

Université de Lille Faculté des Sciences et Technologies
École Doctorale Sciences Pour L'ingénieur

THÈSE DE DOCTORAT

Spécialité **Informatique**

présentée par
YANNIS FLET-BERLIAC

APPRENTISSAGE PAR RENFORCEMENT PROFOND EFFICACE POUR LE CONTRÔLE, L'EXPLORATION ET LA SÛRETÉ

sous la direction de **Philippe Preux**.

Soutenue publiquement le **6 octobre 2021** à **Villeneuve d'Ascq** devant le jury composé de

M ^{me} Ann Nowé	AI Lab, Vrije Universiteit Brussel	Rapporteuse
M. Bruno Scherrer	Inria, Université de Lorraine	Rapporteur
M ^{me} Luce Brotcorne	Inria	Présidente
M. Anders Jonsson	Universitat Pompeu Fabra	Examineur
M ^{me} Joëlle Pineau	McGill University	Examinatrice
M. Adam White	University of Alberta	Examineur
M. Philippe Preux	Inria, Université de Lille	Directeur de thèse

Centre de Recherche en Informatique, Signal et Automatique de Lille (CRISTAL),
UMR 9189 Équipe Scool/SequeL, 59650, Villeneuve d'Ascq, France



Lille University Faculty of Science and Technology
Doctoral School Engineering Sciences

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in **Computer Science**

submitted by

YANNIS FLET-BERLIAC

**SAMPLE-EFFICIENT DEEP REINFORCEMENT LEARNING
FOR CONTROL, EXPLORATION AND SAFETY**

under the supervision of **Philippe Preux**.

Publicly defended on the **6th of October, 2021** at **Villeneuve d'Ascq, France**, to the doctoral committee

Ms. Ann Nowé	AI Lab, Vrije Universiteit Brussel	Thesis Referee
Mr. Bruno Scherrer	Inria, Université de Lorraine	Thesis Referee
Ms. Luce Brotcorne	Inria	Committee Chair
Mr. Anders Jonsson	Universitat Pompeu Fabra	Thesis Jury
Ms. Joëlle Pineau	McGill University	Thesis Jury
Mr. Adam White	University of Alberta	Thesis Jury
Mr. Philippe Preux	Inria, Université de Lille	Thesis Supervisor

Research Center in Computer Science, Signal and Automatic Control of Lille (CRIStAL),
UMR 9189 Scool/SequeL team, 59650, Villeneuve d'Ascq, France



Résumé

Un des défis majeurs de l'apprentissage par renforcement est d'explorer efficacement un environnement afin d'apprendre une politique optimale par une méthode à base d'essai-erreur. Pour y parvenir, l'agent doit être capable d'apprendre efficacement de ses expériences passées, ce qui lui permet d'estimer la performance de certaines actions par rapport à d'autres. En outre, une problématique évidente mais centrale est que ce qui n'est pas connu doit être exploré, et la nécessité d'explorer d'une manière sûre ajoute un autre niveau de difficulté au problème. Ce sont les principales questions que nous abordons dans cette thèse de doctorat. En déconstruisant la méthode acteur-critique et en développant des formulations alternatives du problème d'optimisation sous-jacent via la notion de variance, nous explorons comment les algorithmes d'apprentissage par renforcement profond peuvent résoudre plus efficacement les problèmes de contrôle continu, les environnements d'exploration difficiles et les tâches exposées au risque. La première partie de la thèse se concentre sur la composante du critique de l'approche acteur-critique, ou fonction de valeur, et sur la façon d'apprendre plus efficacement à contrôler les agents dans les domaines de contrôle continu par des utilisations distinctes de la variance dans les estimations de la fonction de valeur. La deuxième partie de la thèse s'intéresse à la composante acteur de l'approche acteur-critique, aussi appelée politique. Nous proposons l'introduction d'un troisième élément au problème d'optimisation que les agents résolvent, en introduisant un adversaire. L'adversaire est de même nature que l'agent RL mais il est entraîné à suggérer des actions qui imitent celles de l'acteur ou qui vont à l'encontre des contraintes de notre problème. Il est représenté par une distribution de politique moyenne avec laquelle l'acteur doit différencier son comportement, encourageant finalement l'acteur à mieux explorer dans les tâches où une exploration efficace constitue la difficulté majeure, ou à prendre des décisions de façon moins risquée.

Abstract

One major challenge of reinforcement learning is to efficiently explore an environment in order to learn optimal policies through trial and error. To achieve this, the agent must be able to learn effectively from past experiences, enabling it to form an accurate picture of the benefit of certain actions over others. Beyond that, an obvious but central issue is that what is not known must be explored, and the necessity to explore in a safe way adds another layer of difficulty to the problem. These are the main issues that we address in this Ph.D. thesis. By deconstructing the actor-critic framework and developing alternative formulations of the underlying optimization problem using the notion of variance, we explore how deep reinforcement learning algorithms can more effectively solve continuous control problems, hard exploration environments and risk-sensitive tasks. The first part of the thesis focuses on the critic component of the actor-critic framework, also referred to as value function, and how to learn more efficiently to control agents in continuous control domains through distinct uses of the variance in the value function estimates. The second part of the thesis is concerned with the actor component of the actor-critic framework, also referred to as policy. We propose the introduction of a third element to the optimization problem that agents solve by introducing an adversary. The adversary is of the same nature as the RL agent but trained to suggest actions that mimic the actor or counteract the constraints of our problem. It is represented by some averaged policy distribution with which the actor must differentiate his behavior by maximizing its divergence with it, eventually encouraging the actor to explore more thoroughly in tasks where efficient exploration is a bottleneck, or to act more safely.

Contents

I	Learning to Control	1
1	Introduction	3
1.1	Reinforcement Learning	4
1.2	Deep Learning Representations	4
1.3	Deep Reinforcement Learning	5
1.4	Choosing what to Learn	6
1.5	Outline and Contributions	8
2	Background	13
2.1	Markov Decision Processes	14
2.2	Trajectory, Policy, Return	14
2.3	RL Optimization Problem	16
2.4	Value Function	16
2.5	Learning with Deep Learning	17
2.6	What to Learn	21
2.7	RL in the Real World	28
II	Variance in the Value Function estimates	33
3	Dynamic Control Problems	35
3.1	Physics Engines for Robotics Applications	36
3.2	General Solutions for Continuous Control tasks	37
4	Use Variance in the Value Function estimates as an auxiliary loss	39
4.1	Motivation	40
4.2	Preliminaries	41

Contents

4.4	MERL: Framework for Self-Performance Assessment	44
4.5	Experimental Study	46
5	Use Variance in the Value Function estimates to filter information	53
5.1	Motivation	54
5.2	Preliminaries	55
5.4	SAUNA: Dynamic Transition Filtering	58
5.5	Experimental Study	59
5.6	Discussion	64
6	Use Variance in the Value Function estimates as an objective function	67
6.1	Motivation	68
6.3	Preliminaries	70
6.4	AVEC: Actor with Variance Estimated Critic	71
6.5	Experimental Study	74
	Part conclusion	83
III	Diversity in the Policy candidates	85
7	Hard-Exploration and Real-World Features Problems	87
7.1	Hard-Exploration Problems	88
7.2	The Real-World RL Challenge	90
7.3	Adversarial Prior in the Actor-Critic Framework	91
8	Use Repulsive Priors to motivate conservatively diversified policies	93
8.1	Motivation	94
8.3	Preliminaries	96
8.4	AGAC: Adversarially Guided Actor-Critic	97
8.5	Experimental Study	100
9	Use Repulsive Priors to motivate risk-sensitive policies	109
9.1	Motivation	110
9.2	Preliminaries	111
9.4	SAAC: Safe Adversarial Soft Actor-Critics	115

9.5 Experimental Study	120
Part conclusion	127
IV Conclusion	129
10 General Conclusion and Perspectives	131
10.1 Epilogue	131
10.2 Frontiers	133
Bibliography	135
A MERL <i>or</i> Using Variance in the Value Function estimates as an auxiliary task	159
A.1 Implementation of MERL coupled with DDPG	159
A.2 Implementation Details	160
A.3 Additional Results	161
A.4 Environment Details	164
B SAUNA <i>or</i> Using Variance in the Value Function estimates to filter information	164
B.1 Additional Results	164
C AVEC <i>or</i> Using Variance in the Value Function estimates to learn value functions	165
C.1 Proof of Section 6.4.1 results	165
C.2 Additional Results	167
C.3 Experiment Details	172
C.4 Implementation Details	173
C.5 Environment Details	173
D AGAC <i>or</i> Using Adversarial Priors to motivate conservatively diversified policies	177
D.1 Additional Results	177
D.2 Illustration of AGAC	179
D.3 Experiment Details	180
D.4 Implementation Details	182
D.5 Proof of Section 8.4.1 results	182
E Experimentation and Learning in Deep Reinforcement Learning	185

Contents

E.1 Reproducible Experiments	185
E.2 Contributions to Open Research and Education	185
List of Figures	187
List of Algorithms	191
List of Tables	192

Part I

Learning to Control

Chapter 1

Introduction

*Of several responses made to the same situation,
those which are accompanied or closely followed by satisfaction,
are more firmly connected with the situation,
so that, when it recurs,
they will be more likely to recur.*

The Law of Effect, Edward Thorndike (1911).

Where to begin? In this chapter, we shall cover the motivations behind the problem of Reinforcement Learning. We will touch on the rise of Deep Learning over the last several years and what characterize the improvements it can bring to our work. We will then consider what we would like our computers to learn considering the problems of main interest in this thesis, before discussing the contributions that will be presented in the following chapters, with their results and the potential new questions they raise.

Contents

1.1 Reinforcement Learning	4
1.2 Deep Learning Representations	4
1.3 Deep Reinforcement Learning	5
1.4 Choosing what to Learn	6
1.5 Outline and Contributions	8

1.1 Reinforcement Learning

Reinforcement Learning (RL) is a discipline of Machine Learning (ML) concerned with learning to make a sequence of decisions to maximize some score, later described as rewards, in different situations. Machine software can employ this technique to find the best possible strategy to solve any problem that could be formulated as an RL problem. Some examples of applications with immediate use include healthcare problems (Schaefer, Bailey, Shechter, et al., 2005; Yu, Liu, and Nemati, 2019), general visual question answering on complex scenes (Antol, Agrawal, Lu, et al., 2015; de Vries, Strub, Chandar, et al., 2017), energy management problems (Dimeas and Hatziaargyriou, 2007; Levent, Preux, Pennec, et al., 2019) and task scheduling problems in high performance computing systems (Mao, Alizadeh, Menache, et al., 2016; Grinsztajn, Beaumont, Jeannot, et al., 2020). Other notable achievements include board games (Tesauro, 1995; Silver, Huang, Maddison, et al., 2016), video games (Mnih, Kavukcuoglu, Silver, et al., 2013; Berner, Brockman, Chan, et al., 2019; Vinyals, Babuschkin, Czarnecki, et al., 2019), or robot control (Kober, Bagnell, and Peters, 2013; Heess, Tirumala, Sriram, et al., 2017; Andrychowicz, Baker, Chociej, et al., 2020).

The general RL problem considers an agent taking the decisions and an environment where the agent operates. At each timestep, the agent takes an action and gets a reward and an observation. As an illustrative example, Figure 1.1 depicts the agent as a dog that must complete a sequence of actions to return the Frisbee to its owner, who plays the role of the environment. The dog observes its owner's movements and is motivated by the satisfaction of playing and receiving a treat at the end. In this setting, an RL algorithm uses a trial-and-error learning process to maximize a decision-making agent's total reward in a previously unknown environment. As an example, in robotics, the observations would be the camera images or joint angles, the actions would be the joint torques, and the rewards would involve navigating to a target location, successfully reaching it and staying balanced.

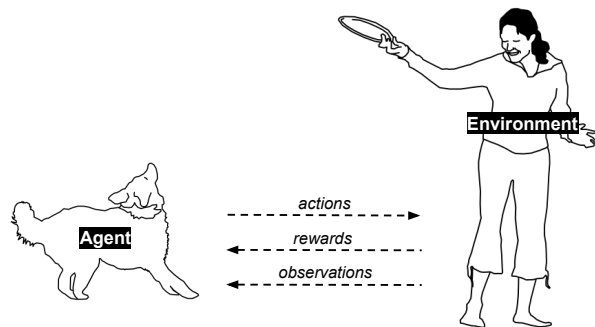


Figure 1.1 – Agent-environment interaction.

