estimation of state and action values. The core idea behind Dueling DQN is the decomposition of the Q-value estimation into two distinct components: one for the state value V^π and another for the action advantage A^π. The architecture comprises two separate streams that converge into a single output, formulated as:

$$Q_\phi(S_t, A_t) = V_\phi(S_t) + A_\phi(S_t, A_t) \tag{2.80}$$

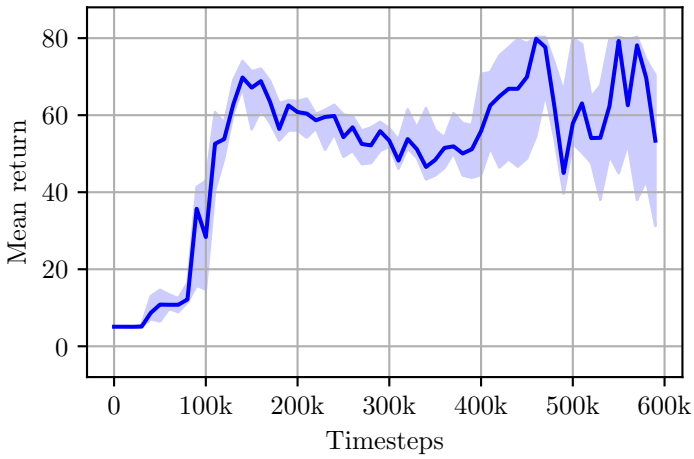


Figure 2.5: Performance of Deep Q-Networks (DQN) on the CartPole-v1 environment. This figure illustrates the effectiveness of DQN, integrating a target network and experience replay, in stabilizing the learning process.

The Dueling DQN architecture intuitively allows the network to learn the value of being in each state, independent of the specific actions taken. This is particularly beneficial in environments where the quality of a state does not significantly depend on the action chosen, enabling the network to discern which states are valuable without needing to evaluate the effect of each action in every state.

However, a challenge arises in terms of *unidentifiability* when attempting to decompose Q_ϕ into V_ϕ and A_ϕ, as it is not possible to uniquely determine V_ϕ and A_ϕ from Q_ϕ alone due to the numerous potential combinations that can sum up to the same Q_ϕ value. To address this, Dueling DQN introduces a constraint: it forces the mean advantage value to be zero. This is achieved by adjusting the advantage function as follows:

$$Q_\phi(S_t, A_t) = V_\phi(S_t) + \left(A_\phi(S_t, A_t) - \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} A_\phi(S_t, a) \right) \tag{2.81}$$

This approach stabilizes the decomposition of Q_ϕ into V_ϕ and A_ϕ.

Multi-step Learning As previously discussed in Section 2.2.3, using the 1-step return $(R_t + V(S_{t+1})$ is a low-variance but biased estimator of the value. Using the n-steps return instead, as defined in Definition 2.2.1, can limit this problem. The use of the n-steps return generally leads to faster learning.

Distributional DQN DQN approximate the expected value of the Q-function, which only captures the mean of the return distribution. In contrast, Distributional DQN, particularly the C51 algorithm introduced in [23], represents a significant shift by modeling the entire distribution of possible returns. Instead of estimating a single expected value, C51 discretizes the range of possible returns into 51 distinct bins, each representing a possible value the return can take, and

learns the probability associated with each bin. The Q-value in this approach is computed as the expectation of this distribution:

$$Q_\phi(s,a) = \sum_{i=1}^{51} p_\phi(s,a,i) z_i \tag{2.82}$$

where $p_\phi(s,a,i)$ is the learned probability of the i-th bin for a state-action pair (s,a), and z_i is the corresponding return value for that bin. By capturing the full distribution of returns, Distributional DQN provides a more robust understanding of the potential effects of actions, leading to more robust inference.

Noisy Nets In DQN, exploration is handled by an ϵ-greedy policy (see definition 2.1.26), where the choice between exploration and exploitation is governed by a randomly chosen threshold. However, this method can be inefficient and oversimplified for complex environments. Noisy Net, as introduced in [84], presents an innovative approach to driving exploration through the network architecture itself. In Noisy Net, the network weights contain a stochastic component, effectively introducing noise into the decision making process. Each network weight is changed to:

$$\phi_i = \mu_i + \sigma_i \odot \epsilon \tag{2.83}$$

where μ and σ are learnable parameters, and ϵ is a noise term sampled from a normal distribution. This addition of noise allows the network to explore different strategies even in the absence of explicit exploration mechanisms such as ϵ greedy. In addition, the exploration behavior is learned and adapted as part of the training process, with the network learning when and how much to explore based on feedback from the environment.

Rainbow DQN

The individual improvements we have discussed, including Double Q-learning, Prioritized Experience Replay, Dueling DQN, Multi-Step Learning, Distributional DQN, and Noisy Nets, have each demonstrated significant performance improvements, particularly in the context of the Atari 2600 [21]. Building on these singular successes, Hessel et al. [110] proposed a combination of these improvements into a cohesive algorithm called Rainbow DQN. As shown in Figure 2.6, this algorithm achieves significantly better results and has set a strong new baseline for the Atari benchmark.

2.3.4 Next steps in distributional RL

C51 marked a significant advance in distributional RL by estimating the full distribution of outcomes. It achieves this by using 51 discretely defined support points, each representing a potential outcome. However, this method is not without its limitations. The rigid, discrete framework of C51 can potentially oversimplify the complexity inherent in the distribution of returns.

In contrast, Quantile Regression-DQN (QR-DQN) [57] builds on the fundamental principles of C51 to provide a more refined approach through quantile regression. This technique allows for a dynamically determined set of support points, providing a more flexible representation of the return distribution. Such

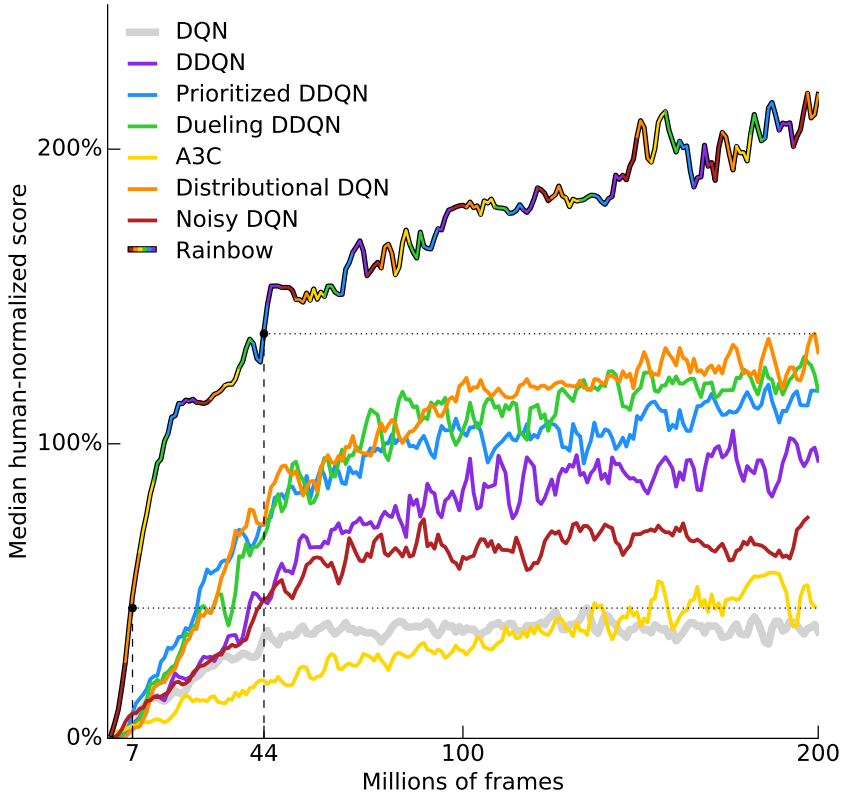


Figure 2.6: Median human-normalized performance across 57 Atari games. Figure from [110].

an approach provides a more granular and adaptive estimate of the range and probabilities associated with returns. This is particularly beneficial in environments characterized by significant variability and the presence of extreme outcomes, where an accurate assessment of the distribution is as important as understanding the average expected return. On the Atari benchmark, QR-DQN outperforms C51[2].

Building upon the advancements made by QR-DQN, Implicit Quantile Networks (IQN) [56] represent a further refinement of distributional RL. IQN extends the idea of quantile regression by introducing a more flexible and comprehensive approach to modeling the return distribution. While QR-DQN estimates a fixed number of quantiles, IQN innovates by allowing for an arbitrary number of quantiles, determined implicitly by the network. During each training or evaluation step, the network samples a different set of quantile fractions from a uniform distribution. These sampled fractions are then input into the network, which

[2]QR-DQN was developed at the same time as Rainbow, which is why QR-DQN isn't built on Rainbow and doesn't use it as a baseline.

produces the corresponding quantile estimates. As a result, IQN can adaptively represent the entire return distribution with a granularity and precision that surpasses that of QR-DQN. Empirical evaluations on standard benchmarks, such as the Atari suite, demonstrate that IQN outperforms QR-DQN, and achieves close performance with respect to Rainbow, while being simpler.

The Fully Parametric Quantile Function (FQF) [288] represents a noteworthy evolution of IQN. Diverging from IQN's strategy of sampling an arbitrary but fixed set of quantiles, FQF introduces a dual learning mechanism. It not only learns the values of the quantiles but also, through a fraction proposal network, identifies their optimal locations within the distribution. This network effectively maps the state to the most informative quantile fractions and optimizes this mapping by minimizing the opposite of the Wasserstein loss. The key innovation of FQF lies in its ability to actively learn which portions of the quantile distribution to focus on, enhancing the model's ability to adapt to the most relevant aspects of the return distribution.

2.3.5 Conclusion on value-based Deep RL

This section has explored the advancements in value-based Deep RL, from the foundational DQN to more recent innovations like IQN and Rainbow. These methods have significantly enhanced the ability to estimate and utilize value functions in complex decision-making tasks, leading to notable successes in various domains.

However, a primary issue with value-based Deep RL lies in its inherent limitation in handling continuous action spaces. By design, value-based methods aim to converge to the optimal value function by bootstrapping with a max operation over the action space, as shown in Equation (2.77). This approach is well-suited to discrete action spaces, where the action set is limited and enumerable. In continuous action spaces, however, this max operation becomes problematic due to the infinite number of possible actions, making it impractical to find the maximum value directly.

$$L(\phi) = \mathbb{E}_\pi \left[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_\phi(S_{t+1}, a') - Q_\phi(S_t, A_t) \right] \qquad ((2.77))$$

This limitation necessitates alternative approaches or approximations for dealing with continuous actions, leading to the exploration of other methods such as policy-based or actor-critic algorithms, which can operate more naturally in such environments.

Transitioning from this limitation, it is worth noting that there are situations where deriving an effective policy does not require a highly accurate model of the value function. The agent can learn to make effective decisions even with a value function that is approximated or not fully captured. This distinction underscores a key insight in RL: the primary goal is to learn a good policy, and a highly detailed value function, while useful, is not absolutely necessary for this purpose. This perspective argues for a focus on policy optimization, especially in contexts where the value function raises significant modeling issues.

2.4 Policy-Based Deep RL

After discussing value-based Deep RL, which primarily focuses on evaluating and maximizing the value function, we move on to discussing policy-based methods. Unlike value-based approaches, policy-based methods directly learn the agent's policy without explicitly estimating value functions. In this section, we describe how these methods work, what their unique advantages are and how they differ from the value-based strategies.

2.4.1 Some additional definitions and tools

Policy

Definition 2.4.1. A *parametrized policy*, is defined as a policy whose behavior is determined by a parameter vector in $\mathbb{R}^d$, where d denotes the number of parameters. This parameter vector is commonly represented by θ, and the corresponding parametrized policy is denoted by $\pi_\theta(\cdot \mid \cdot)$. An alternative notation often used is $\pi(\cdot \mid \cdot, \theta)$.

Definition 2.4.2. A *differentiable policy* refers to a parametrized policy that is differentiable with respect to its parameters. In such instances, the expression

$$\nabla_\theta \pi_\theta$$

is well-defined. In the following, all policies are assumed to be differentiable.

The objective function

Consider a parameterized policy π_θ. The objective of the learning process is to identify the policy that maximizes the expected return. Hence, the goal is to find the parameter θ that maximizes the expected return. Similar to value-based methods, we need to define an *objective function* (see Definition 2.3.11) that is a function of θ. As opposed to value-based methods, the goal is not to minimize the approximation error of the value function, but rather to directly maximize the amount of reward the agent gets following π.

Definition 2.4.3. The *objective function*, denoted J, is defined as the expectation of the value of the initial state.

$$J(\theta) \doteq \mathbb{E}_{\pi_\theta}\left[V^{\pi_\theta}(S_0)\right] \tag{2.84}$$

Remark 40. This definition is quite natural, since the value function is equal to the average return under the policy π, where the initial state is sample by the environment[3].

Theorem 2.4.1.

$$J(\theta) = \sum_{s \in \mathcal{S}} \rho_0(s) V^{\pi_\theta}(s) \tag{2.85}$$

Proof. Immediate consequence of the Definitions 2.1.18 and 2.1.4. □

[3]Sometimes, the initial state is always the same (let's note this state s_0). In such cases, the objective function have a much simpler expression: $J(\theta) = V^{\pi_\theta}(s_0)$.

A reasonable approach to find the parameter θ that maximizes $J(\theta)$ is the use of gradient ascent. The category of RL algorithms that use gradient ascent for policy optimization is known as *policy gradient* methods. This class of algorithms relies on the assumption that an estimate of the gradient is available. In the next section, we introduce a method for approximating $\nabla_\theta J(\theta)$.

The policy gradient theorem

Theorem 2.4.2 (Policy gradient theorem)**.**

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(S_t, A_t)\nabla_\theta \log \pi_\theta(A_t \mid S_t)\right] \tag{2.86}$$

To prove this theorem, we need the following lemma.

Lemma 2.4.2.1. Let $\chi : \mathcal{S} \to \mathbb{R}$ and $\psi : \mathcal{S} \to \mathbb{R}$ be functions defined on a state space $\mathcal{S}$. Suppose $\psi(S^-) = \{0\}$. (In other words, for any terminal state s, then $\psi(s) = 0$.) Additionally, assume ψ satisfies to a specified recursive formula.

$$\forall s \in \mathcal{S} \quad \psi(s) = \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1(s \to s')\psi(s') \tag{2.87}$$

then,

$$\forall s \in \mathcal{S} \quad \psi(s) = \sum_{s' \in \mathcal{S}} \sum_{t=0}^{T-1} \rho_k^\pi(s \to s')\chi(s') \tag{2.88}$$

Proof of the lemma. In this proof, we unroll the sum until the end of the episode. Here is the first unrolling.

$$\begin{aligned}
\psi(s) &= \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s')\psi(s') && ((2.87)) \\
&= \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s') \left(\chi(s') + \sum_{s'' \in \mathcal{S}} \rho_1^\pi(s' \to s'')\psi(s'')\right) \\
&= \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s')\chi(s') + \sum_{s'' \in \mathcal{S}} \underbrace{\sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s')\rho_1^\pi(s' \to s'')}_{=\rho_2(s \to s'')} \psi(s'') \\
&= \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s')\chi(s') + \sum_{s'' \in \mathcal{S}} \rho_2^\pi(s \to s'')\psi(s'') \\
&= \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s')\chi(s') + \sum_{s' \in \mathcal{S}} \rho_2^\pi(s \to s')\psi(s')
\end{aligned}$$

The expression in red is structurally identical to the initial summation, allowing for a repeated expansion up to T:

$$\psi(s) = \chi(s) + \sum_{s' \in \mathcal{S}} \rho_1^\pi(s \to s')\chi(s') + \cdots + \sum_{s' \in \mathcal{S}} \rho_T^\pi(s \to s')\psi(s')$$

On the other hand,

$$\sum_{s' \in \mathcal{S}} \rho_T^\pi(s \to s')\psi(s') = \sum_{s' \in \mathcal{S}^+} \underbrace{\rho_T^\pi(s \to s')}_{=0}\psi(s') + \sum_{s' \in \mathcal{S}^-} \rho_T^\pi(s \to s')\underbrace{\psi(s')}_{=0} = 0$$

Consequently,

$$\begin{aligned}\psi(s) &= \chi(s) + \sum_{t=1}^{T-1} \sum_{s' \in \mathcal{S}} \rho_t^\pi(s \to s')\chi(s') \\ &= \sum_{s' \in \mathcal{S}} \underbrace{\rho_0^\pi(s \to s')}_{=\mathbb{1}_{s=s'}} \chi(s') + \sum_{t=1}^{T-1} \sum_{s' \in \mathcal{S}} \rho_t^\pi(s \to s')\chi(s') \\ &= \sum_{s' \in \mathcal{S}} \sum_{t=0}^{T-1} \rho_t^\pi(s \to s')\chi(s') \qquad ((2.88))\end{aligned}$$

□

Proof of the policy gradient theorem.

$$J(\theta) = \sum_{s \in \mathcal{S}} \rho_0(s) V^{\pi_\theta}(s) \qquad ((2.85))$$

$$\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} \rho_0(s) \nabla_\theta V^{\pi_\theta}(s) \qquad (2.89)$$

We evaluate the gradient of the value function $\nabla_\theta V^{\pi_\theta}(s)$.

$$\begin{aligned}V^{\pi_\theta}(s) &= \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) Q^{\pi_\theta}(s, a) \qquad ((2.23)) \\ \nabla_\theta V^{\pi_\theta}(s) &= \sum_{a \in \mathcal{A}} \nabla_\theta (\pi_\theta(a \mid s) Q^{\pi_\theta}(s, a)) \\ &= \sum_{a \in \mathcal{A}} (Q^{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a \mid s) + \pi_\theta(a \mid s) \nabla_\theta Q^{\pi_\theta}(s, a)) \\ &= \underbrace{\sum_{a \in \mathcal{A}} Q^{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a \mid s)}_{\doteq \chi_\theta(s)} + \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta Q^{\pi_\theta}(s, a) \\ &= \chi_\theta(s) + \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(r, s' \mid s, a)(r + V^{\pi_\theta}(s')) \\ &\qquad \text{(from (2.21))} \\ &= \chi_\theta(s) + \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \sum_{(s', r) \in \mathcal{S} \times \mathcal{R}} p(r, s' \mid s, a) \nabla_\theta V^{\pi_\theta}(s') \\ &= \chi_\theta(s) + \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(r, s' \mid s, a) \nabla_\theta V^{\pi_\theta}(s')\end{aligned}$$

Using:

$$\sum_{r \in \mathcal{R}} p(r, s' \mid s, a) = p(s' \mid s, a) \qquad (2.90)$$

It comes:

$$\begin{aligned}
\nabla_\theta V^{\pi_\theta}(s) &= \chi_\theta(s) + \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) \sum_{s'\in\mathcal{S}} p(s' \mid s,a) \nabla_\theta V^{\pi_\theta}(s') \\
&= \chi_\theta(s) + \sum_{s'\in\mathcal{S}} \underbrace{\sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) p(s' \mid s,a)}_{=\rho_1^{\pi_\theta}(s\to s')} \nabla_\theta V^{\pi_\theta}(s') \qquad \text{(See (2.58))} \\
&= \chi_\theta(s) + \sum_{s'\in\mathcal{S}} \rho_1^{\pi_\theta}(s\to s') \nabla_\theta V^{\pi_\theta}(s') \qquad (2.91)
\end{aligned}$$

Equation (2.91) is a recursive property on $\nabla_\theta V^{\pi_\theta}(s)$:

$$\nabla_\theta V^{\pi_\theta}(s) = \chi_\theta(s) + \sum_{s'\in\mathcal{S}} \rho_1^{\pi_\theta}(s\to s') \nabla_\theta V^{\pi_\theta}(s') \qquad ((2.91))$$

Using the result of the Lemma 2.4.2.1, we get:

$$\nabla_\theta V^{\pi_\theta}(s) = \sum_{s'\in\mathcal{S}} \sum_{t=0}^{T-1} \rho_t^{\pi_\theta}(s\to s') \chi_\theta(s') \qquad (2.92)$$

This reformulation enables the exclusion of $\nabla_\theta V^{\pi_\theta}(s')$ from the expression, facilitating the sampling of $\nabla_\theta V^{\pi_\theta}(s)$. We now return to the analysis of the objective function:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} \rho_0(s) \nabla_\theta V^{\pi_\theta}(s) && ((2.89)) \\
&= \sum_{s\in\mathcal{S}} \rho_0(s) \sum_{s'\in\mathcal{S}} \sum_{t=0}^{T-1} \rho_t^{\pi_\theta}(s\to s') \chi_\theta(s') && \text{(from (2.92))} \\
&= \sum_{s'\in\mathcal{S}} \sum_{t=0}^{T-1} \underbrace{\sum_{s\in\mathcal{S}} \rho_0(s) \rho_t^{\pi_\theta}(s\to s')}_{=\rho_t^{\pi_\theta}(s')} \chi_\theta(s') && \text{(from (2.60))} \\
&= \sum_{s'\in\mathcal{S}} \sum_{t=0}^{T-1} \rho_t^{\pi_\theta}(s') \chi_\theta(s') \\
&= \sum_{s'\in\mathcal{S}} T \rho^{\pi_\theta}(s') \chi_\theta(s') && \text{(from (2.65))} \\
&\propto \sum_{s\in\mathcal{S}} \rho^{\pi_\theta}(s) \chi_\theta(s)
\end{aligned}$$

Hence,

$$\nabla_\theta J(\theta) \propto \sum_{s\in\mathcal{S}} \rho^{\pi_\theta}(s) \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a \mid s) \qquad (2.93)$$

This quantity can only be sampled if it is expressed as the expectation of a

random variable. We now apply what is often called the *likelihood ratio trick*.

$$\begin{aligned}\nabla_\theta J(\theta) &\propto \sum_{s\in\mathcal{S}} \rho^{\pi_\theta}(s) \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \pi_\theta(a \mid s) \\ &= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left(Q^{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \right) \underbrace{\pi_\theta(a \mid s)\rho^{\pi_\theta}(s)}_{=\rho^{\pi_\theta}(s,a)} \qquad \text{(from (2.67))} \\ &= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left(Q^{\pi_\theta}(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \right) \rho^{\pi_\theta}(s,a)\end{aligned}$$

Therefore, we obtain an expression of the form

$$\sum_x f(x)P(X=x) = \mathbb{E}\left[f(X)\right]$$

Consequently,

$$\begin{aligned}\nabla_\theta J(\theta) &\propto \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(S_t,A_t) \frac{\nabla_\theta \pi_\theta(A_t \mid S_t)}{\pi_\theta(A_t \mid S_t)} \right] \\ &= \mathbb{E}_{\pi_\theta}\left[Q^{\pi_\theta}(S_t,A_t) \nabla_\theta \log \pi_\theta(A_t \mid S_t) \right] \qquad ((2.86))\end{aligned}$$

□

The result of the theorem gives an expression for the gradient that can be sampled, thus allowing the use of gradient ascent optimization methods. As explored in Section 2.2, there are several estimators for $Q^{\pi_\theta}(S_t, A_t)$. The following sections will examine the use of these estimators in the development of various algorithms.

Remark 41. The gradient of the objective function is not equal to the expected value, but proportional to it. The proportionality constant is approximately equal to the average duration of an episode. Knowledge of this constant is not essential, because if a value maximizes a function, it also maximizes any positive scalar multiple of that function.

2.4.2 On-policy methods

From REINFORCE to vanilla policy gradient

In this section, we study how to use the concept of maximizing the objective function introduced in the previous section. Equation (2.86) enables the estimation of the gradient of the objective function through sampling. Consequently, any gradient ascent method can be applied to maximize the objective function, thereby determining the policy that maximizes the expectation of the gain, as discussed in Section 2.4.1. The REINFORCE algorithm [284], also known as *MC gradient*, employs the MC estimate for the value function.

Theorem 2.4.3.

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\pi_\theta}\left[G_t \nabla_\theta \log \pi_\theta(A_t \mid S_t) \right] \qquad (2.94)$$

Proof. In Equation (2.86), the term $Q^{\pi_\theta}(S_t, A_t)$ is substituted by $\mathbb{E}_{\pi_\theta}[G_t \mid S_t = s, A_t = a]$, where the expectation is conditioned on $S_t = s$ and $A_t = a$. However, considering that G_t depends on future states and actions, which are independent of past events given S_t and A_t, the conditioning on S_t and A_t can be omitted for G_t. Consequently, this leads to a simplification of the expression to:

$$\begin{aligned}\nabla_\theta J(\theta) &\propto \mathbb{E}_{\pi_\theta}\left[\mathbb{E}_{\pi_\theta}[G_t \mid S_t = s, A_t = a]\nabla_\theta \log \pi_\theta(A_t \mid S_t)\right] \\ &\qquad\qquad \text{(from (2.86) and (2.19))} \\ &= \mathbb{E}_{\pi_\theta}\left[G_t \nabla_\theta \log \pi_\theta(A_t \mid S_t)\right] \qquad ((2.94))\end{aligned}$$

□

Under policy π, states, actions, and rewards (S_t, A_t, R_t) are sampled. Upon the end of an episode, the returns are computed as discounted sums of rewards, in accordance with Equation (2.12). The gradient of the objective function can then be approximated by the equation (2.94). Algorithm 14 shows the resulting algorithm using SGD (ascending) as the optimization method.

Algorithm 14: REINFORCE.

Input: a differentiable policy π_θ parametrized by θ
randomly initialize θ
repeat forever
  sample $\{S_0, A_0, R_1, \ldots, A_{T-1}, R_T\} \sim \pi_\theta$
  for *each time* $t \leq T$ *of the episode* **do**
    $G_t \leftarrow R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
    $\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log \pi_\theta(A_t \mid S_t)$

As we have seen in Section 2.2, MC estimates have high variance, which can make the learning process unstable. A viable strategy to reduce this variance is to subtract a *baseline* from G_t. While beyond our current scope, it can be shown that adding a baseline that depends only on states does not introduce bias. An optimal baseline is defined as a baseline that minimizes variance. Such a baseline is denoted b^* and we can show that:

$$b^*(s) = V^{\pi_\theta} + \frac{\operatorname{cov}_{\pi_\theta}\left(Q^{\pi_\theta}(s, A_t), \zeta_i(s, A_t)^2\right)}{\mathbb{E}_{\pi_\theta}\left[\zeta_i(s, A_t)^2\right]} \tag{2.95}$$

where

$$\forall i \in \mathbb{N}_d \quad \zeta_i(s, a) \doteq \frac{\partial}{\partial \theta_i} \log \pi_\theta(a \mid s) \tag{2.96}$$

This formulation has been demonstrated in [280]. If $\zeta_i(s, A_t)^2$ and $Q^{\pi_\theta}(s, A_t)$ are independent, then the optimal baseline for state s is simply $V^{\pi_\theta}(s)$. Although this condition is usually not met, V^{π_θ} is often used as the baseline in practice due to its simplicity. In doing so, it is in fact the advantage A^{π_θ} that is used in the optimization.

It is hypothesized that the features needed to learn an effective policy may be different from those needed to accurately approximate its value function. Consequently, a distinct set of parameters, separate from the policy parameters, is proposed to approximate V^{π_θ}. The resulting methods are known as *actor-critic* approaches, where *actor* is the policy and *critic* is the value function.

Incorporating this concept into REINFORCE leads of the vanilla policy gradient (VPG) algorithm, shown in Algorithm 15.

Algorithm 15: Vanilla Policy Gradient (VPG).

Input: a differentiable policy π_θ parametrized by θ
Input: a differentiable value function V_ϕ parametrized by ϕ
randomly initialize θ and ϕ
repeat forever
  sample $\{S_0, A_0, R_1, \ldots, A_{T-1}, R_T\} \sim \pi_\theta$
  for *each time* $t \leq T$ *of the episode* **do**
    $G_t \leftarrow R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
    $\theta \leftarrow \theta + \alpha \left(G_t - V_\phi(S_t)\right) \nabla_\theta \log \pi_\theta(A_t \mid S_t)$
    $\phi \leftarrow \phi + \alpha (V_\phi(S_t) - G_t)^2$

Advantage Actor-Critic

Although integrating a learned value estimate as a baseline in the REINFORCE algorithm reduces variance, as discussed earlier, its practical implementation frequently encounters significant noise that hampers the training. Additionally, the algorithm tends to struggle with effective exploration, leading to sub-optimal performance.

Starting from this point, Mnih et al. [189] introduced two significant enhancements. To stabilize the learning, they proposed to replace the MC estimator with the n-steps TD estimator, (similarly to the method described in Section 2.2.3). They also propose adding an entropy regularization originally proposed by Williams and Peng [285] to discourage premature convergence to a sub-optimal policy. Hence, the optimization is performed in the direction of the gradient expressed as:

$$(G_{t:t+n} - V_{\phi'}(S_t)) \nabla_\theta \log \pi_\theta(A_t \mid S_t) + \beta \nabla_\theta \mathcal{H}(\pi_\theta(\cdot \mid S_t)) \tag{2.97}$$

where $\mathcal{H}$ is the entropy and β the hyperparameter that controls the weight of the entropy regularization term. These modifications leads to the *Advantage Actor-Critic* (A2C) algorithm[4].

Proximal Policy Optimization

Proximal Policy Optimization (PPO) significantly builds upon the A2C framework by introducing four key enhancements that improve performance and stability[5]:

[4] A2C is actually the synchronous version of *Asynchronous Advantage Actor-Critic* (A3C). A2C is often preferred over A3C because of its computational efficiency on modern hardware and its more stable learning process, as it is updated synchronously by multiple agents, reducing problems with stale gradients and simplifying implementation. In the end, A2C performed better than its asynchronous equivalent, and the noise introduced by asynchronism doesn't seem to provide any performance advantage.

[5] PPO was not originally conceptualized as an extension of A2C, subsequent research has demonstrated that A2C can be viewed as a special case of PPO [117].

Algorithm 16: Advantage Actor-Critic (A2C).

Input: a differentiable policy π_θ parametrized by θ
Input: a differentiable value function V_ϕ parametrized by ϕ
Input: a fixed number of interaction steps n
randomly initialize θ and ϕ
repeat forever
  sample n steps $\{S_{t_0}, A_{t_0}, R_{t_0+1}, \ldots, A_{t_0+n-1}, R_{t_0+n}\} \sim \pi_\theta$
  for *each step* $t \leq t_0 + n$ **do**
    compute $G_{t:t_0+n}$ ($(t_0 + n - t)$-steps return)
    $\theta \leftarrow \theta + \alpha \left(G_{t:t_0+n} - V_\phi(S_t)\right) \nabla_\theta \log \pi_\theta(A_t \mid S_t) + \beta \nabla_\theta \mathcal{H}(\pi_\theta(\cdot \mid S_t))$
    $\phi \leftarrow \phi - \alpha(V_\phi(S_t) - G_{t:t_0+n}) \nabla_\phi V_\phi(S_t)$
    $t_0 \leftarrow t_0 + n$

Generalized Advantage Estimation (GAE) GAE [238] is a method for estimating the advantage, functionally analogous to the λ-return (Definition 2.2.2), GAE blends the ideas of MC and TD estimates to strike a balance between bias and variance. The GAE formula is:

$$A_t^{\mathrm{GAE}(\lambda)} \doteq \sum_{k=0}^{n-1} (\gamma\lambda)^k (R_{t+k} + \gamma V_\phi(S_{t+k+1}) - V_\phi(S_{t+k})) \tag{2.98}$$

where, λ is the parameter that balances the bias-variance trade-off. This approach leads to more stable and efficient policy updates [10].

Multiple optimization epochs using minibatches PPO differs from the single-update method of A2C by using multiple optimization epochs over mini-batches for policy refinement. This approach, by iterating over multiple epochs, results in a scenario where the policy in subsequent iterations is not the same as the one initially used for action sampling. To address this divergence, PPO uses importance sampling as a scaling mechanism to adjust for the changes in policy across iterations. Overall, this method allows the algorithm to more effectively refine its policy updates.

Normalization of advantage PPO normalizes the advantage estimates to maintain a consistent range and variance.

Value function clipping PPO introduces value function clipping to maintain stability during learning. The clipping operation for the value function is defined as:

$$V^{clip}(s) = \mathrm{clip}(V(s), V_{old}(s) - \epsilon, V_{old}(s) + \epsilon) \tag{2.99}$$

where $V(s)$ is the current estimated value, $V_{old}(s)$ is the old value estimate, and ϵ is a hyperparameter defining the clipping range. This technique limits the extent of the update to the value function, ensuring gradual and stable improvements.

Surrogate objective clipping Finally, PPO incorporates a clipped surrogate objective, a critical element for moderating the magnitude of policy updates. The basic concept is to regulate the ratio of the new policy to the old policy,

denoted $\frac{\pi_{\text{new}}(a|s)}{\pi_{\text{old}}(a|s)}$, by clipping it within a defined range set by the parameter ϵ. This clipping acts as a safeguard, ensuring that updates do not deviate significantly from the previous policy. The efficacy of this clipped objective is notable; it has been shown to deliver performance comparable to Trust Region Policy Optimization (TRPO) [237], but does so with reduced computational requirements [115].

These algorithmic advancements are complemented by key implementation details that are critical for their success [115]. Further elements are discussed in Chapter 5.2.

2.4.3 Off-policy Methods

Deterministic Policy Gradient

The policy gradient methods introduced so far typically work with stochastic policies, where the policy outputs a probability distribution over actions. However, when dealing with environments that have continuous action spaces, stochastic policy gradients can face challenges in efficiency and performance.

The DPG algorithm [249] addresses these challenges by adopting a deterministic approach. Instead of a stochastic policy, DPG uses a deterministic policy (see Remark 15) that maps states directly to specific actions, simplifying the gradient computation.

The new nature of policy requires us to revisit the policy gradient theorem (see Theorem 2.4.2) and give it a version suitable for deterministic policy.

Theorem 2.4.4 (Deterministic policy gradient theorem)**.**

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \pi_\theta(S_t) \nabla_a Q^{\pi_\theta}(S_t, a)|_{a=\pi_\theta(S_t)}\right] \tag{2.100}$$

Proof.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\pi_\theta}\left[V^{\pi_\theta}(S_0)\right] && ((2.3.11)) \\
&= \nabla_\theta \sum_{s\in\mathcal{S}} \rho_0(s) V^{\pi_\theta}(s) \\
&= \sum_{s\in\mathcal{S}} \rho_0(s) \nabla_\theta V^{\pi_\theta}(s) \\
&= \sum_{s\in\mathcal{S}} \rho_0(s) \nabla_\theta Q^{\pi_\theta}(s, \pi_\theta(s)) && \text{(deterministic policy)} \\
&= \sum_{s\in\mathcal{S}} \rho_0(s) \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s)} && \text{(chain rule)} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \pi_\theta(S_t) \nabla_a Q^{\pi_\theta}(S_t, a)|_{a=\pi_\theta(S_t)}\right] && ((2.100))
\end{aligned}$$

□

Formulated in this way, the gradient can indeed be approximated, and hence used for a gradient ascent method.

In stochastic policy gradient methods, exploration is inherently achieved through the random selection of actions. In contrast, the DPG method lacks this intrinsic exploration mechanism due to its deterministic nature. To address this, DPG algorithms often integrate explicit exploration strategies, such as

adding noise to action outputs, promoting diverse experiences and effective learning. Alternatively, sample collection can be conducted using a different policy, known as the behavioral policy (see Section 2.1.5). Consequently, in both cases, the approach aligns with an off-policy algorithm framework. Considering the inapplicability of the importance sampling technique to a deterministic policy, Silver et al. [249] suggest an alternative objective formulation.

$$J(\theta) \doteq \mathbb{E}_{\beta}\left[V^{\pi_\theta}(S_0)\right] \tag{2.101}$$

This formulation is based not on the distribution under π but on β (as discussed in Remark 37).

Designed specifically for problems involving continuous actions, DPG represents a significant advance in policy gradient methods. However, DPG is not without its challenges; it exhibits instability and requires precise optimization of parameters for effective implementation. In the following section, we discuss strategies for stabilizing DPG using the techniques introduced earlier in this document.

Deep Deterministic Policy Gradient and its evolution

Building upon the foundation of DPG, the Deep Deterministic Policy Gradient (DDPG) algorithm [164] addresses the inherent instability seen in DPG, akin to the challenges faced by online Q-learning (see 2.3.2). This instability arises primarily from the correlations present in the sequential data and the continual updates to the policy and value estimates.

DDPG, inspired by DQN [188] approach, introduces two key concepts to mitigate these issues:

- **Experience Replay** DDPG utilizes a replay buffer to store and randomly sample experience, breaking the correlation in the observation sequence. This random sampling stabilizes the training process.
- **Target Networks** DDPG employs target networks for both the actor and critic, similar to DQN. However, unlike DQN, DDPG updates these target networks using *soft updates*. The target networks are updated using a mix of the main network parameters and the target network parameters:

$$\phi_{\text{targ}} \leftarrow \tau\phi + (1-\tau)\phi_{\text{targ}} \tag{2.102}$$
$$\theta_{\text{targ}} \leftarrow \tau\theta + (1-\tau)\theta_{\text{targ}} \tag{2.103}$$

  where τ, usually small, is a parameter controlling the update rate, ensuring slow and stable changes.

Additionally DDPG uses the learned policy augmented by Gaussian noise as the behavioral policy.

$$\beta(\cdot \mid s) = \mathcal{N}(\pi(s), \Sigma) \tag{2.104}$$

This addition of noise ensures sufficient exploration of the action space.

In summary, DDPG extends DPG by incorporating experience replay and softly updated target networks to stabilize the learning process. The integration of exploration through additive noise allows the algorithm to effectively explore, making it suitable for complex environments with continuous action spaces.

Algorithm 17: Deep Deterministic Policy Gradient (DDPG).

Input: a differentiable actor π_phi parametrized by θ
Input: a differentiable critic Q_ϕ parametrized by ϕ
randomly initialize θ and $\theta_{\text{targ}} \leftarrow \theta$
randomly initialize ϕ and $\phi_{\text{targ}} \leftarrow \phi$
initialize a empty replay buffer $\mathcal{D}$
repeat forever
  initialize the environment and get the initial state S_0
  repeat
    select action $A_t \sim \mathcal{N}(\pi_\theta(S_t), \Sigma)$
    take action A_t and get the reward R_{t+1} and the new state S_{t+1}
    store the transition $(S_t, A_t, R_{t+1}, S_{t+1})$ into $\mathcal{D}$
    uniformly sample a batch of transitions $B = \{(s, a, r, s')\}$ from $\mathcal{D}$
    update ϕ by one step of gradient descent in the direction of

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \left(r + \gamma Q_{\phi_{\text{targ}}}\left(s', \pi_{\theta_{\text{targ}}}\left(s'\right)\right) - Q_\theta(s,a)\right)^2$$

    update θ by one step of gradient ascent in the direction of

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi\left(s, \pi_\theta(s)\right)$$

    Update the targets parameters

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

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

  until *the end of the episode*;

Building upon the success of DDPG, the Twin Delayed DDPG (TD3) algorithm, introduced in [86], further refines the approach to mitigate the overestimation bias, a notable issue in methods derived from Q-learning (see Section 2.2.3). TD3 introduces three critical enhancements: twin critics, delayed policy updates, and target policy smoothing. It employs two separate critic networks (hence *twin*) and uses their minimum value to estimate the Q-value, effectively reducing overestimation. The delayed policy updates involve updating the policy less frequently than the value networks, mitigating the risk of value estimates leading policy updates astray. Target policy smoothing, a novel technique, adds noise to the target action, thereby smoothing out the value estimate against small changes in action space. These improvements collectively result in more stable and reliable training, especially in environments with continuous action spaces, positioning TD3 as a more robust alternative to DDPG.

Concurrently, Barth-Maron et al. [20] integrated the distributional approach originally proposed by Bellemare et al. [23] into the DDPG framework (see Section 2.3.3) and adapted it for a distributed setting, resulting in the creation of Distributed Distributional Deep Deterministic Policy Gradients (D4PG). This

extension enables D4PG to effectively handle probability distributions of returns, leading to improved stability and performance.

Truncated Quantile Critics (TQC), introduced in [146], represents a further advancement in the class of actor-critic methods for continuous control. Building on the foundations laid by TD3, TQC integrates the concept of distributional RL, specifically through quantile regression as seen in QR-DQN and IQN (Section 2.3.4). TQC uses multiple quantile critics networks to provide a distributional perspective on value estimation. A key innovation in TQC is the truncation strategy, where the highest quantiles are truncated to mitigate the overestimation bias. This truncation results in a more conservative estimate, leading to more stable and efficient learning. TQC has been shown to outperform existing algorithms such as TD3 and SAC (Soft Actor-Critic, will be presented immediately afterwards). The success of TQC in various benchmarks highlights the effectiveness of the distributional approach to continuous control task.

Soft Actor-Critic

DDPG encounters the common overestimation bias, and its deterministic policy falls short when extensive exploration is required. Building on DDPG, Soft Actor-Critic (SAC) [105], aims to address both the exploration and overestimation problems.

The main innovation of SAC is the replacement of the deterministic policy by a stochastic one. Its policy generates a probability distribution over actions instead of a single deterministic action. This fundamental shift in policy design greatly enhances the algorithm's ability to explore the action space more comprehensively and systematically. To learn this policy, SAC incorporates an entropy term into its objective function. The entropy bonus encourages the policy to maintain a degree of randomness in its action selection, thus facilitating an effective exploration-exploitation trade-off.

On the other hand, SAC uses a twin critic network structure like TD3. In this setup, two separate critics networks are used, and the minimum of their value estimates is used to update the value and actor parameters. This technique plays a crucial role in mitigating overestimation bias, resulting in more accurate and reliable value estimates. Overall, for the target calculation, we use:

$$r + \gamma \min_{i \in \{1,2\}} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}' \mid s'), \quad \tilde{a}' \sim \pi_\theta(\cdot \mid s') \tag{2.105}$$

And for actor loss, the terms in the summation become:

$$\min_{i \in \{1,2\}} Q_{\phi_i}(s, \tilde{a}_\theta) - \alpha \log \pi_\theta(\tilde{a}_\theta \mid s), \quad \tilde{a}_\theta \sim \pi_\theta(\cdot \mid s) \tag{2.106}$$

Note that in the Equation (2.106), $\tilde{a}_\theta$ is differentiable with respect to θ (hence the subscript θ).

As a result, SAC delivers significantly improved performance, particularly in terms of sampling efficiency and robustness. Its ability to effectively balance exploration and exploitation, coupled with its mechanisms to reduce overestimation bias, enables SAC to demonstrate superior capabilities in complex environments characterized by continuous action spaces.

2.4.4 Conclusion on Policy-Based Deep RL

Policy-based and actor-critic methods are now the most widespread RL algorithms. However, they face challenges such as sample inefficiency, requiring large datasets for effective learning, and stability and convergence issues, particularly in complex environments. Additionally, these methods struggle with scalability in high-dimensional spaces and often falter in balancing exploration and exploitation. We further discuss some of these issues in Section 2.6.

The following section presents more specialized approaches specific to particular paradigms, some of which address the issues mentioned above. This section completes the partial overview of the current RL research landscape.

2.5 Specialized Approaches and Paradigms

2.5.1 Model-based Reinforcement Learning

Model-based RL is a significant branch of RL where the agent learns a model of the environment's dynamics and uses this model for planning and decision-making. This approach contrasts with the model-free RL methods discussed above in this section, which learn a policy or value function directly from interactions with the environment without an explicit model. Model-based RL has garnered substantial interest due to its potential for increased sample efficiency over model-free methods. Early work in this area, as highlighted by Sutton [259], involves learning a model of the environment and using it to simulate future states for planning. This approach, known as Dyna-Q [260], combines real and simulated experiences to improve learning efficiency.

Recent advances have focused on integrating deep learning with model-based approaches for better scalability and performance in complex environments. Ha and Schmidhuber [104] introduced World Models, demonstrating how learning a compact representation of the environment can facilitate policy learning in visually rich settings. A significant breakthrough in model-based RL was achieved with the development of MuZero [236]. MuZero extends the concept of Monte Carlo Tree Search (MCTS), used in AlphaGo [250], to environments where the rules are not known a priori. It learns a model of the environment's dynamics and combines it with MCTS for effective planning and decision-making, showcasing remarkable performance in board games and Atari games.

Nagabandi et al. [192] further showed that even simple neural network-based models could effectively bootstrap policy learning in robotic control tasks. Another critical area of research in model-based RL is dealing with model inaccuracies. Deisenroth and Rasmussen [61] addressed this by using Gaussian processes to model uncertainty in the environment, leading to more robust planning. Kurutach et al. [143] proposed Model-Ensemble Trust-Region Policy Optimization (ME-TRPO), which uses an ensemble of models to mitigate the impact of model errors on policy optimization.