1.2 Deep Learning Representations

Representation learning is the process of learning to transform or extract features from input data in order to solve a task. Machine Learning is mainly concerned with function learning from data. Deep learning is concerned with coupling function learning from data with representation learning. Deep learning has the same practical purpose as Machine Learning, except that it

benefits from a generally more expressive function approximator (a feature that has been measured in previous work using the notion of trajectory length (Raghu, Poole, Kleinberg, et al., 2017)), namely a deep neural network trained by successive steps of gradient descent. A deep neural network is an input-to-target mapping composed of a sequence of simple data transformations called projection layers (simple matrix multiplications) aggregated together and combined with non-linearities.

Such deep learning models often involve tens or, at times, hundreds of successive layers of representations learned from exposure to training data where long causal chains of computational stages transform the aggregate activation of the neural network. Some remarkable empirical findings have arisen from this technique, particularly in speech recognition (Dahl, Yu, Deng, et al., 2012), image recognition (Krizhevsky, Sutskever, and Hinton, 2012), and natural language processing (Vaswani, Shazeer, Parmar, et al., 2017).

1.3 Deep Reinforcement Learning

Where simpler ML models with fewer parameters and lack of compositionality may fail, deep learning can be the appropriate technique for complex tasks involving highly dimensional data such as natural language or images and videos. Deep Reinforcement Learning (deep RL) is the discipline of Reinforcement Learning using neural networks as function approximators and is appropriate for sequential decision-making problems where the agent's inputs and outputs (observations and actions) involve high-dimensional data. For example, Tesauro's TD-Gammon (Tesauro, 1995) combined an RL algorithm with a neural network to learn to play backgammon, a stochastic game with approximately 10^{20} states, and played at the level of top human players. Around the same period of time, Rummery and Niranjan (1994) learnt a semi-gradient Sarsa with function approximation adding to the work of Gullapalli (1990) and Lin's and Tham's PhD thesis (Lin, 1992a; Tham, 1994) which explored the combination of various RL algorithms with neural networks.

Two decades after Tesauro's seminal work, deep RL emerged as a promising approach for experience-driven autonomous learning due to their ability to acquire complex strategies and process high-dimensional complex sensory inputs (Jaderberg, Mnih, Czarnecki, et al., 2017). Such algorithms could learn to play several *Atari 2600* video games at a superhuman level solely from image pixels (Mnih, Kavukcuoglu, Silver, et al., 2013). Some other achievements have been the development of a Monte-Carlo Tree Search (MCTS) planning system coupled with a deep RL module (Silver, Huang, Maddison, et al., 2016) that defeated a world champion *Go* player, or also the learning of control policies for robots directly from camera inputs in the real world (Levine, Finn, Darrell, et al., 2016; Zhu, Mottaghi, Kolve, et al., 2017; Levine, Pastor, Krizhevsky, et al., 2018).

In deep RL, the neural networks are used to approximate functions that implement a mapping from states to probabilities of selecting each possible action (called *policies*), functions that estimate how good it is for the agent to be in a given state (called *value functions*), dynamics models or other functions needed by the RL algorithm. In particular, the multi-step bootstrap targets (Sutton, 1988) used in asynchronous advantage actor-critic (Mnih, Badia, Mirza, et al., 2016) has shown strong results using gradient policy on a wide range of tasks. Distributional Q-learning (Bellemare, Dabney, and Munos, 2017) learns a categorical distribution of discounted returns instead of estimating the mean. Rainbow (Hessel, Modayil, Hasselt, et al., 2018) meticulously combines several improvements to the DQN (Mnih, Kavukcuoglu, Silver, et al., 2013) algorithm to provide improved performance on the *Atari 2600* benchmark in terms of data efficiency and final performance. Schulman, Levine, Abbeel, et al. (2015), Schulman, Wolski, Dhariwal, et al. (2017), Lillicrap, Hunt, Pritzel, et al. (2016), Haarnoja, Zhou, Abbeel, et al. (2018) and Fujimoto, Hoof, and Meger (2018) explored different kinds of policy gradient methods with a focus on high performance, low sample utilization, and stability improvements.

1.4 Choosing what to Learn

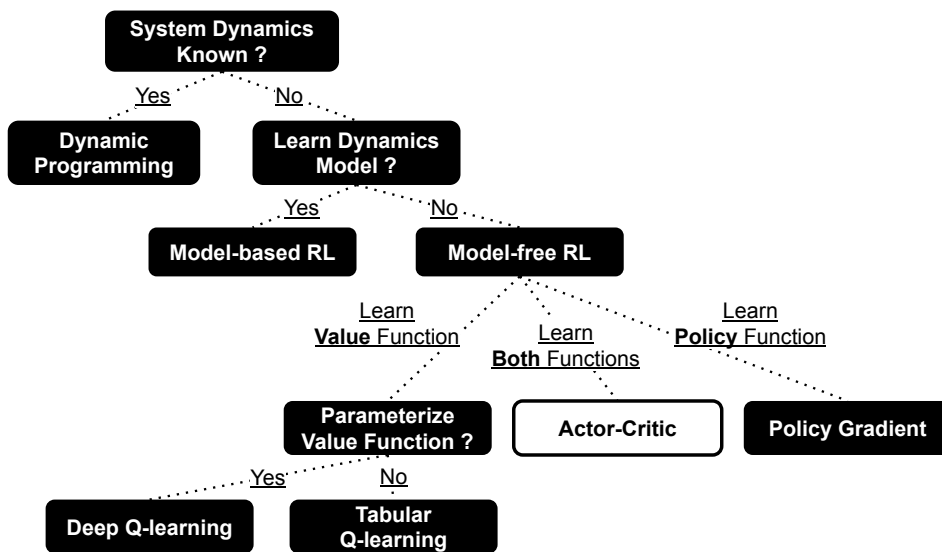


Figure 1.2 – Taxonomy of RL algorithms.

Usually, in Reinforcement Learning, the agent’s actions are based on the most recent version of the policy being learned. During updating, data from the interaction with the environment is used to derive sample-based objective functions, in which the policy and value approximators are updated by gradient descent. In deep RL, the approximators are deep neural networks. The success of those algorithms depends on the trajectories discovered during the interaction phase: if the data includes trajectories with high rewards, then those are reinforced by the

update and become more likely under the newly updated policy. As such, the interaction with the environment and the updating of the approximators are closely related and highly dependent. Therefore, a central question when designing new deep RL algorithms is *what should be approximated*, and *how*. Figure 1.2 shows a high-level taxonomy of RL algorithms. At the top level, we have Dynamic Programming (DP) algorithms that can be used to compute optimal policies given a perfect model of the environment. DP algorithms (*e.g.* Policy Iteration and Value Iteration) are, in fact, archetypal model-based algorithms: these all use the model's predictions or distributions of the next state and reward in order to calculate optimal actions. Specifically, in Dynamic Programming, the model must provide state transition probabilities and expected reward from any state-action pair. Note that, contrary to most other model-based RL algorithms, the model is rarely a learned model.

Conversely, model-free RL algorithms do not estimate the underlying system dynamics and aim to optimize a policy directly. Policy-based methods *explicitly* build and learn a policy mapping states to probabilities of selecting possible actions and store the policy approximator in memory during learning for later use. Value-based methods do not store explicit policies but instead learn a value function. The policy is *implicit* and derived from the value function by picking the action with the best value. As to actor-critic methods, they are part of a framework combining elements from both value- and policy-based methods.

The choice of which method to use depends mainly on the specification of the problem (*e.g.* system dynamics complexity), the context in which it is to be solved (*e.g.* policy optimality), and the experimental specifications (*e.g.* time or resources budget). For instance, model-based RL methods usually speed up learning at the cost of a lack of scalability to problems where the dynamics are complex. They generally learn a system dynamics model, the controller, and use it for planning. Such methods can learn successful controllers with high sample efficiency in low-dimensional continuous control problems (Deisenroth and Rasmussen, 2011; Moldovan, Levine, Jordan, et al., 2015; Zhang, Vikram, Smith, et al., 2019). Another application of such approach is AlphaGo (Silver, Huang, Maddison, et al., 2016; Silver, Schrittwieser, Simonyan, et al., 2017), which effectively tackled the problem of Computer Go by using a Monte-Carlo Tree Search (MCTS) planning module to capitalize on the knowledge of the game dynamics.

In this thesis, we focus our research efforts on the data-efficiency of model-free methods which directly learn a stochastic policy function using gradient-based methods in the actor-critic framework. An advantage with stochastic policies is that they allow for infinitesimal small changes in the policy when moving in the parameter space, whereas a comparable shift would potentially drastically change the policy in case of deterministic policies. The coupling between parameters and policy seems therefore to be more controllable in general and especially for a discrete action space. Another asset of stochastic policies is their inherent exploration nature by essentially sampling Gaussian noise to add to a deterministic base policy. Lastly, the complex dynamic characteristics of the problems of main interest in this thesis (continuous control

tasks, procedurally-generated tasks, and continuous control tasks with safety constraints) also encouraged us to adopt a model-free setting, without the need for assumptions about the environment, specifications or domain knowledge.

1.5 Outline and Contributions

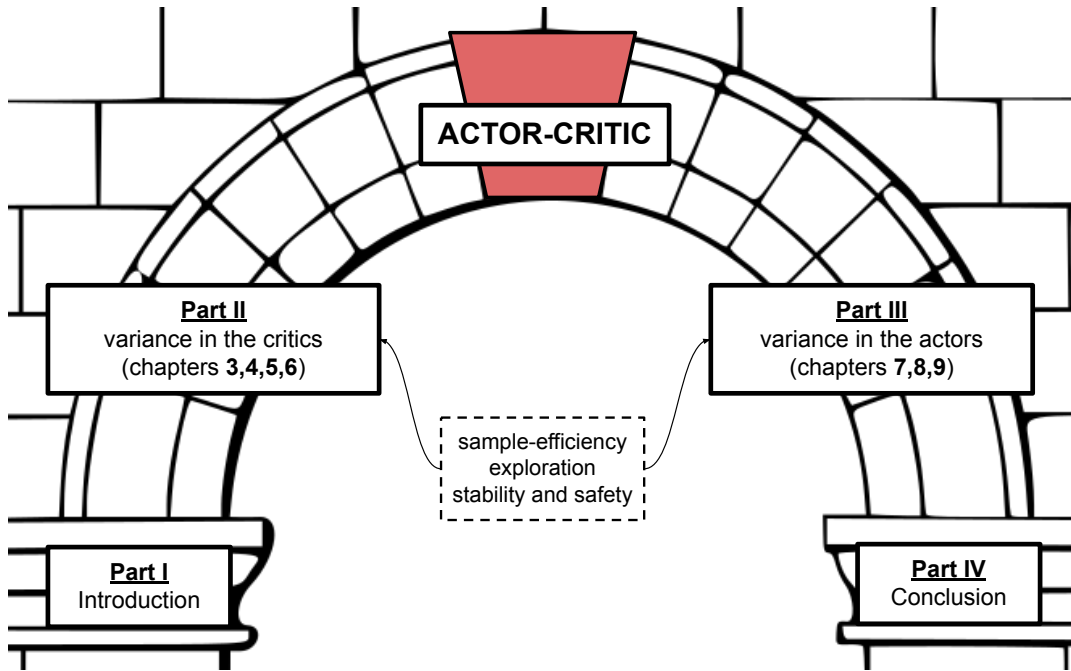


Figure 1.3 – Outline of this thesis structured around the actor-critic components.

Before diving into this thesis, we must ask ourselves what problems we would like to address and what questions remain to be answered. Some sources of difficulty that motivate the work of this thesis can be described as follows:

- Optimization and evaluation of (deep) RL methods are usually based solely on a proxy for the sum of future rewards. Alternative statistics from supervised or statistical learning could be leveraged as additional performance metrics.
- In some continuous control problems or tasks with sparse rewards, policy gradient estimates may have low amplitude and be unstable, potentially leading to sample inefficiency. An RL agent may learn more effectively from some transitions than others, therefore filtering transitions seems a natural idea to consider.
- Variance reduction methods, such as baseline subtraction, exhibit discrepancies between what motivates the conceptual framework of these algorithms and what is implemented in practice to estimate the critic component in the actor-critic framework. More efficient

and robust objective functions are needed to estimate the value function represented by the critic.

- States with rewards often have to be visited many times, especially with on-policy methods in tasks with sparse rewards, for the agent to learn anything of significance. The value function estimation must be sensitive to these extreme values and capture the (sometimes rare) signals corresponding to the rewards as efficiently as possible.
- With exploration induced by stochastic policies, the likelihood of visiting states with rewards in sparse rewards tasks will be infinitesimally small if these states are far from the departure point. Some form of memory needs to be maintained by using, for example, a moving average of previous policies and thus avoid repeating the same trajectories that did not result in relevant learning.
- Building on the same idea, an interesting question is whether a similar prior could be constructed by learning, instead of a mixture of previous policies, how to break the safety constraints to represent a probabilistic unsafe region that the agent should avoid.

All these cases fall under the same umbrella: in this thesis, we attempt to develop policy gradient methods that are more stable and sample-efficient than previous methods by (i) leveraging the information given by self-performance statistics and using alternative ways of learning function estimates which are more adapted to policy gradient methods, and (ii) introducing a third protagonist to the actor-critic duo which serves as a repulsive average distribution from which the policy must distance itself. The keystone of this thesis is the actor critic framework, illustrated in Figure 1.3. We tackle both sides of it, starting by the critic and then the actor, through the prism of variance: variance in the value function estimates calculated using the variance explained and the residual variance, and variance in the policy candidates derived from an adversarial prior maintaining an averaged mixture of policies.

This thesis concludes the research contributions of four previously published papers. The organization follows the order in which the work was published with some reorganized elements. In order to give the thesis a more coherent structure and improve its readability, we have divided it into two parts. **Part I** introduces the problem of Reinforcement Learning from a general point of view. We develop the perspective adopted by this thesis regarding some of the difficulties of the RL problem and detail the issues we have chosen to address as part of this thesis and the motivation behind their study.

Part II is devoted to more efficiently learning to control agents in continuous control problems. In **Chapter 3**, we introduce the problem of learning continuous control policies and present the inference scheme for learning deep neural network representations in high-dimensional continuous state and action space. In **Chapter 4**, we propose the use of more statistical objects as auxiliary losses when learning to solve a task. In particular, we identify

the explained variance of the value function estimates as a tool with interesting properties and propose a generally applicable framework with encoder sharing to speed up the learning of policy gradient agents. **Chapter 5** develops the simple but effective idea that an RL agent will learn more effectively from some experience data than others. We employ the statistics of self-performance assessment introduced in **Chapter 4** to develop a modification to policy gradient algorithms where samples are filtered out when estimating the policy gradient. In **Chapter 6**, motivated by recent studies indicating that traditional actor-critic algorithms do not succeed in fitting the value function and calling for the need to identify a better objective for the critic, we introduce a method to improve the learning of the critic in the actor-critic framework.

Part III concerns the other side of the keystone in **Figure 1.3**: a formulation of the variance in the context of actor policies by introducing a third protagonist to the actor-critic framework. This new protagonist serves as an adversarial prior by maintaining an averaged mixture of policies from which the policy distribution should be repulsed. After the introduction of the problem of learning in environments with more real-world features such as safety constraints or where efficient exploration is a bottleneck in **Chapter 7**, in **Chapter 8** and **9** we develop a form of variance in the policy candidates in maintaining an adversarial prior as a mixture of previous policies (**Chapter 8**) and as a mixture of risk-seeking policies (**Chapter 9**).

Finally, we give an epilogue to the thesis in **Part IV**, with a discussion of progress and future horizons.

List of publications

Publications in international conferences with proceedings

- Yannis Flet-Berliac, Reda Ouhamma, Odalric-Ambrym Maillard, and Philippe Preux (2021). Learning Value Functions in Deep Policy Gradients using Residual Variance. *In International Conference on Learning Representations*
- Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist (2021). Adversarially Guided Actor-Critic. *In International Conference on Learning Representations*
- Yannis Flet-Berliac and Philippe Preux (July 2020). Only Relevant Information Matters: Filtering Out Noisy Samples To Boost RL. *In Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, pp. 2711–2717

Workshop presentations in international conferences or preprints

- Yannis Flet-Berliac and Philippe Preux (2019b). MERL: Multi-Head Reinforcement Learning. In *Deep Reinforcement Learning Workshop of the 33rd conference on advances in Neural Information Processing Systems*
- Yannis Flet-Berliac and Debabrota Basu (2021). SAAC: Safe Reinforcement Learning as an Adversarial Game of Actor-Critics. *Preprint*

Publications in international digital magazines

- Yannis Flet-Berliac (2019). The Promise of Hierarchical Reinforcement Learning. *The Gradient - Stanford AI Lab*

Software

- Omar Darwiche Domingues, Yannis Flet-Berliac, Edouard Leurent, Pierre Ménard, Xuedong Shang, and Michal Valko (2021). *rlberry - A Reinforcement Learning Library for Research and Education*. <https://github.com/rlberry-py/rlberry>

Collaborations not presented in this thesis

- Jacques Demongeot, Yannis Flet-Berliac, and Hervé Seligmann (2020). Temperature Decreases Spread Parameters of the New Covid-19 Case Dynamics. *Biology* 9.5, p. 94
- Yannis Flet-Berliac and Philippe Preux (2019a). High-Dimensional Control Using Generalized Auxiliary Tasks. *Tech. rep. hal-02295705*
- Thomas Depas and Yannis Flet-Berliac (2019). Princess of Parallelograms. *Exhibition Panorama 21 - Le Fresnoy National Studio of Contemporary Arts*

Chapter 2

Background

The only thing that makes life possible is permanent, intolerable uncertainty; not knowing what comes next.

Dialogue between Faxee and Genry, Ursula Le Guin (1969).

After a general introduction to the concepts involved in this work, Chapter 2 charts the technical landscape that will serve us throughout this thesis. We begin with reviewing core concepts of reinforcement learning and deep learning, which will enable us to introduce the discipline of deep reinforcement learning and its different problem formulations. The remainder of the chapter is devoted to generalization and minimum experimental protocol requirements for the application of RL in the real world.

Contents

2.1	Markov Decision Processes	14
2.2	Trajectory, Policy, Return	14
2.3	RL Optimization Problem	16
2.4	Value Function	16
2.5	Learning with Deep Learning	17
2.6	What to Learn	21
2.7	RL in the Real World	28

2.1 Markov Decision Processes

The general framework of Reinforcement Learning is typically formalized using a *Markov Decision Process* (MDP) (Puterman, 1994). An MDP is a mathematical object to model discrete-time sequential decision making problems in which an agent interacts with a stochastic environment. It is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ where:

- \mathcal{S} is a set of states;
- \mathcal{A} is a set of actions;
- \mathcal{T} is a transition probability distribution mapping state-action pairs to the conditional probability distribution on the resulting state. If we write $\mathcal{P}(\mathcal{S})$ the set of probability distributions on the set \mathcal{S} , we have:

$$\begin{aligned} \mathcal{T} : \quad \mathcal{S} \times \mathcal{A} &\rightarrow \mathcal{P}(\mathcal{S}) \\ (s, a) &\mapsto \mathbf{Pr}(\cdot \mid s, a) \end{aligned}$$

- \mathcal{R} is the reward function, mapping state-action-next-state tuples to real-valued rewards:

$$\begin{aligned} \mathcal{R} : \quad \mathcal{S} \times \mathcal{A} \times \mathcal{S} &\rightarrow \mathbb{R} \\ (s, a, s') &\mapsto \mathcal{R}(s, a, s') \end{aligned}$$

A fundamental feature of an MDP is the Markov property, which states that for a sequence of random variables $\{X_n\}$, $\mathbf{Pr}(X_k = s_k \mid X_{k-1} = s_{k-1}, X_{k-2}, \dots, X_1) = \mathbf{Pr}(X_k = s_k \mid X_{k-1} = s_{k-1})$. In other terms, this indicates that when taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, the transition probability to $s' \in \mathcal{S}$ is independent of previously encountered state-action pairs and only conditioned by the pair (s, a) . This enables us to predict the next state and expected next reward given the current state-action pair.

Remark 2.1 (Theoretical Guaranties). *The theoretical guarantees given by the MDP framework on the performance of algorithms are systematically lost when using nonlinear function approximations, and successfully combining such approximations and theoretical guarantees is still a major challenge facing our field.*

2.2 Trajectory, Policy, Return

In an MDP, the description of the dynamics of the interaction process between an agent and an environment yields a sequence of states and actions. Later in this thesis, we refer to this

sequence as a *trajectory*, or an *episode*, or a *rollout*. By referring to $s_t, a_t \in \mathcal{S} \times \mathcal{A}$ as the state-action pair of time step t , we denote $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ a trajectory experienced by an agent acting in an MDP. We also define *terminal states* as specific states that have the property to end the realization of a trajectory before reaching the horizon of the MDP.

A *policy* π is a decision rule. A *stochastic policy* gives a probability distribution over actions and an agent may take a decision by sampling an action according to this probability distribution. In general terms, a policy π is conditioned on a history of observations of the agent-environment interactions. However, in this thesis, we will restrict ourselves to stationary policies that depend only on the current state. By definition, a *stationary stochastic policy* is a mapping:

$$\begin{aligned} \pi : \quad \mathcal{S} &\rightarrow \mathcal{P}(\mathcal{A}) \\ s &\mapsto \pi(\cdot | s). \end{aligned}$$

From here, let $\mu(s)$ denote the probability of starting in state s and T the length of trajectory τ , meaning the time at which the trajectory ends. The probability distribution over trajectories, which depends on both the *policy* (the agent) and the *MDP* (the environment) is given as:

$$P(\tau | \pi) = \mu(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t) \mathcal{T}(s_{t+1} | s_t, a_t). \quad (2.1)$$

The *return* is the cumulative reward over many timesteps of interaction. Let us now denote $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$. It is usual to define the return of a trajectory $R(\tau)$ as the sum of all collected reward along trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ of finite horizon trajectory that ends at time T :

$$R(\tau) = \sum_{t=0}^{T-1} r_{t+1}.$$

In the infinite horizon case, where $T = \infty$, trajectories are assumed to have infinite length. In order to properly define a non-diverging series for the return, we generally introduce a discount factor $\gamma \in [0, 1)$ which acts as a weighting parameter in the now discounted return:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_{t+1}.$$

For all time steps t , we may also write $R_t(\tau)$ the return received after time step t :

$$R_t(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

2.3 RL Optimization Problem

In RL, we are interested in finding a policy π that maximizes the expected return over trajectories. First, let us express the expected return over trajectories:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} [R(\tau)],$$

with $\tau \sim \pi$ indicating that the distribution over τ is given by Equation 2.1. Denoting Π the space of policies, the optimization problem RL aims to solve is:

$$\pi^* = \arg \max_{\pi \in \Pi} J(\pi). \quad (2.2)$$

2.4 Value Function

The *value* of a state or of a state-action pair is the expected return if the agent starts in that state or state-action pair, and then acts according to its policy forever after. The *state value* function $V^\pi(s)$ of a policy π is defined as:

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) \mid s_0 = s].$$

Similarly, the *state-action value* function $Q^\pi(s, a)$ of a policy π is defined as:

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) \mid s_0 = s, a_0 = a].$$

Using the definition of $R_t(\tau)$, a fundamental property of *value functions* in RL is that they satisfy recursive relationships. Taking the example of the *state value function*, the following consistency condition holds between the value of all $s \in \mathcal{S}$ and the value of its possible successor states:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{\tau \sim \pi} [R_t(\tau) \mid s_t = s] \\ &= \mathbb{E}_{\tau \sim \pi} [r_{t+1} + \gamma R_{t+1}(\tau) \mid s_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \mathcal{T}(s' \mid s, a) \left[r_{t+1} + \gamma \mathbb{E}_{\tau \sim \pi} [R_{t+1}(\tau) \mid s_{t+1} = s'] \right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \mathcal{T}(s' \mid s, a) [r_{t+1} + \gamma V^\pi(s')]. \end{aligned} \quad (2.3)$$

Equation 2.3 is the *Bellman equation* (Bellman and Kalaba, 1957) for V^π and expresses a relationship between the value of a state and the value of its successor states. The two value functions (*state value function* and *state-action value function*) are connected through the following

equations:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)],$$

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s, a)} [\mathcal{R}(s, a, s') + \gamma V^\pi(s')].$$

Advantage Function. Let us define the *advantage function*, which quantifies how an action a is better than the average action in state s (following policy π):

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s).$$

2.5 Learning with Deep Learning

In Reinforcement Learning, agents can approximate a policy or a value function using a variety of Machine Learning methods, ranging from SVMs, to decision trees, to neural networks. In this thesis, we will focus our attention on neural networks only: we study *Deep* Reinforcement Learning where *deep* comes from *Deep* Learning. The ultimate goal in RL is to *learn* an optimal policy for the MDP. The learning procedure is to learn from experience by using the collected data to *learn* the shape of certain functions to that purpose. Such functions are for instance defined in Section 2.2, 2.3 and 2.4. Neural networks are versatile function approximators that are composed of layers of parameterized transformations: networks with more layers are comparatively deeper (Goodfellow, Bengio, Courville, et al., 2016). They are not necessarily the best solution to every problem: neural networks are very data intensive and difficult to interpret. However, neural networks are also one of the most powerful function approximations available, and their performance is often the best in large scale problems or when the vector representations becomes computationally intractable. For instance, deep neural networks have been successful at advancing research in many areas including natural language processing (Malinowski, Rohrbach, and Fritz, 2015; Brown, Mann, Ryder, et al., 2020), image classification (He, Zhang, Ren, et al., 2016; Mahajan, Girshick, Ramanathan, et al., 2018), speech recognition (Hannun, Case, Casper, et al., 2014; van den Oord, Dieleman, Zen, et al., 2016), and neural machine translation (Cho, Merriënboer, Gulcehre, et al., 2014; Sutskever, Vinyals, and Le, 2014). In RL, neural networks will be used when a linear or a tabular representation is not adequate to the task at hand.