In parallel, significant work has been done on integrating model-based and model-free methods. Janner et al. [124] showed that combining short-horizon model-based rollouts with model-free policy optimization leads to both sample efficiency and asymptotic performance. This hybrid approach has become a promising direction, as it leverages the strengths of both paradigms.

While offering significant benefits, model-based RL struggles with the complexity of accurately modeling environments where small errors can significantly impact policy performance. Another critical issue is the balance between model detail and computational practicality. The integration of model-based and model-free approaches is also challenging and requires research into optimal practices for hybrid systems.

2.5.2 Batch Reinforcement Learning

Batch RL is an approach characterized by the agent learning from a pre-collected, static dataset of experiences. This paradigm differs from traditional online RL, where agents continuously interact with the environment to acquire new data. Batch RL is particularly relevant in contexts where real-time interaction is either impractical, expensive, or risky.

In Batch RL, the key challenge is to derive an effective policy from a fixed dataset that may not fully represent the state-action space. This scenario requires strategies to maximize learning from limited data and to account for potential distributional shifts. Lange et al. [150], provide foundational insights into the capabilities and constraints of learning from fixed datasets in Batch RL.

Addressing the issue of exploration in a static data context is a critical focus of Batch RL research. Since the agent cannot explore to gather new information, it may encounter states during deployment that are not well represented in the training data. Fujimoto et al. [87] address this challenge and propose methods to mitigate the effects of distributional shifts in the absence of exploration. Similarly, Levine et al. [162] explore offline RL, a closely related concept, and discuss effective techniques for learning from fixed datasets.

Efficient use of data is another focus of batch RL. Techniques such as importance sampling (refer to Section 2.2.4), as discussed by Precup et al. [217], are used to optimize learning from available experience. Kumar et al. [140] extend this by introducing conservative Q-learning, a method designed to learn effective policies from offline data, while carefully addressing the extrapolation error common in Batch RL.

2.5.3 Imitation Learning

Imitation Learning (IL), also known as Learning from Demonstration (LfD), is an approach where the learning process is guided by examples provided by a teacher or expert. This approach is useful in scenarios where it is difficult to define a reward function or where learning from scratch is inefficient. The core concept of IL is to use expert demonstrations to guide the agent's policy development. Early work in this area includes the algorithmic framework proposed by Sammut et al. [231], where they introduced the concept of learning by observing an expert's actions. IL gained prominence with the work of Argall et al. [13], who provided a comprehensive review of different approaches to LfD, categorising them into direct behavior cloning (BC) and inverse reinforcement learning (IRL).

BC, a simple form of IL, involves directly mapping states to actions observed by an expert, as demonstrated by Pomerleau [214] in the ALVINN autonomous driving system. However, this approach can suffer from problems such as compounding errors and lack of adaptability to unseen situations. We'll discuss this in more detail in the Chapter 4.

IRL, another branch of IL, focuses on inferring the underlying reward function that the expert is implicitly optimizing. This approach was notably advanced by Ng and Russell [194], who introduced algorithms for IRL that enable agents to understand and replicate complex behaviors that are difficult to specify with a traditional reward function.

An important development in IL has been its integration with deep learning, allowing it to handle high-dimensional state spaces. This advancement is exemplified by Ho and Ermon [112] in their work on Generative Adversarial Imitation Learning (GAIL). GAIL combines the principles of IRL with the generative adversarial network (GAN) framework, and has demonstrated strong performance on complex control tasks.

The effectiveness of IL heavily relies on the quality and variety of expert demonstrations. Capturing the intricate intentions of experts in complex scenarios remains a challenge. Research is actively directed towards enhancing the robustness of policies derived from IL and extending its applicability to complex, multi-agent, and real-world settings.

2.5.4 Multi-Agent Reinforcement Learning

Multi-Agent RL (MARL) is a paradigm that focuses on environments where multiple agents interact, cooperate, compete, or coexist [167, 184, 157]. In this context, agents must learn to make decisions in environments where other agents are also learning and adapting. This adds layers of complexity due to the non-stationarity and dynamics of multi-agent interactions. Fundamental work in this area includes Littman's paper [167] on Markov games, which extended single-agent Markov decision processes to multi-agent settings.

Recent advances in MARL are often driven by deep learning, which facilitates the handling of complex, high-dimensional state and action spaces. A prominent example is DeepMind's work on AlphaStar, which achieved the grandmaster level in the complex multi-agent environment of StarCraft II [276]. This achievement demonstrated the potential of deep MARL algorithms to solve highly complex real-time strategy games.

Another important area of MARL research is the exploration of cooperative strategies between agents. Foerster et al. [82] introduced a differentiable inter-agent learning model, which allows agents to learn to cooperate in a shared environment. This approach is particularly relevant in scenarios where agents need to work together towards a common goal.

MARL is challenged by the exponentially increasing complexity of joint action spaces and the need for scalable algorithms. Issues such as managing agent cooperation and competition, ensuring equitable and stable learning experiences, and dealing with partial observability are central to ongoing MARL research. The work of Lowe et al. [173] on multi-agent actor-critic for mixed cooperative-competitive environments is a notable contribution to addressing these issues.

2.5.5 Hierarchical Reinforcement Learning

Hierarchical Reinforcement Learning (HRL) involves decomposing complex tasks into simpler, more manageable sub-tasks, inspired by how humans approach problems. The foundational paper by Sutton et al. [262] on the options framework is a seminal work in the field, introducing the concept of temporal abstraction,

allowing agents to perform temporally extended actions or subroutines, which is crucial to understanding the hierarchical decomposition of tasks.

Central to HRL is the creation of a hierarchy of decision-makers or controllers, where each level operates at a different temporal or spatial scale. The development of the Option-Critic Architecture by Bacon et al. [17] advances this concept by enabling the simultaneous learning of intra-option policies and termination conditions, addressing a key challenge in HRL. In addition, the MAXQ value function decomposition, as outlined by Dietterich [65], provides a crucial method for hierarchical decomposition of the value function. This approach plays an important role in simplifying the learning process and improving efficiency in complex environments.

The challenges lie in designing the hierarchical structure and the problem of credit assignment in hierarchical settings. The introduction of Feudal Networks [275] represents a significant advance. By separating the learning process into distinct modules that operate at different levels of abstraction, Feudal Networks facilitate more efficient learning in complex environments.

2.5.6 Meta-Reinforcement Learning

Meta-Reinforcement Learning (Meta-RL) differs from traditional RL approaches in that the agent learns how to learn. The idea is to create agents that can quickly adapt to new tasks using past experience.

A major focus of Meta-RL research is to develop algorithms that can generalize across tasks. Early contributions to this field, such as the work by Schmidhuber [233], introduced the concept of learning at multiple levels, where one level learns to optimize the learning process of another. Researchers are exploring various methods, such as context-based approaches, where the agent learns to infer the context of a task and adjust its strategy accordingly. For example, the work of Wang et al. [277] introduced a model where a meta-learner guides a base-learner, allowing the base-learner to adapt to new tasks using previous experience. Another approach is gradient-based meta-learning, exemplified by the algorithm introduced by Finn et al. [80], known as model-agnostic meta-learning (MAML). This technique optimizes the initial parameters of the model, enabling it to quickly adapt to new tasks with only a few gradient steps.

Meta-RL faces issues primarily related to sampling efficiency, as these algorithms often require extensive data for effective learning, which poses practical challenges. Another critical area is ensuring the robustness and generalizability of learned strategies across a wide range of tasks, which remains a focus of ongoing research in the field.

2.5.7 Reinforcement Learning from Human Feedback

Reinforcement Learning from Human Feedback (RLHF) involves training a reward model using human feedback, which is then used as a reward function to optimize an agent's policy through RL. RLHF is particularly valuable in scenarios where traditional reward functions prove problematic, such as those that are sparse, noisy, or deceptive. It is also highly effective in contexts where it is difficult to define a clear reward function, but where a human can easily judge the quality of the output, such as the fine-tuning of large language models (LLMs). Although a relatively young methodology, RLHF has found its main

application in the improvement of LLMs such as GPT-4 [196] and Claude [12], following the standard methodology introduced by Christiano et al. [46].

The reward model aims to quantify the quality of an output from the main model being fine-tuned by assigning it a reward value. It is usually learned using human annotators who evaluate and rank the results generated by different models. Fine-tuning of the main model is then guided by this reward signal using a policy gradient method, most commonly PPO. An essential aspect of this optimization process is the inclusion of a penalty mechanism that discourages the model from deviating too far from the original pre-tuned model. This is crucial, as it prevents the model from drifting into areas of the reward function that, according to the learned function, may yield high rewards, but which in fact correspond to side effects, out-of-domain generalizations, leading to nonsensical or linguistically irrelevant results [149].

RLHF has significantly improved the fine-tuning of LLMs, making them more aligned with human goals and more secure. However, challenges remain, as these models can still exhibit problems such as inadvertently revealing sensitive private information or producing false content (hallucination). In response to these concerns, the AI research community is actively pursuing solutions to improve the reliability and integrity of these models. Alongside these efforts, there's a growing push for alternative and complementary methods [35].

2.6 Conclusion and Foundation for Subsequent Research

This chapter has introduced the basic concepts underlying RL and the algorithms that are, at the time of writing, commonly used as baseline. Additionally, it has reviewed more specialised approaches to RL, providing a comprehensive understanding of methodologies that define current RL practice.

While the most advanced algorithms achieve impressive results in a variety of scenarios, they assume a sufficiently informative reward to converge. As a result, they have notable shortcomings in environments where rewards are sparse or misleading, hindering their ability to discover optimal policies. This difficulty is well illustrated by the example of the game Montezuma's Revenge, which, although easily mastered by humans, poses a significant challenge to the algorithms seen so far, mainly due to the exploration problem. We discuss this problem in more detail in Chapter 3.

Taking a step back, we can also see that the approaches we have discussed above are primarily focused on mastering a single, specific task. This limits their applicability in more dynamic, real-world situations where adaptability and generalization are key. This is in sharp contrast to human learning capabilities, where an individual can acquire skills across a wide range of tasks. We discuss this limitation further in Chapter 4.

Chapter 3

Cell-Free Latent Go-Explore

In this chapter, we introduce Latent Go-Explore (LGE), a simple and general approach based on the Go-Explore paradigm [70, 71] for exploration in RL. Go-Explore was initially introduced with a strong domain knowledge constraint for partitioning the state space into cells. However, in most real-world scenarios, drawing domain knowledge from raw observations is complex and tedious. If the cell partitioning is not informative enough, Go-Explore can completely fail to explore the environment. We argue that the Go-Explore approach can be generalized to any environment without domain knowledge and without cells by exploiting a learned latent representation. Thus, we show that LGE can be flexibly combined with any representation learning method. Our results indicate that LGE, although simpler than Go-Explore, is more robust and outperforms state-of-the-art algorithms in terms of pure exploration on multiple hard-exploration environments including Montezuma's Revenge. The LGE implementation is available as open-source at `https://github.com/qgalloudec/lge`.

3.1 Introduction

RL algorithms aim to learn a policy by maximizing a reward signal. In some cases, the rewards from the environment are sufficiently informative for the agent to learn a complex policy, and therefore achieve impressive results, including world level in Go [250], StarCraft II [276], or learning sophisticated robotic tasks [154]. However, many real-world environments provide extremely sparse [22], deceptive [156] rewards, or none at all. In such environments, unstructured exploration, on which many current RL approaches rely, may not be sufficient to collect data that is diverse and informative enough for the agent to learn anything. In these cases, the agent must adopt an efficient exploration strategy to reach high reward areas, which may require a significant amount of interactions.

Recent work by Ecoffet et al. [71] has introduced the *return-then-explore* algorithm family, exemplified by Go-Explore, which significantly outperforms state-of-the-art results on challenging exploration tasks such as Montezuma's Revenge by using a unique cell-based approach to grouping observations and planning trajectories. A detailed explanation of Go-Explore, including its critical reliance on the cell design for effective exploration, is provided in Section 3.2.2.

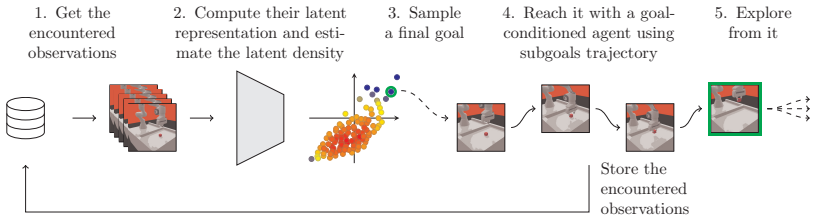


Figure 3.1: LGE exploration workflow. The encountered observations are encoded in a latent space. A latent density is estimated. A final goal is sampled from the states already reached, by skewing the distribution with the density. A goal-conditioned agent is trained to reach this goal by pursuing a sequence of subgoals, derived from the experiment that led to the final goal. Once the agent has reached the final goal, it explores from it with any exploration strategy.

In this chapter, we present Latent Go-Explore (LGE), a new algorithm derived from Go-Explore which operates without cells. This new algorithm meets the definition of a *return-then-explore* family of algorithms since the agent samples a final goal state at the frontier of the achieved states, returns to it, and then explores further from it. Our main contribution consists of three major improvements.

- A latent representation is learned simultaneously with the exploration of the agent to provide the most up-to-date and informative representation possible.

- Sampling of the final goal is based on a non-parametric density model in latent space. This leverages the learned latent representation for sampling the states of interest to be reached.

- The subgoal path pursued by the agent is reduced using a characteristic latent distance.

These three modifications, detailed in Section 3.3, allow us to generalize the Go-Explore approach to any continuous high-dimensional environment. It also enables the automation of the encoding of observations into an informative latent representation, eliminating the need for manual cell design. The full LGE exploration workflow is presented in Figure 3.1.

To evaluate LGE, we conducted experiments in the context of reward-free exploration in various hard-exploration environments including a maze, a robotic arm interacting with an object, and two Atari games known for their high exploration difficulty: Montezuma's Revenge and Pitfall.

LGE can use various types of latent representation learning methods. In this study, we demonstrate the use of three such methods, including inverse dynamics, forward dynamics, and auto-encoding mechanism. We show in Section 3.4.4 that for the environments studied, LGE outperforms all state-of-the-art algorithms studied in this chapter, and in particular Go-Explore for the exploration task.

3.2 Preliminaries and Related Work

Goal-conditioned MDP We extend the formalism introduced in Section 2.1.2 by noting that any MDP can be augmented into a goal-conditioned MDP with a goal space $\mathcal{G}$ and an initial goal distribution ρ_g. At each timestep, the observation is augmented with a goal and the reward function depends on this goal. A goal-conditioned policy [130], denoted $\pi(\cdot \mid \cdot, \cdot)$ also depends on the goal.

3.2.1 Measuring the exploration

In this work, we focus on the agent's ability to explore its environment in a pure exploration context, i.e. in the absence of extrinsic reward. This step is particularly important because, in the case of an environment with very sparse rewards, the agent can interact a large number of times with the environment without getting any reward. It is therefore necessary to follow an efficient exploration strategy to discover the few areas of the state space where the agent can get a reward. To be able to compare the results obtained by different methods in this context, it is necessary to use a common metric for the quality of exploration.

The literature uses various metrics. Some papers use the average reward on a hard-exploration task [8], the zero-shot performance on a predefined task [241] or monitor specific identifiable events in the environment that indirectly informs the degree of exploration [103]. We argue that these indirect measures are unsatisfactory as they rely on the subsequent learning ability of an online and offline agent respectively. For simplicity, we use the number of visited cells as the metric, whose construction strategy is explained in Section 3.4.2. Therefore, the figures represent the number of cells explored, although Go-Explore is the only algorithm to explicitly maximize this metric. Following the guidelines of Agarwal et al. [3], we use for all plots in this chapter the Interquartile Mean (IQM) with the 95% Confidence Interval (CI).

3.2.2 Go-Explore

Before giving a broader overview of exploration strategies in RL, we give a detailed explanation of Go-Explore. Ecoffet et al. [71] introduced a new paradigm in which a goal-conditioned agent is trained to reach states it has already encountered, and then explore from there. The agent thus iteratively pushes back the frontier of its knowledge of the environment. We call this family of algorithms *return-then-explore*. Ecoffet et al. [71] provide Go-Explore, an algorithm of this new family, that outperforms by several orders of magnitude the state-of-the-art scores on the game Montezuma's Revenge, known as a hard-exploration problem.

Go-Explore relies on a grouping of observations into *cells*. Figure 3.3 shows the cell generation process for the Atari environment. These cells are used both to select target observations at the frontier of yet undiscovered states and to build a subgoal trajectory for the agent to follow to reach the final goal cell. As Ecoffet et al. [71] initially spotted, the cell design is not obvious. It requires detailed knowledge of the observation space, the dynamics of the environment, and the subsequent task. If any important information about the dynamics of the environment is missing from the cell representation, the agent may fail to explore at all. For example, in Montezuma's Revenge, possession of a key is

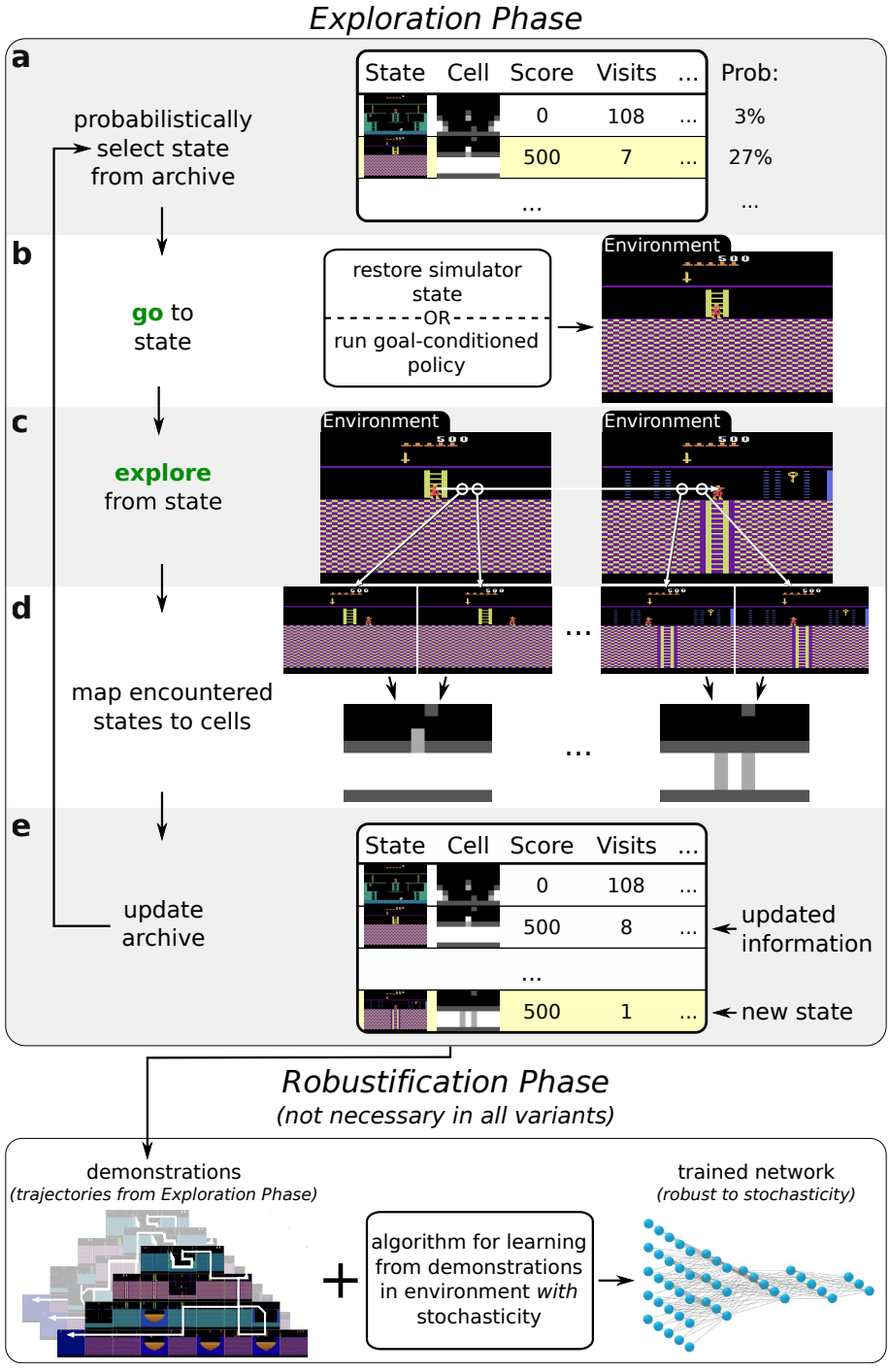


Figure 3.2: Go-Explore selects states from an archive, prioritising promising cells, returns to these states using methods such as simulator restoration, and explores using random actions or policy sampling. Found states are mapped to a cell representation, and new or updated states are added to the archive. Figure from [71].

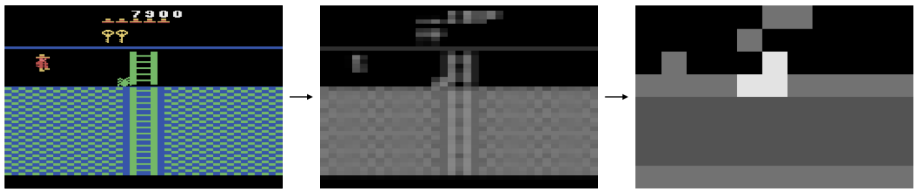


Figure 3.3: An example of cell representation in Go-Explore. The image is first converted to grayscale, then sub-sampled to an 11×8 image containing 8 distinct levels of pixel intensity. The resulting image is the cell representation. Figure from [70].

a crucial piece of information that, when included in the cell representation, increases exploration by several orders of magnitude. We also demonstrate in Section 3.4.6 that the cell design has a major influence on the results.

3.2.3 RL exploration

Exploration in RL usually divided into three types [148, 14]: unstructured exploration[1], intrinsic rewards-based methods, and goal-based methods.

Unstructured exploration

In unstructured exploration, the agent does not adhere to a predetermined exploration plan and instead takes actions randomly or according to a simple heuristic. These actions may be sampled uniformly from the action space, or in the continuous case, they may be augmented by exploration noise that is parametrized by the current state [105] or not [164]. Unstructured exploration can be effective in some environments, but it may not be sufficient to explore more complex or sparsely rewarded environments.

Intrinsic reward-based methods

Intrinsic rewards-based methods are inspired by the concept of intrinsic motivation in cognitive science [201, 200]. Instead of solely relying on extrinsic rewards provided by the environment, this approach introduces *intrinsic* rewards that encourage the agent to explore states with potential learning value. These intrinsic rewards can be derived from various sources, each offering unique advantages.

Count-based methods Bellemare et al. [22] introduced a pioneering pseudo-count method that addressed the challenge of countable yet extensive state spaces by replacing traditional visitation counts with state density estimates and computing intrinsic rewards from them. While their primary focus was on discrete state spaces, they laid the foundation for future research in continuous state spaces. Building on this work, Ostrovski et al. [199] extended the pseudo-count concept by utilizing a PixelCNN model to estimate pseudo-counts when dealing with image observations, leading to significant improvements in exploration

[1]We replace the terminology of Ładosz et al. [148] *random exploration* by *unstructured exploration* that we think is more accurate.

efficiency. Concurrently, Tang et al. [263] extended count-based exploration methods to high-dimensional and continuous state spaces by introducing Locally Sensitive Hashing (LSH). Machado et al. [178] introduced an extension of count-based methods that leverages a learned successor representation to more effectively bridge the gap between tabular and continuous settings.

Knowledge-based methods Early work such as [234] implemented a reward system based on the Euclidean distance between the output of the world model and the actual output of the environment. This approach rewarded the curiosity unit when the model had insufficient knowledge about a particular state, thus encouraging agents to explore unfamiliar areas. In a theoretical exploration framework, Schmidhuber [235] explored the optimal implementation of basic computational principles for intrinsically motivated agents. While theoretically optimal, the methods described in this work are notable for their lack of algorithmic feasibility. Variational Information Maximizing Exploration (VIME) by Houthooft et al. [114] focuses on maximizing information gain about the agent's beliefs about environmental dynamics, which involves rewarding actions that effectively reduce uncertainty about these dynamics. In a different vein, Achiam and Sastry [2] introduced the concept of *surprisal* defined as the error of a learned world forward dynamics model and used as an intrinsic reward. Concurrently, Pathak et al. [205] introduce the Intrinsic Curiosity Module (ICM), which formulates curiosity as an error in predicting the consequences of the agent's actions in a visual feature space designed to accommodate high-dimensional continuous state spaces. Importantly, ICM justifies the use of inverse dynamics over forward dynamics by recognizing that some elements of dynamics may be unpredictable and should not be the focus of the agent's exploration. In addition, the work of Pathak et al. [206] explores the concept of disagreement between ensemble models as a source of intrinsic reward, highlighting its resilience in the face of stochastic environments. Burda et al. [34] offer a different perspective by introducing Random Network Distillation (RND), a straightforward exploration bonus method that effectively addresses the scaling challenges associated with information or prediction gain.

Impact-based methods These methods allow agents to maximise their interactions with the environment by rewarding certain actions based on the impact and influence these actions have on the state of the environment. Mutual Information-based State Intrinsic Control (MUSIC) [295] introduces an intrinsically motivated agent that maximizes mutual information between the agent's state (e.g., position and velocity) and the environment's state (excluding the agent's state) to encourage exploration. In contrast, Rewarding Impact-Driven Exploration (RIDE) [223] addresses exploration in sparse-reward procedurally generated environments (PGE). RIDE mitigates the risks of detachment by learning a state representation via forward and inverse dynamics models that capture elements influenced by the agent.

Empowerment-based methods These methods aim to maximise an agent's ability to control its environment. Klyubin et al. [135] introduce the concept of empowerment as an intrinsic reward that encourages the agent to seek control over its environment. Mohamed and Rezende [190] and Rezende and Mohamed

[225] further explore the idea of intrinsic rewards based on empowerment, where empowerment is defined as the ability between states and actions. Agents seeking to maximise this empowerment value are motivated to approach states from which they can influence the largest number of future states within a given planning horizon, thereby motivating them to target states of maximum influence. Gregor et al. [99] also incorporate intrinsic rewards based on empowerment, which reinforce the agent's desire to control its environment by maximising the mutual information between pairs of start and end states.

Goal-based methods

Methods that modify the reward of the environment provide no mechanism for distilling the knowledge gained from visiting various states. Agents may visit new states, but they quickly forget about them when other states become newer. To address this issue, recent work has suggested the use of a goal-conditioned autotelic agent specifically trained for the exploration task. This approach allows for the use of the knowledge gained during exploration to realize new user-specified goals [161, 51]. During the exploration phase, the reward signal is ignored, and after the exploration phase, the data collected by the agent is used to learn one or more subsequent tasks [127]. Goal-based methods condition the agent with a goal that is used to guide exploration towards unknown areas. These methods rely on a goal generator to create goals for the agent. We divide goal-based methods into two categories: *exploratory goal* methods and *goals to explore from* methods (called *post-exploration* in [290]).

Exploratory goal methods follow the intuition that the agent discovers new areas of the observation space by pursuing goals that have been little or not achieved before. The challenge of these methods is to choose the goal to be neither too easy nor too hard. The literature contains several ways to approach this trade-off. While [36] leverage a model of goal reaching capabilities to select goals of intermediate difficulty, some other methods sample goals that either maximize learning progress [49, 216] or value disagreement [294]. Alternatively, [215] sample goals from the least visited areas using a parametric density model on the visited states. It is also possible to imagine goals that have never been reached using a language model [50], a generative model [219] or a GAN [81].

In *goals to explore from* methods the agent samples a goal from previously visited states. It returns to it, either by teleportation [70, 183], or using a goal-conditioned policy [71]. The challenge of these methods is to choose a goal that is of high exploratory interest. Similarly, some methods estimate the density of the encountered states, using either parametric methods [210] or non-parametric methods [71, 183], to target the low-density areas.

In summary, methods based on a goal reaching policy should facilitate scalable RL. The stunning results of Go-Explore illustrate this point but remain circumscribed to few environments and require a lot of domain knowledge to work. By bridging with concepts already used in the intrinsic reward literature, we show a way to make this approach more general and simpler.

3.3 Latent Go-Explore

LGE meets the definition of the *return-then-explore* family of algorithms. First, a final goal state is sampled from the replay buffer, then the agent learns a goal-conditioned policy to reach this goal. When the agent reaches the goal, the agent starts to explore. LGE learns a latent representation of observations and samples the goal pursued by the goal-conditioned agent in priority in low latent density areas. In Section 3.3.1, we present how the latent representation of observations is learned. In Section 3.3.2, we show how the latent density is estimated and how the final goal state pursued by the agent is sampled. Then, in Section 3.3.3, we show how to build a subgoal trajectory from the final goal to increase the agent's performance, in particular in far-away goal situations. The pseudo-code of the resulting algorithm is presented in Algorithm 18. Finally, in Section 3.3.4, we take a closer look at Go-Explore and LGE, highlighting their main differences.

Algorithm 18: Latent Go-Explore (LGE).

Input: Number of iterations in the exploration phase T
Output: The goal-conditioned policy π and the dataset D
Initialize: Replay buffer $D = \emptyset$; Encoding module; Goal-conditioned policy π
while $t < T$ **do**
  Sample a final goal state with Equation (3.8)
  Build the subgoal trajectory τ^g using Equation (3.9)
  Initialize the subgoal index: $i \leftarrow 0$
  while *the last goal of* τ^g *is not reached* **do**
    Collect interaction using $\pi(\cdot \mid \cdot, \tau_i^g)$ and store it into dataset D
    if *subgoal* τ_i^g *is reached, i.e.,* $||\phi(s_t) - \phi(\tau_i^g)|| < d$ **then**
      Move to the next subgoal: $i \leftarrow i + 1$
  Explore until the end of the episode with any exploration strategy
  Update π with any off-policy algorithm and HER
  Every `update_encoder_freq` timesteps, update encoder ϕ with any representation learning algorithm

3.3.1 Learning a latent representation

The literature contains several latent representation learning methods for RL. Learning such a representation is orthogonal to our approach. Hence, LGE can be combined with any learning method without the need for further modifications. Choosing the best representation learning method given the environment is out of the scope of this work. In this work, we present three methods of representation learning that have been found to work well with our test environments. Two of these methods are inspired by the literature on intrinsic reward-based methods,

Inverse dynamic representation learning Pathak et al. [205] proposed an intrinsic reward calculated based on the agent's prediction error of the consequence of its own actions. The representation is learned using two submodules, as presented in Figure 3.4. The first encodes the observation into a latent

representation $\phi(s_t)$. The second takes as input $\phi(s_t)$ and $\phi(s_{t+1})$ and outputs the prediction of the action taken by the agent at timestep t. The parameters θ of the inverse model $\mathcal{P}_\theta^{\text{inv}}$ are optimized by minimizing the loss function:

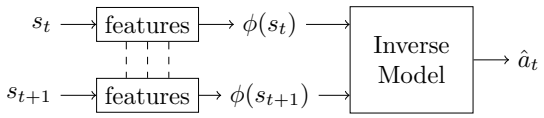


Figure 3.4: Inverse dynamic to learn representation.

$$L = \frac{1}{|N|} \sum_{(s_t, a_t, s_{t+1}) \sim D} \frac{1}{2} \| a_t - \mathcal{P}_\theta^{\text{inv}}(s_t, s_{t+1}) \|_2^2 \tag{3.1}$$

The inverse dynamics representation learning allows getting a latent representation of the states containing only the aspects of the state on which the agent can have an influence.

Forward dynamic representation learning In [2], the intrinsic reward is calculated based on the prediction error of a model approximating the transition probability function of the MDP. Two submodules are used, as presented in Figure 3.5.

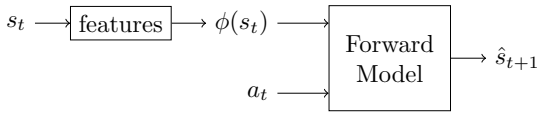


Figure 3.5: Forward dynamic to learn representation.

The first one encodes the observation to a latent representation $\phi(s_t)$. The second takes as input $\phi(s_t)$ and a_t and outputs the prediction of the next state $\hat{s}_{t+1}$. The model parameters θ are optimized by minimizing the loss function:

$$L = -\frac{1}{|N|} \sum_{(s_t, a_t, s_{t+1}) \sim D} \log \mathcal{P}_\theta(s_{t+1} \mid s_t, a_t) \tag{3.2}$$

Vector Quantized Variational Autoencoder (VQ-VAE) Autoencoding [111] aims to train a neural network to reconstruct its input by learning a compressed representation of the data. This approach is known to be effective in extracting useful features from the input, especially images. For Atari environments, we use a VQ-VAE [272], a technique that combines autoencoding with vector quantization, and has shown good results, while being simple to train. We use the coordinates of the embeddings in the embeddings table as the latent representation.

3.3.2 Density estimation for intrinsic goal sampling

The success of the proposed method relies on the agent's ability to generate for itself goals that it will be able to reach and then explore from there. For the

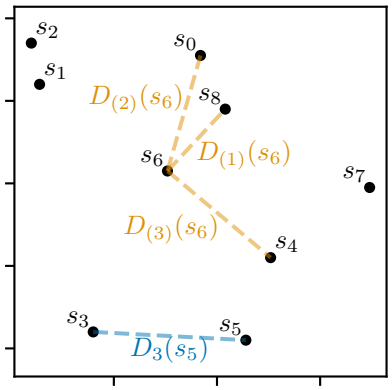


Figure 3.6: Illustration of the notation used in the simple case of $d = 2$ and $n = 9$. $D_{(1)}(s_6)$ denotes the distance between s_6 and its closest neighbor (except itself), i.e. s_8.

agent to progress in the exploration of the environment, these goals must be at the edge of the yet unexplored areas. To identify these areas, we use an estimator of the density of latent representations of the encountered states. Moreover, we require the goal to be reachable. The set of reachable states is a subset of the state space that we assume to be unknown. The easiest way to satisfy the previous requirement is therefore to sample among the states that have already been reached.

We estimate the density of latent representations of the encountered states (called latent density) using the particle-based entropy estimator originally proposed by Kung et al. [142] and used in the literature on intrinsically motivated RL [169, 168]. This estimator has the advantage of being non-parametric and thus does not hinge on the learning capabilities of a learned model.

Let $s_1, \ldots, s_n$ be a sample of event locations[2] in an $\mathbb{R}^d$-space, where d is an arbitrary integer. Assume that the event location s follows a common distribution with density function $f(s)$. For any sample s_i and s_j in this sample, assume that $D_i(s_j) = ||s_i - s_j||$, denotes the euclidean distance between s_i and s_j. For any $k \leqslant n$, let $D_{(k)}(s_i)$ be the distance with k-th nearest neighbors of s_i with respect to the euclidean distance. Figure 3.6 shows an basic example with $d = 2$ and $n = 9$.

Kung et al. [142] propose an optimal unbiased estimator $\hat{f}$ for the density:

$$\hat{f} = \frac{kU_k^*}{k-1} \tag{3.3}$$

where

$$U_k^* = \frac{(k-1)}{nC_d D_{(k)}^d} \tag{3.4}$$

[2]Here, s does not necessarily denote a state, but we deliberately choose this notation, as we believe it improves understanding, insofar as the estimator is easy to visualize when s indeed denotes a state.

and

$$C_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)} \tag{3.5}$$

Hence we have

$$\hat{f} = \frac{k}{nC_d} D_{(k)}^{-d} \tag{3.6}$$

We follow the recommendation of Kung et al. [142] to take

$$k = 2n^{1/d} \tag{3.7}$$

The sampling of the final goal state follows a geometric law on the rank in the latent density sort R_i. The probability to draw s_i as the final goal state is

$$\mathbb{P}(G = s_i) = (1-p)^{R_i - 1} p \tag{3.8}$$

where G is the random variable corresponding to the final goal state, and $0 \leqslant p \leqslant 1$ is a hyperparameter controlling the bias towards states with a low latent density.

This method has the advantage of being robust to approximation errors in the density evaluation, which can be particularly important in low density areas. In doing so, we only focus on the ability of the model to correctly order the observations according to their latent density.

The representation is jointly learned with the exploration of the agent. Therefore, the latent density must be regularly recomputed to take into account the most recent representation on the one hand, and the recently visited states on the other hand. However, considering the slow evolution of this value, we choose to recompute the latent density only once every 5k timesteps for maze and robotic environments, and every 500k timesteps for Atari environments. This allows us to significantly reduce the computation needs while having a low empirical impact on the results.

3.3.3 Subgoal trajectory

As learning progresses, the sampled final goal states are increasingly distant. However, reaching a distant goal is challenging because it implies a sparse reward problem.

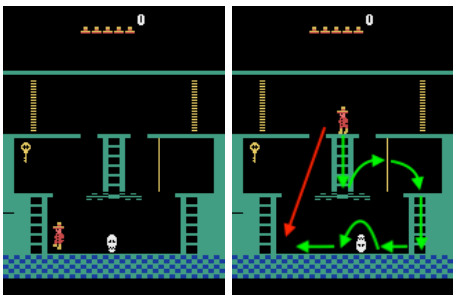
To overcome this problem, we condition the agent to successive intermediate goals $\tau^g = (g_0, g_1 ..., g_L)$ that should guide it to the final goal state g_L. These intermediate goals are chosen from the trajectory that led the agent to the final goal state $(s_0, s_1, \ldots, s_T)$.

The trajectory that led the agent to the final goal state is unlikely to be optimal. Plus, if the agent is conditioned by the whole trajectory, it may fail to reach all of them, even though some of them may not be necessary to reach the final goal state. To allow the agent to find a better path to the final goal state, we remove some subgoals from this trajectory. To decide whether a subgoal should be removed from the trajectory, we evaluate the latent distance to the previous subgoal. If the distance is less than the threshold, then the goal is removed.

$$\forall i \leqslant L-1, \quad ||\phi(g_i) - \phi(g_{i+1})|| > d \tag{3.9}$$

Unlike Go-Explore, LGE doesn't use the best known trajectory that leads to the sampled goal area (cell). The main reason is that the best known trajectory may be particularly difficult to reproduce, due to the dynamics and the stochasticity of the environment, or cause the early termination of the episode.

For example, in Montezuma's Revenge, there are two ways to reach the bottom of the left ladder (a necessary step to get the first key). An illustration of the situation is shown in Figure 3.7. The first one consists in jumping right from the promontory (red arrow), but causes the death of Panama Joe (the character) and as a result ends of the episode. The second one, longer, consists in going around by the right ladder (green arrows). Therefore, if the agent always chooses the shortest path (like Go-Explore), it will most likely fail to reach the first key and to further explore the environment.



(a) Goal state.

(b) Initial state with the two possible paths to the final state.

Figure 3.7: An example of how naively choosing the shortest path to a target state can cause Go-Explore to fail. If the agent chooses the red path, this will cause the death of the character and thus the end of the episode. To reach the goal state, the agent must take the green path, necessarily longer.

Once the goal is reached, the agent explores using any exploration strategy. For the sake of simplicity, we choose a random exploration strategy for our experiments. We also impose that the agent repeats the previous action with a probability of 90%. This technique has been shown to increase the results significantly [71].

3.3.4 Comparing Go-Explore and LGE

The Go-Explore algorithm as presented by Ecoffet et al. [71] has many components. All these components allow to obtain good results on test environments. In this article, we implement our own version of Go-Explore. We have tried to stick as much as possible to the initial implementation and to improve some aspects. We keep the essence of Go-Explore, but our implementation is not intended to be equivalent to the initial implementation. The main goal here is to compare LGE and Go-Explore. Thus, the two implementations differ only in the elements that make them unique. To the best of our knowledge, all the components that we did not implement are compatible with LGE. It is likely that

they improve LGE and Go-Explore in a similar way. In this section, we describe the implementation of LGE and Go-Explore. We explain their differences if any.

Policy-based Go-Explore The initial implementation of Go-Explore distinguishes between the case where the environment can be reset to any desired state and the case where this is not possible. In this work, we choose the general setting where the environment can't be reset to any desired state, and we therefore work with the so-called policy-based implementation of Go-Explore.

Exploration after returning In the original implementation of Go-Explore, once a cell is returned, exploration proceeds with random actions for a certain number of timesteps. For both LGE and Go-Explore, we set this number of timesteps to 50 for all environments. Note that the agent can interrupt this exploration beforehand if the maximum number of interactions with the environment is reached. Ecoffet et al. [71] shows that action consistency generally allows for more effective exploration, especially in the robotic environment. For LGE and Go-Explore, we use the same trick: the agent chooses the previous action with a probability of 90%, and uniformly samples an action with a probability of 10%.

Cell design The original implementation of Go-Explore provides two methods for generating the cell representation.

1. When the observation is an image, the observation is grayscaled and downscaled. The image produced is the cell representation. The parameters to get this representation (downscaling width and height and number of shades of gray) are optimized during training to maximize an objective function that depends on a target split factor.

2. When the observation is a vector, each component of the vector is discretized separately by hand before learning.

For all the environments presented in this work, we use a naive method of cell generation corresponding to a discretization of the observation. The granularity of the discretization is a hyperparameter. The choice of this hyperparameter is crucial, we develop it in more detail in the Section 3.4.6.

Goal-conditioning In the original implementation of Go-Explore, the agent is conditioned by the cell representation of the goal. We note that this representation can vary during learning, and even in size (see previous paragraph). It is not clear how to structure the agent's network when the size of the input varies during the learning process.

In our implementation, we choose to condition the agent by the goal observation rather than by the representation of its cell. We also condition the agent by the goal observation in LGE.

RL agent In the original implementation of Go-Explore, the goal-conditioned agent is based on the on-policy PPO algorithm [239]. For both LGE and Go-Explore, we rather chose to use an off-policy algorithm (SAC or DDPG, see Section 2.4.3) to use a Hindsight Experience Replay (HER) [9] relabelling, which has shown to perform better in a sparse reward environment.

Reward In the original implementation of Go-Explore, the agent gets +1 reward for reaching intermediate cells and +5 reward for reaching the final cell of a path. The rest of the time, it receives a 0 reward. For both LGE and Go-Explore, following the suggestions of Tang and Kucukelbir [264], we rather choose the following structure for the reward: the agent gets a reward of 0 for a success (target cell reached for Go-Explore and latent distance with the goal state below the distance threshold for LGE) and −1 the rest of the time. As noted by Eysenbach et al. [76], in this setting, an optimal agent tries to terminate the episode as quickly as possible. We therefore set `done = True` only at the end of the post-exploration, and not when the agent reaches a final state.

3.4 Experiments

To demonstrate the effectiveness of our method, we apply it to a range of pure exploration tasks. We focus on environments for which naive random exploration is not sufficient to explore the rich variety of reachable states. We compare the results obtained with LGE with the results obtained using several algorithms based on intrinsic curiosity and others based on goal-directed strategies. For each environment, LGE uses the representation method that empirically gives the best results. Consequently, we use the forward dynamics for the maze environment, the inverse dynamics for the robotic environment, and the VQ-VAE for Atari.

In terms of infrastructure, each run was performed on a single worker machine equipped with one CPU and one NVIDIA® V100 GPU + 120 Gb of RAM.

3.4.1 Environments

Continuous maze We train an agent to navigate in a continuous 2D maze. The corresponding configuration is shown in Figure 3.8. The agent starts every episode in the center of the maze. At each timestep, the agent receives the current coordinates as an observation and chooses an action that controls its location change. If the agent collides with a wall, it returns to its previous position. The reachable space is a square of 12×12 and the agent's action is limited to $[-1, 1]$ horizontally and vertically. The agent can interact 100 times with the environment (which is just enough to explore all the maze), after which the episode ends.

Robotic environment Robotic environments are interesting and challenging application cases of RL, especially since the reward is often sparse. We simulate a Franka robot under the PyBullet physics engine using PandaGym [91]. The robot can move and interact with an object. The agent has access to the position of the end-effector and the position of the object, as well as to the opening of the gripper. The agent interacts 50 times with the environment and then the object and the robot arm are reset to their initial position.

Atari We train LGE on two high-dimensional Atari 2600 environments simulated through the Arcade Learning Environment (ALE) [21] that are known to be particularly challenging for exploration: Montezuma's Revenge and Pitfall. Details of the settings used are presented in Section 3.4.3.

3.4.2 Baselines

Random exploration

Most RL methods from the literature do not follow any structured exploration strategy. In a reward-free context, the performance of the latter is often equivalent to a random walk. We take as a reference a random agent, whose actions are uniformly sampled over the action space at each timestep, SAC [105] and DDPG [164] for continuous action space environments.