2.5.1 Neural Networks

In this thesis, we use Artificial Neural Networks (ANN) as multi-layered non-linear function approximators loosely inspired by the biological neural networks in animal brains. An ANN is

Background

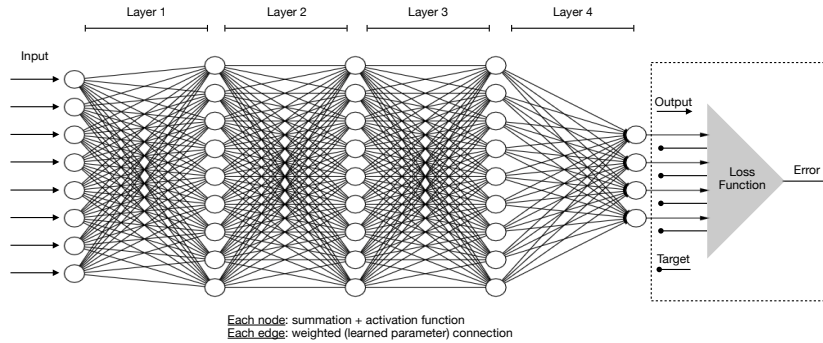


Figure 2.1 – Illustration of a feed-forward multi-layer network.

not an algorithm, but a structure composed of multiple layers of mathematical transformations applied to input values. A characteristic example of a neural network is a Multi-Layer Perceptron (MLP) or multi-layer neural network. First, let us consider the most classical case of a single hidden layer neural network, from vector-valued inputs to vector-valued outputs (*e.g.* for regression):

$$g(x) = b + W a(c + V x).$$

In this equation,

- x is a d -vector (the input);
- V is an $k \times d$ matrix (called input-to-hidden weights);
- c is a k -vector (called hidden units offsets or hidden unit biases);
- b is an m -vector (called output units offset or output units biases);
- W is an $m \times k$ matrix (called hidden-to-output weights);
- a is a threshold-like (non-linear) *activation function* differentiable almost everywhere (*e.g.* the Sigmoid, Tanh, or ReLU function) applied element-wise.

The vector-valued function $h(x) = a(c + V x)$ is called the output of the *hidden layer* and we call *hidden units* the elements of the hidden layer. The weights and biases in the neural network form the set of all *learnable parameters*. Fundamentally, the type of operation calculated by $h(x)$ can be applied to $h(x)$ itself, but with different parameters (different biases and weights). This would give rise to a feed-forward multi-layer network with *two hidden layers*. More generally, one can build a deep neural network by stacking more such layers and each of these layers may have a different dimension (k above). For instance, Figure 2.1 illustrates a feed-forward neural network with 4 layers. In this thesis, we will usually denote the full set of learnable parameters in a function approximator by a lowercase greek letter, *e.g.* θ or ω . A common

variant is to have skip connections, *i.e.* a layer can take as input not only the layer at the previous level but also some of the lower layers. *Residual neural networks* accomplish this by using skip connections to jump over some layers, for instance, ResNets (He, Zhang, Ren, et al., 2016), HighwayNets (Srivastava, Greff, and Schmidhuber, 2015) or DenseNets (Huang, Liu, Maaten, et al., 2017). Silver, Schrittwieser, Simonyan, et al. (2017) and Espeholt, Soyer, Munos, et al. (2018) are examples of RL methods using a deep residual network structure.

The layer parameters are learnable: they are adjusted by an algorithm until the network approximates a target function at an acceptable level of fidelity. This adjustment process is referred to as *learning* or *training*. Although many algorithms have been proposed for training MLPs, the most widely used in practice are based on *gradient descent* (Goodfellow, Bengio, Courville, et al., 2016).

2.5.2 Training

In the standard set-up for *gradient descent*, we consider the general problem of minimizing a *loss function* $L : \mathbb{R}^m \rightarrow \mathbb{R}$:

$$\min_{\theta} L(\theta),$$

which takes as input the parameter of the neural network f_{θ} and returns a scalar value measuring how well the network represents the target function. Given access only to first-order evaluations of L , the parameters θ are iteratively updated by stepping in the opposite direction of the gradient until convergence:

$$\theta_{k+1} \leftarrow \theta_k - \alpha_k \nabla_{\theta_k} L.$$

Here, k is the index of the update iteration and α_k is the *learning rate*. The procedure for computing gradients in the case of feed-forward multi-layer networks is called the *back-propagation algorithm* (Kelley, 1960; Werbos, 1974; Rumelhart, Hinton, and Williams, 1985; LeCun, 1988). In Deep Learning, and thereby in Deep Reinforcement Learning, we often put ourselves in the *stochastic* setting (Robbins and Monro, 1951), where only noisy gradient evaluations are given. The issue of how to choose the *learning rate* is less clear than in the exact gradient setting (Bertsekas, 1997; Nocedal and Wright, 2006). There are different guidelines for setting the learning *schedule*: the classical Robbins/Monro theory (Robbins and Monro, 1951) asserts that if the *learning rate* is chosen such that:

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty,$$

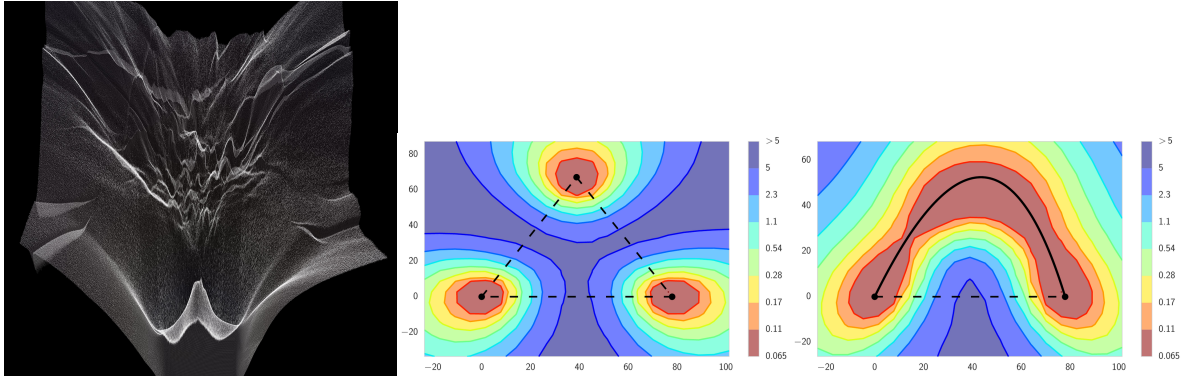


Figure 2.2 – Loss surfaces as a function of network weights in a two-dimensional subspace (Garipov, Izmailov, Podoprikin, et al., 2018). Left (3D): for ResNet-20 (He, Zhang, Ren, et al., 2016) on the ImageNet dataset (Deng, Dong, Socher, et al., 2009). Middle (2D): three optima for independently trained networks. Right (2D): a quadratic Bezier curve connecting the lower two optima on the middle figure along a path of near-constant loss.

and if the *loss function* is sufficiently smooth, then $\lim_{k \rightarrow \infty} \mathbb{E} [\|\nabla L(\theta_k)\|^2] = 0$ (Bottou, Curtis, and Nocedal, 2018). In addition, if the *loss function* is strongly convex, then the stochastic gradient (g_k) update $\theta_{k+1} \leftarrow \theta_k - \alpha_k g_k$ will converge in expectation to the minimizer.

Generally, two fundamental concerns guide the different strategies employed for *MLP training*: (i) training as efficiently as possible, *i.e.* reducing the training error while avoiding getting stuck in narrow valleys or poor local minima of the cost function (a visualization of such cost function valleys is shown in Figure 2.2), and (ii) controlling the expressiveness of the neural network subject to the amount of training data so as to avoid overfitting, *i.e.* minimize the generalization error.

2.5.3 Optimization

Properly adjusting the neural network structure, fine-tuning the hyperparameters, defining a suitable regularization of the loss function are some key elements to consider before the training of neural networks. Such critical design choices are essential, yet subtle and potentially undeclared in scientific papers.

For instance, the difficult task of training deep neural networks is strongly affected by the choice of *network initialization* (Goodfellow, Bengio, Courville, et al., 2016). In other terms, correctly defining the initial values for the parameters in neural network models. Some popular initialization techniques include Glorot and Bengio (2010) and He, Zhang, Ren, et al. (2015), and all depend on the neural network design choices (network structure and *activation functions*).

Moreover, as mentioned in Section 2.5.2, there exists different guidelines for setting the *learning rate*, and in most instances it is not set fixed since problems can occur when gradients

become either too large or too small. By automatically adjusting the *learning rate*, one can hope to keep the gradients in favorable ranges which avoid problems of plateaus in the error landscape (gradient is too small), or may overstep local minima preventing the goal of finding good solutions (gradient is too large). *Adam* optimization (Kingma and Ba, 2015) builds upon the idea of *adaptive learning rates* from AdaGrad (Duchi, Hazan, and Singer, 2011) and RMSProp (Tieleman and Hinton, 2012). Formally, the adaptive gradient update rule is defined as follows:

$$\begin{aligned}\mathbf{m}_{k+1} &= \beta_1 \cdot \mathbf{m}_k + (1 - \beta_1) \cdot \nabla_{\theta_k} L \\ \mathbf{s}_{k+1} &= \beta_2 \cdot \mathbf{s}_k + (1 - \beta_2) \cdot \nabla_{\theta_k} L \odot \nabla_{\theta_k} L \\ \theta_{k+1} &= \theta_k - \eta \cdot \frac{\mathbf{m}_{k+1}}{1 - \beta_1^k} \oslash \sqrt{\frac{\mathbf{s}_{k+1}}{1 - \beta_2^k} + \varepsilon}\end{aligned}$$

where \odot denotes the point-wise multiplication and \oslash the point-wise division between vectors. η is the initial learning rate, $\beta_1, \beta_2 \in [0; 1[$ are respectively decay rates for first-order and second-order moments of the gradient, β_1^k should be understood as “ β_1 to the power k ” and $\varepsilon \in \mathbb{R}^+$ is a smoothing term. Note that intuitively, \mathbf{m}_k and \mathbf{s}_k are interpreted as exponentially moving averages of the first and second raw moment of the gradient. A more detailed discussion of Adam as well as a comparison to other optimization methods can be found in Ruder (2016). When not specified otherwise, we use Adam as our optimization method for all the course of this thesis.

2.6 What to Learn

Now that we understand *how to learn* a function using a neural network for function approximation, we need to choose *which function* to learn, with a common final objective to learn an optimal policy of the MDP. In other terms, with the availability of modern computational resources, the question is not so much how to use large and deep neural network representations but for what purpose. The choice of the *function to be learned* and the *objective function to train it* forms an important part of the work presented in this thesis.

First, one could seek to learn directly an optimal policy π^* by optimizing the expected reward with respect to the policy’s parameters. Examples are Derivative-Free Optimization algorithms (Rastrigin, 1963; Goldberg and Holland, 1988; Sun, Wierstra, Schaul, et al., 2009) and policy gradient methods (Williams, 1992; Sutton, McAllester, Singh, et al., 2000; Kakade, 2002). We refer to these methods as *policy-based*.

A second approach focuses on learning optimal value functions which predict how much rewards an agent will obtain from a state (V^*) or from a state-action pair (Q^*), then derive an optimal policy from those functions. Examples include value iteration algorithms benefiting

Background

from learning Q-functions (Watkins and Dayan, 1992; Mnih, Kavukcuoglu, Silver, et al., 2015; Hessel, Modayil, Hasselt, et al., 2018). We refer to these methods as *value-based*.

Third, some methods primarily learn the dynamics model, *i.e.* the transition and reward functions of the MDP. They usually use that model of the environment to plan in the estimated MDP by deriving an optimal policy π^* . For instance, Ghadirzadeh, Maki, Kragic, et al. (2017) train perception and behaviour networks in simulations and learn only a low-dimensional intermediate layer from real-world interactions. Other examples include Deisenroth and Rasmussen (2011), Moldovan, Levine, Jordan, et al. (2015), and Levine, Finn, Darrell, et al. (2016) when the use of such methods is relevant where the dynamics are relatively simple but the optimal policy is complex. We refer to these methods as *model-based*.

Finally, *actor-critic methods* refer to methods that both learn a policy and a value function. Namely, the *critic* updates the value function parameters, be it the state value function V or the state-action value function Q . On the other hand, the *actor* updates the policy parameters in a direction recommended by the critic.

2.6.1 Value-based Methods

A common value-based method to exactly solve an MDP with a finite number of states and actions is *value iteration*. This process eventually finds the optimal value function by iteratively computing and updating the state-action value function, or *Q-value function*, represented by $Q(s, a)$. However, for more complex MDPs, value functions are facing a different scale, up to thousands or millions of dimensions, including images. In addition, in most real-world scenarios, the agent does not know the state transition probabilities or rewards. In such problems, these algorithms can be combined with function approximation in different ways, adding to the splendour of *Q-learning*, exemplified by the highly popular *deep Q-network* (Mnih, Kavukcuoglu, Silver, et al., 2013, 2015) and its variants double DQN (Hasselt, Guez, and Silver, 2016), IQN (Dabney, Ostrovski, Silver, et al., 2018) and TQC (Kuznetsov, Shvechikov, Grishin, et al., 2020). Inherently, Q-learning is designed to find deterministic policies. Thus, for environments with a continuous action space requiring a policy with a distribution over actions as an output from which to sample, it is preferable to use other methods that fit more naturally under a continuous action space, such as policy gradient or actor-critic methods. Furthermore, the inherent randomness of a stochastic policy leads to exploration, so although we often prefer a deterministic policy at convergence, a stochastic policy is desirable for exploration, which is crucial for many learning problems.

2.6.2 Policy-based Methods

In this thesis, we will typically use parameterized stochastic policies. Stochastic policies have several advantages. For instance, the inherent randomness of stochastic policies leads to exploration, which is crucial for most learning problems. In other RL methods where the policy is deterministic, randomness usually has to be artificially added in some other way, *e.g.* using the naive exploration policy ε -greedy, which takes random actions at a fixed frequency. In addition, while policy-based methods simply increase the probability of actions associated with high rewards whilst decreasing the probability of actions associated with low rewards, value-based methods need the value function and thus also need to actually *learn* and *compute* values, adding additional layers of potential estimation errors. Moreover, Q-functions exhibit problematic issues such as overestimation bias (Thrun and Schwartz, 1993) requiring mitigation strategies such as learning two Q-value estimates (van Hasselt, 2010; Hasselt, Guez, and Silver, 2016) or regularization (Bahdanau, Brakel, Xu, et al., 2017). When choosing a parameterized model for the policy π_θ , Equation 2.2 becomes an optimization problem with respect to $\theta \in \mathbb{R}^d$:

$$\pi_\theta^* = \arg \max_{\theta} J(\pi_\theta). \quad (2.4)$$

Derivative-Free Policy Optimization Methods. There exists a kind of methods where the parameters of the policy are perturbed incrementally in different directions where the performance improves. Remarkably, these methods provide a means to train agents with high return without estimating any gradient at all. Nevertheless, only a few of these methods (Mania, Guy, and Recht, 2018; Pourchot and Sigaud, 2019) succeed in competing against gradient-based methods. Indeed, in practice, the policies of the derivative-free methods require few parameters and often suffer from a lack of scalability as the number of parameters increases. For instance, the cross-entropy method (Szita and Lőrincz, 2006; Gabillon, Ghavamzadeh, and Scherrer, 2013) is a derivative-free optimization method treating the problem of transforming a Gaussian policy parameter $\theta = (\mu, \sigma)$ into a reward R as a black-box process. We illustrate in Algorithm 1 how an evolutionary algorithm is used to find good candidate parameters using the cross entropy method.

Other methods include natural evolution strategies (Wierstra, Schaul, Peters, et al., 2008) and covariance matrix adaptation (Wampler and Popović, 2009; Wang, Fleet, and Hertzmann, 2010). This class of algorithms is discussed with more details by Deisenroth, Neumann, and Peters (2013) and Sigaud and Stulp (2019).

Policy Gradient Methods. Policy-based methods use an estimator function from a space of parameterized stochastic policies $\Pi_\theta = \{\pi_\theta \mid \theta \in \mathbb{R}^d\}$ mapping states to action probabilities. Such algorithms iteratively estimate the policy’s performance with respect to its parameters

Background

Algorithm 1 Cross Entropy Method (CEM).

- 1: **Input** number of parameter vectors n , proportion $\rho \leq 1$, noise η
 - 2: **Initialize** $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$
 - 3: **for** iteration = 1, 2, ... **do**
 - 4: Generate a random sample of n parameter vectors $\theta_i \sim \mathcal{N}(\mu, \text{diag}(\sigma))$
 - 5: For each θ_i perform one episode and collect reward R_i
 - 6: Select $\lfloor \rho n \rfloor$ parameters with the highest score $\theta'_1 \dots \theta'_{\lfloor \rho n \rfloor}$
 - 7: Update $\mu(j) = \frac{1}{\lfloor \rho n \rfloor} \sum_{i=1}^{\lfloor \rho n \rfloor} \theta'_i(j)$ and $\sigma^2(j) = \frac{1}{\lfloor \rho n \rfloor} \sum_{i=1}^{\lfloor \rho n \rfloor} [\theta'_i(j) - \mu(j)]^2 + \eta$
 - 8: **Return** μ
-

and update the weights according to Equation 2.4 by gradient ascent. In practice, the *vanilla policy gradient* algorithm alternates between two phases: (i) exploration where trajectories are generated from actions sampled from the current policy and (ii) update where the trajectories are used to estimate the gradient of policy performance and perform a gradient ascent step.

In the following, we derive a simple formulation of the vanilla policy gradient algorithm (VPG). Let us denote x a random variable with probability density $p(x|\theta)$ and f a scalar-valued function f , which will be the reward in our case, but can more generally denote a *score function*. We would like to compute the score function gradient $\nabla_{\theta} \mathbb{E}_x [f(x)]$:

$$\begin{aligned}
 \nabla_{\theta} \mathbb{E}_x [f(x)] &= \nabla_{\theta} \int p(x|\theta) f(x) dx \\
 &= \int \nabla_{\theta} p(x|\theta) f(x) dx \\
 &= \int p(x|\theta) \nabla_{\theta} \log p(x|\theta) f(x) dx \\
 &= \mathbb{E}_x [f(x) \nabla_{\theta} \log p(x|\theta)].
 \end{aligned}$$

Let us now apply the expectation above to the problem of Reinforcement Learning in which the random variable x translates into a sequence of state and actions resulting in a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T)$. $p(x|\theta)$ will be the probability occurrence of trajectory τ under policy parameters θ , and the score function $f(x)$ will be the total reward $R(\tau)$ of the trajectory. Next, we derive $p(\tau|\theta)$:

$$p(\tau|\theta) = \mu(s_0) \prod_{t=0}^T \mathcal{T}(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t),$$

where μ is the initial state distribution. The expectation above can be estimated with a sample mean, that is, by taking the logarithm of $p(\tau|\theta)$ we obtain the following *score function gradient*

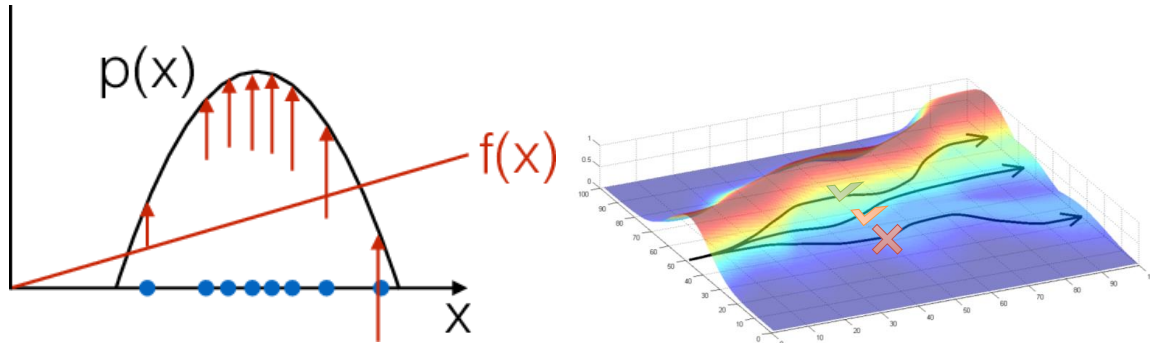


Figure 2.3 – Illustrations of the score function gradient estimator mechanics. From Schulman (2017) and Levine (2017).

estimate \hat{g} :

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau), \quad (2.5)$$

with $\mathcal{D} = \{\tau_i\}_{i=1, \dots, N}$ a set of N trajectories collected in an environment by an agent acting following π_{θ} . Intuitively, $R(\tau)$ gives a clue about if and how much one should move in parameter space in the direction that increases $\log \pi_{\theta}(a_t | s_t)$ (see Figure 2.3). Notably, there is no need to compute the value of states or state-action pairs exactly, contrary to value-based methods. Learning such a parameter representation of the policy can be relevant in practice: in the *Breakout Atari game* for instance, aiming to follow the ball appears to be much more efficient compared to going left, right or still because it will eventually give a return of 3.5, while other actions have returns 2.5 and 0.

From here, we would like to reinforce actions based on their consequences and not based on all rewards obtained in the trajectory. Equation 2.5 can be rewritten as:

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^T r_{t'+1}.$$

Now, to improve stability, Weaver and Tao (2001) show that subtracting a *baseline* (Williams, 1992) can be very beneficial in reducing variance without damaging the bias. Intuitively, adjusting an action's probability based on a trajectory performance without comparing it to previous returns means that if a particular action was in a path to a positive return, you will still increase its probability even if other actions might have done better. Instead, it is less naive to compare the return to what you might have done: this is the purpose of a baseline. We

Background

subtract $b(s_t)$ from the empirical returns:

$$\nabla_{\theta} \mathbb{E}_{\tau} [R(\tau)] = \mathbb{E}_{\tau} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left(\sum_{t'=t}^T r_{t'+1} - b(s_t) \right) \right].$$

The equality in the last equation follows from the fact that $\mathbb{E}_a \nabla_{\theta} \log \pi_{\theta}(a | s) = 0$ (Williams, 1992). The most-common choice for $b(s_t)$ is the state-value function $V^{\pi}(s_t)$ which near-optimally reduces the variance of the sample estimate of the policy gradient (Greensmith, Bartlett, and Baxter, 2004). The resulting term, $\sum_{t'=t}^T r_{t'+1} - V^{\pi}(s_t)$, is an estimate of the *advantage function* $A^{\pi}(s_t, a_t)$ defined previously, and the resulting policy gradient estimate \hat{g} formulation enjoys faster and more stable policy learning. In practice, $V^{\pi}(s_t)$ needs to be estimated: a canonical way of doing it is using a neural network to approximate it and minimizing the mean-squared-error objective against the empirical returns. Algorithm 2 illustrates the resulting vanilla policy gradient algorithm where the policy and value function parameters are typically updated via stochastic gradient ascent/descent or Adam (Kingma and Ba, 2015) optimization.

Algorithm 2 Vanilla Policy Gradient (VPG).