Intrinsic reward-based exploration

In this work, we take as reference two widely used intrinsic reward-based methods combined with either SAC or DDPG. These methods stand out from the others because, despite their simplicity, they have demonstrated good performance on a wide variety of tasks.

Intrinsic Curiosity Module (ICM) [205] The intrinsic reward is computed as the mean square error between the true latent representation and the one predicted by a learned dynamic model given the action taken. The encoder is trained jointly with an inverse dynamics model.

Surprise [2] The intrinsic reward is the approximation of the KL divergence between the actual transition probabilities and a learned transition model.

Goal-directed exploration

Go-Explore [70, 71] The agent divides the observation space into cells, prioritizes the cells that have been visited the least, returns to them using a goal-conditioned policy, and then continues exploring from that point. This is the policy-based and without domain knowledge variant of Go-Explore, but we refer to it simply as *Go-Explore*.

The observations in continuous environments are converted into cell representations by discretizing them. In the maze environment, we use a 24×24 grid, and in the robotic environment, we use a grid with a 0.1m resolution for the position of the gripper and object. For Atari, we use the same fixed cell representation as proposed by [71] in the policy-based case: the observation is converted to grayscale and reduced to the size of 8×11 pixels. The depth is then reduced from 256 to 8 values according to $\lfloor \frac{8p}{255} \rfloor$ where p is the pixel value. The resulting image is the representation of the cell. Go-Explore is the closest baseline to our algorithm. Section 3.3.4 details the differences between LGE and Go-Explore.

Diversity Is All You Need (DIAYN) [77] The agent is conditioned by a *skill* and a discriminator predicts the skill pursued by the agent. The more the discriminator predicts with certainty the skill pursued, the bigger the reward. Conjointly, the discriminator is trained to maximize the distinguishability of skills.

Skew-Fit [215] The agent's goal sampling is skewed to maximize the entropy of a density model learned on the achieved states.

For the goal-directed methods, we use HER, [9] relabeling which has shown to significantly increase learning.

To nullify the variation in results due to different implementations, we implement all algorithms in the same framework: Stable-Baselines3 [222]. The set of intrinsic reward-based methods and goal-directed methods are underpinned by the same off-policy algorithm. The hyperparameters for this algorithm are identical. For the maze environment, we use SAC, while for the robotic environment, we use DDPG as it gives better results for all methods. For Atari environments, we use QR-DQN [57], as it commonly considered to be a strong baseline on it. For Atari, we only compare LGE to Go-Explore as it far outperforms the others. To negate the influence of a bad choice of hyperparameter on the results, the method-specific hyperparameters are optimized. Section 3.4.3 details the optimization process and the resulting hyperparameters.

3.4.3 Hyperparameters and environments settings

To limit the impact of the large variability of results depending on the hyperparameters, we chose to optimize the hyparameters for each experiment. For maze and robotic environments, we selected 100 unique sets of hyperparameters from a search space presented in Table 3.1 using Optuna [5]. For each hyperparameter set, we train the model with 3 different seeds and keep the median score. For Atari, we selected 10 unique sets of hyperparameters and train the agent just once. The method-specific parameters that have not been optimized are presented in Table 3.2. The hyperparameters used for the off-policy agent are identical for all algorithms. They are presented in Table 3.3. For Atari, we mainly use the setting recommended by Machado et al. [177]. Like Ecoffet et al. [71], we use both sticky actions and start no-ops.

Table 3.1: Search space and resulting hyperparameters after optimization.

Method	Hyperparameter	Search space	Maze	Robotic	Atari
LGE	Latent distance threshold	$[0.1, 0.2, 0.5, 1.0, 2.0]$	1.0	1.0	2.0
	Latent dimension	$[4, 8, 16, 32, 64]$	16	8	$8 \times 8 \times 8$ [c]
	Geometric parameter	$[0.001, 0.002, 0.005, 0.01, 0.02, 0.05]$	0.05	0.01	0.001
Go-Explore	Cell size	$[0.2, 0.5, 1.0, 2.0, 5.0]$	2.0	0.2	$11 \times 8 \times 8$ [b]
ICM	Scaling factor	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{-1}	10^{-2}	N/A
	Actor loss coefficient	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{2}	10^{-3}	N/A
	Inverse loss coefficient[a]	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{1}	10^{-3}	N/A
	Forward loss coefficient[a]	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{1}	10^{2}	N/A
Surprise	Feature dimension	$[2, 4, 8, 16, 32]$	2	16	N/A
	Desired average bonus	$[10^{-2}, 10^{-1}, 10^{0}, 10^{1}]$	10^{-2}	10^{-2}	N/A
	Model train frequency	$[2, 4, 8, 16, 32, 64, 128]$	64	8	N/A
	Model learning rate	$[10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$	10^{-5}	10^{-5}	N/A
DIAYN	Number of skills	$[4, 8, 16, 32, 64, 128]$	32	32	N/A
Skew-Fit	Number of models	$[5, 10, 20, 50, 100, 200]$	50	50	N/A
	Density power	$[-5.0, -2.0, -1.0, -0.5, -0.2, -0.1]$	-1.0	-0.2	N/A
	Number of pre-sampled goals	$[64, 128, 256, 512, 1024, 2048]$	64	128	N/A
	Success distance threshold	$[0.05, 0.1, 0.2, 0.5, 1.0]$	0.5	0.2	N/A

[a] In the original paper, the sum of the forward and inverse loss coefficients is 1. We get better results without this constraint.
[b] Width × Height × Number of grayscale values. Not optimized, taken from the original paper.
[c] Width × Height × Number of embedding vectors.

Table 3.2: Hyperparameters specific to each method. Their value is identical for all experiments and have not been optimized.

Method	Hyperparameter	Value
LGE	Encoder training frequency	5k steps (Maze and Robotic) 500k steps (Atari)
	Learning rate	0.001
	Batch size	32
	Gradient steps	500 (Maze and Robotic) 5k (Atari)
	Exploration strategy	Random
	Repeat action probability	0.9
Go-Explore	Exploration strategy	Random
	Repeat action probability	0.9
ICM	Feature size	16
	Networks	$[64, 64]$
	Activation function	ReLU
Surprise	Networks	$[64, 64]$
	Activation function	ReLU
DIAYN	Discriminator networks	$[256, 256]$
	Activation function	ReLU
Skew-Fit	Gradient steps	100
	Batch size	2048
	Learning rate	0.01

Table 3.3: Hyperparameters of the off-policy agent. These hyperparameters are identical for all methods and for all experiments. The hyperparameters related to HER relabeling only apply to the methods for which the agent is goal-conditioned (DIAYN, Go-Explore, Skew-Fit and LGE).

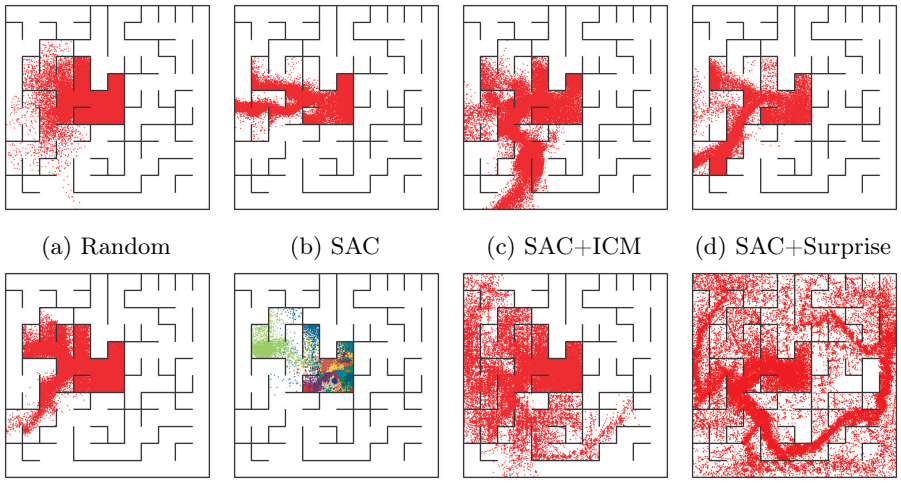
Hyperparameter	SAC	DDPG	QR-DQN
Networks	$[300, 400]$	$[300, 400]$	CNN from [188]
Learning rate	3×10^{-4}	10^{-3}	5×10^{-5}
Learning starts after N timesteps	100	100	1M
Batch size	256	100	32
Discount factor (γ)	0.99	0.99	0.99
Polyak update coefficient (τ)	0.005	0.005	1.0
Target entropy	2.0	N/A	N/A
Target update every N timesteps	N/A	N/A	10k
ϵ decreases during N timesteps	N/A	N/A	4M
Initial ϵ	N/A	N/A	1.0
Final ϵ	N/A	N/A	0.05
Train every N timesteps	1	1	10
Gradient steps	1	1	1
HER sampling probability	0.8	0.8	0.8
HER relabeling strategy	Future	Future	Future

Table 3.4: Atari setting.

Parameter	Value
Reset on life loss	Yes
Start no-ops	From 1 to 30
Action repetitions	4
Sticky action probability σ	0.25
Observation preprocessing	84×84, grayscale
Action set	Full (18 actions)
Max episode length	100k
Max-pool over last N action repeat frames	2

3.4.4 Main results

The exploration results for the maze environment are presented in Figure 3.9. A rendering of the positions explored by the agent is presented in Figure 3.8.



(a) Random (b) SAC (c) SAC+ICM (d) SAC+Surprise

(e) Skew-Fit (f) DIAYN (g) Go-Explore (h) LGE (ours)

Figure 3.8: Space coverage of the maze environment after 100k timesteps. In f, the different colors are the different skills.

We note that only LGE and Go-Explore significantly outperform the results obtained with random exploration. This demonstrates the effectiveness of the *return-then-explore* paradigm in this environment. We note that exploration based on intrinsic curiosity does not yield significantly better results than those obtained by random exploration. We hypothesize that the simple dynamics of the environment makes the intrinsic reward to quickly converge to 0.0. Surprisingly, neither Skew-Fit nor DIAYN performs significantly better than random exploration. For DIAYN, we find that most of the skills were concentrated in the initial position area of the agent. We hypothesize that this is the consequence of the lack of *post-exploration* described by Yang et al. [290]. Finally, we note

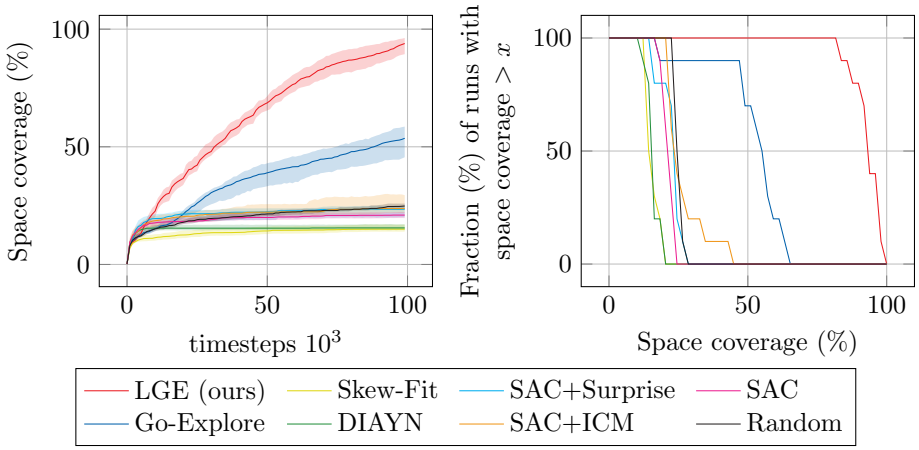


Figure 3.9: Comparison of the space coverage of the maze environment. Each experiment is run 10 times. The left plot represents the space coverage (number of cell divided by the total number of reachable cells) across timesteps. The solid lines are the IQMs and the shaded areas are the 95% CIs. The right plot is the final performance profile (higher is better).

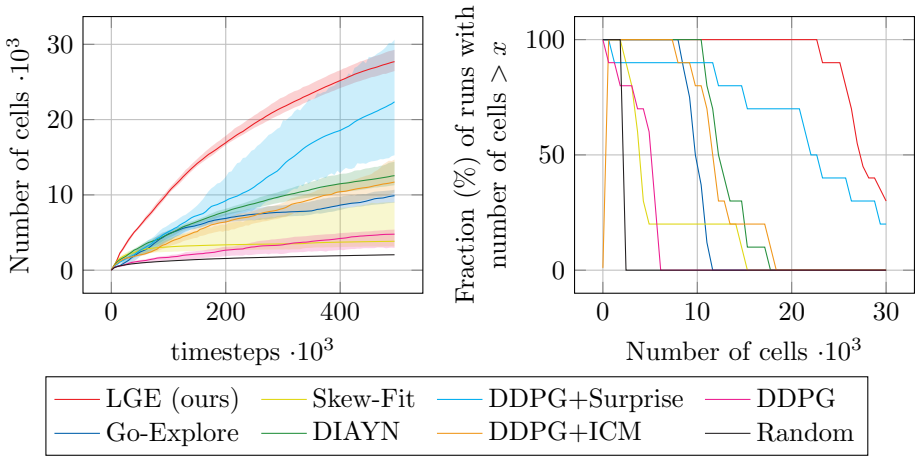


Figure 3.10: Comparison of exploration with the robotic environment. Each experiment is run 10 times. The left plot represents the number of explored bins across timesteps. The solid lines are the IQMs and the shaded areas are the 95% CIs. The right plot is the final performance profile (higher is better).

that, although the cell size has been optimized, LGE significantly outperforms Go-Explore. LGE manages to cover almost the entire reachable space at the end of the runs while exhibiting low variability in the results.

The exploration results for the robotic environment are presented in Figure 3.10. We notice that LGE significantly outperforms all other methods. Notably, Go-Explore performs only slightly better than random exploration. We note that Go-Explore does not learn to grasp the object throughout the learning process. The results presented on a robotic environment by Ecoffet et al. [71] are much better. We presume this is mainly due to the meticulous work done on the state space examination and the induced cell design. Here, we use a naive grid-like cell design. Although the grid parameter is optimized, it does not yield good exploration results with this environment. We thus demonstrate the benefit of using a learned representation to automatically capture important features of the environment's dynamics.

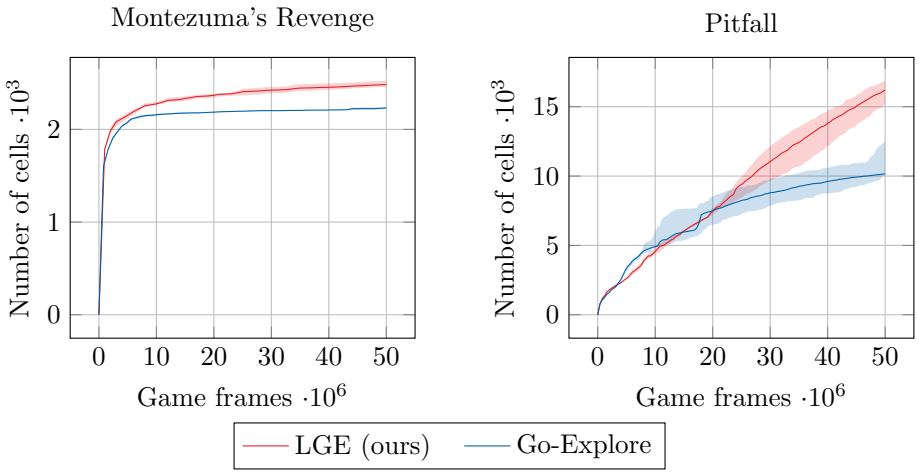


Figure 3.11: Comparison of exploration on the Atari environments. Each experiment is run 3 times. The solid lines are the IQMs and the shaded areas are the 95% CIs.

The exploration results for Atari are presented Figure 3.11. We see that both LGE and Go-Explore quickly discover a large number of cells, then continue their exploration by regularly discovering new cells. LGE slightly outperforms Go-Explore on both Pitfall and Montezuma's Revenge. Nevertheless, we note that the number of discovered cells is much lower than that of Go-Explore in its full configuration (around 5k for Montezuma's Revenge), including domain knowledge and the ability to reset the environment in any state. This shows the criticality of these settings for exploring these particular environments.

3.4.5 Ablation study

We study the impact of the ablation of three key elements of LGE. (1) LGE without further exploration (as exploratory goal methods, see Section 3.2): the environment is reset once the agent reaches the final goal instead of performing the exploration random interactions. (2) LGE without skewing the final goal

distribution in favor of low latent density areas, the final goals are sampled uniformly among the reached goals. (3) LGE without subgoals trajectory reduction: the agent is just conditioned by the final goal state. We perform these ablation studies on the maze environment and use the same hyperparameters as in Section 3.4.4. The results are shown in Figure 3.12.

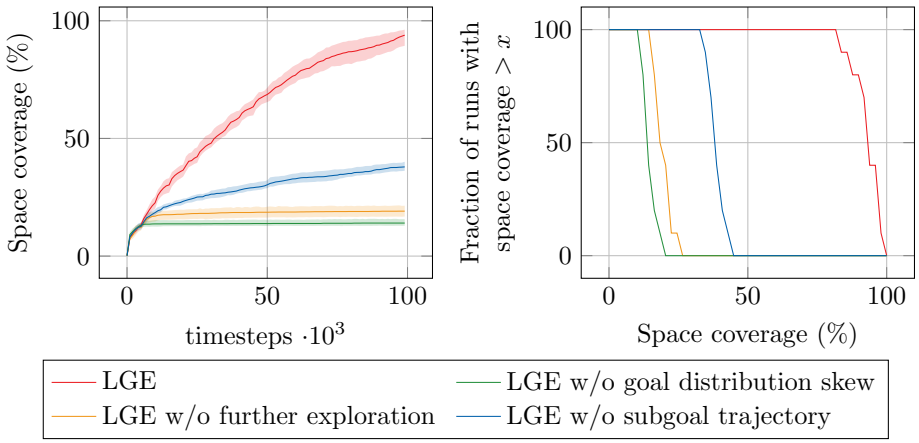


Figure 3.12: Result of ablation study on the maze environment. Each experiment is run 10 times. The left plot represents the space coverage across timesteps. The solid lines are the IQMs and the shaded areas are the 95% CIs. The right plot is the final performance profile (higher is better).

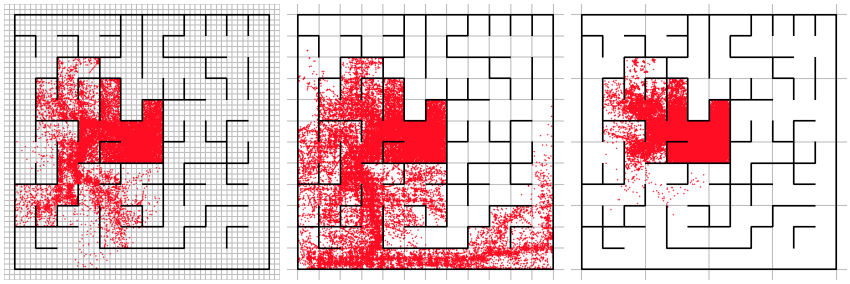
The impact of the three ablations on the outcome is significant. We find that exploration after reaching the final goal is crucial, confirming the results of Yang et al. [290]; without it, the agent reaches the limits of its knowledge but has little chance to explore further. Additionally, sampling goals with low latent density can significantly improve results by directing exploration to states with high exploratory value. Furthermore, we observe that conditioning the agent with successive subgoals greatly improves its exploration.

3.4.6 On the criticality of cell representation in Go-Explore

In Go-Explore, similar observations are grouped into cells and each cell encountered is stored in an archive. The cell representation is a critical aspect of Go-Explore. In the Montezuma's Revenge environment, a slight variation in cell representation results in an order of magnitude difference in the results. The cells are used to (1) estimate the density of states encountered in the observation space and sample a target cell against it; (2) divide this goal reaching task into a sequence of subgoals.

We argue that building a cell representation to capture the relevant components of an environment to perform the desired task requires a significant amount of domain knowledge. In general, this cell representation cannot be generalized to other tasks or to other environments.

To support our claim, we present in Figure 3.13 the space coverage in a continuous maze for different cell design. We show that even in this simple environment, a small variation in cell design has a significant impact on the result.



(a) Cell size = 0.5×0.5 (b) Cell size = 2.0×2.0 (c) Cell size = 6.0×6.0

Figure 3.13: Go-Explore scene coverage after 100k timesteps. The cell design is represented by the gray grid. We show the results for 3 different cell widths and shift. The red dots represent the visited states.

On the left, the cells are small, and the agent must visit each of them. If the agent interacts long enough with the environment, it should eventually explore the whole space. On the right, the cells are large. We can observe some detached areas, because the agent has not visited the cell enough to discover the next one, but enough so that this cell is no longer listed as a target cell.

3.5 Discussion

3.5.1 Limitation and future work

Goal-achievement functions In LGE, an agent is considered to have reached a goal (whether final or intermediate) when the latent distance between its state and the goal is below a threshold. This is a naive way of defining a goal achievement function [51] that depends crucially on the latent representation. We believe that the results could be improved by envisioning a more informative and suitable goal achievement function for our method.

The initial state must remain the same across episodes The approach we propose is based on the assumption that the agent is always initialized in the same state. This assumption guarantees that at the beginning of each episode, all the states previously reached are reachable and that the subgoal trajectory starts with the initial state of the agent. However, in some environments, especially in PGE, this assumption is not fulfilled [145]. In this situation, trying to follow the subgoal trajectory may be counterproductive in reaching the final goal. It is also possible that the final goal is not even reachable. However, note that even the pursuit of an unreachable goal can foster exploration. We believe that an approach inspired by generative networks such as [219] may be appropriate to overcome this problem.

High-dimension environments and representation learning Our main contribution consists in the generalization of the Go-Explore approach by using a latent representation. LGE is notably effective in high-dimensional environments, specifically those with image observations. Representation learning is the

keystone of the method. We provide a proof of concept for a forward model, an inverse model, and a VQ-VAE. We believe that the results can be greatly improved by choosing more finely the representation learning method for each environment by taking advantage of the many works dealing with this subject [158]. Representations are expected to encapsulate transitional proximity between observations, a feature not guaranteed by most learning methodologies. Nonetheless, in practice, such transitional proximity is often exhibited in learned representations.

The representation used by Search on Replay Buffer (SoRB) [76] is directly that of the critic. Using the same reward structure as LGE, the critic thus has the nice property of basically learning the negative distance of the shortest directed path between two states. Overall, we believe that the use of SoRB in the "Go" phase can be a substantial improvement of LGE and is a promising way to solve the three limitations mentioned above.

Finally, we believe that the community should endeavour to find a relevant metric for exploration, especially for image-based environments. We expect that such a metric would allow a more accurate comparison of different methods.

3.5.2 Conclusion

We introduce LGE, a new exploration method for RL. In this method, our agent explores the environment by selecting its own goals based on a jointly learned latent representation. LGE can be used as pretraining in environments where rewards are sparse or deceptive. Our main contribution is to generalize the Go-Explore algorithm, allowing us to benefit from representation learning algorithms for exploration. We present statistically robust empirical results conducted on diverse environments, including robotic systems and Atari games, that demonstrate our approach's significant improvement in exploration performance.

As we conclude our study of the LGE method, it is important to recognize that its framework, while more general, differs significantly from the ideal of a general RL agent. Such an agent would be able to cope with all kinds of environments and perform a variety of tasks using a single neural network. This direction remains largely unexplored in current research. In the following chapter, we attempt to fill this gap by laying the foundations for a more versatile RL approach.

Chapter 4

Jack of All Trades, Expert of Some, a Multi-Purpose Transformer Agent

In the previous chapter, we introduced a novel approach to enhance the exploration capabilities of RL agents under conditions of weak supervision, which is particularly vital in scenarios where the reward signal is sparse or inadequate to guide the agent's decision process. In the quest of a general-purpose model, this approach addresses only one facet of the challenge. The critical task of designing an agent capable of operating seamlessly across multiple domains remains a largely unexplored topic. The prevailing methodology in RL typically limits models to a single task within a unimodal framework, a limitation that contrasts with the broader vision of a versatile, multi-domain model. In this chapter, we present Jack of All Trades (JAT), a transformer-based model with a unique design optimized for handling sequential decision-making tasks and multimodal data types. The JAT model demonstrates its robust capabilities and versatility by achieving strong performance on very different RL benchmarks, along with promising results on CV and NLP tasks, all using a single set of weights. The JAT model marks a step towards more general, cross-domain AI model design, and notably it is the first model of its kind to be fully open-sourced[1], including a pioneering general-purpose dataset.

The JAT project, initiated by Edward Beeching, represents a collaborative effort between Hugging Face and the LIRIS laboratory at Ecole Centrale de Lyon in which I assumed a leading role and was primarily involved in every facet. Ed Beeching was deeply involved in developing the code, playing a crucial part in the project's technical progression. Clément Romac's contributions were invaluable, particularly in building the dataset. Our regular discussions and collaborative approach significantly shaped the project's direction. This work resulted in a paper [92], currently under review for ICML 2024.

[1] Codebase: `https://github.com/huggingface/jat` (private until release, in the meantime: `https://github.com/qgallouedec/jat`)
Model: `https://huggingface.co/jat-project/jat`
Dataset: `https://huggingface.co/datasets/jat-project/jat-dataset`

4.1 Introduction

Machine learning researchers have long aimed to develop versatile models that can adapt seamlessly to different domains. The recent success of Transformers [274] in NLP, CV, and to some extent in RL, has opened new avenues in this quest.

In this work, we attempt to extend the boundaries of this success by proposing a single, unified model capable of operating across a wide range of NLP, CV, and RL tasks using a single set of parameters. This effort not only seeks to challenge the conventional compartmentalization of AI tasks into distinct domains, but also aims to establish a more holistic approach to AI model design.

While the blending of visual and textual tasks has been extensively explored in recent research, integrating RL tasks has been less studied and presents unique challenges. RL tasks are inherently diverse and heterogeneous, making their combination among themselves and with other domains a highly complex exercise. This integration requires dealing with a landscape of different modalities, task complexities, and data volumes across domains and tasks. New questions that arise include: (1) How to design a model and learning method that effectively handles different modalities and data types (sequential and non-sequential)? (2) How to formulate a learning objective that appropriately balances and harmonizes the different modalities, tasks, and domains without bias toward any particular domain or task? (3) How to design a learning strategy that can accommodate the different levels of complexity inherent in different tasks?

These goals are concurrent and, to our knowledge, have only been addressed together by Reed et al. [224] with their model Gato. Our contributions are characterized by three major advances: (1) Our model features an innovative structure optimized for sequential tasks. It uniquely assigns each timestep to a corresponding token embedding, resulting in a simpler design. This approach significantly expands the attention window in terms of timesteps compared to Gato (e.g., it is 19 times larger for Atari and more than 25 times larger for Meta-World). (2) In the spirit of open source, we release our code, dataset, and model to the research community. (3) We add observation prediction as an auxiliary task to our model. We prove that this integration significantly contributes to learning a more efficient agent.

Ultimately, our JAT model achieves competitive results on the tasks studied, while being more than 6 times smaller than Gato and relying on a significantly lower training budget. As mentioned above, this new paradigm raises a number of open questions and paves the way for new research. We present a first milestone in this emerging framework and acknowledge the significant potential for improved results.

4.2 Related Work

4.2.1 Transformer for RL

Transformer models [274] are designed to model sequences and in particular sequences of words in natural language. However, sequence modeling problems span over a much larger set of domains than only NLP. In this work, we focus on modeling RL trajectories (i.e., sequences of observations, actions and rewards).

Modeling such sequences with a Transformer was introduced by Chen et al. [39] and the Decision Transformer (DT) model. In DT, a Transformer model is trained with offline RL to take sequences of transitions as input and predict the next action. In particular, Chen et al. [39] proposed to use returns-to-go (i.e. the return from the current state) to condition actions' generation on both the previous observation and the desired return-to-go. While this has the advantage of explicitly modeling the relations between action selection and return, using the model at inference requires providing at every step a desired return. Liu and Abbeel [170] proposed to extend this by using hindsight relabelling to better exploit sub-optimal trajectories. Zheng et al. [297] also extended the DT approach by mixing offine pretraining and online finetuning. Finally, Lee et al. [155] studied how the DT approach scales to a multi-task RL setup where a single policy is learned for multiple games. Our work lies in this line of work as it also leverages Transformers to model trajectories. However, our approach (1) uses standard Behavior Cloning (BC) instead of conditional BC, relaxing the need to condition the agent by the return-to-go and (2) models a multi-task dataset in which sequences come from very different domains (e.g. control, Atari, visual question answering, see Section 4.3.2).

4.2.2 Multi-Modal Transformer

Apart from being widely used in NLP, Transformers also strive in vision and vision-and-language domains. As one of the first works leveraging Transformers for vision, Dosovitskiy et al. [67] introduced Vision Transformer (ViT), a Transformer model using image patches for recognition. Following this, a line of work aiming to train multi-modal Transformers using both text and images emerged, including works such as Flamingo [6], PaLI [40] or IDEFICS [151]. All these models imply the use of an image encoder allowing to obtain image tokens or embeddings that can be given to the Transformer alongside text tokens.

While all the aforementioned approaches produce text as output and were solely trained to solve vision-and-language tasks such as Visual Q&A, our work spans both vision-and-language and embodied decision-making (e.g. control). Closer to our approach, a recent series of multi-modal Transformers trained for decision-making appeared. For instance, Jiang et al. [126] trained a Transformer-based robot agent with IL on tasks where a robot arm must place objects on a table to match an instruction. Their model uses multi-modal prompts as input (allowing them to define various objectives with a unified approach e.g. visual goals, demonstrations, text instructions, object naming) and directly produces motor actions for the robot arm. Similarly, RT-1 [31] leverages a multi-modal Transformer and IL to produce motor actions for robots but this time using expert demonstrations coming from real-world robots. Palm-E [68] and later RT-2 [30] proposed to study the impact of pretrained Transformers when learning decision-making robotics tasks with IL. Both works proposed to leverage a Visual Language Model (VLM) trained on vision-and-language tasks and finetune it for robotics tasks. While RT-2, being an extension of RT-1, directly outputs motor actions, Palm-E still produces, as its backbone VLM, sequences of text instructions (i.e. plans) that are then executed by low-level control policies.

Finally, our approach is largely inspired by Gato [224], which proposed to train a Transformer on both vision-and-language and decision-making tasks without relying on any pretrained model. The resulting model is therefore smaller

than the ones leveraging large VLMs (e.g. Palm-E, RT-2) while still being able to perform both vision-and-language and decision-making tasks. In this work, we first propose to build a dataset that resembles Gato's dataset except we only use open-source data sources and release all demonstrations as well as expert policies we used to obtain these demonstrations. Then, we also leverage a multi-modal Transformer along with IL for our model, but introduce several improvements, especially in the way observations are mapped to embeddings, allowing us not to design prompts and improving the use of continuous observations (see Section 4.3.1).

4.2.3 Multi-Task RL

The quest for a general agent has long been a goal of RL [21]. However, most works have chosen to use a different neural network for each environment. Recent research has revived interest in this objective and explores it through several approaches.

One such approach involves directly extending online learning to multi-task environments [74, 291, 254]. These works highlight the potential for positive transfer in multi-task learning, meaning that learning across tasks can be mutually beneficial due to underlying commonalities. However, they also acknowledge the risk of negative transfer, where inter-task interference can impair training. Studies have investigated methods to limit this risk, such as that by Yang et al. [289], which proposed refined gradient management techniques to mitigate these detrimental effects.

An alternative approach is policy distillation, which involves condensing the behaviors of expert agents into a singular, unified policy [230, 203]. While these studies also report positive transfer across tasks [230], they also identify instances of negative transfer. Subsequent research has focused on strategies to minimize this negative transfer [266]. One limitation of policy distillation is its reliance on the availability of policies to distill. This constraint is notably addressed in [39], which proposes conditioning the distilled policy on the desired return thus allowing the use of any policy, including those from non-expert agents. This strategy has been adapted to the multi-task setting by Lee et al. [155].

Despite the diversity of research in this area, most studies are limited to multi-task learning within a single domain, such as Atari or Meta-World, and thus involve semantically related tasks (although this is somewhat less true for Atari). The only notable exception we found is the Gato model [224], which learns a large number of domains in a single network. It is the closest baseline to our work.

4.2.4 Spotlight on Gato

In this section we briefly introduce Gato [224], a model which, although quite different, is the closest to our work. Gato is intended to be multi-modal, multi-tasking and multi-domain, in the sense that it addresses RL, NLP and CV tasks in a single network. It is a transformer-based model consisting of three key stages: tokenization and embedding, transformer encoding, and decoding with loss computation. Gato's main contribution lies in these initial and final stages.

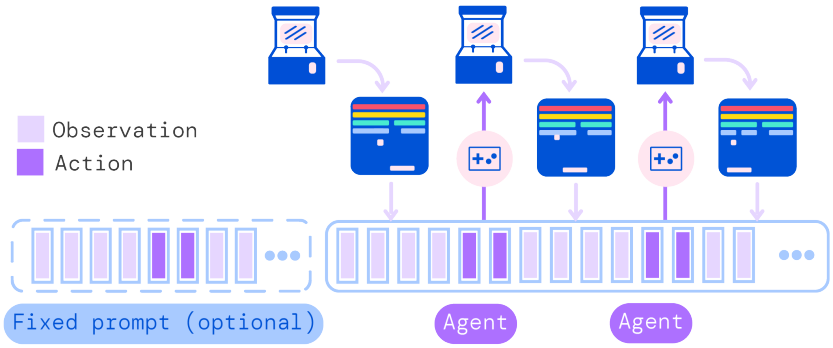


Figure 4.1: Operation of Gato as a control policy. Gato takes in a sequence that consists of tokenized observations, separators, and actions. It uses this input to generate the next action following a autoregressive process. Figure from [224].

Overview

In the initial tokenization and embeddings stage, all modalities are tokenized and embedded, with the exception of images, which are divided into patches and directly embed. Text undergoes standard SentencePiece tokenization [137], while discrete values are shifted by 30,000 to prevent overlap with text tokens. Continuous vectors are mapped into the interval $[-1, 1]$ using the μ-law companding algorithm, discretized into 1,024 bins, and shifted by 30,000. Token count per timestep varies based on observation and action structure, for example, Pendulum results in 6 tokens per timestep, while Humanoid uses 394 tokens. Positional embeddings vary by modality and type, e.g., images use uniformly sampled positions within patch windows and discretize over 128 values horizontally and vertically. For actions, a dedicated local position embedding is employed. Gato uses an autoregressive approach for predicting and optimizing token logits, while masking tokens unrelated to text or action. In terms of training, 25% of the sequences within each batch receive a prepended prompt sequence. These prompts are sourced from episodes generated by the same agent on the same task. Half of the prompts are extracted from the episode's end, while the other half are uniformly sampled from the episode. During evaluation, the agent consistently relies on prompting.

Limitations adressed

In summary, Gato exhibits complexity and is expected to be sensitive to parameters such as discretization granularity, which is likely to be domain-dependent. Discretization introduces discontinuity in the observations, which may affect generalization. Furthermore, the fact that the timestep-based attention window depends on observation size may lack intuitiveness. While Gato predicts tokens for all modalities and types, it only allows decoding and prediction for a subset of them, despite the potential benefits of full prediction for learning. Finally, the heavy reliance on prompts adds further complexity to both the learning and evaluation processes, raising questions regarding parameter selection and optimality, which are also likely to be domain-dependent.

In the next section, we will see how the JAT model addresses these challenges. JAT simplifies the approach by introducing dedicated encoders for each modality, ensuring a consistent mapping between timesteps and tokens. This modification is intended to enhance generalization, particularly for continuous modalities. Furthermore, we illustrate how JAT is capable of decoding and optimizing all inputs, including observations—a critical auxiliary objective that significantly improves agent learning. Lastly, while we acknowledge the motivation behind the use of prompts in multi-task scenarios, JAT takes a divergent approach by eliminating prompts. Our findings reveal minimal impact on results, resulting in a substantial simplification of both the training and deployment processes.

4.3 Methodology

In this section, we introduce the JAT model, detailing our architectural choices that underpin its effectiveness and highlighting its ability to handle different modalities in both sequential control and text-centric tasks. We present the associated dataset, which is notable for its groundbreaking diversity across domains and modalities. Finally, we discuss in depth the learning strategy used.

4.3.1 Model architecture

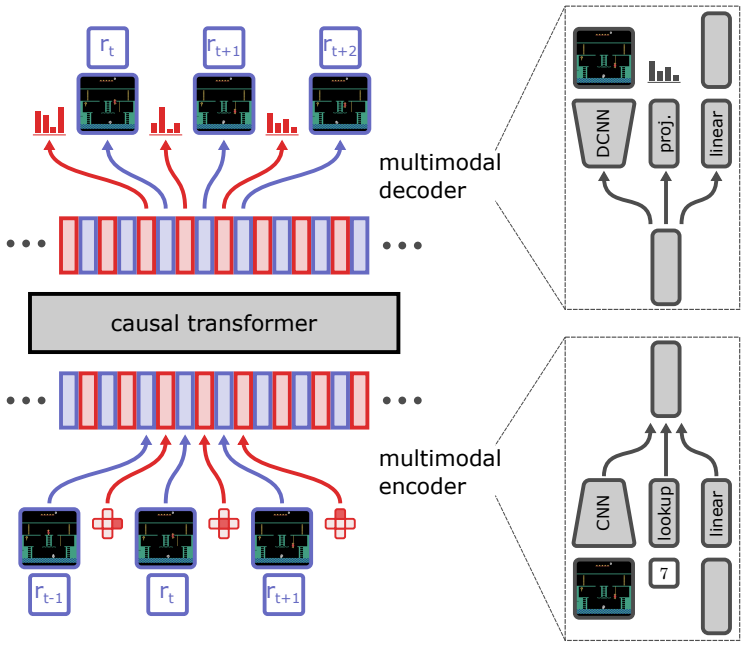


Figure 4.2: Architecture of the JAT network. For sequential decision-making tasks, the observations joined to the rewards are encoded and interleaved with the action embeddings. Each modality has its own encoder and decoder. The model generates the next embedding autoregressively with a causal mask, and decodes according to expected modality.

Embedding mechanism

The model is designed to handle two main categories of data: tasks involving sequential decision-making and non-sequential tasks. In non-sequential tasks, the model currently supports two modalities: text and image. Although the current version of the model supports image generation, we focus only on tasks that involve text generation. To ease reading, we will refer to it as *text-centric tasks* in the remainder of this work. Each of these two categories requires a slightly different approach to the embedding process.

In both cases, the resulting sequence is truncated to match the maximum permissible input size of the inner Transformer model. Any truncated portion is not discarded; instead, it forms the basis of a new sample. This process may be repeated if necessary, ensuring that no valuable information is lost.

Sequential decision-making tasks For tasks involving sequential decision-making, the data comprises a sequence of observations, actions, and rewards. At the embedding stage, these sequences are processed to produce an interleaved sequence of observation embeddings (augmented with the corresponding reward) and action embeddings, denoted as $[\phi(s_0, 0.0), \phi(a_0), \phi(s_1, r_1), \phi(a_1), \ldots]$. Unlike DT [39] and Gato [224], each timestep is consistently associated with two embeddings: one for the observation and the other for the action, regardless of the modality. This enables JAT to better handle high-dimensional observations, and to provide a much wider, constant attention window in terms of timesteps. As an example, this multiplies the size of the attention window in terms of timesteps by more than 25 for Meta-World. The embedding method employed at a specific timestep is modality-dependent (with H the hidden size of the model):

- Continuous observation: The reward value is appended to the observation vector. This augmented vector is then padded to achieve a uniform length of 377, corresponding to the maximum augmented observation size in the dataset. The embedding vector is subsequently obtained by passing this padded vector through a linear layer with an output size of H. This layer is consistently used across all timesteps.

- Discrete observation: The observation consists of a vector of integers, each of which is encoded into a continuous vector of size H using a lookup table. Subsequently, a linear layer is applied to reduce the dimensionality to $\lfloor H/50 \rfloor$. Following vector flattening, another linear layer is applied, resulting in an output size of $H - 1$. Lastly, the reward is added to the resulting vector.

- Image observation: The input image is first resized to a uniform dimension of 84×84 using bicubic approximation, normalized, and padded to ensure 4 channels. The image encoder consists of a series of three blocks, each consisting of a convolutional layer, an instance normalization layer, and an attention layer. The output of the last block is flattened and passed through a linear layer, resulting in an embedding vector of size H.

- Continuous action: The process is similar to that of continuous observations, with the exception of the reward component. Notably, the linear layer is shared with the one used for continuous observations.

- Discrete action: In the case of discrete actions, the process is slightly different due to the nature of the input: a discrete action is represented by a single integer, as opposed to a vector of integers for discrete observations. The input is directly mapped to a continuous vector of size H using the same lookup table employed for discrete observations.

Text-Centric Tasks For text-centric tasks, each sample includes text, accompanied or not by an image.

- Image data: We employ the ViT architecture, as originally proposed by Dosovitskiy et al. [67]. The image is first cropped to its central square, and resized to 224×224. The image is then normalized and divided into non-overlapping patches of 16×16. Each patch is linearly embedded in a vector of size H.

- Text data: We use the GPT-2 tokenization strategy [220], utilizing a byte-pair encoding (BPE) [243] specifically designed for unicode characters. The tokenizer produces a vocabulary of 50,257 tokens. For efficient implementation, we use the Hugging Face integration [191]. Each token is mapped to an embedding vector using a lookup table, where each unique token in the vocabulary is associated with a distinct vector. Notably, this lookup table is shared with the one employed for discrete values in sequential decision-making tasks.

When a sample includes both images and text, the embeddings are arranged so that the image embeddings precede the text embeddings. This specific order is essential for image captioning task because of the causal masking applied by the model's internal Transformer. The concatenated image-text embeddings form a unified representation for subsequent processing steps.

Transformer architecture

The JAT model is based on a Transformer architecture using EleutherAI's implementation of GPT-Neo [27]. It takes as input the embedding sequence whose computation was described in the previous section. The model uses a dual attention mechanism whose design is inspired by the Longformer [24]: global attention with a window size of 512 tokens for full context understanding, and local attention with a fixed window of 256 tokens. The Transformer's feed-forward components consist of 12 layers and 12 heads with an intermediate dimensionality of 8192 and a hidden size of 768. They are designed to be causal, meaning a causal mask is applied during training and inference.

Output processing and loss

The internal causal Transformer outputs a sequence of embeddings, each encoding the basis for predicting subsequent elements in different data modalities. As we predict multiple modalities within a single sequence, we use the appropriate decoders and corresponding loss functions for each modality. When an embedding encodes an image, we use a transposed convolutional neural network [292] for prediction. When an embedding represents a continuous vector, we use a continuous linear layer for prediction. For both image and continuous vector

prediction, we compute the loss using Mean Square Error (MSE). When embedding represents a discrete value, we use a linear projection layer to score each discrete candidate and compute loss via cross-entropy. Notably, this projection layer is shared between text tokens and discrete sequential values (e.g., actions for Atari and BabyAI). The sequence's overall loss is determined by averaging the individual losses computed for each element. In sequential decision-making tasks, we assign a weight to balance losses associated with observations and actions. In Section 4.4.3, we demonstrate that predicting observations does help in learning, addressing a common open question in [39] and [224].

4.3.2 Datasets

In this work, we have collected a wide range of datasets, classified into two main groups: sequential decision-making datasets and textual datasets. The former include a series of interaction sequences, each consisting of observations, actions and a subsequent rewards, generated by so-called expert agents, details of which are given in the following sections. The latter includes large corpora of textual data and image-text pairs. In order to promote the emerging field of general-purpose AI models, we have made these datasets, together with the expert agents and the full set of code required to generate them, available to the public as open resources in our Hugging Face repository, accessible at the following address `https://huggingface.co/jat-project`. To the best of our knowledge, this compilation is unprecedented in terms of the variety of tasks and the volume of data, representing a valuable new contribution to the field.

Sequential decision-making datasets

For each decision-making environment, we collect a set of interactions using expert agents. Detailed scores are available in the Section 4.4.2.

Atari We use the 57 games from the ALE [21] as a benchmark in our research, amassing roughly 500,000 interactions per game. Episode lengths varied significantly depending on the specific game. For each game, we trained a dedicated agent using the asynchronous implementation of Proximal Policy Optimization [239] from Sample Factory [208]. While the expert agents demonstrated commendable performance across many games, they fell short of human benchmarks in 14 tasks[2]. Five environments of this benchmark are displayed in Figure 4.3.