- 1: **Initialize** policy parameter θ and value function parameter ϕ
 - 2: **for** iteration = 1, 2, ... **do**
 - 3: Collect a set of trajectories $\mathcal{D} = \{\tau_i\}_{i=1, \dots, N}$ by running the current policy
 - 4: At each timestep in each trajectory compute $R_t = \sum_{t'=t}^T r_{t'+1}$ and $\hat{A}_t = R_t - V_{\phi}(s_t)$
 - 5: Fit the value function estimate by minimizing $\|V_{\phi}(s_t) - R_t\|^2$ over all timesteps in \mathcal{D}
 - 6: Compute the policy update by summing $\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \hat{A}_t$ over all timesteps in \mathcal{D}
-

Some disadvantages of such defined policy-based methods (Williams, 1992; Jaakkola, Jordan, and Singh, 1994; Sutton, McAllester, Singh, et al., 2000) is that they generally take longer to converge and tend to converge to local optima rather than the global optimum due to premature convergence to a near-deterministic policy that obtains sub-optimal rewards. Fortunately, this problem can be alleviated by using alternative ways of estimating the advantage function, or by using entropy regularization in the policy update.

Remark 2.2 (Control Variates). *This technique of adding a baseline is sometimes referred to as an additive control variate (Nelson, 1990; Greensmith, Bartlett, and Baxter, 2004), a generic approach to reducing the variance of Monte Carlo estimators by using a function with known mean whose behavior is correlated with a function of interest. A second application of control variates is developed in the following under the name of actor-critic methods and consists of using a learnt value function instead of the discounted value function estimate.*

2.6.3 Actor-Critic Methods

When policy-gradient methods use an estimation of the value function in the estimation of the returns, they generally are granted with another name: *actor-critic methods*. In this case, the policy and value function are intertwined and optimized jointly: the policy is referred to as the *actor*, and the value function as the *critic*. This joint optimization formulation often combined with bootstrapping is preferred when working with large-scale RL problems where training either of the estimators to convergence is at best challenging.

Recall that an unbiased estimator of the policy gradient can be written as:

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^T \hat{A}^\pi(s_t, a_t) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t),$$

where \hat{A}^π is an estimation of the advantage function. For very long trajectories, we can further reduce the variance by using a discount factor, which reduces variance at the cost of bias: $\hat{A}^\pi(s_t, a_t) = \sum_{t'=t}^T \gamma^{t'-t} r_{t'+1} - \hat{V}^\pi(s_t)$ where in this case the value function should estimate the *discounted sum of rewards* by minimizing some notion of distance between the value function and the true discounted value function. Some examples of such approximation techniques include TD (Sutton, 1988), LSTD (Bradtke and Barto, 1996) or more recently Generalized Advantage Estimators (Schulman, Moritz, Levine, et al., 2016).

Advantage Function Estimation. Following an idea first encountered in the literature in Kimura and Kobayashi (1998) and Wawrzyński (2009), and popularized as Generalized Advantage Estimators (GAE) (Schulman, Moritz, Levine, et al., 2016), we can form a k -step advantage function estimator:

$$\hat{A}^{(k)}(s_t, a_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k \hat{V}(s_{t+k}) - \hat{V}(s_t),$$

where $k \in \{1, 2, \dots, \infty\}$ and $\hat{V}(s)$ is a value function estimator. Early adoption of such estimators in the policy gradient are observed in advantage actor-critic methods (Mnih, Badia, Mirza, et al., 2016; Schulman, Moritz, Levine, et al., 2016) such as A2C and A3C. Since the value function will often have a non-negligible estimation error, there will always be some bias in the gradient estimator, and a delicate choice of k will further balance the bias-variance ratio.

Exploration and Off-Policy. Other actor-critic methods address the problem of premature convergence to sub-optimal policies due to a lack of exploration: they use entropy regularization in the policy update. For instance, Soft Actor-Critic (SAC) (Haarnoja, Zhou, Abbeel, et al., 2018; Haarnoja, Zhou, Hartikainen, et al., 2018) is a method based on the maximum entropy

Background

framework which objective encourages policy stochasticity by augmenting the reward with the entropy at each step. Notably, SAC is an off-policy actor-critic algorithm, meaning it is able to efficiently reuse old data leading to potential instability. On the contrary, on-policy methods estimate the return for state-action pairs assuming the current policy continues to be followed. Deep Deterministic Policy Gradient (DDPG) (Silver, Lever, Heess, et al., 2014; Lillicrap, Hunt, Pritzel, et al., 2016) and Twin Delayed DDPG (TD3) (Fujimoto, Hoof, and Meger, 2018) are other off-policy algorithms which interleave learning a Q-function and a policy, with TD3 tackling the overestimation of the Q-values in DDPG.

Trust Region Optimization. Similar to SAC, Trust Region Policy Optimization (TRPO) (Schulman, Levine, Abbeel, et al., 2015) can be viewed as a policy iteration scheme but where the greedy step is penalized with a Kullback-Leibler (KL) penalty between two consecutive policies (generally a previous “old” policy and the current policy), instead of the entropy of the current policy. This mechanism, known as *trust region optimization*, allows the policy to be improved as much as possible by taking a gradient step in the policy parameter space without deviating too much from the current policy and avoiding a potentially irrecoverable drop in performance. While TRPO uses a resource-intensive second-order method for trust region optimization, Proximal Policy Optimization (PPO) (Schulman, Wolski, Dhariwal, et al., 2017) takes its inspiration from TRPO, natural policy gradient (Kakade, 2002; Peters and Schaal, 2008a) and conservative policy iteration (Kakade and Langford, 2002) to introduce a simpler to implement first-order method. TRPO and PPO also resemble Mirror Descent (Beck and Teboulle, 2003) in that they use a linearization of the objective function with a proximity term which restricts two consecutive updates to be “close” to each other. In the same vein, Actor Critic using Kronecker-Factored Trust Region (ACKTR) (Wu, Mansimov, Grosse, et al., 2017) uses Kronecker-factored natural gradient approximations to tackle the scalability shortcomings of TRPO in learning in very high-dimensional observation space, *e.g.* directly from raw pixel observations.

2.7 RL in the Real World

Although we do not address the application of RL to real-world conditions until the final part of this thesis, it is generally admitted that this ultimate goal must, from the outset, be part of the way in which RL research is advanced. Furthermore, the possibility of re-use of our work by fellow researchers should serve as a guiding framework for the experimental protocol, scientific paper writing and access to code. We will discuss this topic further next. On a more practical note, applying the current state-of-the-art in RL directly in the real world is challenging, in part because most of the algorithms cannot be safely deployed in real-world systems. For instance, when all training occurs in a simulator, the translation of behaviour to the real world,

or *sim-to-real* transition, often fails because of the inherent domain-shift between the simulation and the real world. This can be prone to causing damage to the physical world or not being suitable for the application environment. One might then legitimately ask why not train our agents directly in the real world. Indeed, part of the problem is the large sample sizes required by many RL algorithms, requiring tens of thousands of trials in the real world, which can be technically unsustainable and potentially very expensive.

In this thesis, we specifically work on methods to improve the sampling efficiency of deep RL algorithms through the prism of actor-critic methods. So far, all the methods we have described in this chapter are referred to as *model-free* methods. But there are other approaches to make RL more efficient in terms of sampling, such as *model-based* methods, which mostly rely on the estimated dynamics to derive the corresponding optimal controls, and imitation learning (or learning from demonstration) methods, where an agent learns from an expert who demonstrates the desired behaviour rather than directly learning the policy by trial-and-error. In the following sections, we discuss how achieving a level of generalization in deep RL is a big step towards applying it to the real world, and we address the question of the extent to which the promises of Hierarchical RL could help. Appendix E includes additional examples of projects not presented in the thesis that aim to make a positive contribution to open research and education.

2.7.1 Generalization in Deep RL

Several works (Packer, Gao, Kos, et al., 2018; Zhang, Ballas, and Pineau, 2018; Zhang, Vinyals, Munos, et al., 2018) acknowledge there is a sort of a “replication crisis” in the domain of Deep Reinforcement Learning and observe a growing demand for generalization in particular in robotics via the concept of reality gap where real-world problems certainly involve intrinsic noise and uncertainty with novel conditions encountered (Sünderhauf, Brock, Scheirer, et al., 2018). Although this thesis does not focus on generalization in deep RL, we provide directions towards the development of new exploration methods that can generalize well to unseen scenarios by evaluating our agents on procedurally-generated environments in Chapter 8, and we propose an approach to learning risk-averse agents in real-world inspired tasks in Chapter 9.

Standardized and Reproducible Evaluations. A well-designed evaluation protocol is the cornerstone of research and is naturally inseparable from research in Artificial Intelligence. In 2019, NeurIPS conference was the first to introduce a Machine Learning Reproducibility checklist (Pineau, Vincent-Lamarre, Sinha, et al., 2020) as part of the submission process and other conferences including ICLR and ICML have followed by introducing reproducibility programs. Those programs are designed to improve the standards of machine learning research. As an example, most of the time, deep RL algorithms are compared based on training return.

Background

In fact, standard deviation of returns and average return are generally considered to be the most stable measures used to compare the performance of the algorithms being studied (Islam, Henderson, Gomrokchi, et al., 2017). Nevertheless, they can be of high variance hence a non-rigorous experimental protocol will lead to uninterpretable results (Henderson, Islam, Bachman, et al., 2018). One necessary but not sufficient step is to use a sufficient number of random seeds to allow reliable comparison among algorithms (Colas, Sigaud, and Oudeyer, 2018; Pineau, 2018; Zhang, Ballas, and Pineau, 2018). Another important component for contributing meaningfully to a study is to use a standardized evaluation protocol. In the context of deep RL, this means choosing standard environment models and benchmarks, and preferably open source code bases whenever possible.

Experimental Protocol and Best Practices. In this part, we consider *a few* useful good practices developed by the research community for the research community which we learned to apply during the course of this thesis. In general, a safe way to ensure that one compares two algorithms fairly is to keep all other things equal. This can be done for example by using similar neural network architectures for our method and baselines. For the case of one-dimensional input vectors, researchers usually pick a two- or three-layered Multi-Layer Perceptron with maximum hidden sizes of 256 or 512. For the case of two-dimensional input images a structure equivalent to the convolutional architecture of the Nature paper of DQN (Mnih, Kavukcuoglu, Silver, et al., 2015) seems to be preferred. In any cases, it is generally a good practice to ensure this information is given in the paper, or the code provided (open-source software is a good practice to help reproducible research). Another critical aspect of research is to engage in fair comparisons between one’s new method and baselines. Whether it is a personal implementation or (official) code published on GitHub, it is always worth checking the performance of the code against that reported in the original paper. In addition, it is paradoxically quite complicated to keep a method simple. Sometimes, when a new method finally shows interesting results, it is almost more challenging to dry it out and make it as lean as possible. One way of doing this is to dissect the components of the method and do ablative studies to see the impact of each component on performance. Being able to justify each part of the method will then become easier and make the work stronger as a whole. Needless to say that a final good practice is to not consider random seeds as hyperparameters. These elements of best practices are discussed in more details in the following works (Whiteson, Tanner, Taylor, et al., 2011; Baker, 2016; Islam, Henderson, Gomrokchi, et al., 2017; François-Lavet, Henderson, Islam, et al., 2018).

2.7.2 Hierarchical Reinforcement Learning

Real-world applications also calls for more interpretable methods. The Hierarchical Reinforcement Learning setting is often based on the *options framework* (Sutton, Precup, and Singh, 1999)

or contextual policies (Kupcsik, Deisenroth, Peters, et al., 2013; Schaul, Horgan, Gregor, et al., 2015). Both use the concept of *temporal abstraction* which allows representing knowledge about courses of action that take place at different time scales. In this case, we talk about temporally extended actions. The options framework combines low-level policies with a top-level policy that invokes individual sub-policies and contextual policies generalize options to continuous goals. For more details on Hierarchical RL methods, refer to Schmidhuber and Wahnsiedler (1993), Dayan and Hinton (2000), Barto and Mahadevan (2003), Nachum, Gu, Lee, et al. (2018), and Nachum, Tang, Lu, et al. (2019). Fundamentally, such methods are interesting for real-world applications because of their interpretability: high-level actions often correspond to more semantically meaningful behaviors. In addition, many real-world tasks may be decomposed into natural hierarchical structures (Flet-Berliac, 2019), further opening the door to reusable models, multi-task learning and pre-training of general-purpose neural networks (Haarnoja, Hartikainen, Abbeel, et al., 2018; Shu, Xiong, and Socher, 2018; Li, Wang, Tang, et al., 2019; Zhou, Yu, Chen, et al., 2019). However, in addition to a generally increased sample complexity, challenges such as the non-stationarity generated by updating the levels of a hierarchical agent require more effort in implementation and introduce additional hyperparameters.