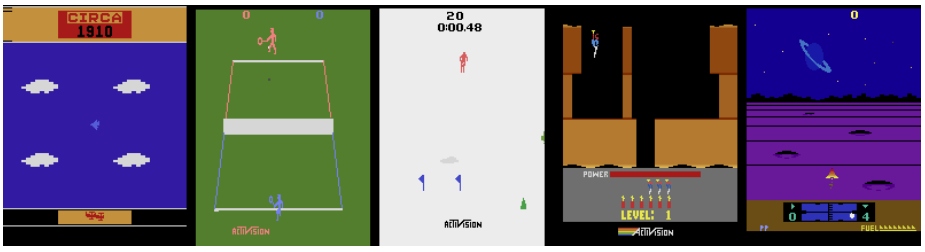


Figure 4.3: Examples of environments from the Atari benchmark.

[2]Asterix, Bowling, Centipede, Fishing Derby, Kangaroo, Montezuma's Revenge, Ms. Pacman, Pitfall, Private Eye, River Raid, Seaquest, Skiing, Solaris, Venture

BabyAI BabyAI stands out in our study due to its unique characteristic of being partially observable and its dual-modality observations [43, 44]. Using the bot provided with the BabyAI paper [43], we gathered 100,000 episodes for 39 of its available settings. Each interaction consists of a text observation (mission), a discrete observation (7×7 symbolic representation of the agent's field of view), an action, and a reward. Five environments of this benchmark are displayed in Figure 4.4.

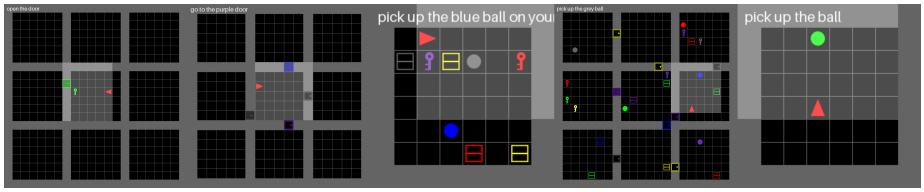


Figure 4.4: Examples of environments from the BabyAI benchmark.

Meta-World Meta-World's MT50 benchmark provides a set of 50 diverse and challenging robot manipulation tasks [291]. Similar to the methodology used for Atari, we trained one agent per task using the asynchronous PPO [239] implementation of [208]. The trained agents solved most of the tasks, except for Assembly and Disassemble, where they failed to reach the expected performance. We limit the number of timesteps per episode to 100, which proved to be sufficient for solving the tasks. Without this limit, much of the subsequent dataset would consist of the stabilization phases of the agents after goal attainment, reducing its relevance. We then used the trained agents to generate 10,000 episodes per environment. Five environments of this benchmark are displayed in Figure 4.5.

Figure 4.5: Examples of environments from the Meta-World benchmark.

MuJoCo We included the MuJoCo locomotion benchmark suite [268, 29] comprising 11 continuous control tasks into our study due to its diverse challenges in domain complexity and task difficulty, and its wide recognition in the research literature. Following our methodologies for Atari and Meta-World, we individually trained agents for each task using asynchronous PPO [239] from Sample Factory [208]. These agents successfully solved all tasks, achieving scores that meet or exceed the current highest standards. Subsequently, we employed these agents to generate 10,000 episodes per environment. Five environments of this benchmark are displayed in Figure 4.6.

Text-centric datasets

Oscar Common Crawl-based text documents have been widely used in the past to create datasets for Language Modeling [220, 32, 221]. We chose to leverage

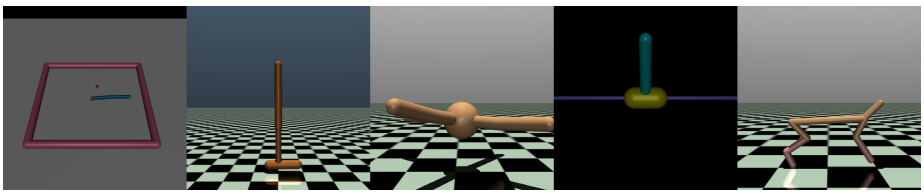


Figure 4.6: Examples of environments from the MuJoCo benchmark.

the unshuffled deduplicated English subset of the OSCAR[3] corpus [198] for our Language Modeling objective. As such crawled internet data needs to be cleaned before using it for training Language Models (e.g. deduplication, filtering out machine-generating content), we reused both the cleaning and deduplication pipeline from the ROOTS corpus [152]. The initial dataset was shuffled, split into a training (95%) and test (5%) set, and evenly split into 30 shards on which the cleaning and deduplication pipelines were applied to reduce the memory needs. Shards were then concatenated back together, leading to a final dataset of 245 million documents (compared to 304 million documents in the initial dataset).

Conceptual-Captions We include the Conceptual-Captions dataset [245], as it is a key resource for image captioning and visual understanding tasks. It contains over 2.6 million training examples and over 12,000 test examples, with a wide range of web-sourced images, each paired with a descriptive caption.

OK-VQA We include the OK-VQA dataset [181] because it is an essential resource for visual question answering tasks that focus on the intersection of visual perception and knowledge-based reasoning. With over 14,000 samples, it contains a wide range of images, each associated with questions that require not only visual understanding, but also external knowledge for an accurate answer.

Wikipedia The Wikipedia dataset, built from the Wikipedia dump [85], contains over 6 million English language samples as of March 1, 2022. It offers a wide range of topics and a wealth of information. By using this dataset, we aim to improve the language processing capabilities of our model and provide access to extensive reservoir of encyclopedic knowledge.

4.3.3 Training

Overall training procedure

The model was trained for 250,000 steps. We distributed the training across 8 GPUs NVIDIA V100 using the Trainer from the Hugging Face Transformers library [286] in conjunction with Accelerate [102]. This training lasted approximately 9 days. For practical reasons, each batch is made up of data from a single dataset. We use a constant batch size of 20 and accumulate over 2 steps, resulting in an effective batch size of 320. Consequently, each optimizer step is performed on data from up to 16 datasets. We utilize the AdamW optimizer

[3]Its original version from 2019: `https://huggingface.co/datasets/oscar`

with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate starts at $5 \cdot 10^{-5}$ and linearly decays to zero throughout the training process.

Task-specific weight adjustments

Each task presents a unique training challenge. To allow for balanced learning of all tasks, we introduced custom weight modifications. The choice of these weights is made heuristically. The learning would surely benefit from a more precise and systematic method for choosing these weights.

Sample Weight Some tasks required more updates for effective convergence. To allow proportionate progress of all tasks during learning, these tasks are sampled more frequently. Specifically, Oscar, Conceptual-Captions and Wikipedia was assigned a sample weight of 10.0 while other have a sample weight of 1.0.

Loss Weight Some control tasks require increased accuracy of actions. To allow for more strongly penalizing the error for these tasks, we assigned loss weights. In MuJoCo tasks, the loss weight is typically set at 10.0, except for the Pendulum task (20.0) and the Double Pendulum task (50.0). In Meta-World tasks, a uniform loss weight of 50.0 is used.

CO_2 emission related to experiments

Experiments on the Jean Zay supercomputer (France), with a carbon efficiency of 57.3 gCO_2eq/kWh [255], consumed 2000 GPU hours on Tesla V100-SXM2-32GB hardware (TDP[4]: 482 W). Total emissions were estimated at 57.2 $kgCO_2eq$, or 29.7 $kgCO_2eq$ when considering heat recovery. Estimations used the Machine-Learning Impact calculator from [147].

4.4 Experiments and Results

In this section we discuss the results of our experiments. First, we provide a brief overview of the model's performance on text-centric tasks. We then present the results of the sequential decision tasks, showing the different levels of mastery across the different domains within our study. Finally, we provide a comprehensive analysis highlighting the benefits of incorporating the prediction of the next observation as an auxiliary task during the learning process.

4.4.1 Text-centric tasks

We present a showcase of JAT's capabilities, with a particular focus on text completion and image captioning. It is important to note that JAT is much smaller and has a much lower training budget than the specialized models for these tasks. Therefore, instead of comparing it to these expert systems, we want to demonstrate its intrinsic capabilities. Figure 4.7 shows a selection of captioning results to illustrate how the model interprets and describes visual data. Figure 4.8, meanwhile, shows a series of text-based interactions that provide

[4]Estimate for 11/2020: `http://www.idris.fr/media/jean-zay/jean-zay-conso-heure-calcul.pdf`

the flag was removed from the building after the fire broke out

: and beverage type at the beach. here are some of the most beautiful things i have ever seen.

: the food is good for the body! :) - a-baked chicken. person. photo by author

tattoo on the left inner forearm. ∼ artist. ∼ ∼ photo sharing website

- shaped cloud formation over a city. ∼ photo by person. #zn. - #zn.0 # christmas

and illustration of the new year. photo by person.

s and drawings on the ceiling of a building.

- cut emerald - cut diamonds are a perfect addition to any home.

day in the green forest.

for the first time! by person, the man who is now on the right.

s are part of the annual event. organisation

ging it up : the model was spotted wearing a pair of black jeans, a white t - shirt and black t - shirt

s and other souvenirs at the market.

s of the day : flowers

dog in a bedroom.

Figure 4.7: JAT image captioning examples. The theme is usually correct, although the relevance is sometimes limited.

insight into its ability to complete text prompts. These examples were chosen to highlight the model's basic capabilities in these areas, providing a realistic view of its current state of development and potential for future enhancements. In addition, we provide a demo[5] for direct interaction and experimentation, allowing users to experience its functionalities.

Input	Model completion
The weather today is	a great time to the city of New York City. The city is a great place to stay in.
In the future, cars will	be able to drive cars to the market. The new car will be built in the new market for the new car.
My favorite book is	a book by the author of the book.
The first official	website of the United States was held on November 6, 1891, at the behest of the United States Secretary of State.

Figure 4.8: JAT text completion examples. The syntax is generally correct, the completion is on-topic, although the generated text may be wrong.

4.4.2 RL tasks

We save checkpoints regularly during training. We evaluate each checkpoint on all the tasks on which it has been trained. Unlike Gato, the evaluation does not require any data to be used as a prompt. We show empirically in Section 4.4.4 that despite the absence of a prompt, and even in the worst case of our study, the agent still manages to identify the requested task. For each task, we collect 10 evaluation episodes and normalize by the average expert score of the dataset for this task. For the final checkpoint, we use 100 evaluation episodes. We then aggregate the results by domain. The score of the random agent for Atari games is sourced from [188]. In other domains, this score is approximated by averaging the returns from 1,000 episodes, where the agent selects actions uniformly across its action space. The expert scores represent the average return in the dataset for the task. Meanwhile, the raw score is the average return achieved by the trained agent, based on 100 evaluation episodes. Both these scores, along with the trained agent, are accessible as open-source[6]. The normalized score is derived by comparing the agent's return to the expert's, calculated using the formula: $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$. It is important to note that in instances where the *expert*, inaccurately named, does not fully master the task and thus scores similarly or lower than the random agent, the normalized score must be interpreted cautiously. Specifically, if this score falls below that of the random agent, as in the case of Bowling, normalization is not applied. The full results for Atari are presented in Table 4.1, for BabyAI in Table 4.2, for Meta-World in Table 4.3, and for MuJoCo in Table 4.4. Figure 4.9 focuses on Atari, showing the human normalized score for each environment, and Figure 4.10 shows the evolution of the aggregate score for each domain during learning.

[5] `https://huggingface.co/spaces/jat-project/text-completion`
[6] `https://huggingface.co/jat-project/jat`

Table 4.1: Comparison of performance scores across tasks on Atari 57. The table presents the episodic return (score) achieved by a random agent (from [188]), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT	JAT (normalized)
Alien	227.8	16912.5 ± 7087.4	1085.9 ± 396.4	0.05 ± 0.02
Amidar	5.8	2164.7 ± 1229.5	41.3 ± 28.6	0.02 ± 0.01
Assault	222.4	15699.1 ± 9572.1	772.9 ± 59.3	0.04 ± 0.00
Asterix	210.0	3699.6 ± 2421.3	778.5 ± 429.0	0.16 ± 0.12
Asteroids	719.0	177011.1 ± 35334.2	1423.6 ± 538.8	0.00 ± 0.00
Atlantis	12850.0	320679.6 ± 418247.4	23541.0 ± 10376.7	0.03 ± 0.03
Bank Heist	14.2	1322.4 ± 60.8	685.5 ± 157.9	0.51 ± 0.12
Battle Zone	236.0	295592.6 ± 161961.0	12950.0 ± 4306.7	0.04 ± 0.01
Beam Rider	363.9	29589.3 ± 16133.0	762.0 ± 243.3	0.01 ± 0.01
Berzerk	123.7	57085.3 ± 13104.5	523.9 ± 161.9	0.01 ± 0.00
Bowling	23.1	20.4 ± 7.3	30.0 ± 11.5	N/A
Boxing	0.1	98.0 ± 3.8	87.0 ± 22.6	0.89 ± 0.23
Breakout	1.7	703.0 ± 203.6	9.2 ± 5.8	0.01 ± 0.01
Centipede	2090.9	11624.3 ± 4918.3	4461.7 ± 2188.8	0.25 ± 0.23
Chopper Command	811.0	90990.6 ± 270876.9	1497.0 ± 723.1	0.01 ± 0.01
Crazy Climber	10780.5	179296.9 ± 39862.1	52850.0 ± 31617.9	0.25 ± 0.19
Defender	2874.5	351958.3 ± 40466.8	10627.5 ± 4473.2	0.02 ± 0.01
Demon Attack	152.1	92195.2 ± 26174.8	315.1 ± 279.0	0.00 ± 0.00
Double Dunk	-18.6	20.9 ± 3.6	0.1 ± 11.6	0.47 ± 0.29
Enduro	0.0	2292.2 ± 147.5	111.5 ± 27.4	0.05 ± 0.01
Fishing Derby	-91.7	7.2 ± 25.1	-55.2 ± 19.4	0.37 ± 0.20
Freeway	0.0	33.9 ± 0.3	24.1 ± 1.6	0.71 ± 0.05
Frostbite	65.2	13196.1 ± 4341.0	617.3 ± 686.1	0.04 ± 0.05
Gopher	257.6	81676.2 ± 46329.5	2947.2 ± 1448.3	0.03 ± 0.02
Gravitar	173.0	3986.6 ± 1729.0	1030.5 ± 719.2	0.22 ± 0.19
H.E.R.O.	1027.0	44677.4 ± 1754.4	6997.9 ± 2562.5	0.14 ± 0.06
Ice Hockey	-11.2	25.2 ± 5.8	-3.8 ± 3.1	0.20 ± 0.09
James Bond	29.0	27786.9 ± 33819.2	187.5 ± 72.2	0.01 ± 0.00
Kangaroo	52.0	574.0 ± 636.9	124.0 ± 156.9	0.14 ± 0.30
Krull	1598.0	11439.8 ± 1218.3	8933.0 ± 1358.6	0.75 ± 0.14
Kung-Fu Master	258.5	32392.8 ± 10006.6	100.0 ± 142.1	-0.00 ± 0.00
Montezuma's Revenge	0.0	393.5 ± 50.4	0.0 ± 0.0	0.00 ± 0.00
Ms. Pacman	307.3	6896.1 ± 2032.0	1516.3 ± 376.7	0.18 ± 0.06
Name This Game	2292.3	22991.2 ± 2473.1	3798.6 ± 1361.6	0.07 ± 0.07
Phoenix	761.5	424583.2 ± 97649.2	1267.5 ± 1013.7	0.00 ± 0.00
PitFall	-229.4	-1.4 ± 4.5	-287.4 ± 492.8	-0.25 ± 2.16
Pong	-20.7	21.0 ± 0.2	-11.0 ± 11.3	0.23 ± 0.27
Private Eye	24.9	100.0 ± 0.0	96.0 ± 19.6	0.95 ± 0.26
Q*Bert	163.9	42971.4 ± 85070.7	1701.8 ± 1912.6	0.04 ± 0.04
River Raid	1338.5	14800.9 ± 7924.6	2793.1 ± 693.8	0.11 ± 0.05
Road Runner	11.5	77942.8 ± 6088.6	7699.0 ± 3446.6	0.10 ± 0.04
Robotank	2.2	80.5 ± 13.3	16.4 ± 5.2	0.18 ± 0.07
Seaquest	68.4	2597.3 ± 386.1	515.2 ± 141.5	0.18 ± 0.06
Skiing	-17098.0	-10738.1 ± 111.1	-29396.1 ± 3289.8	-1.93 ± 0.52
Solaris	1236.3	1353.7 ± 517.0	988.2 ± 487.4	-2.11 ± 4.15
Space Invaders	148.0	29425.3 ± 23623.9	339.5 ± 164.1	0.01 ± 0.01
Star Gunner	664.0	360588.6 ± 49207.7	978.0 ± 638.4	0.00 ± 0.00
Surround	-10.0	9.4 ± 0.8	-8.2 ± 1.2	0.09 ± 0.06
Tennis	-23.8	11.1 ± 7.6	-22.4 ± 2.2	0.04 ± 0.06
Time Pilot	3568.0	69583.3 ± 29838.7	9534.0 ± 2577.8	0.09 ± 0.04
Tutankham	11.4	291.2 ± 30.4	40.2 ± 14.5	0.10 ± 0.05
Up and Down	533.4	429418.3 ± 7187.4	6072.0 ± 2283.3	0.01 ± 0.01
Venture	0.0	0.0 ± 0.0	0.0 ± 0.0	N/A
Video Pinball	0.0	441507.9 ± 283264.6	7943.0 ± 8351.2	0.02 ± 0.02
Wizard of Wor	563.5	49333.3 ± 16157.1	1306.0 ± 1139.8	0.02 ± 0.02
Yars Revenge	3092.9	270262.9 ± 161816.0	8597.4 ± 4291.8	0.02 ± 0.02
Zaxxon	32.5	73097.2 ± 14825.8	896.0 ± 1172.7	0.01 ± 0.02

Table 4.2: Comparison of performance scores across tasks on BabyAI. The table presents the episodic return (score) achieved by a random agent (averaged over 1,000 episodes), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT	JAT (normalized)
Action Obj Door	0.37 ± 0.39	0.99 ± 0.01	0.95 ± 0.13	0.94 ± 0.22
Blocked Unlock Pickup	0.00 ± 0.02	0.95 ± 0.01	0.95 ± 0.01	1.00 ± 0.01
Boss Level	0.06 ± 0.21	0.94 ± 0.05	0.48 ± 0.45	0.48 ± 0.51
Boss Level No Unlock	0.06 ± 0.19	0.94 ± 0.05	0.44 ± 0.45	0.43 ± 0.51
Find Obj S5	0.08 ± 0.23	0.95 ± 0.04	0.95 ± 0.03	1.00 ± 0.04
Go To	0.13 ± 0.29	0.92 ± 0.07	0.80 ± 0.27	0.85 ± 0.35
Go To Door	0.45 ± 0.38	0.99 ± 0.00	0.99 ± 0.01	1.00 ± 0.01
Go To Imp Unlock	0.07 ± 0.22	0.83 ± 0.13	0.50 ± 0.44	0.56 ± 0.59
Go To Local	0.16 ± 0.30	0.93 ± 0.04	0.88 ± 0.14	0.94 ± 0.18
Go To Obj	0.13 ± 0.27	0.93 ± 0.03	0.93 ± 0.04	0.99 ± 0.05
Go To Obj Door	0.53 ± 0.39	0.99 ± 0.01	0.98 ± 0.04	0.97 ± 0.08
Go To Red Ball	0.17 ± 0.30	0.93 ± 0.04	0.91 ± 0.08	0.98 ± 0.11
Go To Red Ball Grey	0.12 ± 0.27	0.92 ± 0.05	0.91 ± 0.06	0.99 ± 0.08
Go To Red Ball No Dists	0.14 ± 0.28	0.93 ± 0.03	0.93 ± 0.03	1.00 ± 0.04
Go To Red Blue Ball	0.12 ± 0.27	0.92 ± 0.05	0.88 ± 0.11	0.96 ± 0.13
Go To Seq	0.08 ± 0.23	0.94 ± 0.05	0.73 ± 0.34	0.75 ± 0.40
Key Corridor	0.00 ± 0.00	0.91 ± 0.01	0.88 ± 0.10	0.97 ± 0.11
Mini Boss Level	0.07 ± 0.21	0.89 ± 0.10	0.69 ± 0.35	0.76 ± 0.43
Move Two Across S8N9	0.00 ± 0.00	0.96 ± 0.01	0.03 ± 0.15	0.03 ± 0.16
One Room S8	0.08 ± 0.21	0.92 ± 0.03	0.92 ± 0.03	1.00 ± 0.04
Open	0.10 ± 0.24	0.95 ± 0.05	0.93 ± 0.11	0.97 ± 0.13
Open Door	0.23 ± 0.34	0.99 ± 0.00	0.99 ± 0.00	1.00 ± 0.01
Open Doors Order N4	0.16 ± 0.30	0.99 ± 0.01	0.96 ± 0.11	0.97 ± 0.13
Open Red Door	0.08 ± 0.21	0.92 ± 0.03	0.92 ± 0.02	1.00 ± 0.03
Open Two Doors	0.08 ± 0.20	0.98 ± 0.00	0.98 ± 0.00	1.00 ± 0.00
Pickup	0.08 ± 0.22	0.92 ± 0.07	0.72 ± 0.34	0.77 ± 0.40
Pickup Above	0.02 ± 0.09	0.91 ± 0.07	0.92 ± 0.06	1.01 ± 0.07
Pickup Dist	0.10 ± 0.24	0.86 ± 0.21	0.88 ± 0.13	1.03 ± 0.18
Pickup Loc	0.08 ± 0.23	0.91 ± 0.04	0.84 ± 0.20	0.91 ± 0.24
Put Next S7N4	0.00 ± 0.03	0.96 ± 0.01	0.82 ± 0.26	0.86 ± 0.27
Put Next Local	0.00 ± 0.05	0.92 ± 0.03	0.60 ± 0.36	0.65 ± 0.39
Synth	0.11 ± 0.26	0.93 ± 0.06	0.68 ± 0.39	0.69 ± 0.47
Synth Loc	0.13 ± 0.29	0.94 ± 0.06	0.82 ± 0.31	0.85 ± 0.38
Synth Seq	0.07 ± 0.20	0.95 ± 0.04	0.57 ± 0.44	0.57 ± 0.50
Unblock Pickup	0.08 ± 0.22	0.91 ± 0.08	0.76 ± 0.33	0.82 ± 0.39
Unlock	0.03 ± 0.15	0.87 ± 0.10	0.55 ± 0.42	0.63 ± 0.50
Unlock Local	0.01 ± 0.09	0.98 ± 0.01	0.98 ± 0.01	1.00 ± 0.01
Unlock Pickup	0.00 ± 0.00	0.75 ± 0.04	0.76 ± 0.03	1.01 ± 0.04
Unlock To Unlock	0.00 ± 0.00	0.96 ± 0.00	0.86 ± 0.29	0.89 ± 0.30

Table 4.3: Comparison of performance scores across tasks on Meta-World. The table presents the episodic return (score) achieved by a random agent (averaged over 1,000 episodes), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT	JAT (normalized)
Assembly	45.3 ± 4.1	246.0 ± 3.5	238.3 ± 33.0	0.96 ± 0.16
Basketball	2.8 ± 1.2	628.0 ± 2.0	1.6 ± 0.4	-0.00 ± 0.00
BinPicking	1.9 ± 0.4	425.6 ± 101.9	374.2 ± 168.2	0.88 ± 0.40
Box Close	76.4 ± 17.9	512.5 ± 107.8	510.1 ± 117.5	0.99 ± 0.27
Button Press	31.7 ± 5.2	643.1 ± 12.8	556.8 ± 198.9	0.86 ± 0.33
Button Press Topdown	29.0 ± 10.4	490.2 ± 27.2	265.2 ± 77.9	0.51 ± 0.17
Button Press Topdown Wall	29.0 ± 10.5	497.2 ± 31.4	260.1 ± 67.7	0.49 ± 0.14
Button Press Wall	9.0 ± 4.0	675.4 ± 15.0	621.7 ± 137.1	0.92 ± 0.21
Coffee Button	31.7 ± 6.4	731.1 ± 29.3	250.5 ± 266.9	0.31 ± 0.38
Coffee Pull	4.1 ± 0.4	259.9 ± 88.5	55.1 ± 97.0	0.20 ± 0.38
Coffee Push	4.2 ± 0.8	496.8 ± 118.2	269.2 ± 237.8	0.54 ± 0.48
Dial Turn	29.6 ± 16.7	793.6 ± 80.1	738.2 ± 168.4	0.93 ± 0.22
Disassemble	40.3 ± 7.5	42.8 ± 6.3	39.1 ± 11.9	-0.47 ± 4.70
Door Close	5.3 ± 1.3	529.7 ± 27.2	528.2 ± 29.9	1.00 ± 0.06
Door Lock	112.3 ± 28.6	811.5 ± 34.1	676.5 ± 192.7	0.81 ± 0.28
Door Open	56.4 ± 11.2	581.9 ± 19.7	572.8 ± 57.5	0.98 ± 0.11
Door Unlock	94.2 ± 15.6	802.9 ± 17.1	654.9 ± 260.6	0.79 ± 0.37
Drawer Close	116.7 ± 253.1	867.9 ± 4.5	663.0 ± 214.5	0.73 ± 0.29
Drawer Open	126.8 ± 25.2	493.0 ± 2.5	489.1 ± 21.3	0.99 ± 0.06
Faucet Close	253.1 ± 22.9	753.9 ± 13.4	361.3 ± 72.3	0.22 ± 0.14
Faucet Open	244.1 ± 23.3	705.8 ± 7.1	637.9 ± 134.5	0.85 ± 0.29
Hammer	95.3 ± 9.0	693.2 ± 34.6	691.7 ± 25.3	1.00 ± 0.04
Hand Insert	2.8 ± 3.5	740.5 ± 36.7	719.6 ± 99.3	0.97 ± 0.13
Handle Press	80.4 ± 110.2	855.9 ± 72.7	731.9 ± 261.9	0.84 ± 0.34
Handle Press Side	57.0 ± 39.5	861.1 ± 20.0	84.3 ± 113.3	0.03 ± 0.14
Handle Pull	10.3 ± 13.5	669.4 ± 24.8	501.3 ± 209.5	0.74 ± 0.32
Handle Pull Side	2.1 ± 2.8	384.7 ± 102.9	233.1 ± 199.5	0.60 ± 0.52
Lever Pull	60.3 ± 15.8	612.0 ± 38.9	250.2 ± 228.6	0.34 ± 0.41
Pec Insert Side	1.7 ± 0.4	315.2 ± 140.1	288.0 ± 157.9	0.91 ± 0.50
Pec Unpluc Side	4.7 ± 2.8	456.1 ± 81.7	68.5 ± 125.3	0.14 ± 0.28
Pick Out Of Hole	1.5 ± 0.2	219.6 ± 88.9	2.1 ± 0.1	0.00 ± 0.00
Pick Place	1.6 ± 1.0	419.1 ± 98.2	264.2 ± 195.7	0.63 ± 0.47
Pick Place Wall	0.0 ± 0.0	450.6 ± 64.1	6.9 ± 45.0	0.02 ± 0.10
Plate Slide	74.6 ± 13.8	527.0 ± 155.3	497.4 ± 168.7	0.93 ± 0.37
Plate Slide Back	33.5 ± 11.2	718.2 ± 87.4	196.8 ± 1.7	0.24 ± 0.00
Plate Slide Back Side	34.3 ± 11.5	729.6 ± 69.1	697.5 ± 137.8	0.95 ± 0.20
Plate Slide Side	22.6 ± 17.4	662.8 ± 102.8	122.6 ± 24.5	0.16 ± 0.04
Push	5.5 ± 2.4	750.6 ± 44.0	604.2 ± 261.9	0.80 ± 0.35
Push Back	1.2 ± 0.2	85.0 ± 107.1	91.4 ± 115.0	1.08 ± 1.37
Push Wall	6.1 ± 3.2	748.9 ± 10.6	116.5 ± 208.1	0.15 ± 0.28
Reach	149.7 ± 44.7	681.4 ± 133.7	325.3 ± 159.2	0.33 ± 0.30
Reach Wall	143.3 ± 36.6	746.1 ± 104.2	634.6 ± 231.4	0.81 ± 0.38
Shelf Place	0.0 ± 0.0	241.3 ± 24.6	124.6 ± 112.8	0.52 ± 0.47
Soccer	5.7 ± 4.6	375.2 ± 140.2	364.5 ± 175.4	0.97 ± 0.47
Stick Pull	2.6 ± 1.4	523.6 ± 18.9	398.6 ± 205.6	0.76 ± 0.39
Stick Push	2.8 ± 1.0	627.9 ± 10.2	158.3 ± 264.6	0.25 ± 0.42
Sweep	11.2 ± 7.3	494.8 ± 43.3	15.6 ± 9.3	0.01 ± 0.02
Sweep Into	12.5 ± 10.7	799.2 ± 19.1	775.3 ± 119.0	0.97 ± 0.15
Window Close	57.5 ± 7.1	591.3 ± 38.6	423.3 ± 203.9	0.69 ± 0.38
Window Open	43.4 ± 2.1	590.8 ± 57.1	593.1 ± 54.8	1.00 ± 0.10

Table 4.4: Comparison of performance scores across tasks on MuJoCo. The table presents the episodic return (score) achieved by a random agent (averaged over 1,000 episodes), scores of the expert agent (as averaged from the dataset), scores of the learned agent, and the expert normalized score calculated as $\frac{\text{score}-\text{random\_score}}{\text{expert\_score}-\text{random\_score}}$.

Task	Random agent	Expert	JAT	JAT (normalized)
Ant	-59.9 ± 99.6	5846.4 ± 942.6	5268.0 ± 1495.4	0.90 ± 0.25
Inv. Double Pendulum	57.5 ± 17.5	9338.7 ± 352.6	4750.1 ± 931.2	0.51 ± 0.10
Half Cheetah	-285.0 ± 79.8	7437.8 ± 173.3	6659.7 ± 409.7	0.90 ± 0.05
Hopper	18.4 ± 17.1	1858.7 ± 534.1	1835.9 ± 532.2	0.99 ± 0.29
Humanoid	122.0 ± 35.3	6281.0 ± 1795.8	697.4 ± 108.1	0.09 ± 0.02
Inverted Pendulum	6.1 ± 3.5	475.4 ± 179.0	116.3 ± 20.2	0.23 ± 0.04
Pusher	-149.7 ± 7.4	-25.2 ± 6.7	-26.3 ± 6.3	0.99 ± 0.05
Reacher	-43.0 ± 3.9	-5.7 ± 2.5	-6.1 ± 2.6	0.99 ± 0.07
Humanoid Standup	33135.8 ± 2481.9	273574.2 ± 85253.3	118125.2 ± 24880.3	0.35 ± 0.10
Swimmer	0.8 ± 10.7	92.2 ± 4.4	93.3 ± 3.8	1.01 ± 0.04
Walker 2d	2.7 ± 6.1	4631.2 ± 1059.0	4662.4 ± 762.7	1.01 ± 0.16

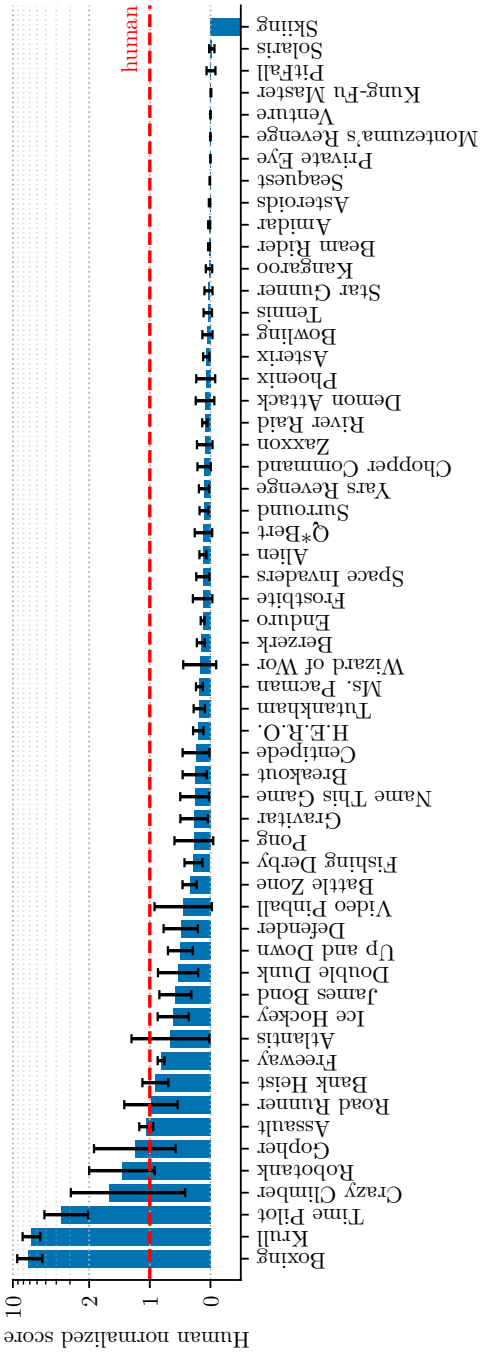


Figure 4.9: Human normalized scores for the JAT agent on the Atari 57 benchmark.

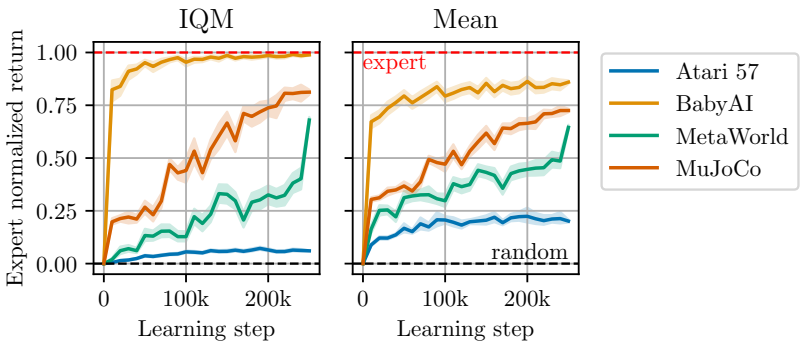


Figure 4.10: Aggregated expert normalized scores with 95% CIs for each RL domain as a function of learning step.

The final agent achieves a mean expert normalized IQM score of 63.5%, demonstrating the network's ability to effectively mimic expert agents across a wide range of tasks. The agent achieves 6.1% of the expert's score on the Atari 57 benchmark, corresponding to 16.8% of human performance, and exceeding the average human level in 10 games. For the BabyAI benchmark, JAT achieved a normalized score of 98.9%. However, this score falls below 50% for two tasks, namely Boss Level No Unlock and Move Two Across S8N9. For this benchmark, however, there is no guarantee that the expert score can be achieved, since the bot used for dataset collection has access to the full state of the environment, while the interacting agents only have access to a partial observation. Finally, in the MuJoCo and Meta-World, JAT records scores of 81.2% and 68.0%, respectively. Although JAT reaches the expert level for a fair number of Meta-World tasks, we note that a few, such as Basketball, fail completely to be learned. Insofar as the action and observation spaces are identical for all tasks in this benchmark, these failures may be due to task indeterminacy, which we explore in more detail in Section 4.4.4. Future research will have to confirm this hypothesis. We also note that some domains are mastered more quickly than others; in particular, BabyAI achieves a score of 90% after only 30,000 learning steps. We hypothesize that the high semantic similarity of the tasks enables a strong positive transfer, without however providing any proof of this.

Although the results achieved are commendable, for a fair comparison we limit our benchmarking to Gato only, as it is the only truly comparable baseline. Reed et al. [224] present results only for the 1.18 billion parameter version of Gato, which is 6 times larger than JAT. Its results are normalized to expert performance. Since we don't have access to the normalization parameters, we estimated scores for the random agents, which may not be exactly the same as those used by Reed et al. [224], and used our expert scores for normalization, even though they obviously do not match those used by Reed et al. [224]. Therefore, comparisons of these normalized scores should be interpreted with great caution. On the Atari benchmark, JAT achieves an average normalized score of 20.1% approaching Gato, which reports a score of 30.9%. For BabyAI, JAT achieves an average normalized score of 86.0%, also close to the Gato score of 93.2%. Our study, however, is made with 39 tasks versus the 46 used in [224], with the specific seven additional tasks in their study remaining unidentified. Since our

evaluation includes all of the hardest tasks mentioned in their study, the seven missing tasks are likely to be easier, suggesting a harder test scenario in our study. For Meta-World, JAT achieves an average normalized score of 64.5%, which is below the 87.0% reported for Gato. On the MuJoCo benchmark, JAT achieves an average normalized score of 72.5%. While [224] doesn't specifically use MuJoCo, it's noteworthy that they use the DMC [265] and Modular RL [120] benchmarks, which have similarities to MuJoCo. For reference, Gato achieves average scores of 63.6% and 62.9% on the DMC and Modular RL benchmarks, respectively.

4.4.3 Predicting the observations does help

The model's main task is to predict the actions that maximize the sum of future rewards. Its ability to predict future observations is therefore not the main concern. However, can this ability contribute to better prediction of actions or accelerate the learning process? Two contrasting hypotheses emerge: firstly, learning to predict observations could serve as an auxiliary objective, directing the learning process towards a deeper understanding of the environment, which could lead to improved and faster learning. Conversely, this prediction learning could serve as a distracting objective: instead of excelling in action prediction, the model might only achieve moderate performance in both action and observation prediction. This could slow down the learning process, resulting in a lower overall performance score. Reed et al. [224] choose not to predict the observation, but does not study the influence of this prediction on learning.

To answer this question, we use a loss function that combines observation loss ($\mathcal{L}_{\text{obs}}$) and action loss ($\mathcal{L}_{\text{act}}$), balanced by a weighting parameter κ. The function is defined as:

$$\mathcal{L} = \kappa \cdot \mathcal{L}_{\text{obs}} + (1 - \kappa) \cdot \mathcal{L}_{\text{act}} \tag{4.1}$$

We select a range of values for κ and train the model on a subset of 6 dataset tasks from different domains (Freeway, Pong, ButtonPressWall, WindowClose, Ant and DoubleInvertedPendulum). Figure 4.11 shows the evolution of the score at different learning steps for different values of κ. Figure 4.12 compares the results at the end of training for the different values of κ.

In our study of the κ coefficient and its impact on learning, we find an interesting balance. When set to the highest value in our range ($\kappa = 0.5$), the learning process seems to be somewhat hindered by the additional objective. On the other hand, at lower κ values, this added task of predicting observations doesn't significantly impact learning, leading to scores that are similar to the base score of $94.5 \pm 1.1\%$, which we get when predicting observations isn't part of the objective. The sweet spot appears to be around $\kappa = 0.005$. At this point, learning to predict observations doesn't distract but actually improves the agent's learning efficiency, achieving an near-optimal score of $99.1 \pm 0.4\%$. This finding highlights that adding observation prediction into the learning process is beneficial, provided it is balanced correctly.

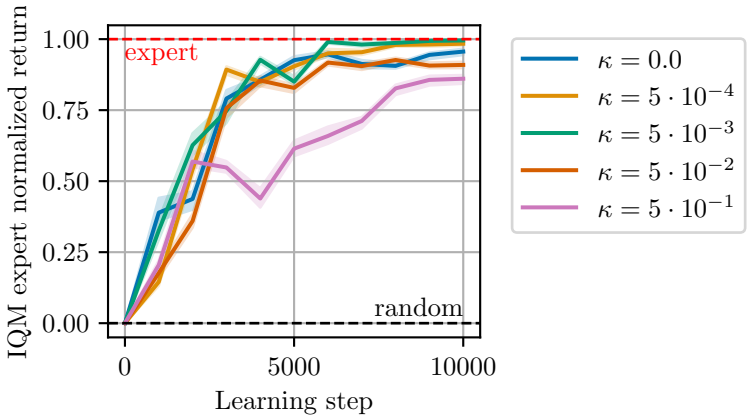


Figure 4.11: Stratified bootstrapped IQM with 95% CIs of expert normalized episodic return as a function of learning steps. This graph displays the progression of learning efficiency over various learning stages with different κ values. It demonstrates that selecting an optimal κ value enables the agent to reach near-optimal scores more rapidly.

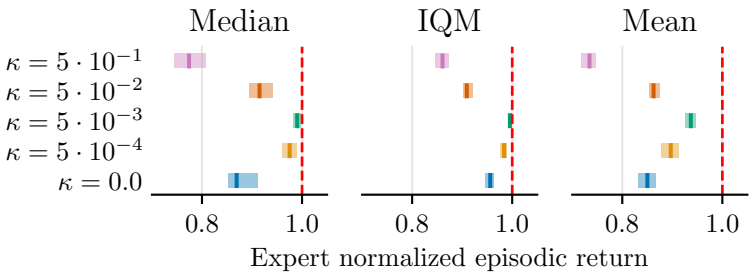
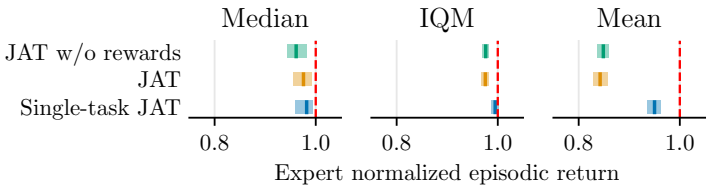


Figure 4.12: Aggregate measures with 95% CIs for the study on the influence of observation prediction learning for selected tasks. The results presented cover the selected range of κ values. These results are based on 100 evaluations per task. The figure shows that a well-chosen κ value can significantly improve the agent's evaluation results.

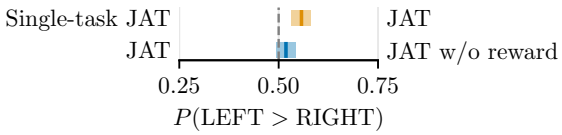
4.4.4 Reward as a task determinant

In multi-task learning, different environments may share identical dynamics and observational structures while differing in their ultimate goals (i.e., reward functions). Initially, the agent cannot distinguish the specific task it is facing. In most cases, this problem does not arise. For BabyAI, for example, the goal is an explicit part of the observation. For Atari, a single frame is sufficient to determine the game, and therefore the goal. In our dataset, the only domain that could be challenging in this respect is Meta-World, for which the structure of observations and dynamics is consistent across tasks. Note also that even in this case, it should be possible for the agent in some instances to infer the task from the initial conditions. We confirm this hypothesis in the following experiment.

To solve the problem of task indeterminacy, Gato introduces a method of pre-empting the sequence with an expert demonstration (prompt) to guide the agent. While this approach is effective, it imposes an important limitation: a demonstration must be available, and this demonstration must be sufficiently complete to clearly define the task. In the JAT model, we adopt a less restrictive and simpler approach by incorporating the reward signal directly into the observation encoding. We believe that this integration can, in most cases, provide the agent with sufficient context to remove ambiguity about the task at hand.



(a) Aggregated metrics.



(b) Probability of improvement.

Figure 4.13: Results of the reward ablation. The vertical bars are the estimated values and the shaded areas are the 95% stratified bootstrap CIs. The experiments were conducted on a selection of 10 tasks from the Meta-World benchmark. Displayed are the results of an ablation study on our JAT model variations: Single-task JAT with each task learned by a dedicated agent; JAT without rewards where the training omits reward signals; and the full JAT model integrating reward signals. Results are based on 100 evaluations per task.

To support our hypothesis on the effectiveness of integrating reward signals into observations, we conducted an experiment with three different settings. First, to create a baseline where task indeterminacy is absent, we trained individual agents, each on a specific task from a random subset of 10 Meta-World tasks. This

single-task training ensures that each agent is perfectly matched to its respective task, without any ambiguity. Next, we introduced a degree of indeterminacy by training a single model on the same 10 tasks without access to the reward signal, presenting a scenario that simulates a worst-case uncertainty condition. We compare these two settings with our full JAT model, i.e. with access to the reward signal, trained on the same selection of tasks. We compared the performance of these three scenarios, with the results detailed in Figures 4.13 following the recommendations of Agarwal et al. [3].

Firstly, it is notable that the JAT model trained on a single task surpasses other settings, thus demonstrating the existence of a negative impact of task indeterminacy. However, this impact is actually very minor, and even in the most unfavorable setting (JAT without reward), the normalized IQM score reaches $97.6 \pm 0.7\%$. This confirms the previously formulated intuition that the task can generally be inferred from the initial conditions. Then, when comparing the JAT model with and without access to the reward, we observe a probability of improvement from the former over the latter of $51.8 \pm 2.5\%$, indicating that the addition of the reward has a significant, albeit small, positive effect on resolving indeterminacy. Lastly, the most significant gap is observed in the average score. This can be attributed to the fact that this metric accounts for outliers. Here, the outliers are the tasks suffering from indeterminacy, for which the agent often fails to resolve the task.

In summary, the key insight from this study is that complex solutions like prompting are often not required to address this task indetermination issue, as it typically presents a minimal challenge. Furthermore, in instances where the problem does manifest, implementing a straightforward strategy like incorporating the reward into the observation proves to be an effective measure for mitigation.

4.5 Conclusion

In this study, we introduce JAT, a novel multimodal framework for general RL agents. JAT features the ability to handle diverse tasks of varying complexity using a single set of parameters. Its innovations include a new transformer-based structure that efficiently addresses sequential control, CV, and NLP tasks. We also show that joint learning of observation prediction significantly improves performance in sequential control tasks. We have open-sourced our training dataset, which includes a wide range of sequential control data as well as extensive language and visual data. We believe that JAT represents an important and valuable step towards general-purpose RL models.

This study reveals several avenues for improvement. A primary challenge is the joint learning of tasks characterized by high heterogeneity. Our dataset features variations in size, task complexity, and accuracy requirements for optimal performance. Our current approach, which uses basic sample and loss weighting, partially addresses this challenge. A refinement of task sampling could potentially account for task difficulty, although quantifying *difficulty* remains a challenge. Another important challenge is imitation learning. While our current method relies on rudimentary behavioral cloning, the use of more advanced imitation learning techniques is likely to yield better results. In addition, improving the quality of expert data is a clear opportunity for improvement. For example, in

the Asterix task, our model's expert score (3699.6) lags significantly behind the scores achieved by agents such as R2D2 (999,153.3) [131]. Using the best RL agent for each specific task could significantly improve the overall scores in our dataset, leading to better results when distilled for the generalist agent.

Chapter 5

Tools for Continuity in RL: Advancing Research Through Established Foundations

In this chapter, we introduce two contributions that complement the previously introduced research: PandaGym and Open RL Benchmark. These two contributions have the particularity of addressing methodological and technical issues, with the aim of enabling more open and efficient research. PandaGym offers a suite of simulated environments adapted to robotic learning in a sparse reward situation. Open RL Benchmark presents a complete set of tracked RL experiments, promoting transparency and reproducibility in RL research. The unique aspect of Open RL Benchmark that we would like to highlight is that it is a shared initiative that has benefited from the contribution of researchers and practitioners from all over the world, and more broadly reflects the philosophy of open research. Together, these tools reflect our commitment to continuous, collaborative advancement in RL research, building on existing frameworks and knowledge. The following sections explore PandaGym and Open RL Benchmark individually.

5.1 PandaGym: Open-Source Goal-Conditioned Environments for Robotic Learning

This section introduces PandaGym, a suite of RL environments that simulates the Franka Emika Panda robot, integrated with OpenAI Gym [29]. PandaGym contains six tasks: Reach, Push, Slide, Pick and Place, Stack, and Flip. Each task offers two types of rewards—sparse and dense—and two control options: end-effector displacement and joint control. This configuration results in a total of 24 distinct environments. They all follow a multi-goal RL framework, allowing to use goal-oriented RL algorithms. To foster open-research, we chose to use the open-source physics engine PyBullet [54]. The implementation chosen for this package allows to define very easily new tasks or new robots. This section also presents a baseline of results obtained with state-of-the-art model-free off-policy algorithms. PandaGym is open-source and freely available at

`https://github.com/qgallouedec/panda-gym`. In 2021, PandaGym was featured in a scholarly publication [91]. Our approach in the initial subections is to present PandaGym as it was first introduced, followed by Subection 5.1.5 that chronicles its subsequent usage and development. This structure offers a unique perspective on a project that isn't merely a static endeavor but rather one with its own evolving narrative, reflecting our view of research as a dynamic and ongoing process.

5.1.1 Introduction

Recent advances in RL applied to robotics have enabled to learn complex manipulation tasks. Nevertheless, current algorithms still struggle to solve tasks for which rewards are very sparse. Recent algorithms have contributed to the advancement in this area, but the number of interactions required to learn a satisfactory model is still very high. For the moment, learning complex tasks with sparse reward functions requires learning in simulation. A large number of physics simulator exists for various applications [52]. For robotic manipulation, the Panda robot arm from Franka Emika is widely used. For this reason, we propose a simulated environment of this robot arm for common tasks used to evaluate RL algorithm. Contrary to what the name of the package suggests, it is possible to easily define new robots, which can be used directly with the tasks already available.

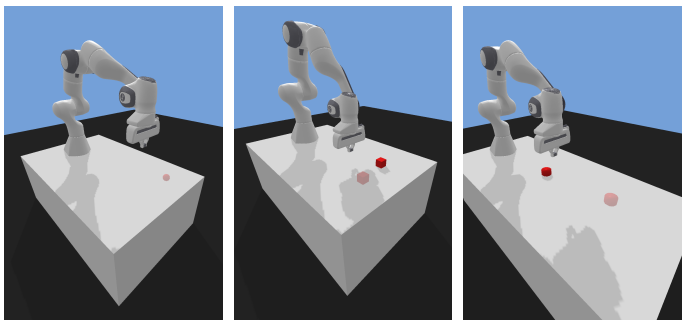
5.1.2 Environments

The environments presented consist of a Panda robotic arm from Franka Emika[1], already widely used in simulation as well as in real in many academic works. It has 7 degrees of freedom and a parallel finger gripper. The robot is simulated with the PyBullet physics engine [54], which has the advantage of being open-source and shows very good simulation performance. The environments are integrated with OpenAI Gym [29], allowing the use of all learning algorithms based on this API. All the tasks presented by Plappert et al. [212] and Andrychowicz et al. [9] have their equivalent in this package. We have also added a Stack task, which is harder to solve than the other tasks, since two objects must be moved (instead of one for the Pick and Place task). The proposed environments all follow the multi-goal RL framework [212]. At each episode, a new goal is randomly generated. The type of this goal depends on the task. For example, for the Reach task, it is the position to be reached with the end-effector which is randomly generated. The observation is therefore augmented with two additional vectors: the *desired goal*, and the *achieved goal*.

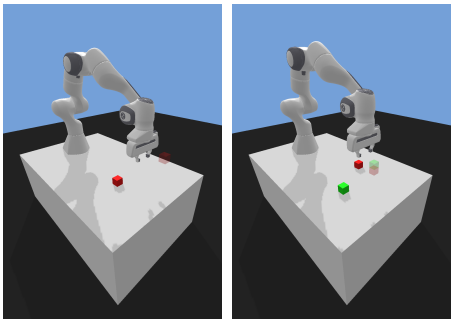
Tasks

A task consists in moving either the gripper or one (or more) object(s) to a target position. A task is completed when the distance between the entity to move and the target position is less than 5 cm. The tasks have an increasing level of difficulty. For each task, a rendering is presented in the Figure 5.1.

[1] `https://www.franka.de/`



(a) Reach (b) Push (c) Slide



(d) Pick and Place (e) Stack

Figure 5.1: Visual rendering of the PandaGym environments. Target positions are shaded (red and green).

`PandaReach-v1` A target position must be reached with the gripper. This target position is randomly generated in a volume of 30 cm × 30 cm × 30 cm.

`PandaPush-v1` A cube, placed on a table, must be pushed to a target position also on the table surface. The gripper is blocked closed. The target position and the initial position of the cube are randomly generated in a 30 cm × 30 cm square around the neutral position of the robot.

`PandaSlide-v1` A flat cylinder (like an ice hockey puck) must be moved to a target position on the surface of a table. The gripper is blocked closed. The target position is randomly generated in a 50 cm × 50 cm square located out of reach of the robot, in front of the neutral position. Thus, is necessary to give an impulse to the object, instead of just pushing it.

`PandaPickAndPlace-v1` A cube must be brought to a target position generated in a volume of 30 cm × 30 cm × 20 cm above the table.

`PandaStack-v1` Two cubes must be stacked at a target position on the table surface. The target position is generated in a square of 30 cm × 30 cm. The stacking must be done in the correct order: the red cube must be under the green cube.

Table 5.1: Detailed breakdown of observation spaces for PandaGym tasks.

	Robot		Task	
	End-effector	Gripper opening	Object 1	Object 2
Size	6	1	12	12
Reach	✓			
Push	✓		✓	
Slide	✓		✓	
Pick and Place	✓	✓	✓	
Stack	✓	✓	✓	✓

Table 5.2: Detailed breakdown of action spaces for PandaGym tasks.

	End-effector displacement	Finger control
Size	3	1
Reach	✓	
Push	✓	
Slide	✓	
Pick and Place	✓	✓
Stack	✓	✓

Observation and action space

The observation space varies depending on the task. For all tasks, the observation contains the position and speed of the gripper (6 coordinates). The control of the gripper does not allow to change its orientation. Its state is therefore completely determined by these 6 coordinates. If the task involves one or more objects, the observation space contains the position, the orientation, the linear and rotational speed (12 coordinates) for each object. When the gripper is not constrained to be closed, the opening of the gripper (i.e. the distance between the fingers) is part of the observation space (1 coordinate). Table 5.1 summarises the observation spaces for the different tasks.

The action space is composed of the end-effector displacement command (3 coordinates, one for each axis of movement x, y and z) and the fingers movement (1 coordinate, corresponding to the variation of the gripper opening). For some tasks, the gripper is blocked closed. For these tasks, the action space is only composed of the end-effector displacement control. Table 5.1 summarises the observation spaces for the different tasks.

After each agent action, the simulator runs 20 timesteps before returning control to the agent and waiting for the next action. On the other hand, one simulator timestep corresponds to 2 ms. The interaction frequency is therefore 25 Hz. An episode consists of 50 interactions, so the duration of an episode is 2 seconds (for the Stack task, an episode lasts 100 interactions, so 4 seconds). These durations are empirically sufficient for the realisation of the corresponding tasks.

Reward

By default, the reward is *sparse*: a reward of 0 is obtained if the entity to move is at the desired position (with a tolerance of 5 cm), and -1 otherwise. For each environment, a variant exists in which the reward is *dense*: this reward is the opposite of the distance between the entity to move and the desired position[2].

In general, a sparse reward function is easier to define, since it is only a question of assessing whether the task is completed in the current state. Conversely, defining a dense reward function can be a tricky process, especially when the task implies several completion criteria. For example, for a task consisting in moving and rotating a cube[3], defining a dense reward function requires to assign a weight of preference to each criterion [89, 193]. These preferences constitute additional hyperparameters.

5.1.3 Design decisions

A robotic environment consists of a robotic arm and a task. Conceptually, a robotic arm can perform different tasks. Similarly, a task can be performed by different robots. To allow for this flexibility, we have separated the task class from the robot class. This allows to easily define a new task without worrying about the robot that will execute it. In the same way, it is possible to define a new robot without worrying about the task to be executed. Figure 5.2 shows the chosen implementation.

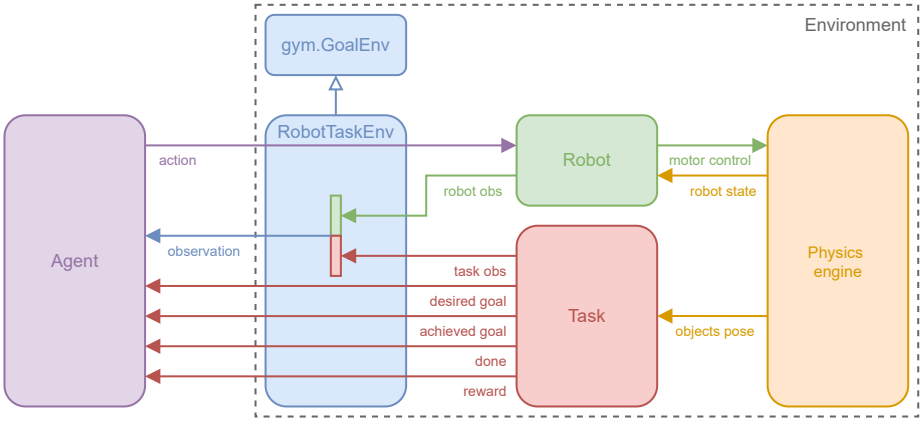


Figure 5.2: PandaGym source code design. The task and the robot are separate, which allows them to be modular. The agent's actions are sent to the robot.

The main class, named `RobotTaskEnv` contains a `robot` attribute, and a `task` attribute. When the agent takes an action, it is transferred directly to the robot. The collected observation is the concatenation of the observations specific to the robot and the observations specific to the task (see Table 5.1). Finally, to follow the Multi-Goal framework, the desired goal and the achieved goal are derived from the task attribute.

[2]For the Stack task, the reward is $-\sqrt{{d_1}^2 + {d_2}^2}$, where d_i is the distance between the object i and its desired position.

[3]This task has been implemented in the next version of the package, see Section 5.1.5

The proposed environments allow fast learning, even on a computer with limited computing capacity. The PyBullet physics engine allows the parallel simulation of several scenes. Thus, the environments are compatible with learning methods that use multiple CPU cores. Tests show that the environments are on average 9.2% faster than their equivalents developed on MuJoCo[4] [268].

5.1.4 Experimental results

The length of a trajectory is 50 timesteps, except for the Stack task, for which we chose a length of 100 timesteps due to its higher complexity. At the end of each trajectory, the environment is reset and a new goal is randomly generated. Learning is distributed over 8 CPU cores. Each core generates trajectories and all these trajectories are stored in a common replay buffer. The result is the success rate, which is evaluated over 80 test episodes, periodically throughout the learning process.

Main results

We give a baseline of the results we obtain for three off-policy algorithms from the recent literature: DDPG [164], SAC [105] and TD3 [86]. We integrate HER [9] to all these algorithms, which has been shown to dramatically increase results on similar robotic tasks. The implementation of DDPG used for the training is the one proposed by Dhariwal et al. [64]. The appropriate modifications have been made to DDPG to implement TD3 and SAC[5]. The hyperparameters are available in Table 5.3. They have been chosen identical to those used in [212]. The learning curves are shown in Figure 5.3. Note that the horizontal axis corresponds to the total number of interactions with the environment. The learning curves are therefore independent of the number of workers used to collect these interactions. Figure 5.4 shows an rendering of the policies at the end of the training for the four task that are solved or partially solved.

The number of timesteps needed to resolve a task depend on the task and the algorithm. For DDPG, the success rate reaches 100% for the Reach task and the Push tasks, after 10^4 and 3×10^4 timesteps, respectively. It reaches about 50% for the Slide and for the Pick and Place tasks, after 6×10^5 and 1.6×10^6 timesteps, respectively. The success rate for the Stack task remains close to 0 after 1.6×10^6 timesteps of training. The presented algorithms do not allow to solve it in this amount of timesteps.

Ablation studies

From the results obtained, we observe that TD3 achieves lower performance than DDPG, despite being its more advanced iteration. We hypothesize that one of the three key modifications in TD3 compared to DDPG might be detrimental in the context of environments with sparse rewards. While two of these modifications - delayed policy updates and target policy smoothing - are designed to stabilize learning, the third addition, clipped double-Q Learning, is intended to limit the overestimation of value estimates. In environments where reward density is low, we speculate that overestimation of value might be minimal or even

[4]We measured the time required to simulate 10^5 timesteps using a single CPU core.

[5]`https://github.com/qgallouedec/baselines`

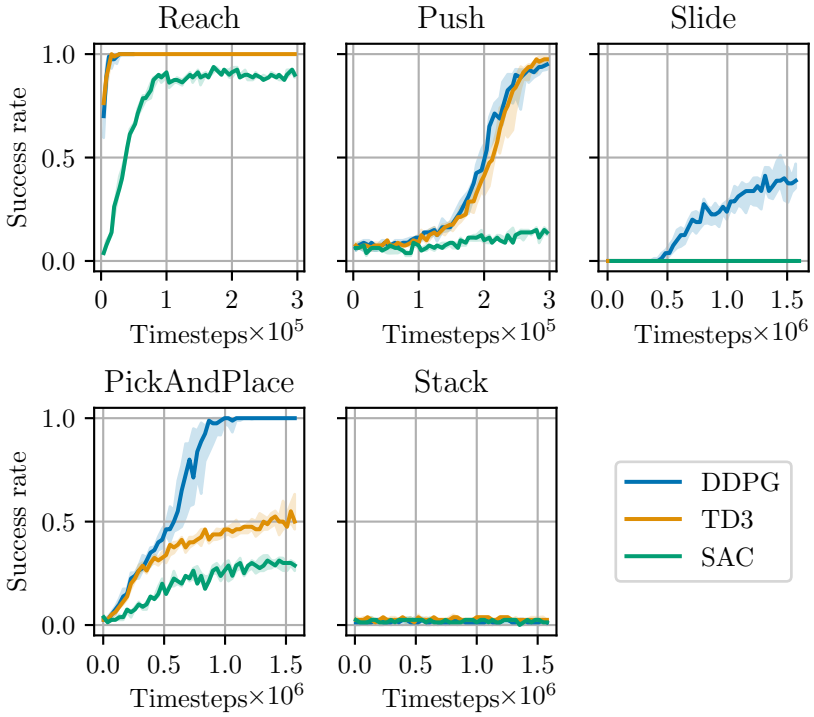


Figure 5.3: Success rates for the five Panda environments. We repeat each experiment with 21 different random seeds. Median rates are solid lines and interquartile range are shaded areas. We represent the results for the DDPG, SAC and TD3 algorithms, all three ran with HER. The horizontal axis corresponds to the total number of interaction with the environment.

Table 5.3: Hyperparameters for PandaGym trainings. Parameters specific to a certain algorithm are noted with the algorithm's name in parentheses.

	Parameter	Value
Agent	Network type	Multi-layer perceptron
	Network size	3 layers of 256 nodes
	Optimizer	Adam [134]
	Learning rate	0.001
	Polyak-averaging [213]	0.95
	L2 normalisation coefficient	1.0
Observation	Clipping	$[-200, 200]$
Action	Clipping	$[-1, 1]$
	Prob. of random (DDPG and TD3)	0.3
	Additive noise type (DDPG and TD3)	Gaussian
	Noise scale (DDPG and TD3)	0.2
Training	Episode length	50 (100 for Stack)
	Testing	Every 80 episodes
	Number of testing episodes	80
	Replay buffer size	10^6 transitions
	Batch size	256
	HER per transition (k)	4
	Policy delay (TD3)	2
	Policy noise (TD3)	0.2
	Policy noise clip (TD3)	$[-0.5, 0.5]$
	α (SAC)	0.2

non-existent. Thus, the clipped double-Q learning could inadvertently lead to an underestimation of value. Without definitive confirmation of this hypothesis, our empirical findings nonetheless suggest that removing this feature significantly improves performance, even surpassing DDPG across all tested environments. Encouraged by these positive results, we further experimented by reversing the trick in TD3. Rather than taking the minimum of the two critics, we opted for the maximum, which resulted in even better performance by the agent. These findings are presented in Figure 5.5. Additionally, we replicate and confirm the results of Andrychowicz et al. [9] by examining the impact of removing HER and demonstrating that its exclusion significantly reduces performance.

5.1.5 PandaGym unleashed: a story of impact and evolution

Developed in 2021 and featured in a publication the same year [91], PandaGym addressed a crucial need in the field of robotics research: the lack of maintained, free, open-source, and well-documented environments that are both challenging and suitable for exploring hard exploration problems, particularly those with sparse rewards. This gap was the primary inspiration behind PandaGym's development.

PandaGym distinguished itself by offering customizable environments. This feature, in particular, has led to widespread adoption and adaptation by re-

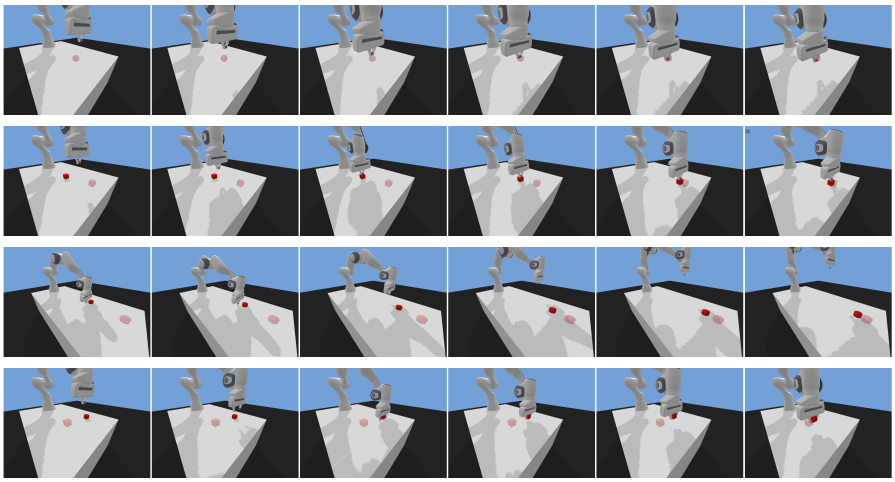


Figure 5.4: Overview of policies at the end of the training. Each line represents a task. From top to bottom: Reach (one timestep between two successive images), Push (two timesteps between two successive images), Slide (four timesteps between two successive images) and Pick and Place (two timesteps between two successive images).

searchers, who have customized their environments to suit specific research needs. The initial goal of PandaGym was to establish a stable set of environments that could serve as benchmarks for algorithms tackling hard exploration challenges. Since its release, PandaGym has undergone significant evolution, now in its third major version, featuring new tasks, improved dynamics, and enhancements inspired by user feedback and contributions. This evolution has been facilitated by an active community of users who have opened issues and contributed to the codebase on GitHub.

The impact of PandaGym on research methodologies and practices has been profound. As a challenging benchmark, it is now widely used within the field, allowing for easy comparison of new algorithms with established baselines, as detailed in Section 5.2. While specific examples of diverse applications will be discussed in the following sections, it is important to note the broad reach and adaptability of PandaGym in various research contexts.

In the following sections, we will delve deeper into various aspects of PandaGym's impact. We will explore how its design and features have facilitated research advancements, look at its evolution through user contributions and feedback, and examine its role as a benchmark in the field. Additionally, we will highlight specific case studies and examples that showcase PandaGym's application in diverse research scenarios. This exploration will not only reflect on the software's success and reach but also provide insights into its future potential and ongoing relevance in robotics research.

Usage statistics and measurable outcomes

PandaGym demonstrates significant impact in RL with over 60,000 downloads, indicating its widespread use. Its GitHub presence, with 86 forks and over 380

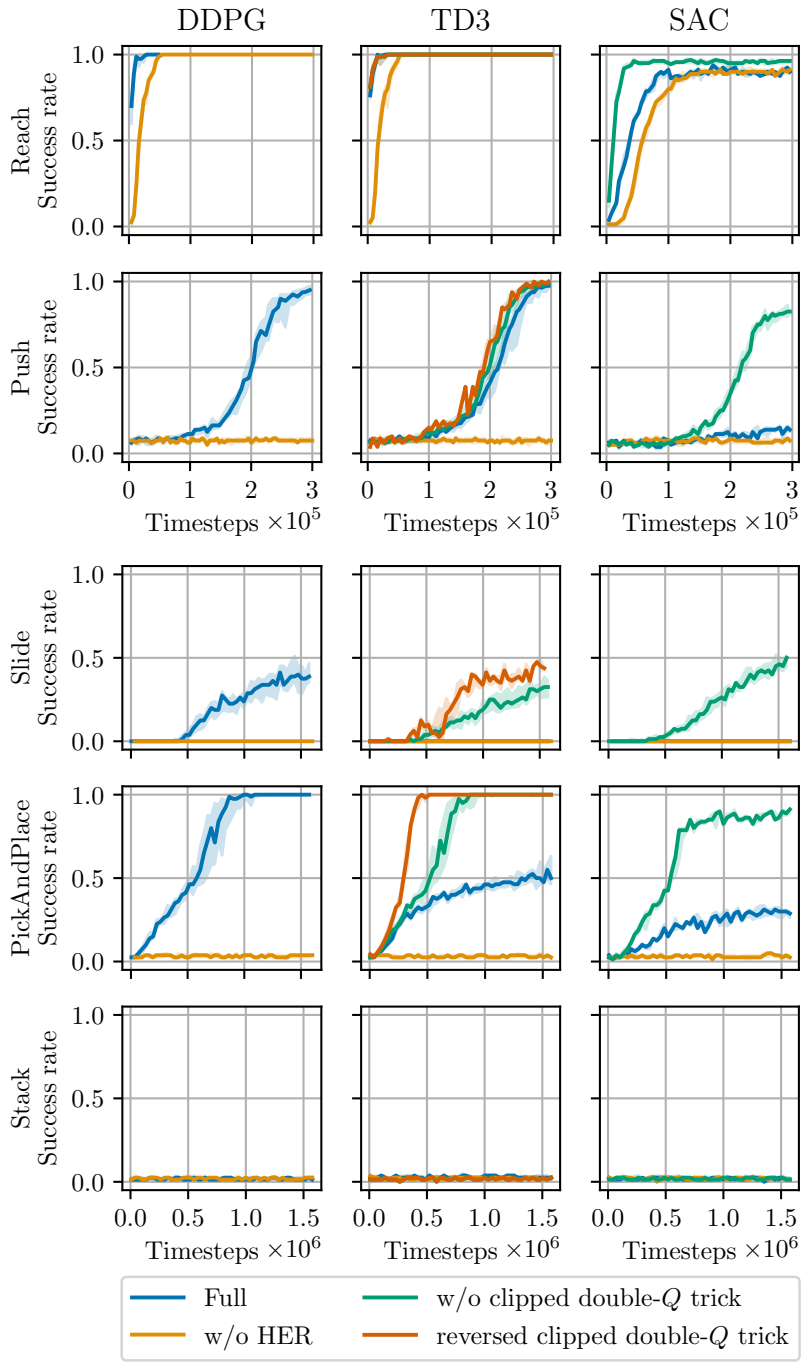


Figure 5.5: Ablation study for HER and for the clipped double-Q trick. We also reverted the clipped double-Q trick. We repeat each experiment with 21 different random seeds. Median success rates are solid lines and interquartile range are shaded areas.

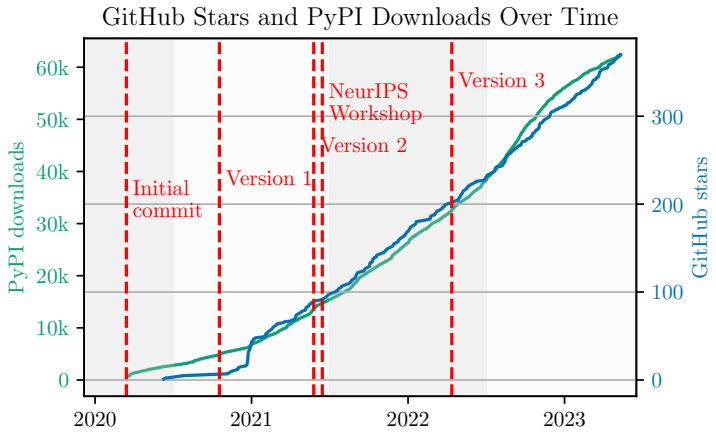


Figure 5.6: Cumulative growth of GitHub stars and PyPI downloads for PandaGym. This graph illustrates the progressive increase in both GitHub stars and PyPI downloads from its early development to the time of writing, highlighting significant development milestones.

stars, reflects robust community engagement. The progressive and still steady increase in both GitHub stars and PyPI downloads, as depicted in Figure 5.6, further illustrates PandaGym's growing influence. This figure also highlights key events in PandaGym's history. The platform's practicality is underscored by over 1,400 pretrained models available on the HuggingFace Hub. In standardized testing, PandaGym is featured in 80 benchmark runs on Open RL Benchmark, illustrating its reliability. Academically, its relevance is affirmed by citations in over 45 scientific papers, cementing its role in advancing RL research.

Usage and case studies

Since its initial release, PandaGym has been widely recognised and cited in numerous research papers as a fundamental benchmark in the RL community [257, 79]. In the following sections, we discuss the various ways in which the academic community has engaged with PandaGym, using it as a versatile benchmark for comparative simulation studies and tailoring it to meet the unique needs of different research questions. As an extension, we would also like to point out that some researchers have identified specific limitations inherent in PandaGym [141], which has led to a number of innovative extensions and adaptations, either building on the original code base or using entirely new methodologies [26, 132].

PandaGym as a Common Benchmark PandaGym has been widely used in numerous studies, largely due to its recognition as a set of benchmark tasks. Researchers have proposed improvements to widely used algorithms in the literature, such as improvements to DDPG [244] and SAC [121], using PandaGym to provide solid validation of their improvements. Others have focused on robustifying existing algorithms [107, 69], with the challenging nature of PandaGym's tasks providing a solid foundation for their experiments. In addition, innova-

tions focused on curriculum learning [248, 58] have highlighted the usefulness of PandaGym in introducing progressively more difficult tasks to improve learning effectiveness. Innovations aimed at improving results with an appropriate number of demonstrations for complex tasks [95, 218, 94] have also been noteworthy. Furthermore, research on pattern efficiency issues has introduced partial learning on low-fidelity clones [247, 246], data augmentation [11], and model-based methods [16]. Finally, PandaGym has found applications in more specific domains such as IL [287], transfer learning [176], and policy fine-tuning [98, 97].

PandaGym for Task-Specific Needs Often chosen for its ease of customization through well-documented and structured code, PandaGym has been tailored by researchers to meet the specific needs of their studies. Some have modified the physical environment of tasks by altering objects [72, 227, 293, 45], while others have introduced elements like wind [45] or made surfaces more slippery [125]. The customization extended to altering goals or reward functions to align with study objectives [66, 133, 53, 128, 90], and direct modifications to the robot itself, such as adding a scraper [211] or a wrist camera [108].

PandaGym in Robotics: The Sim2Real Transition In some cases, PandaGym has been used to simulate the Franka Emika Panda robot, serving as a platform for validating innovations in simulation before deployment on real Franka Emika Panda robots [59]. Notably, successful transfers of policies learned in the PandaGym simulation to real robots have been documented [182, 172].

PandaGym as a Simulation and Visualization Tool An unexpected yet valuable use of PandaGym has been for its visualization capabilities [171] and as a simulation tool. For instance, PandaGym has been instrumental in making teleoperation tools accessible and usable for users without real robots [96].

PandaGym for Education PandaGym has been an invaluable aid to education, with numerous articles, tutorials and courses using it to teach, particularly over the last twelve months. These resources are aimed at a wide audience, from beginners [129, 252, 63] to more advanced learners [42]. Some focus on introducing the environment [63], while others use it to introduce RL topics [129, 252].

Evolutions and community contributions

Since its initial release, PandaGym has undergone several significant developments, each enriched by valuable contributions from the community. One of the most important updates was the introduction of a new, complex task named *Flip*. This new task consist in grasping and rotate a cube to the desired orientation. This new task provides a unique challenge requiring high precision control, particularly due to the need to finely manage surface friction. The development and integration of the Flip task also led to improvements in the simulation's handling of friction, making it more reliable and reflective of real-world physics. Another major improvement to PandaGym was the introduction of a new control mode. In addition to the original simple control mode, which was limited to basic (x, y, z) displacement control and relied on predefined inverse dynamics,

the new control mode introduced the ability to individually manipulate each of the robot's seven joints. This advance expanded the range of possible movements and strategies, adding depth and complexity to any task. As a result, this new control mode was integrated into all tasks, providing a more challenging experience. As a result, PandaGym now offers a diverse range of 24 different environments, categorised into six tasks, two different reward types (dense and sparse) and two control modes. This variety allows for a wide range of difficulty levels and research interests.

In addition to these major updates, the platform has benefited from a number of smaller but powerful enhancements, many of which were user-driven. These include the introduction of state saving and restoration capabilities, which are critical for certain visualisation and methodological needs, and the implementation of robust random seed support, which ensures the reproducibility of results—a critical aspect of scientific research. The active PandaGym community, as shown by the numerous discussions and open questions (over 82 answered in the source code repository), has played a key role in the development of the platform. Feedback from the community has led not only to improvements and clarifications in the documentation, but also to the development of new features and modifications. For example, community suggestions include dual robot environments and the integration of force sensors.

5.1.6 Conclusion

In this section, we have outlined PandaGym, a maintained, open-source set of free simulated robotic environments. The various tasks were introduced, with detailed explanations of their observations, actions, and rewards. We focused on the design choices made to ensure easy customizability, as we aim for PandaGym to be swiftly and effortlessly adapted for individual research needs. This design facilitates the straightforward definition of new tasks and the inclusion of new robots. We have also provided several reference results using state-of-the-art algorithms, illustrating that at the time of its initial release, contemporary algorithms were not able to solve all the tasks presented.

Over its three-year journey, PandaGym has garnered considerable attention and use within the community, particularly in areas it was specifically designed for, such as benchmarking RL control algorithms and as a customizable base for research. We are gratified to see PandaGym widely adopted and utilized, a testament to its relevance and utility, which is further evidenced by various statistics and practical applications. Feedback from the community has led us to reconsider certain enhancements, such as support for image-based learning, due to challenges in implementation and efficiency. Nonetheless, we are encouraged to see alternative platforms like IsaacGym [180] emerging, addressing these gaps and expanding the field's horizons. Despite these limitations, we remain satisfied with the community's engagement with PandaGym and its evolution. The package continues to be actively maintained, evolving through both authorial and community contributions, and we are optimistic about its ongoing evolution to meet the community's evolving needs.

5.2 Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning

In many RL papers, learning curves are useful indicators to measure the effectiveness of RL algorithms. However, the complete raw data of the learning curves are rarely available. As a result, it is usually necessary to reproduce the experiments from scratch, which can be time-consuming and error-prone. We present Open RL Benchmark[6], a set of fully tracked RL experiments, including not only the usual data such as episodic return, but also all algorithm-specific and system metrics. Open RL Benchmark is community-driven: anyone can download, use, and contribute to the data. At the time of writing, more than 25,000 runs have been tracked, for a cumulative duration of more than 8 years. Open RL Benchmark covers a wide range of RL libraries and reference implementations. Special care is taken to ensure that each experiment is precisely reproducible by providing not only the full parameters, but also the versions of the dependencies used to generate it. In addition, Open RL Benchmark comes with a command-line interface[7] (CLI) for easy fetching and generating figures to present the results. In this document, we include two case studies to demonstrate the usefulness of Open RL Benchmark in practice. To the best of our knowledge, Open RL Benchmark is the first RL benchmark of its kind, and the authors hope that it will improve and facilitate the work of researchers in the field.

Open RL Benchmark is intended to be a community-driven collaboration, as evidenced by its initiation and ongoing development. Initiated by Shengyi (Costa) Huang, this work has benefited greatly from numerous contributions. I have made substantial contributions to the benchmark and played a key role in streamlining the community's efforts. It also benefited from the invaluable efforts of Florian Felten for extensive documentation of the CLI, and Mohamad H. Danesh for illustrating the practical capabilities of Open RL Benchmark. To enhance and promote this work, I initiated, organized an serve as the primary author for a paper [118] that will be submitted to the NeurIPS 2024 Datasets and Benchmarks track, with all the 30 contributors listed as authors.

5.2.1 Introduction

RL research is based on comparing new methods to baselines to assess progress. This process implies the availability of the data associated with these baselines or, alternatively, the ability to reproduce them and generate the data oneself. In addition, the ability to reproduce also allows the methods to be compared with new benchmarks and to identify the areas in which the methods excel and those in which they are likely to fail, thus providing avenues for future research.

In practice, the RL research community faces complex challenges in comparing new methods with reference data. The unavailability of reference data requires researchers to reproduce experiments, posing difficulties due to insufficient source code documentation and evolving software dependencies. Implementation intricacies, as highlighted in past research, can significantly impact results [109, 115]. Moreover, limited computing resources play a crucial role, hindering the reproduction process and affecting researchers without substantial access.

[6] `https://wandb.ai/openrlbenchmark`

[7] `https://github.com/openrlbenchmark/openrlbenchmark`

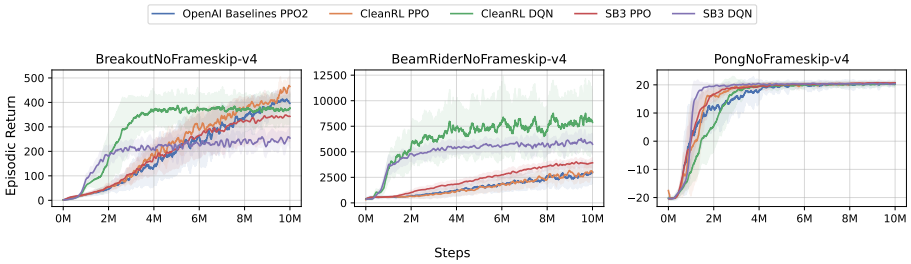


Figure 5.7: Example of learning curves that can be obtained with Open RL Benchmark. These compare the episodic returns achieved by different implementations of PPO and DQN on a number of Atari tasks.

These challenges lead to difficulties in reliably evaluating new methods and hinder efficient comparisons against established ones. Reproducing experiments is a time-consuming and resource-intensive task, or researchers may rely on inconsistently presented paper results. The lack of standardized metrics and benchmarks across studies not only impedes comparison but also results in a substantial waste of time and resources. To address these issues, the RL community must establish rigorous reproducibility standards, ensuring replicability and comparability across studies. Transparent sharing of data, code, and experimental details, along with the adoption of consistent metrics and benchmarks, would collectively enhance the evaluation and progression of RL research, ultimately accelerating advancements in the field.

As a consequence, the RL research community faces significant challenges in reliably assessing the true value and progress of new methods. This difficulty stems largely from the complexities and constraints associated with the reproduction of baseline results. Researchers often face the challenge of reproducing experiments, which can be a time-consuming and resource-intensive task. Alternatively, they may rely on the results presented in papers, which may not consistently use the same metrics or benchmarks. This situation not only hinders the efficient comparison of new methods against established ones but also leads to a considerable waste of time and resources. Furthermore, the uncertainty surrounding the reproducibility of results raises concerns about the reliability of progress claims in the field. To tackle these challenges, the community must establish more rigorous standards for reproducibility, ensuring that results are not only replicable but also comparable across different studies. A concerted effort towards more transparent and comprehensive sharing of data, code, and experimental details, as well as the adoption of consistent metrics and benchmarks across studies, would enhance the collective ability to evaluate and build upon the findings of RL research. This would ultimately accelerate progress in the field.

Open RL Benchmark presents a rich collection of tracked RL experiments and aims to set a new standard by providing a diverse training dataset. This initiative prioritizes the use of existing data over re-running baselines, emphasizing reproducibility and transparency. Our contributions are:

- **Extensive dataset:** Offers a large, diverse collection of tracked RL experiments.

- **Standardization:** Establishes a new norm by encouraging reliance on existing data, reducing the need for re-running baselines.
- **Comprehensive metrics:** Includes diverse tracked metrics for method-specific and system evaluation, in addition to episodic return.
- **Reproducibility:** Emphasizes clear instructions and fixed dependencies, ensuring easy experiment replication.
- **Resource for research:** Serves as a valuable, efficient, and collaborative resource for RL research.
- **Facilitating exploration:** Enables reliable exploration and assessment of new RL methods.

5.2.2 Comprehensive overview of Open RL Benchmark: content, methodology, tools, and applications

This section provides a detailed exploration of the contents of Open RL Benchmark, including its diverse set of libraries and environments, and the metrics it contains. We also look at the practical aspects of using Open RL Benchmark, highlighting its ability to ensure accurate reproducibility and facilitate the creation of data visualizations thanks to its CLI.

Content

Open RL Benchmark data is stored and shared with W&B [25]. They are contained in a common entity named `openrlbenchmark`. Runs are divided into several *projects*. A project can correspond to a library, but it can also correspond to a set of more specific runs, such as `envpool-cleanrl` in which we find CleanRL runs [116] which have the particularity of being launched with the EnvPool implementation [282] of environments. A project can also correspond to a reference implementation, such as TD3 (project `sfujim-TD3`) or Phasic Policy Gradient [48] (project `phasic-policy-gradient`). Open RL Benchmark also includes reports, which are interactive documents designed to enhance the visualization of selected representations. These reports provide a more user-friendly format for practitioners to share, discuss, and analyze experimental results, even across different projects. Figure 5.8 shows a preview of one such report.

At the time of writing, Open RL Benchmark contains nearly 25,000 runs, for a total of 72,000 hours (more than 8 years) of tracking. In the following paragraphs, we present the libraries and environments for which runs are available in Open RL Benchmark, as well as the metrics tracked.

Libraries At the time of writing, Open RL Benchmark contains runs for several reference RL libraries. These libraries are: abcdRL [296], Acme [113], Cleanba [119], CleanRL [116], jaxrl [136], moolib [185], MORL-Baselines [78], OpenAI Baselines [64], rlgames [179] Stable-Baselines3 [222] Stable-Baselines Jax [222] and TorchBeast [144].

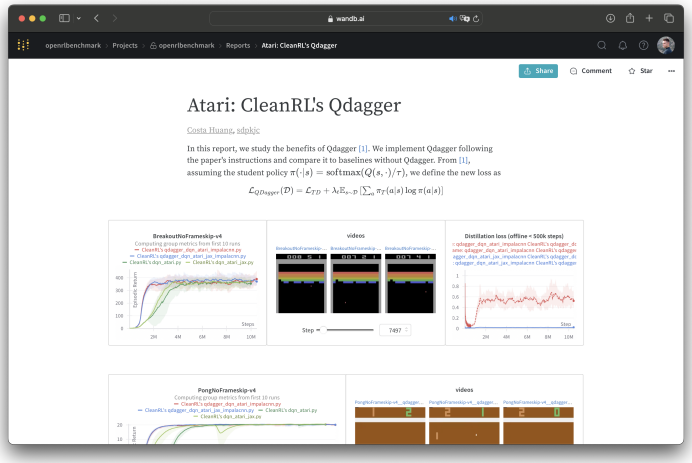


Figure 5.8: An example of a report on the W&B platform, dealing with the contribution of QDagger [4], and using data from Open RL Benchmark. The URL to access the report is `https://wandb.ai/openrlbenchmark/openrlbenchmark/reports/Atari-CleanRL-s-Qdagger--VmlldzoONTg1ODY5`.

Environments The runs contained in Open RL Benchmark cover a wide range of classic environments. They include Atari [21, 177], Classic control [29], Box2d [29] and MuJoCo [268] as part of either Gym [29] or Gymnasium [270] or EnvPool [282]. They also include Bullet [54], Procgen Benchmark [47], Fetch environments [212], PandaGym [91], highway-env [160], Minigrid [44] and MO-Gymnasium [7].

Tracked metrics Metrics are recorded throughout the learning process, consistently linked with a global step indicating the number of interactions with the environment, and an absolute time, which allows for the calculation of the process's relative duration to track elapsed time. We categorize these metrics into four distinct groups:

- **Training-related metrics:** These are general metrics related to RL learning. This category contains, for example, the average returns obtained, the episode length or the number of collected samples per second.
- **Method-specific metrics:** These are losses and measures of key internal values of the methods. For PPO, for example, this category includes the value loss, the policy loss, the entropy or the approximate KL divergence.
- **Evolving configuration parameters:** These are configuration values that change during the learning process. This category includes, for example, the learning rate when there is decay, or the exploration rate (ϵ) in DQN [187].
- **System metrics:** These are metrics related to system components. These could be GPU memory usage, its power consumption, its temperature, system and process memory usage, CPU usage or even network traffic.

The specific metrics available may vary from one library to another. In addition, even where the metrics are technically similar, the terminology or key used to record them may vary from one library to another. Users are advised to consult the documentation specific to each library for precise information on these measures.

Everything you need for perfect repeatability

Reproducing experimental results in computational research, as discussed in Section 5.2.4, is often challenging due to evolving codebases, incomplete hyperparameter listings, version discrepancies, and compatibility issues. Our approach aims to enhance reproducibility by ensuring users can exactly replicate benchmark results. Each experiment includes a complete configuration with all hyperparameters, frozen versions of dependencies, and the exact command, including the necessary random seed, for systematic reproducibility. Furthermore, CleanRL [116] introduces a unique utility that streamlines the process of experiment replication (see Figure 5.9). This utility produces the command lines to set up a Python environment with the necessary dependencies, download the run file, and the precise command required for the experiment reproduction. Such an approach to reproduction facilitates research and makes it possible to study in depth unusual phenomena, or cases of rupture[8], in learning processes, which are generally ignored in the results presented, either because they are deliberately left out or because they are erased by the averaging process.

```
$ python -m cleanrl_utils.reproduce --run openrlbenchmark/cleanrl/runs/c1y1qnz4

# run the following
python3 -m venv venv
source venv/bin/activate
pip install -r https://api.wandb.ai/files/openrlbenchmark/cleanrl/c1y1qnz4/requirements.txt
curl -OL https://api.wandb.ai/files/openrlbenchmark/cleanrl/c1y1qnz4/code/cleanrl/ppo_atari.py
python ppo_atari.py --track --env-id BreakoutNoFrameskip-v4 --seed 3
```

Figure 5.9: CleanRL's module `reproduce` allows the user to generate, from an Open RL Benchmark run reference, the exact command suite for an identical reproduction of the run.