Part II

Variance in the Value Function estimates

Chapter 3

Dynamic Control Problems

*One of the most gratifying results
of intellectual evolution
is the continuous opening up
of new and greater prospects.*

Electrical Possibilities in Coal and Iron, Nikola Tesla (1915).

Part II studies continuous control problems in which we focus on the design of sample-efficient learning algorithms. In this chapter, we introduce the problem of learning continuous control policies and present the inference scheme that allows it to be used with deep neural network representations in high-dimensional continuous state and action spaces.

Contents

3.1	Physics Engines for Robotics Applications	36
3.2	General Solutions for Continuous Control tasks	37

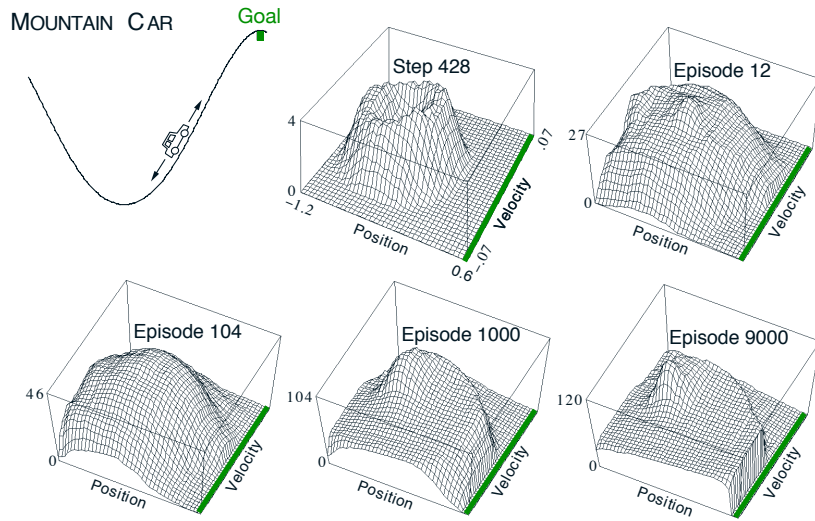


Figure 3.1 – Negative of the empirical value function ($-\max_a \hat{Q}^\pi(s, a)$) learned on a single run by an agent in the Mountain Car task (illustrated in the upper left panel). Each other panel shows what happens while learning to solve the task using semi-gradient Sarsa. Taken from Sutton and Barto (1998).

3.1 Physics Engines for Robotics Applications

Recently, the deep RL community has generally been using a set of benchmark problems inspired by the MuJoCo physics engine (Todorov, Erez, and Tassa, 2012) and integrated with OpenAI Gym (Brockman, Cheung, Pettersson, et al., 2016). In most cases, the optimal control problem for these tasks is to make the simulation of a robot (creature or humanoid) to walk/move as fast as possible (*i.e.* using as few actions as possible) in one direction. In those tasks, the robots interact with environments by the contact force. Some tasks are simpler than others to learn, depending on the degrees of freedom of the simulated robot. For instance, the Mountain Car task is a simplified physics environment illustrated in Figure 3.1 which has only 3 degrees of freedom, meaning the dimension of the action space \mathcal{A} of the task is 3, with the following actions: full acceleration forward, full acceleration reverse and zero acceleration. On the contrary, the Humanoid robot learning task is more challenging since it has 17 degrees of freedom.

MuJoCo (Todorov, Erez, and Tassa, 2012) stands for Multi-Joint dynamics with Contact. It is a proprietary physics engine for efficient rigid body simulations with contacts and is widely used by the deep RL community as a benchmark for robotics applications. PyBullet (Coumans and Bai, 2016) is another rigid body simulation platform that simulate contact dynamics with a difference that it has an open-source *Zlib* license. We will use MuJoCo and PyBullet extensively in this thesis. Roboschool (Klimov and Schulman, 2017) is another environment platform made for controlling robots in simulation. It was initially built as free alternatives to MuJoCo implementations with modifications to make them more physically realistic. Roboschool is also

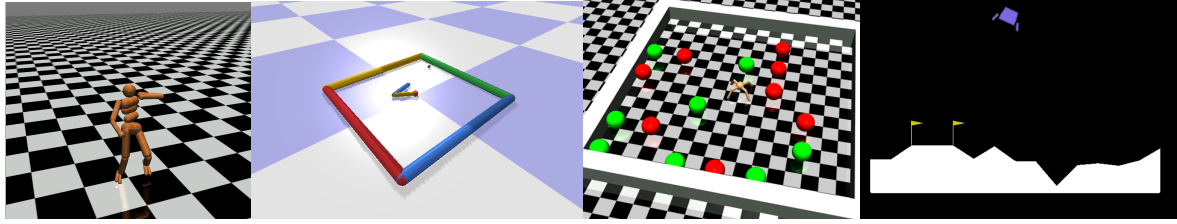


Figure 3.2 – Examples of continuous control tasks. From left to right: Humanoid (MuJoCo), Reacher (PyBullet), AntGather (rllab) and LunarLander (Box2D).

providing several more challenging environments, which we will use in this thesis. Fortunately, RL researchers can still use Roboschool but the project has been deprecated by its creators during the course of this thesis, and now recommend using PyBullet instead¹. Box2D (Catto, 2011) is another rigid body simulation library, mainly for games, and rllab (Duan, Chen, Houthoof, et al., 2016) is another framework similar to OpenAI Gym for developing and evaluating RL algorithms.

3.2 General Solutions for Continuous Control tasks

Many dynamic control tasks have a continuous state space and often continuous action variables. More traditional RL algorithms assume discrete states and actions and are not directly suited to these tasks. One solution is to use discretization of the continuous variables (Millán, Posenato, and Dedieu, 2002; Kimura, 2007). However, depending on the quality of the discretized models, this can lead to unrecoverable errors in the training model, prevent generalization capabilities across similar states or be excessively expensive, especially in very high-dimensional action spaces. Therefore, while these lookup tables work well in an environment with a finite number of states and actions, they are not suitable for MDPs with continuous states and actions due to the curse of dimensionality. In the case of a high-dimensional MDP, function approximation is generally used for its ability to handle multi-dimensional continuous variables and to generalize across similar states.

Value-based methods have remained under-explored RL methods in continuous control, illustrated by the optimization problem $\max_a \hat{Q}^\pi(s, a)$, which is generally difficult to solve if the action space is continuous: the learned value function $\hat{Q}^\pi(s, a)$ may have many local maxima and saddle points (Lim, Joseph, Le, et al., 2019). Policy-based methods have therefore typically been preferred for these kinds of problems. In general, a neural network is used as a function approximation for the policy whose structure depends on the observation (\mathcal{S}) and action (\mathcal{A}) spaces. In practice, the size of the input layer will correspond to the size of the observation space. Moreover, in the context of continuous control problems, the use of the Gaussian distribution

¹Roboschool implementation [GitHub Commit](#).

Dynamic Control Problems

as a stochastic policy has been widely studied and applied since Williams (1992), mainly because the Gaussian distribution is easy to sample and its gradients are simple to calculate. Furthermore, since we generally assume that the action dimensions are independent, we will use fully-factored Gaussian policies where the means of the action distributions will be the outputs of the neural network and the variances will be separate trainable parameters.

Chapter 4

Use Variance in the Value Function estimates as an auxiliary loss

Γνώθι σεαυτόν

–

Know thyself.

Pronaos of the Temple of Apollo, Socrates.

Earlier work in deep RL makes use of domain knowledge to improve the sample efficiency of existing algorithms. While promising, previously acquired knowledge is often costly and challenging to scale up. Instead, we address problem knowledge using signals from quantities relevant to solve any task. In this chapter, we propose to use the coefficient of determination of the value function estimates as an additional objective function (auxiliary loss) in the learning algorithm of policy gradient methods. The first work of this thesis consists in proposing a method theoretically applicable to any deep RL algorithm for learning complementary diagnostic signals of self-performance assessment. The approach is motivated by the prospect of using an auxiliary loss predictive of the variance in the value function estimates to help differentiate the representations of known or unknown areas of the state space¹.

Contents

4.1 Motivation	40
4.2 Preliminaries	41
4.4 MERL: Framework for Self-Performance Assessment	44
4.5 Experimental Study	46

¹This chapter is based on Flet-Berliac and Preux (2019b) presented at the *Deep Reinforcement Learning workshop* at the *33rd conference on advances in Neural Information Processing Systems (NeurIPS)*.

4.1 Motivation

In Part I, we have introduced the problem of learning how to act optimally in an unknown dynamic environment and developed the many research efforts it has generated for decades (Werbos, 1989; Nguyen and Widrow, 1990; Schmidhuber and Huber, 1991) and until recently with work in Deep Reinforcement Learning (Silver, Huang, Maddison, et al., 2016; Burda, Edwards, Storkey, et al., 2018; Espeholt, Soyer, Munos, et al., 2018; Ha and Schmidhuber, 2018). Nonetheless, current algorithms tend to be fragile and opaque (Iyer, Li, Li, et al., 2018): they require a large amount of training data collected from an agent interacting with a simulated environment where the reward signal is often critically sparse. Collecting signals that will make the agent more efficient is, therefore, at the core of the algorithms designers' concerns. Previous work in RL (Clouse and Utgoff, 1992; Lin, 1992b; Ribeiro, 1998; Moreno, Regueiro, Iglesias, et al., 2004) use prior knowledge to reduce sample inefficiency. While promising and unquestionably necessary, the integration of such priors into current methods is likely costly to implement, it may cause undesired constraints and can hinder scaling up. Therefore, we propose a framework to directly integrate non-limiting constraints in current RL algorithms while being applicable to any task. The agent should learn from all interaction data, not just the rewards. Indeed, if the probability of receiving a reward by chance is arbitrarily low, then the time required to learn from it will be arbitrarily long (Whitehead, 1991). This barrier to learning will prevent agents from significantly reducing learning time. One way to overcome this barrier is to learn complementary and task-agnostic signals of self-performance assessment and accurate expectations from different sources, in a similar vein to Schmidhuber (1991) and Oudeyer and Kaplan (2007), whatever the task to master.

Our contributions are the following: from the above considerations and building on existing auxiliary task methods, we design a framework that integrates problem knowledge quantities into the learning process. In addition to providing a method technically applicable to any policy gradient algorithm or environment, the central idea of MERL is to incorporate a measure of the discrepancy between the estimated state value and the observed returns as an auxiliary loss. This discrepancy is formalized with the notion of fraction of variance explained \mathcal{V}^{ex} (Kvålseth, 1985). One intuition that can be developed is that MERL transforms a reward-focused task into a task regularized with dense problem knowledge signals. Figure 4.1 below provides a preliminary understanding of MERL assets: an enriched actor-critic structure with a lightly modified learning algorithm places the agent amidst task-agnostic auxiliary quantities directly sampled from the environment. In the sequel of this chapter, we use two problem knowledge quantities to demonstrate the performance of MERL: \mathcal{V}^{ex} , a measure of self-performance, and future states prediction, commonly used in auxiliary task methods. One objective of this chapter is to encourage the introduction of many other relevant signals. We show that while being able to correctly predict the quantities with the different MERL heads (each minimizing an auxiliary

loss), the agent outperforms the baseline that does not use the MERL framework on various continuous control tasks. We also show that, interestingly, our framework has the potential to take advantage of encoder sharing to allow for better transfer learning on several *Atari 2600* games.

4.2 Preliminaries

We consider a Markov Decision Process (MDP) with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition distribution $s_{t+1} \sim P(s_t, a_t)$ and reward function $r(s, a)$. Let $\pi = \{\pi(a|s), s \in \mathcal{S}, a \in \mathcal{A}\}$ denote a stochastic policy and let the objective function be the traditional expected discounted reward:

$$J(\pi) \triangleq \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (4.1)$$

where $\gamma \in [0, 1)$ is a discount factor (Puterman, 1994) and $\tau = (s_0, a_0, s_1, \dots)$ is a trajectory sampled from the environment. Policy gradient methods aim at modelling and optimizing the policy directly (Williams, 1992). The policy π is generally implemented with a function parameterized by θ . In the sequel, we will use θ to denote the parameters as well as the policy. In deep RL, the policy is represented by a neural network called the policy network and is assumed to be continuously differentiable with respect to its parameters θ . When the policy is represented by such a parameterized function, hence by an approximation of a policy, the MDP theory basically breaks down.