The CLI, for figures in one command line

Open RL Benchmark offers convenient access to raw data from RL libraries on standard environments. It includes a feature for easily extracting and visualizing data in a paper-friendly format, streamlining the process of filtering and extracting relevant runs and metrics for research papers through a single command. The CLI is a powerful tool for generating most metrics-related figures for RL research and notably, all figures in this document were generated using the CLI. The data in Open RL Benchmark can also be accessed by custom scripts, as detailed in Appendix B.2. Specifically, the CLI integrated into Open RL Benchmark provides users with the flexibility to:

[8] Exemplified in `https://github.com/DLR-RM/rl-baselines3-zoo/issues/427`

- Specify algorithms' implementations (from which library) along with their corresponding git commit or tag;
- Choose target environments for analysis;
- Define the metrics of interest;
- Opt for the additional generation of metrics and plots using RLiable [3].

Concrete example usage of the CLI and resulting plots are available in Appendix B.1.

5.2.3 Open RL Benchmark in action: an insight into case studies

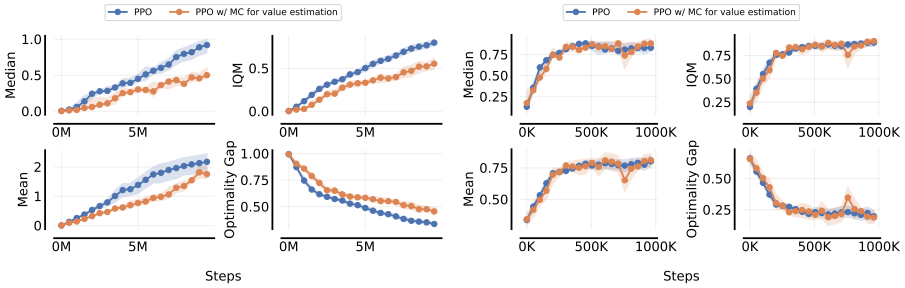
The practical application of Open RL Benchmark is best illustrated through case studies that demonstrate its utility in streamlining the process of comparing and evaluating RL methods. This section delves into two specific case studies that exemplifies the ease and efficiency with which researchers can now propose improvements and benchmark them against existing methodologies using Open RL Benchmark. By leveraging the comprehensive data and metrics available from established RL libraries, these case studies showcases the benchmark's capability to facilitate direct and fair comparisons without the need to recreate existing experiments. This not only saves valuable time and resources but also ensures that comparisons are made on a consistent and reliable basis. First, we propose to investigate the benefit of using TD(λ) for value estimation in PPO [239] versus using MC. This simple study illustrates the use of Open RL Benchmark via a classic research question. Moreover, as far as we know, this question has never yet been studied in the literature. We then present a more unusual approach. We show how Open RL Benchmark is used to demonstrate the speedup and variance reduction of the new IMPALA implementation proposed by Huang et al. [119]. We underline the versatility of Open RL Benchmark, both in the evaluation of classic and more unusual RL approaches. This not only saves valuable time and resources, but also ensures that comparisons are made on a consistent, reliable and fully reproducible basis. All in all, these case studies demonstrate the benchmark's role in providing clear, accessible, and actionable insights, which are crucial for driving forward the field of RL research.

Easily assess the contribution of TD(λ) for value estimation in PPO

In the first case study, we showcase the adaptability and usability of Open RL Benchmark by employing a modification in PPO [239] value function estimation. PPO typically employs GAE [238] to update the actor (see Section 2.4.2) and the target return is computed from the GAE value, by adding the minibatch return [115, point 5]. Empirical studies [207] have highlighted the joint merits of GAE and TD(λ) over MC and n-steps return estimates. Here, we propose a focused investigation into the impact of TD(λ) over MC in estimating the value function, while maintaining GAE for advantage estimation.

The first step is to identify the reference runs in Open RL Benchmark. As PPO is a widely recognized baseline, a large number of runs are available. We chose to use the Stable-Baselines3 runs for this example. We retrieve the

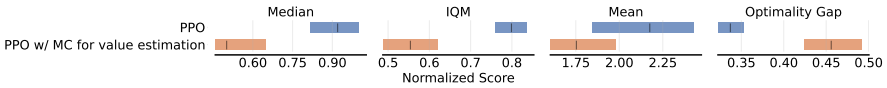
precise source code and command used to generate them, thanks to the pinned dependencies provided in the runs. We apply the appropriate modification to the source code. We select a diverse set of environments, including Atari games (Breakout, Space Invaders, Seaquest, Enduro, Pong, Q*Bert, Beam Rider), Box2D (Lunar Lander), and MuJoCo (Inverted Double Pendulum, Inverted Pendulum, Reacher, Half Cheetah, Hopper, Swimmer, Walker 2d). For each environment selected, we launch 3 learning runs using the same command as the one retrieved and using the modified codebase. The runs are stored in a dedicated project[9]. For fast and user-friendly rendering of the results, we create a W&B report[10]. Using the CLI, we generate Figure 5.10 and 5.11. The command used to generate the figures is given in Appendix C.



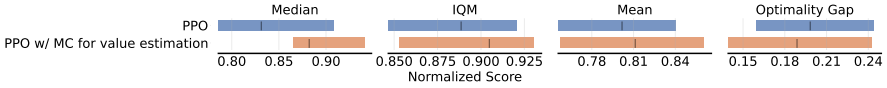
(a) Results for Atari games.

(b) Results for Box2D and MuJoCo environments.

Figure 5.10: Comparing the original PPO and the PPO with MC for value estimation. These experiments were conducted over 15 environments, including Atari games, Box2D, and MuJoCo. The plots represent the episodic return normalized by human performance for Atari and by min-max for other environments. Aggregation follows [3] recommendations, and shaded areas correspond to stratified bootstrap 95% CIs.



(a) Results for Atari games.



(b) Results for MuJoCo and Box2d environments.

Figure 5.11: Comparing the original PPO and the PPO with MC for value estimation. Aggregated minmax normalized final scores with 95% stratified bootstrap CIs.

Figure 5.10 shows the evolution of the aggregated episodic return obtained over the course of learning for the Atari tasks on the one hand, and the Box2d

[9] `https://wandb.ai/modanesh/openrlbenchmark`
[10] `https://api.wandb.ai/links/modanesh/izf4yje4`

and MuJoCo tasks on the other. (This division is chosen on the basis of the proximity of the tasks in each group.) Figure 5.10 shows the final aggregated episodic return using the same division. We find that using the MC estimator for the value gives significantly weaker results for the Atari environment, thus justifying the use of TD(lambda). On the other hand, the difference in results for the other environments is not significant and seems to have virtually no impact. Overall, this case study demonstrates how Open RL Benchmark's can be leveraged for investigating TD(λ) impact on PPO's value function estimation. It streamlines research, reuses computations, ensuring strict reproducibility.

Demonstrating the utility of Open RL Benchmark through the Cleanba case study

This section describes how Open RL Benchmark was instrumental in the evaluation and presentation of Cleanba [119], a new open-source platform for distributed RL implementing highly optimized distributed variants of PPO [239] and IMPALA [74]. Cleanba's authors asserted three points: (1) Cleanba implementations compare favorably with baselines in terms of sample efficiency, (2) for the same system, the Cleanba implementation is more optimized and therefore faster, and (3) the design choices allow a reduction in the variability of results.

To prove these assertions, the evaluation of Cleanba encountered a common problem in RL research: the works that initially proposed these baselines did not provide the raw results of their experiments. Although a reference implementation is available[11], it is no longer maintained. Subsequent works like moolib [185] and TorchBeast [144] have reproduced the IMPALA results, but similarly, the shared results are restricted to the paper's curves, presenting only a smoothed measure of episodic return as a function of interractions step, on a set of Atari tasks, notwithstanding numerous, but not matching to the widespread Atari 57, and for which the raw data that yielded these curves is not available.

Recognizing the lack of raw data for existing IMPALA implementations, the authors reproduced the experiments, tracked the runs and integrated them into Open RL Benchmark. As a reminder, these logged data include not only the return curves, but also the system configurations and temporal data, which are crucial to support the Cleanba authors' optimization claim. Comparable experiments have been run, tracked and shared on Open RL Benchmark with the proposed Cleanba implementation.

Using Open RL Benchmark CLI, the authors generated several figures. The authors have provided the exact commands to reproduce these curves in the source directory. In Figure 5.12, taken from [119], the authors show that the results in terms of sample efficiency compare favourably with the baselines, and that for the same system configuration, convergence was temporally faster with the proposed implementation, thus proving claims (1) and (2). Figure 5.13 demonstrates that Cleanba variants maintain consistent learning curves across different hardware configurations. Conversely, moolib's IMPALA shows marked variability in similar settings, despite identical hyperparameters, affirming the authors' third claim.

[11] `https://github.com/google-deepmind/scalable_agent`

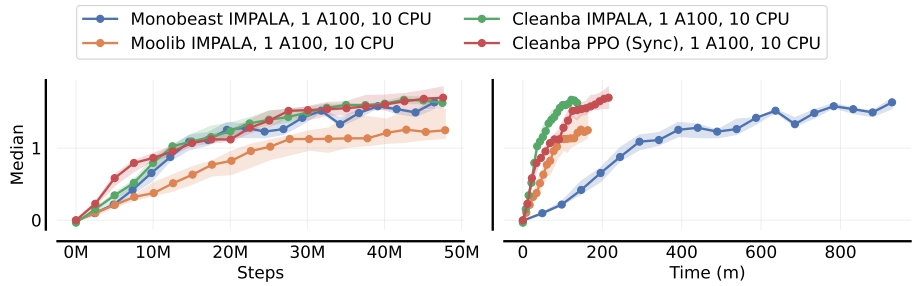


Figure 5.12: Median human-normalized scores with 95% stratified bootstrap CIs of Cleanba [119] variants compared with moolib [185] and monobeast [144]. The experiments were conducted on 57 Atari games [21]. The data used to generate the figure comes from Open RL Benchmark, and the figure was generated with a single command from Open RL Benchmark's CLI. Figure from [119].

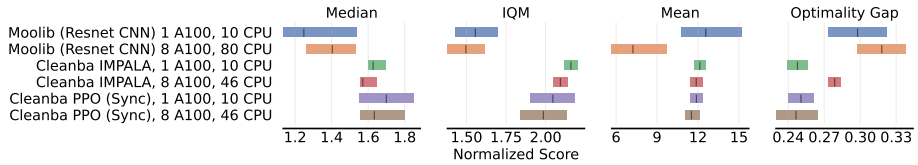


Figure 5.13: Aggregated normalized human scores with stratified 95% bootstrap CIs, showing that unlike moolib [185], Cleanba [119] variants have more predictable learning curves (using the same hyperparameters) across different hardware configurations. Figure from [119].

5.2.4 Current practices in RL: data reporting, sharing and reproducibility

Many new methods have emerged in recent years, with some becoming standard baselines, but current practices in the field make it challenging to interpret, compare, and replicate study results. In this section, we highlight the inconsistent presentation of results, focusing on learning curves as an example. This inconsistency can hinder interpretation and lead to incorrect conclusions. We also note the insufficient availability of learning data, despite some positive efforts, and examine challenges related to method reproducibility.

Analyzing learning curve practices

Plotting learning curves is a common way to show the evolution of an agent's performance as it learns. In this section, we take a closer look at the different components of learning curves. We examine in detail the choices made by a selection of key publications in the field on these different aspects. We show that among these publications, there is no uniformity on any aspect, that the choices of presentation are almost never motivated and that, sometimes, they are not even explicitly stated.

Axis Typically, the y axis measures either the return acquired during data collection or evaluation. Some older papers, like [237, 189, 239], fail to specify the metric, using the vague term *learning curve*. The first approach sums the rewards collected during agent rollout [56, 34]). The second approach suspends training, averaging the agent's return over episodes, deactivating exploration elements [86, 105, 110, 124, 19, 71, 41]. This method is prevalent and provides a more precise evaluation. Regarding the x axis, while older baselines [237, 189] use policy updates and learning epochs, the norm is to use interaction counts with the environment. In Atari environments, it is often the number of frames, adjusting for frame skipping to match human interaction frequency.

Shaded area Data variability is typically shown with a shaded area, but its definition varies across studies. Commonly, it represents the standard deviation [41, 124] and less commonly half the standard deviation [86]. [105] uses a min-max representation to include outliers, covering the entire observed range. This method offers a comprehensive view but amplifies outliers' impact with more runs. [71] adopts a probabilistic approach, showing a 95% bootstrap confidence interval around the mean, ensuring statistical confidence. Unfortunately, [237, 239, 189, 56, 19] omit statistical details or even the shaded area, introducing uncertainty in data variability interpretation, as seen in [110].

Smoothing The variability in results can hinder figure clarity. While many papers present raw curves [237, 86, 105, 56, 124, 41, 19], smoothing is a common practice to address this issue. However, smoothing sacrifices data variability information. Authors should provide clear explanations of this post-treatment to prevent misinterpretation. Unfortunately, most authors don't offer sufficient details to understand and reproduce the smoothing process. For instance, [237, 239] likely use smoothing without explicit mention. [110, 86] briefly mention curve smoothing but lack method details. The exception is [71], which provides a precise smoothing description.

Normalization and aggregation Performance aggregation assesses method results across various tasks and domains, indicating their generality and robustness. Outside the Atari context, aggregation practices are not common due to the absence of a universal normalization standard. In the absence of a widely accepted normalization strategy, scores are typically not aggregated, or if done, it relies on a min-max approach based on extreme study scores, lacking absolute significance and unsuitable for subsequent comparisons. In the case of Atari, early research did not use normalization or aggregate results [187]. However, there has been a notable shift towards generalizing normalization against human performance, although this has weaknesses and may not truly reflect agent mastery [269]. Aggregation methods vary as well. Mean is common but can be influenced by outliers, leading some studies to prefer the more robust median, as in [110], while, many papers now report both mean and median results [56, 106, 18]. Recent approaches like [155] use the Interquartile Mean (IQM) for balanced aggregation, providing a more accurate performance representation across diverse games, as suggested by [3].

Spectrum of data sharing practices

While the mentioned studies often have reference implementations (see Section 5.2.4), the sharing of training data typically extends only to the curves presented in their articles. This necessitates reliance on libraries that replicate these methods, offering benchmarks with varying levels of completeness. Several widely-used libraries in the field provide high-level summaries or graphical representations without including raw data (e.g., Tensorforce [138], Garage [93], ACME [113], MushroomRL [62], ChainerRL [88], and TorchRL [28]). Spinning Up [1] offers partial data accessibility, providing benchmark curves but withholding raw data. TF-Agent [101] is slightly better, offering experiment tracking with links to TensorBoard.dev, though its future is uncertain due to service closure. Tianshou [281] provides individual run reward data for Atari and average rewards for MuJoCo, with more detailed MuJoCo data available via a Google Drive link, but it is not widely promoted. RLLib [163] maintains an intermediate stance in data sharing, hosting run data in a dedicated repository. However, this data is specific to select experiments and often presented in non-standard, undocumented formats, complicating its use. Leading effective data-sharing platforms include Dopamine [37] and Sample Factory [208]. Dopamine consistently provides accessible raw evaluation data for various seeds and visualizations, along with trained agents on Google Cloud. Sample Factory offers comprehensive data via Weights and Biases [25] and a selection of pre-trained agents on the Hugging Face Hub, enhancing reproducibility and collaborative research efforts.

Review on reproducibility

The literature shows variations in these practices. Some older publications like [237, 239, 21, 189, 110] and even recent ones like [224] lack a codebase but provide detailed descriptions for replication[12]. However, challenges arise because certain hyperparameters, important but often unreported, can significantly affect performance [10]. In addition, implementation choices have proven to be critical [109, 119, 115, 73], complicating the distinction between implementation-based improvements and methodological advances.

Recognizing these challenges, the RL community is advocating for higher standards. NeurIPS, for instance, has been requesting a reproduction checklist since 2019 [209]. Recent efforts focus on systematic sharing of source code to promote reproducibility. However, codebases are often left unmaintained post-publication (with rare exceptions [86]), creating complexity for users dealing with various dependencies and unsolved issues. To address these challenges, libraries have aggregated multiple baseline implementations (see Section 5.2.2), aiming to match reported paper performance. However, long-term sustainability remains a concern. While these libraries enhance reproducibility, in-depth repeatability is still rare.

[12]This section uses the taxonomy introduced by [175]: *repeatability* means accurately duplicating an experiment with source code and random seed availability, *reproducibility* involves redoing an experiment using an existing codebase, and *replicability* aims to achieve similar results independently through algorithm implementation.

5.2.5 Discussion and conclusion

Reproducing RL results is difficult due to limited data access and code sharing. Minor implementation variations can lead to performance differences, and verifying implementations lacks tools. Researchers often rely on vague comparisons with paper figures, making reproduction time-consuming and challenging, highlighting reliability and reproducibility issues in RL research. In our paper, we introduce Open RL Benchmark, a vast collection of tracked experiments spanning algorithms, libraries, and benchmarks. We capture all relevant metrics and data points, offering detailed resources for precise reproduction. This tool democratizes access to comprehensive datasets, simplifying valuable information extraction, enabling metric comparisons, and introducing a CLI for easier data access and visualization. Open RL Benchmark is a dynamic resource, regularly updated by both its founders and the user community. User contributions, whether new results or additional runs, enhance result reliability. Sharing trained agents can also offer insights and support offline RL studies.

Despite its strengths, Open RL Benchmark faces challenges in user-friendliness which must be addressed. Inconsistencies across libraries in evaluation strategies and terminology can complicate usage. Scaling community engagement becomes challenging with more members, libraries, and runs. The lack of Git-like version tracking for runs adds to these limitations.

Open RL Benchmark is a key step forward in addressing RL research challenges. It offers a comprehensive, accessible, and collaborative experiment database, enabling precise comparisons and analyses. It enhances data access, promoting a deeper understanding of algorithmic performance. While challenges persist, Open RL Benchmark has the potential to elevate RL research standards.

5.3 Perspective: A Commitment to Continual Relevance

We have introduced PandaGym and Open RL Benchmark, two contributions designed to facilitate easier, more qualitative, and collaborative research in RL. A key difference between these contributions and traditional scholarly outputs is that their value lies largely in their active, long-term maintenance. This principle is exemplified by tools like PyTorch [204], which have undergone continuous development and improvement over the years. Today, the machine learning community relies heavily on such well-maintained tools to advance the field. Similarly, the continued relevance and utility of PandaGym and Open RL Benchmark will depend on our commitment to their ongoing maintenance and development, ensuring that they remain robust and valuable resources for researchers now and in the future.

Chapter 6

Conclusion

6.1 Summary and Conclusion

6.1.1 Summary and contributions

This work aims to extend and redefine the scope of RL. The underlying goal is to get closer to what we conceptualise as an autonomous generalist agent. Such an agent, as envisioned in our research, should have key characteristics, notably autonomy, characterized by reduced dependence on user supervision, and versatility, which includes the ability to handle multiple input modalities, operate in different environments with different dynamics, and achieve a wide range of tasks. Consequently, we have focused on two main research topics: first, improving the agent's ability to operate in environments where goals are either poorly defined or even absent, which implies improving existing methods in terms of exploration; second, designing a framework for a general-purpose agent capable of performing in multitasking, multimodal, and multidomain scenarios. This effort is further extended to a third critical dimension–research methodology, where our goal is to improve the current framework for RL research by providing tools that enhance algorithm comparison and reproducibility.

We first laid the foundations of RL and introduced some of the key algorithms. Our first contribution is LGE, a novel exploration method for RL. A key component of LGE is the ability of the agent to autonomously select goals using a jointly learned latent representation. This innovation eliminates the need for predefined cells, a major limitation of the original Go-Explore method. As a result, our work significantly extends the method's ability to operate in a wide range of environments. In addition, LGE demonstrated significantly better exploration performance than baselines in various environments, including robotic systems and Atari games. The robust empirical results we present confirm the effectiveness of this approach. This advance contributes directly to addressing the initial question of our research: the generalization of RL. By enabling more flexible and efficient exploration methods, LGE represents a significant step toward developing RL algorithms that are adaptable to a wider range of real-world scenarios, beyond the constraints of task-specific environments.

Our second key contribution JAT, a novel model characterized by its multimodal and multitasking design, which allows it to operate in different domains within a single network. This design facilitates the execution of diverse tasks

ranging from standard sequential decision making to non-sequential tasks such as image description and text completion. The first innovation lies in its unique structure, which has been optimized for sequential tasks by allocating each timestep to a corresponding token embedding, resulting in a very simple design. Another key innovation in JAT is the inclusion of an auxiliary objective for predicting future observations, a strategy that accelerates convergence and improves overall model performance. Accompanying JAT is its comprehensive dataset, the first to integrate such a diverse range of tasks, modalities and domains, setting a new precedent in the field. When benchmarked against Gato, its closest comparable baseline, JAT demonstrates competitive performance across its trained domains. This development is particularly significant in the context of our research, as it exemplifies our goal of generalizing RL and demonstrates the practical applicability of JAT beyond traditional task-specific environments. In fact, this work lays the foundations for linking RL to CV and NLP.

We also present several methodological and technical contributions that are integral to our research. These include PandaGym, an open and actively maintained suite of simulated robotic environments designed for complex sparse reward tasks. These environments are closely aligned with our research goals, challenging algorithms to learn autonomously and perform robotic tasks with minimal supervision. The response from the RL community to PandaGym has been largely positive, with widespread adoption and improvement over the three years of its existence. Researchers have adapted these environments, outperforming existing baselines and contributing significantly to its continuous improvement. We also present Open RL Benchmark, an extensive collection of open, thoroughly tracked RL experiments. This benchmark includes runs of a wide range of popular RL libraries in a variety of environments. Its tracking goes beyond mere performance metrics, providing full access to all raw training data and essential resources for rigorous replication of results. We argue that progress towards a generalist RL model depends on a foundation of reproducible research. This benchmark is a testament to our commitment to enabling innovation that builds on previous work rather than redundantly duplicating established findings. Practical examples of its application are provided, reinforcing our aspiration that Open RL Benchmark will evolve alongside ongoing research advances.

While the road to a truly effective generalist RL agent remains long, we believe that our work represents a meaningful step in this direction. By enhancing agent autonomy in contexts of weak supervision and by proposing a structure for a multi-task and multi-domain agent, we have addressed two critical attributes necessary for a generalist RL agent. These advances are fundamental steps in the broader quest to realize the full potential of RL. These contributions, alongside the technical and methodological ones, form the basis for future research, fostering an RL landscape characterized by its broader applicability and adaptability in diverse real-world scenarios.

6.1.2 Perspectives

In this thesis, each topic we cover is accompanied by a specific set of perspectives, which are detailed in the respective chapters. Here, we propose a broader overview, identifying noteworthy general perspectives on RL. Of the many possible ones, we believe three stand out.

Foundation models for RL Foundation models for RL can be understood as large-scale, pretrained models that provide a versatile and adaptable base, enabling RL agents to efficiently learn and generalize across a diverse range of environments and tasks, akin to how foundation models in NLP have revolutionized understanding and generation of human language. The application of foundation models in RL faces unique challenges that differ from their success in NLP. RL environments vary widely, each with unique rules and interactions, in contrast to the consistent structure of language in NLP. This diversity raises critical questions about the adaptability of fundations models in RL, particularly in dealing with different state and action spaces and environment-specific dynamics. Developing such models may require the creation of environment-agnostic learning strategies, but the practicality and effectiveness of this approach remains to be determined. Overcoming these challenges could lead to a new era in RL, characterized by highly adaptive, versatile agents.

Reincarnating RL Traditional RL framework is predominantly characterized by learning from scratch. However, this approach encounters a significant bottleneck in terms of sample efficiency, a limitation unlikely to be overcome in the foreseeable future. This question is partially addressed in a specific framework where the distribution of data evolves during learning [159], but this is however limited by the underlying assumption of constancy of policy structure and data type. Therefore, this necessitates a paradigm shift in our framework, leading us to question how prior computational data can be leveraged to transcend the *learning from scratch* model. This evolving concept has been termed Reincarnating Reinforcement Learning (RRL) [4].

The rationale behind RRL is underpinned by its immense potential, as it recognizes the abundance of existing open data, especially from [258] or simulating the real world [242]. By reducing resource requirements, RRL democratizes access to more complex problem solving, enabling broader participation beyond institutions with substantial resources. Crucially, RRL fosters an environment in which researchers can iteratively improve existing agents. The core challenge of RRL lies in the effective distillation of diverse data sources—ranging from experiences to policies, optimal or not etc.—into a new policy. This challenge has only been sporadically addressed in large-scale training programs, such as those detailed in [197], which often lack comprehensive exploration of this aspect. However, emerging works are beginning to tackle this issue more directly. For example, [4] focuses on the efficient transfer of existing sub-optimal policies to standalone, value-based RL agents, with very encouraging results.

To advance this field, a clear problem statement must be articulated. Then, determining a suitable evaluation protocol is essential to enable coherent and meaningful comparisons of different RRL methods. Key questions arise: What kind of initial resources should be used? (expert experience, sub-optimal network weights, etc.) Which environments are most appropriate for testing? Should the protocol consider the use of data from another environments? (Thus including scenarios in which an action dimension is added, for example.) The groundwork has barely been laid, but we think it is an absolutely essential step towards an RL agent that actually solves real-world problems.

Self Supervised World-Model with Intrinsic Goals RL faces two key challenges that hinder its broader real-world application: low sample efficiency, as evidenced by the need for hours of training for even basic tasks like the inverted pendulum [123], and the difficulties associated with exploration, as discussed in Chapter 3. Learning beings such as animals, including humans, inherently anticipate and visualise future scenarios, suggesting a reliance on an internal world model. The idea of learning a world model and anticipating outcomes without direct interaction with the environment, although previously sidelined, is receiving renewed attention in the literature, notably discussed in [153]. The integration of model learning with structured exploration based on intrinsic goals may be the key to success. This approach's essence, and the key question, is how to formulate goals that balance world understanding with reward maximisation (the exploration-exploitation dilemma). This idea fits elegantly with the vision of a generalist agent capable of reward-free exploration and subsequent task learning through pure planning within its internal world model.

What does the future of RL look like when integrating these three perspectives? We envisage a process where initially, a general foundation agent is retrieved. This agent, inherently multi-modal and equipped with an internal world model, is pre-trained with diverse data from previous computations. It is designed to process any data type efficiently, ensuring there is no risk of unlearning, and at the very least, maintaining its current knowledge level. After this, the agent would engage in interaction with the chosen environment(s), identifying and exploring key areas of interest using its world model. Ultimately, the culmination of this process is the sharing of all the computational data and the resulting agent, allowing the cycle to iterate again, each time refining and enhancing the agent's capabilities.

Finally, let's use this section on the future to revisit the past and note that despite being 74 years old, Alan Turing's quote remains remarkably relevant:

> "Instead of trying to produce a program that simulates the mind of an adult, why not try to produce one that simulates the mind of a child?"
>
> — Alan M Turing (1950)

6.2 Ethical Considerations in Artificial Intelligence Research

6.2.1 Ethics in AI: an all-too-often overlooked necessity

Traditionally, doctoral theses in Artificial Intelligence (AI) have focused on technical advancements, methodological innovations, and experiments. The emphasis is on creating, improving, and validating algorithms, models, and applications. In this context, it is rare to find a section dedicated to the ethical, societal, or philosophical implications of AI. This absence is all the more surprising given that AI, by its very nature, is redefining our world and our place within it.

The title *Doctor of Philosophy* or PhD, although often associated with philosophy, is in fact a reminder of the historical origins of medieval European universities. The term *philosophy* is not used in the strict sense of the academic discipline, but rather to signify the "love of wisdom". This wisdom encompasses a universal quest for knowledge, transcending disciplinary boundaries. Thus, even if the thesis focuses on AI, it is rooted in this tradition of seeking truth and understanding.

I firmly believe that integrating ethical reflection into AI research is not only desirable but necessary. AI is not just a mere technical tool; it shapes our society, influences our decisions, and redefines our interactions. Ignoring the ethical implications of our work means overlooking an essential part of AI's impact.

As researchers, we have a responsibility not only to push the boundaries of knowledge but also to understand and anticipate the consequences of our discoveries. AI, with its transformative potential, demands particular vigilance from us. Issues of fairness, transparency, privacy, and accountability are not mere additions to our work; they are intrinsic components. The AI community seems to be increasingly interested in this issue, as reflected by the recent introduction of the position paper track at the ICML 2024 conference[1].

Therefore, through this section, we not only aim to address the ethical implications of my work but also encourage future AI researchers to do the same. Ethics should not be an afterthought but an integral part of our scientific approach. By integrating ethics into our research, we are not just creating more efficient algorithms; we are working towards an AI that respects and enriches humanity.

We thus invite all future doctors to seriously consider integrating ethical reflection into their work. Only by addressing these issues head-on can we ensure that AI truly serves everyone.

6.2.2 The societal and civilizational challenge

Separating science fiction fears from real-world ethical challenges

Public perceptions of AI threats and ethical issues are often influenced by dramatic depictions in science fiction, such as the AI uprising in *The Terminator* or the ethical complexities of creating sentient AI in *Ex Machina*. While these narratives are compelling, they stem from anthropomorphism, speculative futures, and a fundamental misunderstanding of the current capabilities of AI, which operates within human-defined parameters. AI operates on the basis of programmed algorithms and data processing, without the self-awareness or subjective experiences that are hallmarks of human consciousness as currently understood in cognitive science and philosophy. Therefore, it seems more relevant to examine more pressing and likely real-world problems with AI that involve its societal impact and potential for abuse. For example, deepfakes can be used for political misinformation, and AI in hiring processes can inadvertently perpetuate bias due to flawed training data. In addition, the ethical use of AI in surveillance raises significant privacy concerns. In the following discussion, we will delve into these critical issues, exploring the societal impact of AI, potential misuse scenarios, and ethical considerations, and highlighting the need for solutions and responsible integration of AI into society.

[1] `https://icml.cc/Conferences/2024/CallForPositionPapers`

AI and the labor market

The unfolding reality of AI in the workforce underscores the notion that AI-driven technological advances will lead to significant job displacement. This trend is already visible, as evidenced by IBM's initiative to replace approximately 7,800 jobs with AI over the next five years, particularly in non-customer-facing roles such as back-office functions [83]. Moreover, the broader technology industry is witnessing a similar pattern. According to a report from Challenger, Gray & Christmas, about 5% of job cuts in May 2023 were directly due to AI integration [38]. The central question regarding the impact of AI on the labor market transcends simplistic binary categorizations. Rather, it requires a critical examination of how AI, as a transformative technological entity, is reshaping employment dynamics. This shift in perspective is not only about the technical capabilities of AI, but also about understanding its broader societal implications. Significantly, AI's impact on the labor market is not an inevitable outcome of technological determinism. Rather, it depends largely on collective societal decisions about the application and governance of AI in the workforce.

Deepfakes, scams and spam

Deepfake technology, which leverages AI to create realistic media, brings a significant risk of misinformation and disinformation. Among other things, this technology enables the creation of completely fictitious identities and documents, leading to fraud risks and security threats. AI tools amplify these risks by increasing the scale, effectiveness, and persuasiveness of fraudulent activities, such as the automation of scams or mass phishing campaigns aimed at extracting sensitive information or money. For example, the 2019 CEO impersonation scam reported by the Wall Street Journal illustrates this significant fraud potential [256]. Deepfakes also extend to non-consensual exploitation and pornography, invading privacy and causing serious psychological harm to victims. Politically, given their ease of manipulation and distribution, they can disrupt elections and incite civil unrest by misrepresenting political figures. The evolving nature of deepfakes technology requires advanced detection methods and public education, but the rapid development of deepfakes continues to defy these efforts. Societal responses, like past media manipulations, can disrupt politics, undermine trust in media and democratic institutions, and foster skepticism and conspiracism.

Amplifying stereotypes and inequities

AI has the potential to exacerbate biases and stereotypes, a concern that has been increasingly recognized in recent years. AI systems, trained on data reflecting historical biases, often perpetuate and even amplify these biases, as seen in cases involving facial recognition technologies and hiring algorithms. Studies have already highlighted the significant disparities in the accuracy of facial recognition technology across different demographics [33]. Similarly, in hiring, AI tools can inadvertently favor candidates based on biased historical hiring data, as reported by Dastin [60]. Moreover, AI-driven content recommendation systems, like those used by social media platforms and search engines, can reinforce stereotypes by perpetuating echo chambers and presenting biased viewpoints [202]. The challenge lies in ensuring AI is developed and deployed responsibly, with an emphasis on diverse, inclusive training data and continuous monitoring for biases.

This approach is critical to prevent AI from deepening societal divisions and reinforcing harmful stereotypes.

6.2.3 The ecological challenge

The ecological impact of AI is a growing concern, as the computational demands of developing and operating AI systems contribute significantly to carbon emissions and energy consumption. The training of large-scale AI models requires substantial computational resources.

A critical first step is to understand the scope of the problem. There appears to be an increasing focus on the accurate measurement of energy consumption and carbon emissions due to AI systems, as evidenced by research such as [174]. Concurrently, tools are being developed to assist in the accurate calculation and sharing of these data [147]. In 2022, global data centers consumed 240-340 TWh of electricity, about 1-1.3% of global electricity demand, not including the 0.4% from cryptocurrency mining. Despite the rise in demand for data center services, energy use increased only moderately due to improved efficiency in IT hardware and cooling. Large data centers, however, experienced a 20-40% annual increase in energy use. Notably, combined electricity use by major companies like Amazon, Microsoft, Google, and Meta more than doubled between 2017 and 2021, reaching around 72 TWh in 2021 [122]. The International Energy Agency notes that in countries with expanding data storage sectors, such as Ireland and Denmark, data centers consume a significantly higher percentage of national electricity [186].

Furthermore, the footprint of AI goes beyond energy consumption. The production of hardware for AI systems also contributes to environmental stress. The extraction, refining, and transportation of minerals required to produce AI hardware comes at a significant environmental and human cost. The problem is further exacerbated by the short lifespan and difficult recyclability of this hardware [55].

Efforts to employ AI for optimizing energy use in various sectors, monitoring environmental conditions, and supporting climate research are noteworthy, as detailed in [228]. Additionally, initiatives aimed at sustainable AI practices are receiving increasing attention, with a focus on creating energy-efficient AI algorithms [240]. Paradoxically, this increase in efficiency is likely to lead to an increase in overall energy consumption as a result of the *rebound effect* [267].

6.2.4 Should research be halted? Openness as an imperfect solution

At this point, the reader is likely well aware of the significant ethical challenges posed by AI and the potential for its negative social impact. However, the reality of AI's impact is more nuanced. The next section provides a concrete illustration of how AI tools can benefit research, and more specifically, how they have helped shape this thesis. One question remains, however: How can we ensure that AI acts as a beneficial force rather than a societal threat?

One possible approach is to collectively consider slowing down the development of AI. This would allow time to consider the broader implications of AI and to facilitate its integration into society. However, the feasibility of slowing or pausing AI research is questionable. It depends on a high degree of consensus

and collaboration among all AI stakeholders, who must not only agree that slowing down is beneficial, but also actively engage in this approach. Given the diversity of interests and the rapid pace of progress in AI, achieving such broad agreement and action is challenging.

A more practical and promising way forward is to embrace openness in AI. Openness in this context means transparency of AI models, data, and methodologies. It is a concept that revolves around the idea of making AI accessible and understandable to a wider audience.

The risks of such openness are not trivial. A primary concern is that it might facilitate the development of problematic applications, including those that could be used unethically or maliciously. However, these risks must be weighed against the substantial benefits that openness brings to the table. Openness enable a broader segment of society to engage with and shape these technologies, thus limiting the risk of a few entities monopolizing them to the detriment of many. Additionally, the open nature of these models allows for constant auditing and community-driven improvements, particularly on issues the community deems significant. This continuous review not only improves the effectiveness of AI solutions, but also aligns their development with societal values. Moreover, openness fosters increased transparency, which in turn leads to greater understanding and trust among the general public. This aspect is crucial in building a society that not only uses AI but also understands and trusts its mechanisms and decisions. Numerous technologies have demonstrated the success of an open-source strategy, with Linux, PyTorch, and Git being prime examples. Far from being anecdotal, these technologies have become de facto standards in the AI field itself, demonstrating the effectiveness of openness in driving innovation, ethical development and widespread adoption.

There is no one-size-fits-all answer to the ethical challenges posed by AI. The strategy of openness seems most promising, and is attracting growing interest, even from big technology companies [195].

6.3 The Role of AI in Shaping Research: A Personal Perspective

This section aims to deepen our understanding of how AI tools are reshaping research methodologies. We begin by arguing for the need to tailor our methodological approaches to the significant changes brought about by these emerging technologies. We then discuss the practical applications of some of these tools to the research presented in this document.

6.3.1 Advocating for methodological transparency: the role of AI in research

In the annals of academic research, transparency has always been a cornerstone, ensuring the integrity and credibility of scholarly work. As we navigate the evolving landscape of research in the age of AI, this transparency becomes even more crucial. The use of AI in this thesis has been more than an ordinary tool, and has influenced various facets of the research process. It is, therefore, both a duty and an ethical responsibility to elucidate its role and impact.

The reasons for this are manifold. Firstly, for the sake of reproducibility, it is vital that other scholars understand the influence of AI to either reproduce or build upon this work. Secondly, as AI reshapes the academic paradigm, detailing its use contributes to the ongoing discourse on the evolution of academic norms. This exposition also serves an educational purpose, offering a valuable framework for peers considering AI in their research, thereby fostering peer education.

However, a deeper introspection arises: if AI has played such a pivotal role in this research, does it diminish the worthiness of obtaining a doctoral degree? Historically, the essence of a doctoral thesis has been the demonstration of independent research, critical thinking, and a significant contribution to one's field. While AI has facilitated certain tasks and perhaps even suggested research directions, it was under the guidance, interpretation, and discretion of the human researcher. Throughout history, scholars have embraced new tools, from telescopes to electron microscopes, each met with initial skepticism, only to later become integral to research. The integration of AI is merely the latest, albeit profound, step in this evolution. The skills required to adeptly incorporate AI, ensuring its ethical application and interpreting its outputs, are commendable and align with the spirit of doctoral research. The very act of questioning the role of AI in this context underscores a deep ethical reflection, a quality emblematic of doctoral-level inquiry.

By transparently delineating the use of AI, this work not only acknowledges its influence but also assumes responsibility for its implications, reinforcing both the research's credibility and public trust in academic pursuits in the AI era.

6.3.2 AI's Practical Applications and Enhancing Research Quality

This section provides a detailed description of the use of AI tools in this work, with a particular focus on language models, as they were the only category used. It is worth noting that the use of models such as Copilot[2] and ChatGPT[3], which will be discussed in more detail in the following sections, has not been consistent throughout this thesis. At the time of writing, these models have only been available for the last 12 months. Consequently, the following discussion focuses primarily on this one-year period.

Language models for literature review and self-learning

The task of identifying relevant papers and references was greatly facilitated by the use of language models, particularly ChatGPT. It was able to conduct quick literature reviews and summarize areas that were initially unfamiliar, which was particularly useful in preparing the Section 2.5, among others. In addition, it played a critical role in accelerating the understanding of complex concepts that would otherwise have taken much longer to master. It was particularly effective in highlighting and clarifying details that posed comprehension challenges, thereby enhancing the learning process. An example is shown in Figure 6.1.

It also made it easier to make connections between seemingly disparate concepts, enriching the research with a more integrated perspective. However, it was found that while the answers provided by ChatGPT were often convincing,

[2] `https://www.microsoft.com/fr-fr/microsoft-copilot`
[3] `https://chat.openai.com`

User: In Proximal Policy Optimization (PPO), what is the most straightforward way to enhance the agent's exploration? Please provide a concise answer.

ChatGPT: In Proximal Policy Optimization (PPO), a straightforward way to enhance the agent's exploration is by increasing the entropy term in the loss function. This encourages the agent to explore more diverse actions, reducing the likelihood of premature convergence to sub-optimal policies.

Figure 6.1: Example of the use of ChatGPT for self-training.

they were not infallibly accurate. This highlighted the importance of careful verification and critical analysis

Writing assistance and correspondence

The use of AI tools like ChatGPT and DeepL proved to be invaluable in enhancing the writing style in English for this research. These tools were effective at correcting typos and significantly improved English fluency, ensuring clarity and precision in academic writing as well as in professional communication. Examples of its use in this document is shown in Figures 6.2 and 6.3. Their influence occasionally extended to help with organizing ideas, although this was not consistently relevant. Acting as ongoing English tutors, they not only suggested better phrasing for the research text, but also contributed significantly to the author's personal language development. While there is a concern that such tools may diminish linguistic diversity by promoting uniformity, we believe that this uniformity can be beneficial to a certain extent: it helps to convey complex technical content clearly and concisely, which is paramount in academic and professional communication in this field. In addition, we believe that the standardization of language quality provided by these tools has helped to mitigate bias against non-native English speakers, promoting a more inclusive academic environment.

Coding assistance

The extensive coding effort that underpinned the work presented in this document benefited greatly from the use of AI tools, in particular ChatGPT and Copilot. These tools played a crucial role at several stages of the coding process.

First and foremost, ChatGPT and Copilot dramatically accelerated the process of finding coding solutions. When faced with the question "How do I do this?", these tools provided immediate, accurate and well-explained answers to basic questions. Moreover, Copilot has a very efficient auto-completion function, and in particular saves a considerable amount of time on simple queries. Take, for example, the production of documentation, for which Copilot is particularly effective. This is an important task, not very complex, but time-consuming, which we might be tempted to postpone or neglect without this tool. This level of efficiency and accuracy made them more practical than traditional resources such as forums like Stack Overflow or GitHub Issues. Notably, the launch of

User: Could you please convert the following explanation into a style more suited for a mathematical or academic paper? Emphasize formal mathematical language, concise definitions, and the use of appropriate notation and symbols. Here's the text:
"You might notice that in many cases, the reward of a good action is not immediate. If I play a good chess move, I'm not going to win immediately. Thus, the agent actually tries to maximize the *return* instead of the reward."

ChatGPT: "In numerous scenarios, the reward resulting from an optimal action may not be immediate. Consider a chess game: a strategically sound move does not guarantee an immediate win. Consequently, the agent's objective is to maximize the *return*, a cumulative measure, rather than the immediate reward."

Figure 6.2: Example of the use of ChatGPT for conversion of a casual explanation into formal academic language.

the Copilot and ChatGPT beta coincided with a significant decrease in traffic to Stack Overflow, underscoring the effectiveness and growing preference for AI-assisted coding solutions. There is an important nuance here. For more complex requests, it often happens that part of the response is not correct. Just as in the case of writing assistance, the response may still seem very credible, which makes these cases more difficult to detect. It is precisely for this reason that Stack Overflow has temporarily banned responses generated by these tools[4].

In addition, the use of ChatGPT has sometimes improved the formulation of requests for help in GitHub Issues. As a project manager, the benefit of receiving clear, well-formulated questions rather than vague or poorly described problems is considerable. This clarity of communication not only speeds up problem resolution, but also contributes to a more effective collaborative environment.

In conclusion, the integration of AI tools such as ChatGPT and Copilot in this research has been beneficial. These tools have improved the quality of the code and played a role in improving the author's coding skills. It is important for users to find an appropriate balance in their reliance on these tools. The support provided by these tools is expected to further improve the overall quality of code production and help to standardise coding practices, which is particularly beneficial in software development.

[4] `https://stackoverflow.com/help/ai-policy`

User: For my computer science thesis, I want to ensure that it adheres to American English standards. To ensure accuracy, I am seeking a comprehensive list of words that are frequently misspelled or have different spellings in British English. This will enable me to thoroughly review and verify the correctness of my thesis.

ChatGPT: Creating a comprehensive list of words that are often misspelled or have different spellings in British and American English is a great way to ensure the consistency and accuracy of your thesis. Here's a list of commonly used words with different American and British spellings:

1. Color (US) / Colour (UK)
2. Honor (US) / Honour (UK)
3. Labor (US) / Labour (UK)
4. Organize (US) / Organise (UK)
5. Color (US) / Colour (UK)
6. Honor (US) / Honour (UK)
7. Labor (US) / Labour (UK)
8. Organize (US) / Organise (UK)
9. Realize (US) / Realise (UK)
10. Analyze (US) / Analyse (UK)
11. Traveling (US) / Travelling (UK)
12. Defense (US) / Defence (UK)
13. License (US verb) / Licence (UK noun)
14. Practice (US noun) / Practise (UK verb)
15. Program (US) / Programme (UK)
16. Behavior (US) / Behaviour (UK)
17. Center (US) / Centre (UK)
18. Fiber (US) / Fibre (UK)
19. Tire (US) / Tyre (UK)
20. Aging (US) / Ageing (UK)
21. Dialog (US) / Dialogue (UK)
22. Catalog (US) / Catalogue (UK)
23. Check (US) / Cheque (UK)
24. Aerobic (US) / Anaerobic (UK)

Remember, this list is not exhaustive, but it covers many of the common differences. Additionally, be aware of the differences in punctuation and formatting conventions between American and British English, such as the use of serial commas and quotation marks. For a more comprehensive review, consider using a style guide or editing software that can be set to American English standards.

Figure 6.3: Example of the use of ChatGPT for guidance on consistency improvements to this document.

Appendix A

JAT Dataset In Depth

This appendix completes the presentation of the JAT dataset begun in Section 4.3.2. This dataset, the scripts and expert agent used to generate it are available as open-source[1].

The JAT dataset contains sequential decision-making tasks from 4 domains: Atari, BabyAI, Meta-World and MuJoCo. Each sample in this dataset is an episode. This episode consists of a list of observations, actions and rewards, the nature and size of which depend on the task. In Figure A.1, we represent for each Atari game the average return of the episodes of the dataset normalized by the human score from [188]. Notably, for 43 games the average score is higher than the human score, and for 31 games the average score is more than twice the human score. It should be noted, however, that for 7 games (Bowling, Montezuma's Revenge, PitFall, Private Eye, Seaquest, Solaris and Venture) the average score is less than 10% of the human score.

We also plot the distribution of returns for each task, which provides a more detailed picture than a simple average. Figure A.2 shows this distribution for Atari, Figure A.3 for BabyAI, Figure A.4 for Meta-World and Figure A.5 for MuJoCo.

[1] `https://huggingface.co/datasets/jat-project/jat-dataset`

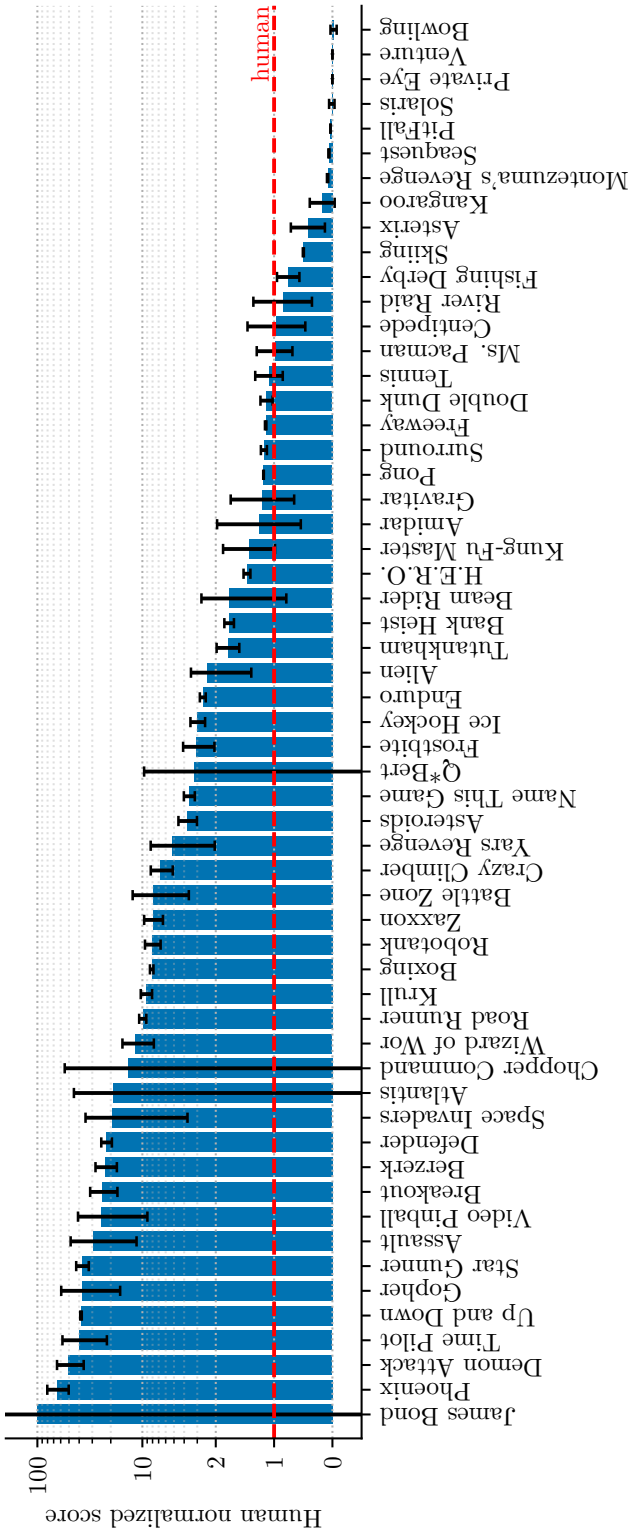


Figure A.1: Human normalized dataset scores for the Atari benchmark.

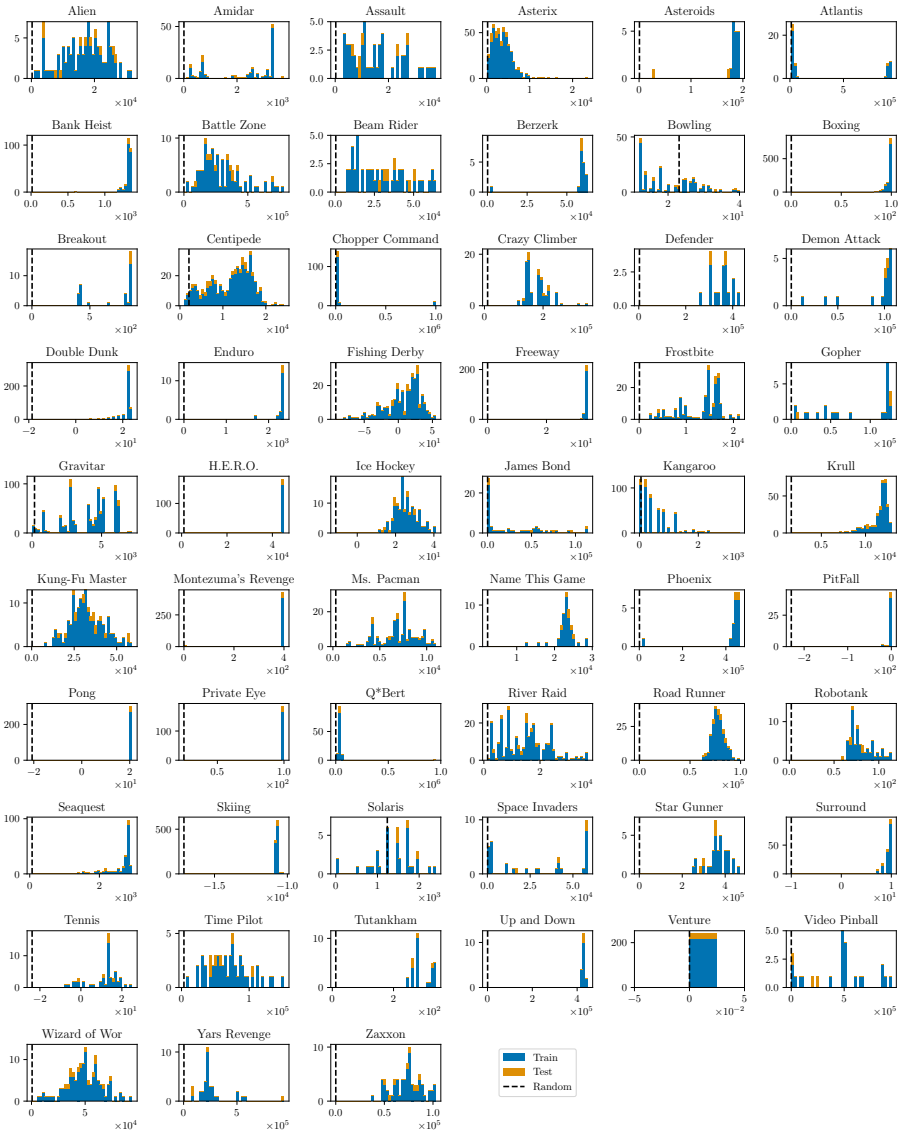


Figure A.2: Atari dataset return distribution.

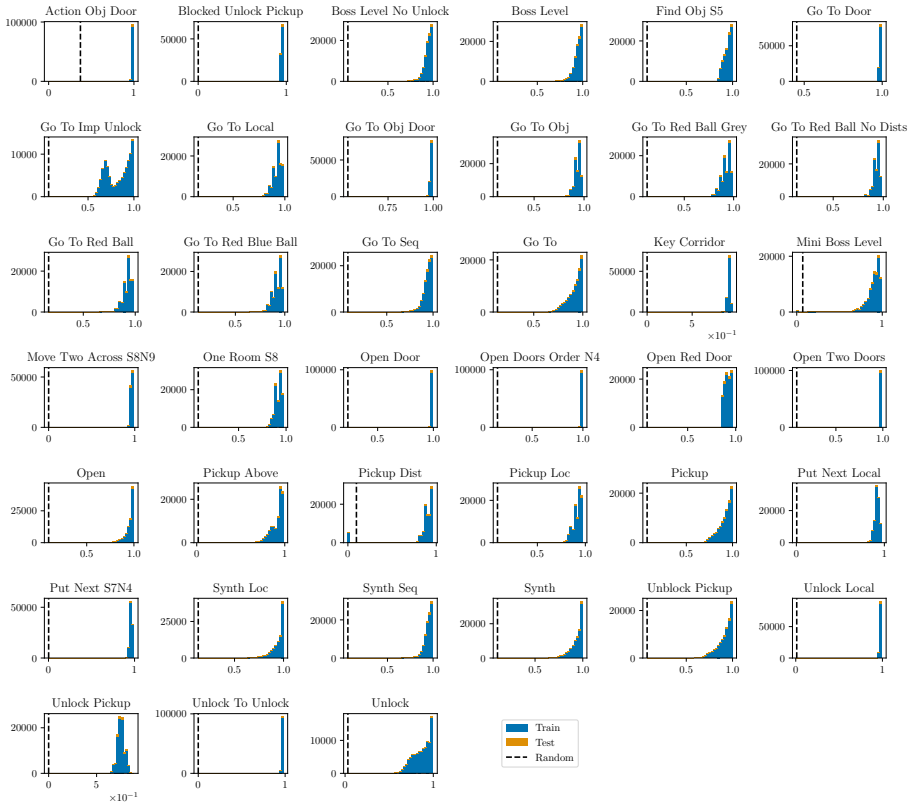


Figure A.3: BabyAI dataset return distribution.

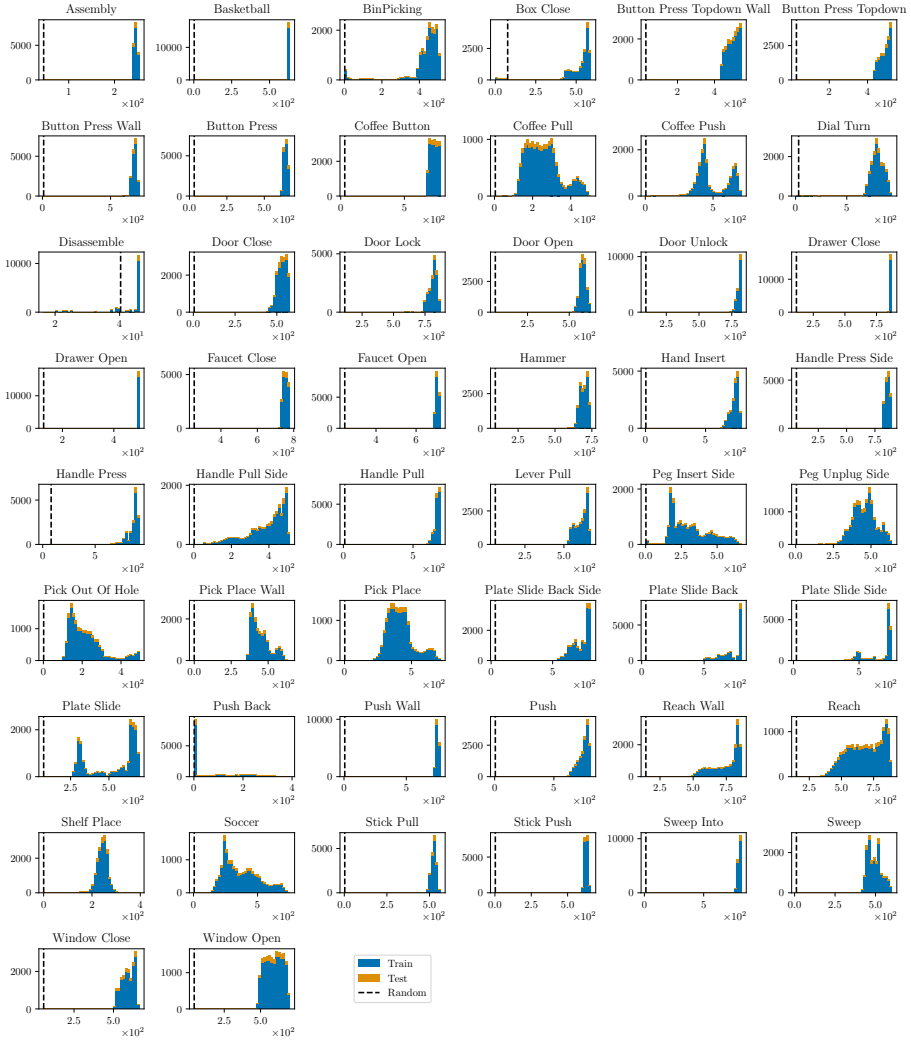


Figure A.4: Meta-World dataset return distribution.

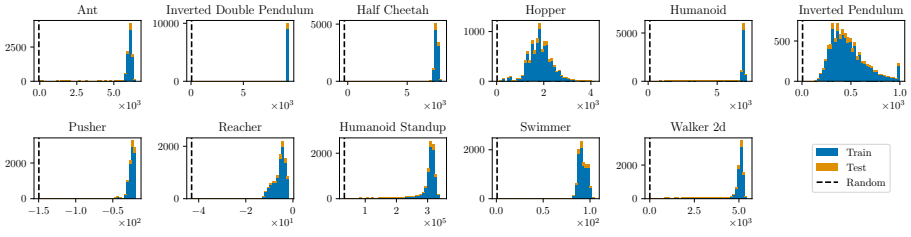


Figure A.5: MuJoCo dataset return distribution.

Appendix B

Plotting Guidelines for Open RL Benchmark

B.1 Using the CLI

This section gives notable additional examples of usage of the provided CLI. A more comprehensive set of examples and manual is available in the README page of the project.

B.1.1 Plotting episodic return from various libraries

First, we showcase the most basic usage of the CLI, that is comparing two different implementations of the same algorithm based on learning curve of episodic return. For example, Figure B.1 and Figure B.2 compare CleanRL's TD3 implementation against the original TD3, both in terms of sample efficiency and time. The command used to generate this plot is listed below.

```
python -m openrlbenchmark.rlops \
    --filters '?we=openrlbenchmark&wpn=sfujim-TD3&ceik=env&cen=policy&metric=charts/
         episodic_return' 'TD3?cl=Official TD3' \
    --filters '?we=openrlbenchmark&wpn=cleanrl&ceik=env_id&cen=exp_name&metric=charts/
         episodic_return' 'td3_continuous_action_jax?cl=Clean RL TD3' \
    --env-ids HalfCheetah-v2 Walker2d-v2 Hopper-v2 \
    --pc.ncols 3 \
    --pc.ncols-legend 2 \
    --output-filename static/td3_vs_cleanrl \
    --scan-history
```

In the above command, `wpn` denotes the project name, typically the learning library name. This allows to fetch results of implementations from different projects. Moreover, it is possible to specify which metric to compare, in this case `charts/episodic_return`. Also, the CLI provides the possibility to select a given algorithm and apply a different name in the plot, e.g. we rename `TD3` to `Official TD3` and `td3_continuous_action_jax` to `Clean RL TD3` Finally, we can also select a set of environments through the `--env-ids` option.

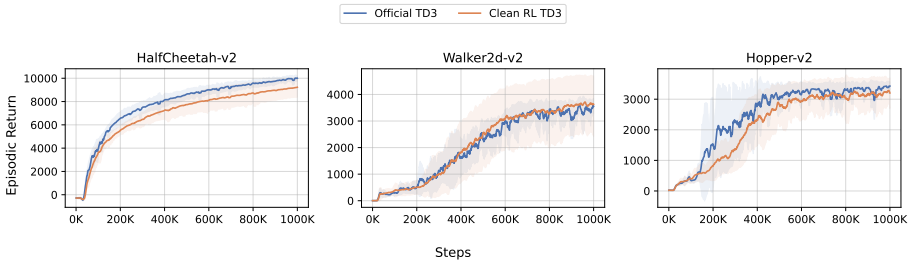


Figure B.1: Comparing CleanRL's TD3 against the original TD3 implementation (sample efficiency).

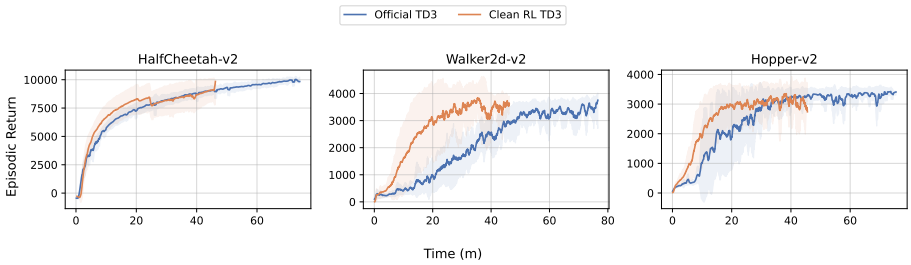


Figure B.2: Comparing CleanRL's TD3 against the original TD3 implementation (time).

B.1.2 RLiable integration

Open RL Benchmark also integrates with RLiable [3]. To enable such plot, the option `--rliable` can be toggled, then additional parameters are available under `--rc`. Figures B.3, B.4, B.5 and B.6 showcase the resulting plots of the following command:

```
python -m openrlbenchmark.rlops \
    --filters '?we=openrlbenchmark&wpn=baselines&ceik=env&cen=exp_name&metric=charts/
         episodic_return' 'baselines-ppo2-cnn?cl=OpenAI Baselines PPO2' \
    --filters '?we=openrlbenchmark&wpn=envpool-atari&ceik=env_id&cen=exp_name&metric=charts/
         avg_episodic_return' 'ppo_atari_envpool_xla_jax_truncation?cl=CleanRL PPO' \
    --env-ids AlienNoFrameskip-v4 AmidarNoFrameskip-v4 AssaultNoFrameskip-v4
         AsterixNoFrameskip-v4 AsteroidsNoFrameskip-v4 AtlantisNoFrameskip-v4
         BankHeistNoFrameskip-v4 BattleZoneNoFrameskip-v4 BeamRiderNoFrameskip-v4
         BerzerkNoFrameskip-v4 BowlingNoFrameskip-v4 BoxingNoFrameskip-v4
         BreakoutNoFrameskip-v4 CentipedeNoFrameskip-v4 ChopperCommandNoFrameskip-v4
         CrazyClimberNoFrameskip-v4 DefenderNoFrameskip-v4 DemonAttackNoFrameskip-v4
         DoubleDunkNoFrameskip-v4 EnduroNoFrameskip-v4 FishingDerbyNoFrameskip-v4
         FreewayNoFrameskip-v4 FrostbiteNoFrameskip-v4 GopherNoFrameskip-v4
         GravitarNoFrameskip-v4 HeroNoFrameskip-v4 IceHockeyNoFrameskip-v4
         PrivateEyeNoFrameskip-v4 QbertNoFrameskip-v4 RiverraidNoFrameskip-v4
         RoadRunnerNoFrameskip-v4 RobotankNoFrameskip-v4 SeaquestNoFrameskip-v4
         SkiingNoFrameskip-v4 SolarisNoFrameskip-v4 SpaceInvadersNoFrameskip-v4
         StarGunnerNoFrameskip-v4 SurroundNoFrameskip-v4 TennisNoFrameskip-v4
         TimePilotNoFrameskip-v4 TutankhamNoFrameskip-v4 UpNDownNoFrameskip-v4
         VentureNoFrameskip-v4 VideoPinballNoFrameskip-v4 WizardOfWorNoFrameskip-v4
         YarsRevengeNoFrameskip-v4 ZaxxonNoFrameskip-v4 JamesbondNoFrameskip-v4
         KangarooNoFrameskip-v4 KrullNoFrameskip-v4 KungFuMasterNoFrameskip-v4
         MontezumaRevengeNoFrameskip-v4 MsPacmanNoFrameskip-v4 NameThisGameNoFrameskip-v4
         PhoenixNoFrameskip-v4 PitfallNoFrameskip-v4 PongNoFrameskip-v4 \
    --env-ids Alien-v5 Amidar-v5 Assault-v5 Asterix-v5 Asteroids-v5 Atlantis-v5 BankHeist-v5
         BattleZone-v5 BeamRider-v5 Berzerk-v5 Bowling-v5 Boxing-v5 Breakout-v5 Centipede-v5
          ChopperCommand-v5 CrazyClimber-v5 Defender-v5 DemonAttack-v5 DoubleDunk-v5 Enduro-
         v5 FishingDerby-v5 Freeway-v5 Frostbite-v5 Gopher-v5 Gravitar-v5 Hero-v5 IceHockey-
         v5 PrivateEye-v5 Qbert-v5 Riverraid-v5 RoadRunner-v5 Robotank-v5 Seaquest-v5 Skiing-
         v5 Solaris-v5 SpaceInvaders-v5 StarGunner-v5 Surround-v5 Tennis-v5 TimePilot-v5
         Tutankham-v5 UpNDown-v5 Venture-v5 VideoPinball-v5 WizardOfWor-v5 YarsRevenge-v5
         Zaxxon-v5 Jamesbond-v5 Kangaroo-v5 Krull-v5 KungFuMaster-v5 MontezumaRevenge-v5
         MsPacman-v5 NameThisGame-v5 Phoenix-v5 Pitfall-v5 Pong-v5 \
    --no-check-empty-runs \
    --pc.ncols 5 \
    --pc.ncols-legend 2 \
    --rliable \
    --rc.score_normalization_method atari \
    --rc.normalized_score_threshold 8.0 \
    --rc.sample_efficiency_plots \
    --rc.sample_efficiency_and_walltime_efficiency_method Median \
    --rc.performance_profile_plots \
    --rc.aggregate_metrics_plots \
    --rc.sample_efficiency_num_bootstrap_reps 50000 \
    --rc.performance_profile_num_bootstrap_reps 2000 \
    --rc.interval_estimates_num_bootstrap_reps 2000 \
    --output-filename static/cleanrl_vs_baselines_atari \
    --scan-history
```

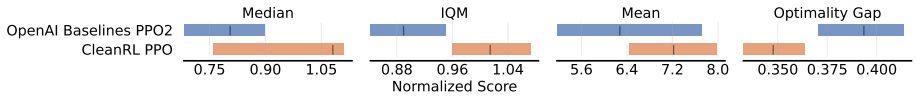


Figure B.3: Clean RL PPO vs. OpenAI Baselines PPO, normalized score (RLiable).

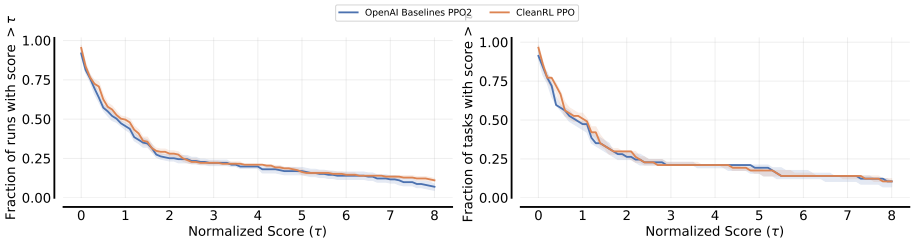


Figure B.4: Clean RL PPO vs. OpenAI Baselines PPO, performance profile (RLiable).

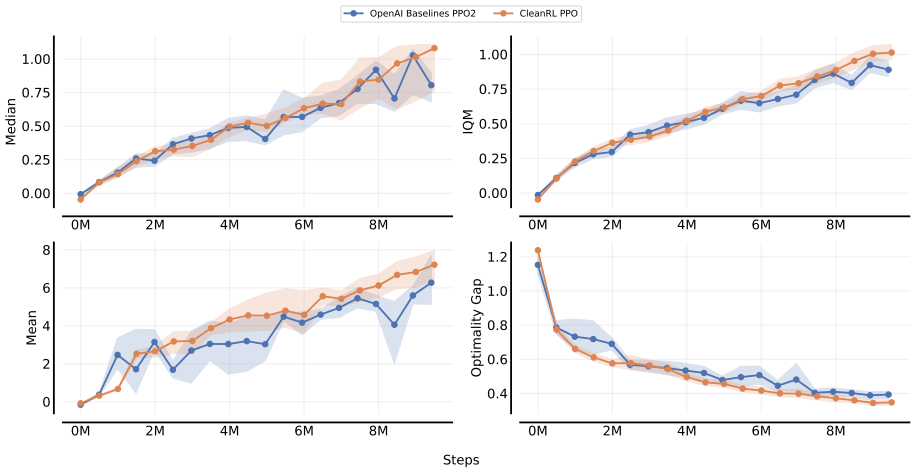


Figure B.5: Clean RL PPO vs. OpenAI Baselines PPO, sample efficiency (RLiable).

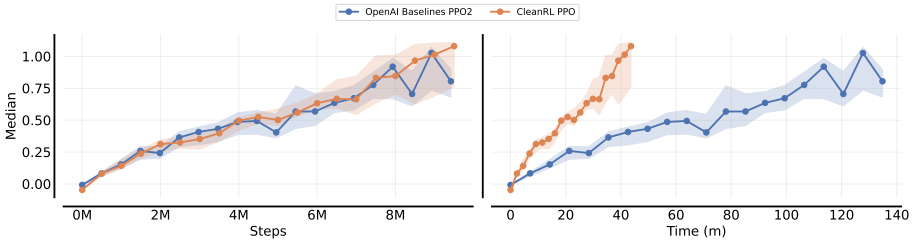


Figure B.6: Clean RL PPO vs. OpenAI Baselines PPO, walltime efficiency (RLiable).

B.1.3 Multi-metrics

Sometimes, such as in multi-objective RL (MORL), it is useful to report multiple metrics in the paper. Hence, the CLI includes an option to plot multiple metrics. Below is an example of CLI and resulting plots (Figure B.7) for multiple MORL algorithms on different environments.

```
python -m openrlbenchmark.rlops_multi_metrics \
  --filters '?we=openrlbenchmark&wpn=MORL-Baselines&ceik=env_id&cen=algo&metrics=eval/
      hypervolume&metrics=eval/igd&metrics=eval/sparsity&metrics=eval/mul' \
  'Pareto Q-Learning?cl=Pareto Q-Learning' \
  'MultiPolicy MO Q-Learning?cl=MPMOQL' \
  'MultiPolicy MO Q-Learning (OLS)?cl=MPMOQL (OLS)' \
  'MultiPolicy MO Q-Learning (GPI-LS)?cl=MPMOQL (GPI-LS)' \
  --env-ids deep-sea-treasure-v0 deep-sea-treasure-concave-v0 fruit-tree-v0 \
  --pc.ncols 3 \
  --pc.ncols-legend 4 \
  --pc.xlabel 'Training steps' \
  --pc.ylabel '' \
  --pc.max_steps 400000 \
  --output-filename morl/morl_deterministic_envs \
  --scan-history
```

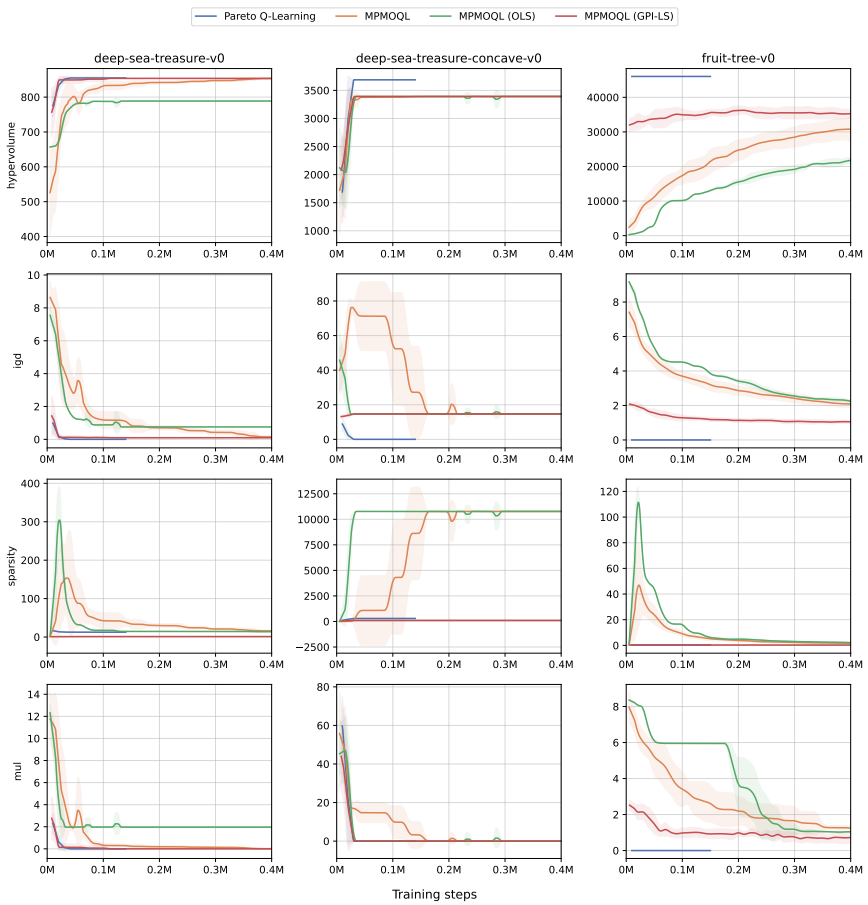


Figure B.7: Plotting different metrics for different environments.

B.2 Using a custom script

Our CLI proves highly beneficial for generating standard RL plots, as demonstrated above. Nevertheless, in certain specialized cases, researchers may wish to expose the data in an alternative format. Fortunately, all the data hosted in Open RL Benchmark is accessible through the W&B API. The following example illustrates how this API can be utilized. From there, researchers can employ any custom script for plotting this data to suit their specific needs. A simple example of such a script is given below, and the corresponding generated plot is shown in Figure B.8.

```
import matplotlib.pyplot as plt
import wandb

project_name = "sb3"
run_id = "0a1kqgev"

api = wandb.Api()
run = api.run(f"openrlbenchmark/{project_name}/{run_id}")
history = run.history(keys=["global_step", "eval/mean_reward"])
plt.plot(history["global_step"], history["eval/mean_reward"])
plt.title(run.name)
plt.savefig("custom_plot.png")
```

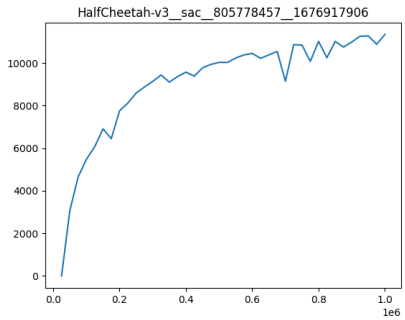


Figure B.8: Example of a plot created with a custom script, by importing data directly from Open RL Benchmark using the W&B API.

Appendix C

Additional Details for the Open RL Benchmark Case Study

This appendix gives additional results related to the first case study presented in Section 5.2.3. Figure C.2 shows the results by environment for the Atari benchmark, and Figure C.1 shows them for the MuJoCo and Box2d benchmarks. The command lines used to generate these figures are as follows.

```
python -m openrlbenchmark.rlops \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'ppo?cl
        =PPO' \
    --filters '?we=modanesh&wpn=openrlbenchmark&ceik=env&cen=algo&metric=eval/mean_reward' '
        ppo?cl=PPO w/ MC for value estimation' \
    --env-ids BreakoutNoFrameskip-v4 SpaceInvadersNoFrameskip-v4 SeaquestNoFrameskip-v4
        EnduroNoFrameskip-v4 PongNoFrameskip-v4 QbertNoFrameskip-v4 BeamRiderNoFrameskip-v4
         \
    --no-check-empty-runs \
    --pc.ncols 3 \
    --pc.ncols-legend 2 \
    --rliable \
    --rc.score_normalization_method atari \
    --rc.normalized_score_threshold 8.0 \
    --rc.sample_efficiency_plots \
    --rc.sample_efficiency_and_walltime_efficiency_method Median \
    --rc.performance_profile_plots \
    --rc.aggregate_metrics_plots \
    --rc.sample_efficiency_num_bootstrap_reps 1000 \
    --rc.performance_profile_num_bootstrap_reps 1000 \
    --rc.interval_estimates_num_bootstrap_reps 1000 \
    --output-filename static/gae_for_ppo_value_atari_per_env \
    --scan-history \
    --rc.sample_efficiency_figsize 7 4

python -m openrlbenchmark.rlops \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'ppo?cl
        =PPO' \
    --filters '?we=modanesh&wpn=openrlbenchmark&ceik=env&cen=algo&metric=eval/mean_reward' '
        ppo?cl=PPO w/ MC for value estimation' \
    --env-ids InvertedDoublePendulum-v2 InvertedPendulum-v2 Reacher-v2 HalfCheetah-v3 Hopper-
        v3 Swimmer-v3 Walker2d-v3 LunarLander-v2 \
    --no-check-empty-runs \
    --pc.ncols 3 \
    --pc.ncols-legend 2 \
    --rliable \
    --rc.normalized_score_threshold 1.0 \
    --rc.sample_efficiency_plots \
```

```
--rc.sample_efficiency_and_walltime_efficiency_method Median \
--rc.performance_profile_plots \
--rc.aggregate_metrics_plots \
--rc.sample_efficiency_num_bootstrap_reps 1000 \
--rc.performance_profile_num_bootstrap_reps 1000 \
--rc.interval_estimates_num_bootstrap_reps 1000 \
--output-filename static/gae_for_ppo_value_mujoco_per_env \
--scan-history \
--rc.sample_efficiency_figsize 7 4
```

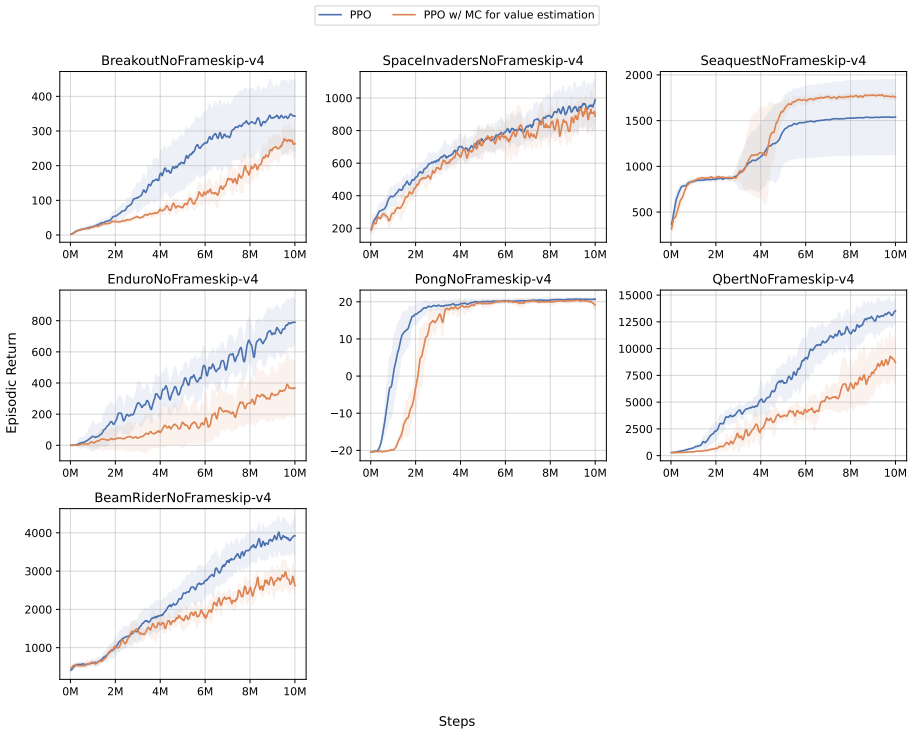


Figure C.1: Comparison between the original PPO and the PPO with MC value estimates in various MuJoCo and Box2D environments. Plots represent the evolution of the episodic return as a function of the number of interactions with the environment, and shaded areas represent the standard deviation.

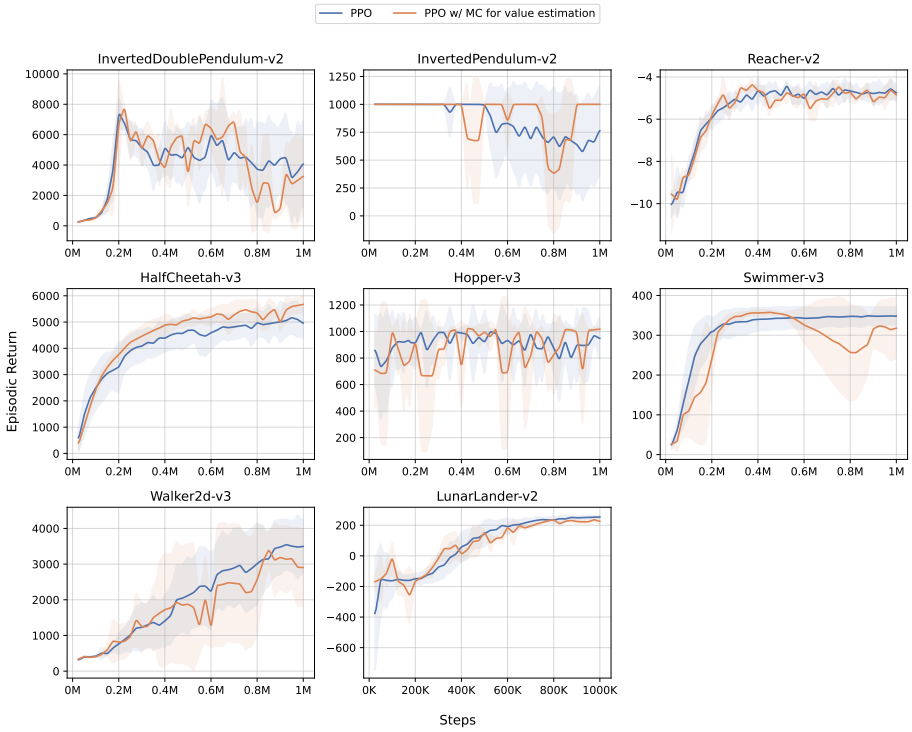


Figure C.2: Comparison between the original PPO and the PPO with MC value estimates in various MuJoCo and Box2D environments. Plots represent the evolution of the episodic return as a function of the number of interactions with the environment, and shaded areas represent the standard deviation.

Appendix D

Refine the MuJoCo benchmark with Stable-Baselines3

In this appendix, we present a brief overview of the learning results for the algorithms implemented in Stable-Baselines3 [222] tested on the MuJoCo benchmark [29, 268], whose data is contained in Open RL Benchmark. At the time of writing, data from 757 runs has been used, unevenly distributed between the different experiments. It is important to emphasise that the optimization of hyperparameters and the training budget vary from one experiment to another. Consequently, the results should be interpreted with caution. All the hyperparameters and raw data used to generate these curves are available on Open RL Benchmark. Figure D.1 shows the aggregation of the final performances following the recommendations of Agarwal et al. [3], and Figure D.2 the corresponding performance profiles. Figure D.3 shows the learning curves as a function of the number of interactions.

The command used to generate Figures D.1, D.2 and D.3 is as follows[1].

```
python -m openrlbenchmark.rlops \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'trpo?
        cl=TRPO' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'ddpg?
        cl=DDPG' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'a2c?cl
        =A2C' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'ppo?cl
        =PPO' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' '
        ppo_lstm?cl=PPO LSTM' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'sac?cl
        =SAC' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'td3?cl
        =TD3' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'ars?cl
        =ARS' \
    --filters '?we=openrlbenchmark&wpn=sb3&ceik=env&cen=algo&metric=eval/mean_reward' 'tqc?cl
        =TQC' \
    --env-ids Ant-v3 BipedalWalker-v3 BipedalWalkerHardcore-v3 HalfCheetah-v3 Hopper-v3
        Humanoid-v3 Swimmer-v3 Walker2d-v3 \
```

[1] For Figure D.3, we're omitting ARS as it was run with many more steps, and its inclusions hinder readability.

```
--no-check-empty-runs \
--pc.ncols 2 \
--pc.ncols-legend 4 \
--rliable \
--rc.normalized_score_threshold 1.0 \
--output-filename static/mujoco_sb3 \
--scan-history
```

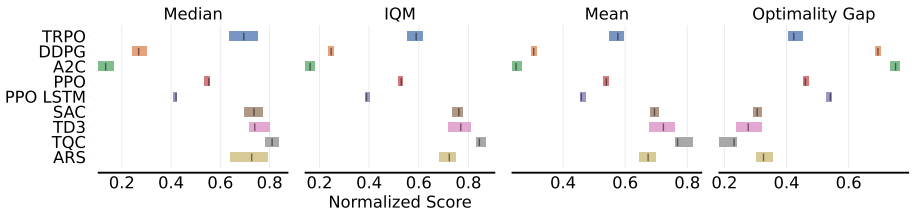


Figure D.1: Aggregated final normalized episodic return with 95% stratified bootstrap CIs on the MuJoCo benchmark of the algorithms integrated into Stable-Baselines3.

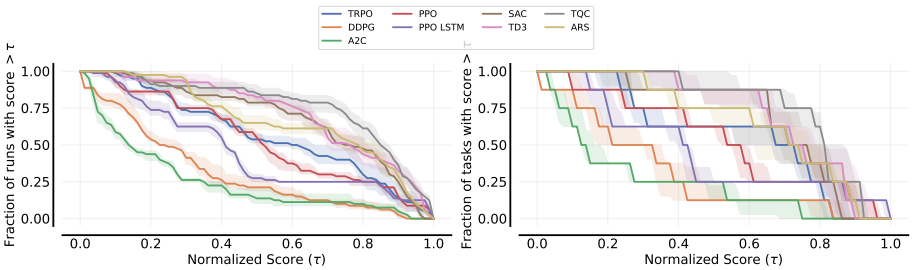


Figure D.2: Performance profile of algorithms implemented using Stable-Baselines3 [222] on the MuJoCo benchmark [268]. Scores are normalized using the min-max method.

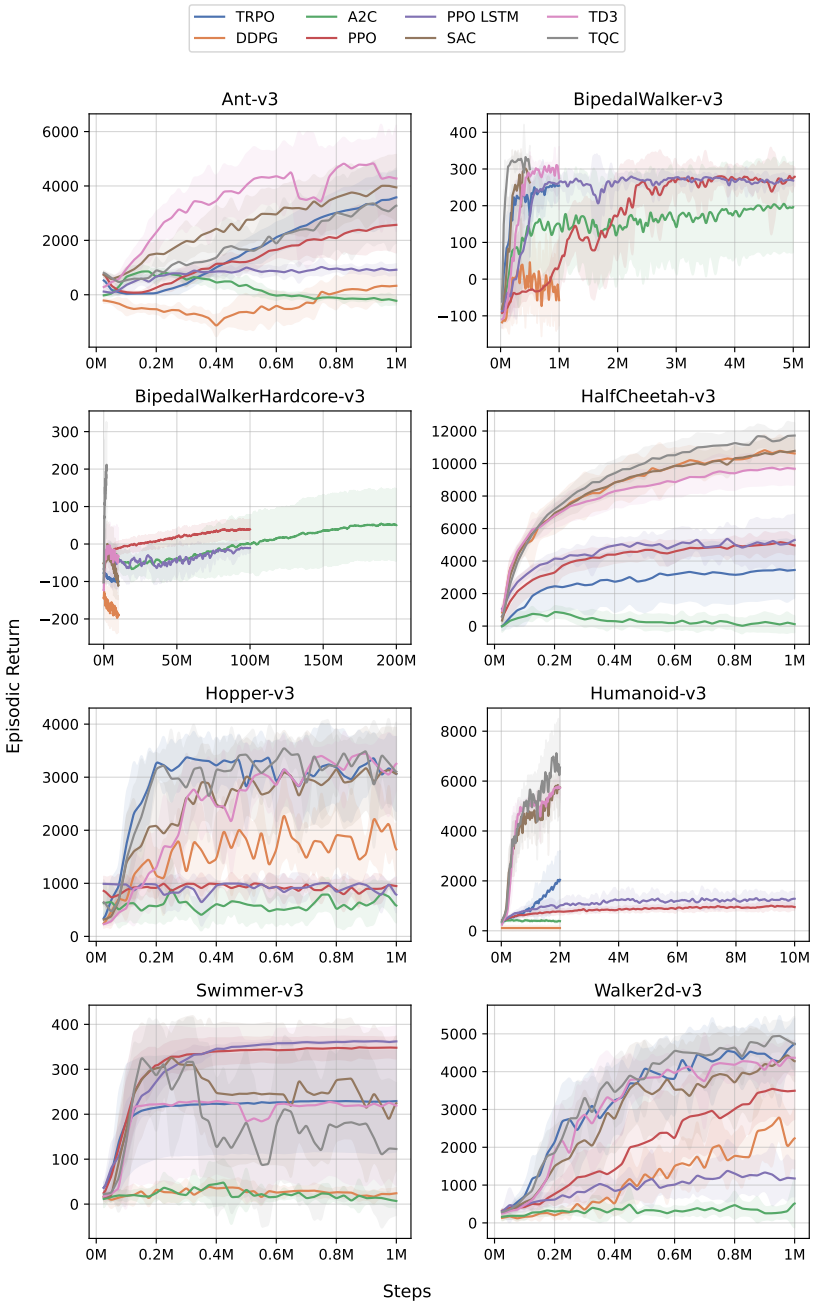


Figure D.3: Sample efficiency curves for algorithms on the MuJoCo Benchmark [268]. This graph presents the mean episodic return for algorithms implemented using Stable-Baselines3 [222], averaged across a minimum of 10 runs (refer to Open RL Benchmark for specific run counts). Data points are subsampled to 10,000 and interpolated for clarity. The curves are smoothed using a rolling average with a window size of 100. The shaded regions around each curve indicate the standard deviation.

Bibliography

[1] Joshua Achiam. Spinning Up in Deep Reinforcement Learning. `https://github.com/openai/spinningup`, 2018. URL `https://github.com/openai/spinningup`.

[2] Joshua Achiam and Shankar Sastry. Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning. *arXiv preprint arXiv:1703.01732*, 2017.

[3] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, volume 34, pages 29304–29320, 2021.

[4] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Reincarnating Reinforcement Learning: Reusing Prior Computation to Accelerate Progress. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/ba1c5356d9164bb64c446a4b690226b0-Abstract-Conference.html`.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-Generation Hyperparameter Optimization Framework. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2623–2631. Association for Computing Machinery, 2019.

[6] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a Visual Language Model for Few-Shot

Learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html.

[7] Lucas N. Alegre, Florian Felten, El-Ghazali Talbi, Grégoire Danoy, Ann Nowé, Ana L. C. Bazzan, and Bruno C. da Silva. MO-Gym: A Library of Multi-Objective Reinforcement Learning Environments. In *Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn 2022*, 2022.

[8] Adrien Ali Taïga, William Fedus, Marlos C. Machado, Aaron C. Courville, and Marc G. Bellemare. On Bonus-Based Exploration Methods in the Arcade Learning Environment. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[9] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5048–5058, 2017.

[10] Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. *arXiv preprint arXiv:2006.05990*, 2020.

[11] Anonymous. Understanding When Dynamics-Invariant Data Augmentations Benefit Model-Free Reinforcement Learning Updates. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. under review.

[12] Anthropic. Introducing Claude. https://www.anthropic.com/index/introducing-claudeAnthropic, 2023. [Accessed 02-01-2024].

[13] Brenna D. Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics Auton. Syst.*, 57(5):469–483, 2009. doi: 10.1016/J.ROBOT.2008.10.024. URL https://doi.org/10.1016/j.robot.2008.10.024.

[14] Arthur Aubret, Laëtitia Matignon, and Salima Hassas. A Survey on Intrinsic Motivation in Reinforcement Learning. *arXiv preprint arXiv:1908.06976*, 2019.

[15] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256,

2002. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

[16] Sriharshitha Ayyalasomayajula. A Deep Hierarchical Variational Autoencoder for World Models in Complex Reinforcement Learning Environments. Master's thesis, Portland State University, 2023.

[17] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1726–1734. AAAI Press, 2017. doi: 10.1609/AAAI.V31I1.10916. URL https://doi.org/10.1609/aaai.v31i1.10916.

[18] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the Atari Human Benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 507–517. PMLR, 2020. URL http://proceedings.mlr.press/v119/badia20a.html.

[19] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never Give Up: Learning Directed Exploration Strategies. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Sye57xStvB.

[20] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed Distributional Deterministic Policy Gradients. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[21] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. doi: 10.1613/JAIR.3912. URL https://doi.org/10.1613/jair.3912.

[22] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, volume 29, pages 1471–1479. Curran Associates, Inc., 2016.

[23] Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. In Doina Precup and Yee Whye Teh,

editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 2017. URL `http://proceedings.mlr.press/v70/bellemare17a.html`.

[24] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[25] Lukas Biewald. Experiment Tracking with Weights and Biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

[26] Christian Bitter, Timo Thun, and Tobias Meisen. Karolos: An Open-Source Reinforcement Learning Framework for Robot-Task Environments. *arXiv preprint arXiv:2212.00906*, 2022.

[27] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021.

[28] Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. TorchRL: A Data-Driven Decision-Making Library for Pytorch. *arXiv preprint arXiv:2306.00577*, 2023.

[29] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[30] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael S. Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong T. Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv preprint arXiv:2307.15818*, 2023.

[31] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan

Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

[32] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[33] Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In Sorelle A. Friedler and Christo Wilson, editors, *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91. PMLR, 2018. URL http://proceedings.mlr.press/v81/buolamwini18a.html.

[34] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

[35] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Ségerie, Micah Carroll, Andi Peng, Phillip J. K. Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca D. Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2307.15217*, 2023.

[36] Nicolas Castanet, Olivier Sigaud, and Sylvain Lamprier. Stein Variational Goal Generation for adaptive Exploration in Multi-Goal Reinforcement Learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3714–3731. PMLR, 2023. URL https://proceedings.mlr.press/v202/castanet23a.html.

[37] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning. *arXiv preprint arXiv:1812.06110*, 2018.

[38] Inc. Challenger, Gray & Christmas. May 2023 Layoffs Jump on Tech, Retail, Auto; YTD Hiring Lowest Since 2016. Technical report, Challenger, Gray & Christmas, Inc., May 2023. URL `https://www.challengergray.com/blog/may-2023-layoffs-jump-on-tech-retail-auto-ytd-hiring-lowest-since-2016/`.

[39] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097, 2021.

[40] Xi Chen, Xiao Wang, Soravit Changpinyo, A. J. Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish V. Thapliyal, James Bradbury, and Weicheng Kuo. PaLI: A Jointly-Scaled Multilingual Language-Image Model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=mWVoBz4W0u`.

[41] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=AY8zfZm0tDd`.

[42] Keith Chester. Reinforcement Learning with a Pick and Place Robotic Arm. Blog Post, August 2023. `https://hlfshell.ai/posts/ppo-pick-and-place/`.

[43] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[44] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. *arXiv preprint arXiv:2306.13831*, 2023.

[45] Era Choshen and Aviv Tamar. ContraBAR: Contrastive Bayes-Adaptive Deep RL. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 6005–6027. PMLR, 2023.

[46] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html`.

[47] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 2020. URL `http://proceedings.mlr.press/v119/cobbe20a.html`.

[48] Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic Policy Gradient. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2020–2027. PMLR, 2021.

[49] Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1331–1340. PMLR, 2019.

[50] Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Dominey, and Pierre-Yves Oudeyer. Language as a Cognitive Tool to Imagine Goals in Curiosity Driven Exploration. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pages 3761–3774. Curran Associates, Inc., 2020.

[51] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: a Short Survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

[52] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. A Review of Physics Simulators for Robotic Applications. *IEEE Access*, 9: 51416–51431, 2021.

[53] Jeremy Collins, Alan Hesu, Cody Houff, and Dane Wang. Learning Robotic Tasks From Video Demonstrations. 2022.

[54] Erwin Coumans and Yunfei Bai. PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016.

[55] Kate Crawford and Vladan Joler. Anatomy of an AI System. `https://anatomyof.ai/`, 2018.

[56] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit Quantile Networks for Distributional Reinforcement Learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1104–1113. PMLR, 2018. URL `http://proceedings.mlr.press/v80/dabney18a.html`.

[57] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional Reinforcement Learning With Quantile Regression. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2892–2901. AAAI Press, 2018.

[58] Mehul Damani and Lerrel Pinto. Continuous Goal Sampling: A Simple Technique to Accelerate Automatic Curriculum Learning, 2023.

[59] Apan Dastider and Mingjie Lin. Non-parametric Stochastic Policy Gradient With Strategic Retreat for Non-stationary Environment. In *18th IEEE International Conference on Automation Science and Engineering, CASE 2022, Mexico City, Mexico, August 20-24, 2022*, pages 1377–1384. IEEE, 2022.

[60] Jeffrey Dastin. Amazon Scraps Secret AI Recruiting Tool That Showed Bias Against Women. In *Ethics of Data and Analytics*, pages 296–299. Auerbach Publications, 2022.

[61] Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 465–472. Omnipress, 2011. URL `https://icml.cc/2011/papers/323_icmlpaper.pdf`.

[62] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. MushroomRL: Simplifying Reinforcement Learning Research. *Journal of Machine Learning Research*, 22(131):1–5, 2021. URL `http://jmlr.org/papers/v22/18-056.html`.

[63] Victor Dey. Exploring Panda Gym: A Multi-Goal Reinforcement Learning Environment. Analytics India Magazine Website, August 2021. `https://analyticsindiamag.com/exploring-panda-gym-a-multi-goal-reinforcement-learning-environment/`.

[64] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI Baselines. `https://github.com/openai/baselines`, 2017. URL `https://github.com/openai/baselines`.

[65] Thomas G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000. doi: 10.1613/JAIR.639. URL `https://doi.org/10.1613/jair.639`.

[66] Han Dongqi. *Self-Organization of Action Hierarchy and Inferring Latent States in Deep Reinforcement Learning With Stochastic Recurrent Neural Networks*. PhD thesis, Okinawa Institute of Science and Technology Graduate University, 2022.

[67] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[68] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An Embodied Multimodal Language Model. *arXiv preprint arXiv:2303.03378*, 2023.

[69] Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. Temporal Disentanglement of Representations for Improved Generalisation in Reinforcement Learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[70] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-Explore: A New Approach for Hard-Exploration Problems. *arXiv preprint arXiv:1901.10995*, 2019.

[71] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First Return, Then Explore. *Nature*, 590(7847):580–586, 2021. doi: 10.1038/S41586-020-03157-9. URL `https://doi.org/10.1038/s41586-020-03157-9`.

[72] Steven Eisinger, Tim de Jager MSc, and Roland Tóth. Deep Learning Based Motion Planning for Joint Position-Controlled Robotic Manipulators. Master's thesis, Eindhoven University of Technology, 2022.

[73] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation

Matters in Deep RL: A Case Study on PPO and TRPO. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1etN1rtPB.

[74] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

[75] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. doi: 10.1007/S11263-009-0275-4.

[76] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, volume 32, pages 15220–15231. Curran Associates, Inc., 2019.

[77] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity Is All You Need: Learning Skills Without a Reward Function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[78] Florian Felten, Lucas Nunes Alegre, Ann Nowe, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno Castro da Silva. A Toolkit for Reliable Benchmarking and Research in Multi-Objective Reinforcement Learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 3, NeurIPS Datasets and Benchmarks 2023*, 2023. URL https://openreview.net/forum?id=jfwRLudQyj.

[79] Sakib Ferdous, Ibne Farabi Shihab, Ratul Chowdhury, and Nigel F. Reuel. Reinforcement Learning-Guided Control Strategies for CAR T-Cell Activation and Expansion. *bioRxiv*, 2023.

[80] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

[81] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1514–1523. PMLR, 2018.

[82] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2137–2145, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html.

[83] Brody Ford. IBM to Pause Hiring for Back-Office Jobs That AI Could Kill. *Bloomberg*, may 2023. URL https://www.bloomberg.com/news/articles/2023-05-01/ibm-to-pause-hiring-for-back-office-jobs-that-ai-could-kill.

[84] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks For Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rywHCPkAW.

[85] Wikimedia Foundation. Wikimedia Downloads.

[86] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html.

[87] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 2019. URL http://proceedings.mlr.press/v97/fujimoto19a.html.

[88] Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. ChainerRL: A Deep Reinforcement Learning Library. *Journal of Machine Learning Research*, 22(77):1–14, 2021. URL http://jmlr.org/papers/v22/20-376.html.

[89] Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári. Multi-criteria Reinforcement Learning. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 197–205. Morgan Kaufmann, 1998.

[90] Quentin Gallouédec and Emmanuel Dellandréa. Cell-Free Latent Go-Explore. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10571–10586. PMLR, 2023.

[91] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

[92] Quentin Gallouédec, Edward Beeching, Clément Romac, and Dellandréa Emmanuel. Jack of All Trades, Expert of Some, a Multi-Purpose Transformer Agent, 2024.

[93] The garage contributors. Garage: A Toolkit for Reproducible Reinforcement Learning Research. `https://github.com/rlworkgroup/garage`, 2019.

[94] Abraham George and Amir Barati Farimani. One ACT Play: Single Demonstration Behavior Cloning With Action Chunking Transformers. *arXiv preprint arXiv:2309.10175*, 2023.

[95] Abraham George, Alison Bartsch, and Amir Barati Farimani. Minimizing Human Assistance: Augmenting a Single Demonstration for Deep Reinforcement Learning. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 5027–5033. IEEE, 2023.

[96] Abraham George, Alison Bartsch, and Amir Barati Farimani. OpenVR: Teleoperation for Manipulation. *arXiv preprint arXiv:2305.09765*, 2023.

[97] Sean Gillen. *Improving Reinforcement Learning for Robotics With Control and Dynamical Systems Theory*. PhD thesis, University of California, Santa Barbara, USA, 2022.

[98] Sean Gillen, Asutay Ozmen, and Katie Byl. Direct Random Search for Fine Tuning of Deep Reinforcement Learning Policies. *arXiv preprint arXiv:2109.05604*, 2021.

[99] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational Intrinsic Control. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Skc-Fo4Yg`.

[100] Marek Grzes. Reward Shaping in Episodic Reinforcement Learning. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 565–573. ACM, 2017. URL http://dl.acm.org/citation.cfm?id=3091208.

[101] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for Reinforcement Learning in TensorFlow. https://github.com/tensorflow/agents, 2018. URL https://github.com/tensorflow/agents.

[102] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and Inference at Scale Made Simple, Efficient and Adaptable. https://github.com/huggingface/accelerate, 2022.

[103] Çaglar Gülçehre, Tom Le Paine, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil C. Rabinowitz, Duncan Williams, Gabriel Barth-Maron, Ziyu Wang, Nando de Freitas, and Worlds Team. Making Efficient Use of Demonstrations to Solve Hard Exploration Problems. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[104] David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

[105] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

[106] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104*, 2023.

[107] Xiangkun He and Chen Lv. Robotic Control in Adversarial and Sparse Reward Environments: A Robust Goal-Conditioned Reinforcement Learning Approach. *IEEE Transactions on Artificial Intelligence*, pages 1–10, 2023.

[108] Jascha Hellwig, Mark Baierl, João Carvalho, Julen Urain, and Jan Peters. A Hierarchical Approach to Active Pose Estimation. *arXiv preprint arXiv:2203.03919*, 2022.

[109] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3207–3214. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11694. URL https://doi.org/10.1609/aaai.v32i1.11694.

[110] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3215–3222. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11796. URL https://doi.org/10.1609/aaai.v32i1.11796.

[111] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[112] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4565–4573, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html.

[113] Matthew W. Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Nikola Momchev, Danila Sinopalnikov, Piotr Stańczyk, Sabela Ramos, Anton Raichuk, Damien Vincent, Léonard Hussenot, Robert Dadashi, Gabriel Dulac-Arnold, Manu Orsini, Alexis Jacq, Johan Ferret, Nino Vieillard, Seyed Kamyar Seyed Ghasemipour, Sertan Girgin, Olivier Pietquin, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, Sarah Henderson, Abe Friesen, Ruba Haroun, Alex Novikov, Sergio Gómez Colmenarejo, Serkan Cabi, Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Andrew Cowie, Ziyu Wang, Bilal Piot, and Nando de Freitas. Acme: A Research Framework for Distributed Reinforcement Learning. *arXiv preprint arXiv:2006.00979*, 2020.

[114] Rein Houthooft, Xi Chen, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational Information Maximizing

Exploration. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, volume 29, pages 1109–1117. Curran Associates, Inc., 2016.

[115] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 Implementation Details of Proximal Policy Optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

[116] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

[117] Shengyi Huang, Anssi Kanervisto, Antonin Raffin, Weixun Wang, Santiago Ontañón, and Rousslan Fernand Julien Dossa. A2C is a special case of PPO. *arXiv preprint arXiv:2205.09123*, 2022.

[118] Shengyi Huang, Quentin Gallouédec, Florian Felten, Antonin Raffin, Rousslan Fernand Julien Dossa, Yanxiao Zhao, Ryan Sullivan, Viktor Makoviychuk, Denys Makoviichuk, Cyril Roumégous, Jiayi Weng, Chufan Chen, Masudur Rahman, João G. M. Araújo, Guorui Quan, Daniel Tan, Timo Klein, Rujikorn Charakorn, Mark Towers, Yann Berthelot, Kinal Mehta, Dipam Chakraborty, Arjun KG, Valentin Charraut, Chang Ye, Zichen Liu, Lucas N. Alegre, Jongwook Choi, and Brent Yi. Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning. May 2023.

[119] Shengyi Huang, Jiayi Weng, Rujikorn Charakorn, Min Lin, Zhongwen Xu, and Santiago Ontañón. Cleanba: A Reproducible and Efficient Distributed Reinforcement Learning Platform, 2023.

[120] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4455–4464. PMLR, 2020. URL `http://proceedings.mlr.press/v119/huang20d.html`.

[121] Conor Igoe, Swapnil Pande, Siddarth Venkatraman, and Jeff G. Schneider. Multi-Alpha Soft Actor-Critic: Overcoming Stochastic Biases in Maximum Entropy Reinforcement Learning. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 7162–7168. IEEE, 2023.

[122] International Energy Agency. Data Centres & Networks, 2022. URL `https://www.iea.org/reports/digitalisation-and-energy/data-centres-and-data-transmission-networks`.

[123] Sardor Israilov, Li Fu, Jesús Sánchez-Rodríguez, Franco Fusco, Guillaume Allibert, Christophe Raufaste, and Médéric Argentina. Reinforcement Learning Approach to Control an Inverted Pendulum: A General Framework for Educational Purposes. *PLoS ONE*, 18(2):e0280071, 2023.

[124] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to Trust Your Model: Model-Based Policy Optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12498–12509, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html`.

[125] Pingcheng Jian, Easop Lee, Zachary Bell, Michael M Zavlanos, and Boyuan Chen. Policy Stitching: Learning Transferable Robot Policies. In *7th Annual Conference on Robot Learning*, 2023.

[126] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General Robot Manipulation with Multimodal Prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[127] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-Free Exploration for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 4870–4879. PMLR, 2020.

[128] Tom Jurgenson and Aviv Tamar. Goal-Conditioned Supervised Learning With Sub-Goal Prediction. *arXiv preprint arXiv:2305.10171*, 2023.

[129] Ankush k Singal. Deep Q-Learning to Actor-Critic using Robotics Simulations with Panda-Gym. Medium, October 2023. `https://medium.com/@andysingal/deep-q-learning-to-actor-critic-using-robotics-simulations-with-panda-gym-ff220f980366`.

[130] Leslie Pack Kaelbling. Learning to Achieve Goals. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1094–1099. Morgan Kaufmann, 1993.

[131] Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent Experience Replay in Distributed Reinforcement Learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=r1lyTjAqYX`.

[132] Jayasekara Kapukotuwa, Brian Lee, Declan Devine, and Yuansong Qiao. MultiROS: ROS-Based Robot Simulation Environment for Concurrent Deep Reinforcement Learning. In *18th IEEE International Conference on Automation Science and Engineering, CASE 2022, Mexico City, Mexico, August 20-24, 2022*, pages 1098–1103. IEEE, 2022.

[133] Byeongjun Kim, Gunam Kwon, Chaneun Park, and Nam Kyu Kwon. The Task Decomposition and Dedicated Reward-System-Based Reinforcement Learning Algorithm for Pick-and-Place. *Biomimetics*, 8(2), 2023.

[134] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[135] Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. Empowerment: A Universal Agent-Centric Measure of Control. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*, pages 128–135. IEEE, 2005. doi: 10.1109/CEC.2005.1554676. URL https://doi.org/10.1109/CEC.2005.1554676.

[136] Ilya Kostrikov. JAXRL: Implementations of Reinforcement Learning algorithms in JAX. https://github.com/ikostrikov/jaxrl, Oct 2021. URL https://github.com/ikostrikov/jaxrl.

[137] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-2012. URL https://doi.org/10.18653/v1/d18-2012.

[138] Alexander Kuhnle, Michael Schaarschmidt, and Kai Fricke. Tensorforce: a TensorFlow library for applied reinforcement learning. https://github.com/tensorforce/tensorforce, 2017. URL https://github.com/tensorforce/tensorforce.

[139] Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[140] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

[141] Swagat Kumar, Hayden Sampson, and Ardhendu Behera. Benchmarking Deep Reinforcement Learning Algorithms for Vision-Based Robotics. *arXiv preprint arXiv:2201.04224*, 2022.

[142] Yi-Hung Kung, Pei-Sheng Lin, and Cheng-Hsiung Kao. An Optimal k-Nearest Neighbor for Density Estimation. *Statistics & Probability Letters*, 82(10):1786–1791, 2012.

[143] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-Ensemble Trust-Region Policy Optimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=SJJinbWRZ`.

[144] Heinrich Küttler, Nantas Nardelli, Thibaut Lavril, Marco Selvatici, Viswanath Sivakumar, Tim Rocktäschel, and Edward Grefenstette. TorchBeast: A PyTorch Platform for Distributed RL. *arXiv preprint arXiv:1910.03552*, 2019.

[145] Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack Learning Environment. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pages 7671–7684. Curran Associates, Inc., 2020.

[146] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry P. Vetrov. Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5556–5566. PMLR, 2020. URL `http://proceedings.mlr.press/v119/kuznetsov20a.html`.

[147] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the Carbon Emissions of Machine Learning. *arXiv preprint arXiv:1910.09700*, 2019.

[148] Paweł Ładosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in Deep Reinforcement Learning: A Survey. *Information Fusion*, 85:1–22, 2022. ISSN 1566-2535.

[149] Nathan Lambert, Louis Castricato, Leandro von Werra, and Alex Havrilla. Illustrating Reinforcement Learning from Human Feedback (RLHF). *Hugging Face Blog*, 2022. URL `https://huggingface.co/blog/rlhf`.

[150] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch Reinforcement Learning. In Marco A. Wiering and Martijn van Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 45–73. Springer, 2012. doi: 10.1007/978-3-642-27645-3_2. URL `https://doi.org/10.1007/978-3-642-27645-3_2`.

[151] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander M. Rush, Douwe Kiela, Matthieu Cord, and Victor Sanh. OBELISC: An Open Web-Scale Filtered Dataset of Interleaved Image-Text Documents. *arXiv preprint arXiv:2306.16527*, 2023.

[152] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao

Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gerard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Alexandra Luccioni, and Yacine Jernite. The BigScience ROOTS Corpus: A 1.6TB Composite Multilingual Dataset. *arXiv preprint arXiv:2303.03915*, 2023.

[153] Yann LeCun. A Path Towards Autonomous Machine Intelligence Version 0.9. 2, 2022-06-27. *Open Review*, 62(1), 2022.

[154] Joonho Lee, Jemin Hwangbo, and Marco Hutter. Robust Recovery Controller for a Quadrupedal Robot Using Deep Reinforcement Learning. *arXiv preprint arXiv:1901.07517*, 2019.

[155] Kuang-Huei Lee, Ofir Nachum, Mengjiao (Sherry) Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, and Igor Mordatch. Multi-Game Decision Transformers. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, volume 35, pages 27921–27936. Curran Associates, Inc., 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/b2cac94f82928a85055987d9fd44753f-Abstract-Conference.html`.

[156] Joel Lehman and Kenneth O Stanley. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2): 189–223, 2011.

[157] Joel Z. Leibo, Vinícius Flores Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 464–473. ACM, 2017. URL `http://dl.acm.org/citation.cfm?id=3091194`.

[158] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-François Goudou, and David Filliat. State Representation Learning for Control: An Overview. *Neural Networks*, 108:379–392, 2018. ISSN 0893-6080.

[159] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Inf. Fusion*, 58:52–68, 2020. doi: 10.1016/J.INFFUS.2019.12.004. URL `https://doi.org/10.1016/j.inffus.2019.12.004`.

[160] Edouard Leurent. An Environment for Autonomous Driving Decision-Making. `https://github.com/eleurent/highway-env`, 2018. URL `https://github.com/eleurent/highway-env`.

[161] Sergey Levine. Understanding the World Through Action. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 1752–1757. PMLR, 2021.

[162] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643*, 2020.

[163] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for Distributed Reinforcement Learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3059–3068. PMLR, 2018. URL `http://proceedings.mlr.press/v80/liang18b.html`.

[164] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control With Deep Reinforcement Learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[165] Long-Ji Lin. Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8(3-4):293–321, 1992.

[166] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48.

[167] Michael L. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In William W. Cohen and Haym Hirsh, editors, *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pages 157–163. Morgan Kaufmann, 1994. doi: 10.1016/B978-1-55860-335-6.50027-1. URL `https://doi.org/10.1016/b978-1-55860-335-6.50027-1`.

[168] Hao Liu and Pieter Abbeel. APS: Active Pretraining with Successor Features. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6736–6747. PMLR, 2021.

[169] Hao Liu and Pieter Abbeel. Behavior From the Void: Unsupervised Active Pre-Training. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, volume 34, 2021.

[170] Hao Liu and Pieter Abbeel. Emergent Agentic Transformer from Chain of Hindsight Experience. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 21362–21374. PMLR, 2023.

[171] Haoxiang Liu, Ren Komatsu, Shinsuke Nakashima, Hiroyuki Hamada, Nobuto Matsuhira, Hajime Asama, and Atsushi Yamashita. Viewpoint Selection for the Efficient Teleoperation of a Robot Arm Using Reinforcement Learning. *IEEE Access*, 11:119647–119658, 2023.

[172] Andrew Lobbezoo and Hyock-Ju Kwon. Simulated and Real Robotic Reach, Grasp, and Pick-And-Place Using Combined Reinforcement Learning and Traditional Controls. *Robotics*, 12(1):12, 2023.

[173] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6379–6390, 2017.

[174] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *Journal of Machine Learning Research*, 24:253:1–253:15, 2023. URL `http://jmlr.org/papers/v24/23-0069.html`.

[175] Nicolai A. Lynnerup, Laura Nolling, Rasmus Hasle, and John Hallam. A Survey on Reproducibility by Evaluating Deep Reinforcement Learning Algorithms on Real-World Robots. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 466–489. PMLR, 2019. URL `http://proceedings.mlr.press/v100/lynnerup20a.html`.

[176] Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. Offline Goal-Conditioned Reinforcement Learning via f-Advantage Regression. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[177] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the Arcade

Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018. doi: 10.1613/JAIR.5699. URL https://doi.org/10.1613/jair.5699.

[178] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-Based Exploration with the Successor Representation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5125–5133. AAAI Press, 2020. doi: 10.1609/AAAI.V34I04.5955. URL https://doi.org/10.1609/aaai.v34i04.5955.

[179] Denys Makoviichuk and Viktor Makoviychuk. rl-games: A High-performance Framework for Reinforcement Learning. https://github.com/Denys88/rl_games, May 2021. URL https://github.com/Denys88/rl_games.

[180] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/28dd2c7955ce926456240b2ff0100bde-Abstract-round2.html.

[181] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3195–3204. Computer Vision Foundation / IEEE, 2019.

[182] Luca Marzari, Ameya Pore, Diego Dall'Alba, Gerardo Aragon-Camarasa, Alessandro Farinelli, and Paolo Fiorini. Towards Hierarchical Task Decomposition Using Deep Reinforcement Learning for Pick and Place Subtasks. In *20th International Conference on Advanced Robotics, ICAR 2021, Ljubljana, Slovenia, December 6-10, 2021*, pages 640–645. IEEE, 2021.

[183] Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. PBCS: Efficient Exploration and Exploitation Using a Synergy Between Reinforcement Learning and Motion Planning. In Igor Farkas, Paolo Masulli, and Stefan Wermter, editors, *Artificial Neural Networks and Machine Learning - ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part II*, volume 12397 of *Lecture Notes in Computer Science*, pages 295–307. Springer, 2020.

[184] Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowl. Eng. Rev.*, 27(1):1–31, 2012. doi:

10.1017/S0269888912000057. URL https://doi.org/10.1017/S0269888912000057.

[185] Vegard Mella, Eric Hambro, Danielle Rothermel, and Heinrich Küttler. moolib: A Platform for Distributed RL. *GitHub repository*, 2022. URL https://github.com/facebookresearch/moolib.

[186] MIT Technology Review. Sustainability Starts With the Data Center. *MIT Technology Review*, 2023. URL https://www.technologyreview.com/2023/11/30/1083909/sustainability-starts-with-the-data-center/.

[187] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*, 2013.

[188] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[189] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1928–1937. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/mniha16.html.

[190] Shakir Mohamed and Danilo Jimenez Rezende. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2125–2133, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/e00406144c1e7e35240afed70f34166a-Abstract.html.

[191] Anthony Moi and Nicolas Patry. HuggingFace's Tokenizers, April 2023.

[192] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 7559–7566. IEEE, 2018. doi: 10.1109/ICRA.2018.8463189. URL https://doi.org/10.1109/ICRA.2018.8463189.

[193] Sriraam Natarajan and Prasad Tadepalli. Dynamic Preferences in Multi-Criteria Reinforcement Learning. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the Twenty-Second International Conference on Machine Learning, ICML 2005, Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 601–608. ACM, 2005.

[194] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 663–670. Morgan Kaufmann, 2000.

[195] Richard Nieva. Meta AI Chief Yann LeCun On His Open Source Mission And The 'Surprise' Of ChatGPT. *Forbes*, November 2023. URL `https://www.forbes.com/sites/richardnieva/2023/11/30/meta-ai-yann-lecun-fair-10th-anniversary/`. Accessed: January 7, 2024.

[196] OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

[197] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019.

[198] Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online, July 2020. Association for Computational Linguistics.

[199] Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-Based Exploration with Neural Density Models. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2721–2730. PMLR, 2017. URL `http://proceedings.mlr.press/v70/ostrovski17a.html`.

[200] Pierre-Yves Oudeyer and Frédéric Kaplan. What Is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers Neurorobotics*, 1:6, 2007. doi: 10.3389/NEURO.12.006.2007. URL `https://doi.org/10.3389/neuro.12.006.2007`.

[201] Pierre-Yves Oudeyer, Frederic Kaplan, and Verena V Hafner. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007. doi: 10.1109/TEVC.2006.890271.

[202] Eli Pariser. *The Filter Bubble: What the Internet Is Hiding From You*. Penguin UK, 2011.

[203] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.06342`.

[204] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`.

[205] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, PMLR, 2017.

[206] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-Supervised Exploration via Disagreement. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5062–5071. PMLR, 2019.

[207] Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical Design in Reinforcement Learning. *arXiv preprint arXiv:2304.01315*, 2023.

[208] Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav S. Sukhatme, and Vladlen Koltun. Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7652–7662. PMLR, 2020. URL `http://proceedings.mlr.press/v119/petrenko20a.html`.

[209] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Hugo Larochelle. Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program). *Journal of Machine Learning Research*, 22:164:1–164:20, 2021. URL `http://jmlr.org/papers/v22/20-303.html`.

[210] Silviu Pitis, Harris Chan, Stephen Zhao, Bradly C. Stadie, and Jimmy Ba. Maximum Entropy Gain Exploration for Long Horizon Multi-Goal Reinforcement Learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7750–7761. PMLR, 2020.

[211] Gabriella Pizzuto, Hetong Wang, Hatem Fakhruldeen, Bei Peng, Kevin S. Luck, and Andrew I. Cooper. Accelerating Laboratory Automation Through Robot Skill Learning for Sample Scraping. *arXiv preprint arXiv:2209.14875*, 2022.

[212] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *arXiv preprint arXiv:1802.09464*, 2018.

[213] Boris T Polyak and Anatoli B Juditsky. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[214] Dean Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*, pages 305–313. Morgan Kaufmann, 1988. URL `http://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network`.

[215] Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-Fit: State-Covering Self-Supervised Reinforcement Learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7783–7792. PMLR, 2020.

[216] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher Algorithms for Curriculum Learning of Deep RL in Continuously Parameterized Environments. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 835–853. PMLR, 2020.

[217] Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. Off-Policy Temporal Difference Learning with Function Approximation. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 417–424. Morgan Kaufmann, 2001.

[218] Aniruddh G Puranic, Jyotirmoy V Deshmukh, and Stefanos Nikolaidis. Signal Temporal Logic-Guided Apprenticeship Learning. *arXiv preprint arXiv:2311.05084*, 2023.

[219] Sébastien Racanière, Andrew K. Lampinen, Adam Santoro, David P. Reichert, Vlad Firoiu, and Timothy P. Lillicrap. Automated Curriculum Generation Through Setter-Solver Interactions. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[220] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9, 2019.

[221] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. ISSN 1533-7928.

[222] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021.

[223] Roberta Raileanu and Tim Rocktäschel. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=rkg-TJBFPB`.

[224] Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL `https://openreview.net/forum?id=1ikK0kHjvj`. Featured Certification, Outstanding Certification.

[225] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015. URL `http://proceedings.mlr.press/v37/rezende15.html`.

[226] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL `https://doi.org/10.1214/aoms/1177729586`.

[227] Frank Röder, Manfred Eppe, and Stefan Wermter. Grounding Hindsight Instructions in Multi-Goal Reinforcement Learning for Robotics. In *IEEE International Conference on Development and Learning, ICDL 2022, London, United Kingdom, September 12-15, 2022*, pages 170–177. IEEE, 2022.

[228] David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Sasha Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla P. Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer T. Chayes, and Yoshua Bengio. Tackling Climate Change with Machine Learning. *ACM Comput. Surv.*, 55(2):42:1–42:96, 2023. doi: 10.1145/3485128. URL `https://doi.org/10.1145/3485128`.

[229] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet: Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/S11263-015-0816-Y.

[230] Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.06295`.

[231] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to Fly. In Derek H. Sleeman and Peter Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning (ML 1992), Aberdeen, Scotland, UK, July 1-3, 1992*, pages 385–393. Morgan Kaufmann, 1992. doi: 10.1016/B978-1-55860-247-2.50055-3. URL `https://doi.org/10.1016/b978-1-55860-247-2.50055-3`.

[232] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[233] Jürgen Schmidhuber. *Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook*. PhD thesis, Technical University of Munich, Germany, 1987. URL `https://mediatum.ub.tum.de/813180`.

[234] Jürgen Schmidhuber. A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 222–227, 1991.

[235] Jürgen Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010. doi: 10.1109/TAMD.2010.2056368. URL `https://doi.org/10.1109/TAMD.2010.2056368`.

[236] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609, 2020. doi: 10.1038/S41586-020-03051-4. URL `https://doi.org/10.1038/s41586-020-03051-4`.

[237] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust Region Policy Optimization. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org, 2015. URL `http://proceedings.mlr.press/v37/schulman15.html`.

[238] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1506.02438`.

[239] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[240] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Commun. ACM*, 63(12):54–63, 2020. doi: 10.1145/3381831. URL `https://doi.org/10.1145/3381831`.

[241] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to Explore via Self-Supervised World Models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 8583–8592. PMLR, 2020.

[242] Ahmed Rida Sekkat, Yohan Dupuis, Pascal Vasseur, and Paul Honeine. The OmniScape Dataset. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 1603–1608. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9197144. URL `https://doi.org/10.1109/ICRA40945.2020.9197144`.

[243] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[244] Yanpeng Shao, Haibo Zhou, Shuaishuai Zhao, Xiaoyan Fan, and Jiayi Jiang. A Control Method of Robotic Arm Based on Improved Deep Deterministic Policy Gradient. In *2023 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 473–478. IEEE, 2023.

[245] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[246] Yash Shukla and Jivko Sinpov. A Framework for Curriculum Schema Transfer From Low-Fidelity to High-Fidelity Environments. In *3rd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics at RSS 2022*, 2022.

[247] Yash Shukla, Kaleb Loar, Robert Wright, and Jivko Sinapov. An Object-Oriented Approach for Generating Low-Fidelity Environments for Curriculum Schema Transfer. *Scaling Robot Learning Workshop at ICRA 2022*, 2022.

[248] Yash Shukla, Abhishek Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. Automaton-Guided Curriculum Generation for Reinforcement Learning Agents. In Sven Koenig, Roni Stern, and Mauro Vallati, editors, *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling, July 8-13, 2023, Prague, Czech Republic*, pages 605–613. AAAI Press, 2023.

[249] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. PMLR, JMLR.org, 2014.

[250] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.

[251] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the Game of Go Without Human Knowledge. *Nature*, 550(7676):354–359, 2017.

[252] Thomas Simonini. Deep Reinforcement Learning Course. HuggingFace Website, feb 2023. `https://huggingface.co/learn/deep-rl-course`.

[253] Satinder Singh, Tommi S. Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. *Machine Learning*, 38(3):287–308, 2000.

[254] H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin A. Riedmiller, and Matthew M. Botvinick. V-MPO: On-Policy Maximum a Posteriori Policy Optimization for Discrete and Continuous Control. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=SylOlp4FvH`.

[255] Statista Search Department. Carbon Intensity Outlook of the Power Sector in France from 2020 to 2050. `https://www.statista.com/statistics/1190067/carbon-intensity-outlook-of-france/`, October 2020. [Infographic].

[256] Catherine Stupp. Fraudsters Used AI to Mimic CEO's Voice in Unusual Cybercrime Case. *The Wall Street Journal*, August 2019. URL `https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402`.

[257] Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Efficient Multi-Task and Transfer Reinforcement Learning With Parameter-Compositional Framework. *IEEE Robotics and Automation Letters*, 8(8): 4569–4576, 2023.

[258] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2443–2451. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00252. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Sun_Scalability_in_Perception_for_Autonomous_Driving_Waymo_Open_Dataset_CVPR_2020_paper.html.

[259] Richard S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In Bruce W. Porter and Raymond J. Mooney, editors, *Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990*, pages 216–224. Morgan Kaufmann, 1990. doi: 10.1016/B978-1-55860-141-3.50030-4. URL https://doi.org/10.1016/b978-1-55860-141-3.50030-4.

[260] Richard S. Sutton. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bulletin*, 2(4):160–163, 1991. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

[261] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[262] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

[263] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2753–2762, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html.

[264] Yunhao Tang and Alp Kucukelbir. Hindsight Expectation Maximization for Goal-Conditioned Reinforcement Learning. In *24th International Con-*

ference on Artificial Intelligence and Statistics, volume 130 of *Proceedings of Machine Learning Research*, pages 2863–2871. PMLR, 2021.

[265] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

[266] Yee Whye Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust Multitask Reinforcement Learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4496–4506, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/0abdc563a06105aee3c6136871c9f4d1-Abstract.html`.

[267] Joan Thiesen, Torben S Christensen, Thomas G Kristensen, Rikke D Andersen, Brit Brunoe, Trine K Gregersen, Mikkel Thrane, and Bo P Weidema. Rebound Effects of Price Differences. *The International Journal of Life Cycle Assessment*, 13:104–114, 2008.

[268] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-Based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 5026–5033. IEEE, 2012.

[269] Marin Toromanoff, Émilie Wirbel, and Fabien Moutarde. Is Deep Reinforcement Learning Really Superhuman on Atari? *arXiv preprint arXiv:1908.04683*, 2019.

[270] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL `https://zenodo.org/record/8127025`.

[271] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236): 433–460, 1950.

[272] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, volume 30, pages 6306–6315. Curran Associates, Inc., 2017.

[273] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100. AAAI Press, 2016.

[274] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[275] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3540–3549. PMLR, 2017. URL `http://proceedings.mlr.press/v70/vezhnevets17a.html`.

[276] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster Level in Starcraft II Using Multi-Agent Reinforcement Learning. *Nature*, 575(7782):350–354, 2019.

[277] Jane Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z. Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt M. Botvinick. Learning to Reinforcement Learn. In Glenn Gunzelmann, Andrew Howes, Thora Tenbrink, and Eddy J. Davelaar, editors, *Proceedings of the 39th Annual Meeting of the Cognitive Science Society, CogSci 2017, London, UK, 16-29 July 2017*. cognitivesciencesociety.org, 2017. URL `https://mindmodeling.org/cogsci2017/papers/0252/index.html`.

[278] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling Network Architectures for Deep Reinforcement Learning. *arXiv preprint arXiv:1511.06581*, 2015.

[279] Christopher John Cornish Hellaby Watkins. *Learning From Delayed Rewards*. PhD thesis, King's College, Cambridge United Kingdom, 1989.

[280] Lex Weaver and Nigel Tao. The Optimal Reward Baseline for Gradient-Based Reinforcement Learning. In Jack S. Breese and Daphne Koller, editors, *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 538–545. Morgan Kaufmann, 2001.

[281] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A Highly Modularized

Deep Reinforcement Learning Library. *Journal of Machine Learning Research*, 23(267):1–6, 2022. URL http://jmlr.org/papers/v23/21-1127.html.

[282] Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A Highly Parallel Reinforcement Learning Environment Execution Engine. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 2, NeurIPS Datasets and Benchmarks 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/8caaf08e49ddbad6694fae067442ee21-Abstract-Datasets_and_Benchmarks.html.

[283] Lilian Weng. A (Long) Peek into Reinforcement Learning. *lilianweng.github.io*, 2018. URL https://lilianweng.github.io/posts/2018-02-19-rl-overview/.

[284] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

[285] Ronald J. Williams and Jing Peng. Function Optimization using Connectionist Reinforcement Learning Algorithms. *Connection Science*, 3(3): 241–268, 1991.

[286] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, October 2020.

[287] Jimmy Xin, Linus Zheng, Jiayi Wei, Kia Rahmani, Jarrett Holtz, Isil Dillig, and Joydeep Biswas. PLUNDER: Probabilistic Program Synthesis for Learning From Unlabeled and Noisy Demonstrations. *arXiv preprint arXiv:2303.01440*, 2023.

[288] Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6190–6199, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f471223d1a1614b58a7dc45c9d01df19-Abstract.html.

[289] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-Task Reinforcement Learning with Soft Modularization. In Hugo Larochelle,

Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/32cfdce9631d8c7906e8e9d6e68b514b-Abstract.html`.

[290] Zhao Yang, Thomas M. Moerland, Mike Preuss, and Aske Plaat. First Go, then Post-Explore: The Benefits of Post-Exploration in Intrinsic Motivation. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, editors, *Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 2, Lisbon, Portugal, February 22-24, 2023*, pages 27–34. SCITEPRESS, 2023.

[291] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 2019.

[292] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Robert Fergus. Deconvolutional Networks. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 2528–2535. IEEE Computer Society, 2010.

[293] Linrui Zhang, Zichen Yan, Li Shen, Shoujie Li, Xueqian Wang, and Dacheng Tao. Safety Correction From Baseline: Towards the Risk-Aware Policy in Robotics via Dual-Agent Reinforcement Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, pages 9027–9033. IEEE, 2022.

[294] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic Curriculum Learning through Value Disagreement. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, volume 33, pages 7648–7659. Curran Associates, Inc., 2020.

[295] Rui Zhao, Yang Gao, Pieter Abbeel, Volker Tresp, and Wei Xu. Mutual Information State Intrinsic Control. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=0thEq8I5v1`.

[296] Yanxiao Zhao. abcdRL: Modular Single-file Reinforcement Learning Algorithms Library. `https://github.com/sdpkjc/abcdrl`, December 2022. URL `https://github.com/sdpkjc/abcdrl`.

[297] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online Decision Transformer. *arXiv preprint arXiv:2202.05607*, 2022.