4.2.1 Fraction of Variance Explained

The fraction of variance that the value function V explains about the returns corresponds to the proportion of the variance in the dependent variable V that is predictable from the independent variable s_t . We define \mathcal{V}_{τ}^{ex} as the fraction of variance explained for a trajectory τ :

$$\mathcal{V}_{\tau}^{ex} \triangleq 1 - \frac{\sum_{t \in \tau} (\hat{R}_t - V(s_t))^2}{\sum_{t \in \tau} (\hat{R}_t - \bar{R})^2}, \quad (4.2)$$

where \hat{R}_t and $V(s_t)$ are respectively the return and the expected return from state $s_t \in \tau$, and \bar{R} is the mean of all returns in trajectory τ . In statistics, this quantity is also known as the coefficient of determination R^2 and it should be noted that this criterion may be negative for non-linear models (Kvålseth, 1985), indicating a severe lack of fit of the corresponding function:

- $\mathcal{V}_{\tau}^{ex} = 1$: V perfectly explains the returns - V and the returns are correlated;

- $\mathcal{V}_\tau^{ex} = 0$ corresponds to a simple average prediction - V and the returns are not correlated;
- $\mathcal{V}_\tau^{ex} < 0$: V provides a worse fit to the outcomes than the mean of the returns.

Following this definition, one can have the intuition that \mathcal{V}_τ^{ex} close to 1 implies that the trajectory τ provides valuable signals because they correspond to transitions sampled from an exercised behavior. On the other hand, \mathcal{V}_τ^{ex} close to 0 indicates that the value function is not correlated with the returns, therefore, the corresponding samples are not expected to provide as valuable information as before. Finally, $\mathcal{V}_\tau^{ex} < 0$ corresponds to a high mean-squared error for the value function, which means for the related trajectory that the agent still has to learn to perform better. In Flet-Berliac and Preux (2019c), policy gradient methods are improved by using \mathcal{V}^{ex} as a criterion to dropout transitions before each policy update. We will show that \mathcal{V}^{ex} is also a relevant indicator for assessing self-performance in the context of MERL agents.

4.2.2 Policy Gradient Method: PPO with Clipped Surrogate Objective

In this chapter, we consider on-policy learning² primarily for its unbiasedness and stability compared to off-policy methods (Nachum, Norouzi, Xu, et al., 2017). On-policy is also empirically known as being less sample efficient than off-policy learning hence this issue emerged as an interesting research topic. However, our method can be applied to off-policy methods as well, and we leave this investigation open for future work.

Introduced by Schulman, Wolski, Dhariwal, et al. (2017), PPO is among the most commonly used and state-of-the-art on-policy policy gradient methods. Indeed, PPO has been tested on a set of benchmark tasks and has proven to produce impressive results in many cases despite a relatively simple implementation. For instance, instead of imposing a hard constraint like TRPO (Schulman, Levine, Abbeel, et al., 2015), PPO formalizes the constraint as a penalty in the objective function. In PPO, at each iteration, the new policy θ_{new} is obtained from the old policy θ_{old} :

$$\theta_{new} \leftarrow \operatorname{argmax}_{\theta} \mathbb{E}_{s_t, a_t \sim \pi_{\theta_{old}}} \left[L^{\text{PPO}}(s_t, a_t, \theta_{old}, \theta) \right]. \quad (4.3)$$

We use the clipped version of PPO whose objective function is:

$$L^{\text{PPO}}(s_t, a_t, \theta_{old}, \theta) = \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A^{\pi_{\theta_{old}}}(s_t, a_t), g(\varepsilon, A^{\pi_{\theta_{old}}}(s_t, a_t)) \right), \quad (4.4)$$

where

$$g(\varepsilon, A) = \begin{cases} (1 + \varepsilon)A, & A \geq 0 \\ (1 - \varepsilon)A, & A < 0. \end{cases} \quad (4.5)$$

²We apply MERL in the off-policy policy gradient setting using the DDPG algorithm in Appendix A.1 and A.3.1.

A is the advantage function, $A(s, a) \triangleq Q(s, a) - V(s)$. The expected advantage function $A^{\pi_{old}}$ is estimated by an old policy and then re-calibrated using the probability ratio between the new and the old policy. In Equation 4.4, this ratio is constrained to stay within a small interval around 1, making the training updates more stable.

4.3 Related Work

Auxiliary tasks have been adopted to facilitate representation learning for decades (Sudarth and Kergosien, 1990; Klyubin, Polani, and Nehaniv, 2005), along with intrinsic motivation (Schmidhuber, 2010; Pathak, Agrawal, Efros, et al., 2017) and artificial curiosity (Schmidhuber, 1991; Oudeyer and Kaplan, 2007). The use of auxiliary tasks to allow the agents to maximize other pseudo-reward functions simultaneously has been studied in a number of previous work (Dosovitskiy and Koltun, 2016; Shelhamer, Mahmoudieh, Argus, et al., 2016; Burda, Edwards, Pathak, et al., 2018; Du, Czarnecki, Jayakumar, et al., 2018; Riedmiller, Hafner, Lampe, et al., 2018; Kartal, Hernandez-Leal, and Taylor, 2019), including incorporating unsupervised control tasks and reward predictions in the UNREAL framework (Jaderberg, Mnih, Czarnecki, et al., 2017), applying auxiliary tasks to navigation problems (Mirowski, Pascanu, Viola, et al., 2016), or for utilizing representation learning (Lesort, Diáz-Rodríguez, Goudou, et al., 2018) in the context of model-based RL. Lastly, in imitation learning of sequences provided by experts, Li, Li, Gao, et al. (2016) introduces a supervised loss for fitting a recurrent model on the hidden representations to predict the next observed state.

Our method incorporates two key contributions: a multi-head layer with auxiliary task signals both environment-agnostic and technically applicable to any policy gradient method, and the use of \mathcal{V}_τ^{ex} as an auxiliary task to measure the discrepancy between the value function and the returns in order to allow for better self-performance assessment and eventually more efficient learning. In addition, MERL differs from previous approaches in that its framework simultaneously addresses the advantages mentioned hereafter: (i) neither the introduction of new neural networks (*e.g.* for memory) nor the introduction of a replay buffer or an off-policy setting is needed, (ii) all relevant quantities are compatible with any task and is not limited to pixel-based environments, (iii) no additional iterations are required, and no modification to the reward function of the policy gradient algorithms it is applied to is necessitated. The above reasons make MERL generally applicable and technically suitable out-of-the-box to most policy gradient algorithms with a negligible computational cost overhead.

From a different perspective, Garcia and Fernández (2015) give a detailed overview of previous work that has changed the optimality criterion as a safety factor. But most methods use a hard constraint rather than a penalty; one reason is that it is difficult to choose a single coefficient for this penalty that works well for different problems. We are successfully addressing

this problem with MERL. In Lipton, Azizzadenesheli, Kumar, et al. (2016), catastrophic actions are avoided by training an intrinsic fear model to predict whether a disaster will occur and using it to shape rewards. Compared to both methods, MERL is more scalable and lightweight while it successfully incorporates quantities of self-performance assessments (e.g. variance explained of the value function) and accurate expectations (e.g. next state prediction) leading to an improved performance.

4.4 MERL: Framework for Self-Performance Assessment

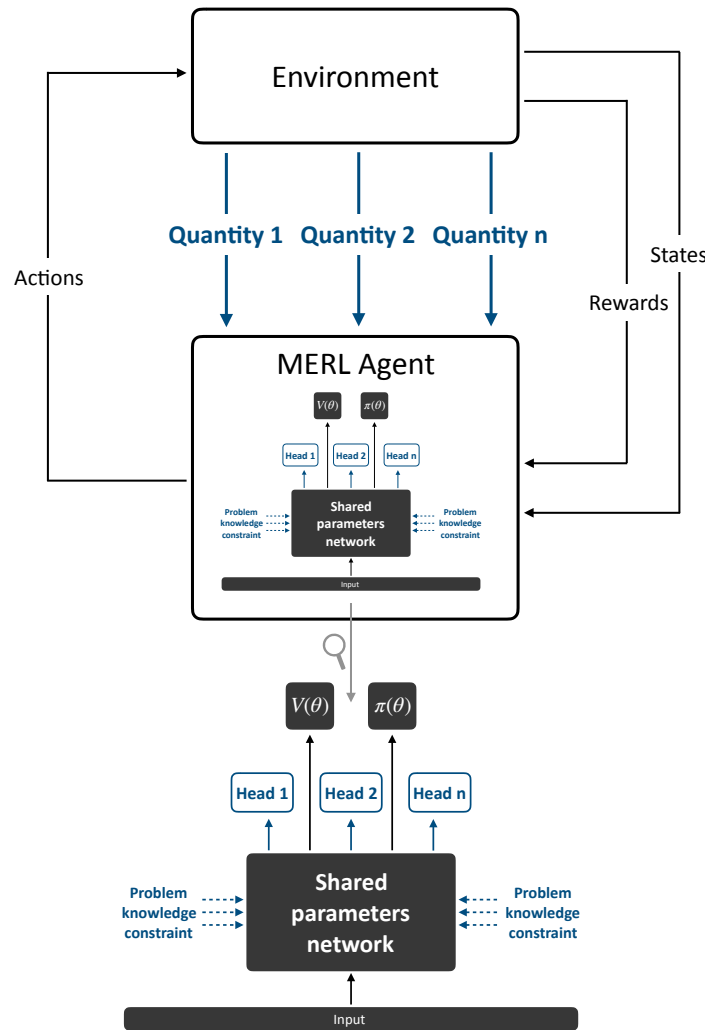


Figure 4.1 – Schematic overview of MERL system.

The multi-head structure and its associated learning algorithm are directly applicable to most state-of-the-art policy gradient methods implying a simple augmentation of the training process. Let h be the index of each MERL head: MERL^h . Below, we review two of the quantities predicted by these heads and derive how to practically incorporate them into PPO.

4.4.1 From Policy and Value Function Representations to MERL^h

In deep RL, the policy is generally represented by a neural network called the policy network, with parameters θ , and the value function is parameterized by the value network, with parameters ϕ . Each MERL head MERL^h takes as input the last embedding layer from the value network and is constituted of only one layer of fully-connected neurons, with parameters ϕ^h . The output size of each head corresponds to the size of the predicted MERL quantity. We introduce below the \mathcal{V}^{ex} estimation and the prediction of future states as two quantity predicted by the method.

4.4.2 Estimation of \mathcal{V}^{ex}

First, let us write MERL^{VE} the MERL head with parameters ϕ^{VE} responsible for predicting an estimate of the fraction of variance explained in trajectory τ . Its objective function is defined as:

$$L^{\text{MERL}^{\text{VE}}}(\tau, \phi, \phi^{\text{VE}}) = \|\text{MERL}^{\text{VE}}(\tau) - \mathcal{V}_\tau^{ex}\|_2^2. \quad (4.6)$$

4.4.3 Estimation of Future States

Auxiliary task methods based on next state prediction are, to the best of our knowledge, the most commonly used in the RL literature. We include such auxiliary task into MERL, in order to assimilate our contribution to the previous work and to provide a enriched evaluation of the proposed framework. At each timestep, one of the agent’s MERL heads predicts a future state s' from s . While a typical MERL quantity can be fit by regression on mean-squared error, we observed that predictions of future states are better fitted with a cosine-distance error. We denote MERL^{FS} the corresponding head, with parameters ϕ^{FS} , and S the observation space size (size of vector s). We define its objective function as:

$$L^{\text{MERL}^{\text{FS}}}(s, \phi, \phi^{\text{FS}}) = 1 - \frac{\sum_{i=1}^S \text{MERL}_i^{\text{FS}}(s) \cdot s'_i}{\sqrt{\sum_{i=1}^S (\text{MERL}_i^{\text{FS}}(s))^2} \sqrt{\sum_{i=1}^S (s'_i)^2}}. \quad (4.7)$$

4.4.4 Problem-Constrained Policy Update

Once a set of MERL heads MERL^h and their associated objective functions L^{MERL^h} have been defined, we modify the gradient update step of the policy gradient algorithms. The objective function incorporates all L^{MERL^h} . Of course, each MERL objective is associated with its coefficient c_h . It is worthy to note that we used the exact same MERL coefficients for all our experiments, which demonstrate the framework’s ease of applicability without the need for further hyperparameter tuning. Algorithm 3 illustrates how the learning is achieved. In Equation 4.9, only the boxed

Use Variance in the Value Function estimates as an auxiliary loss

MERL objectives parameterized by ϕ are added to the value function update and modify the learning algorithm.

Algorithm 3 MERL coupled with PPO.

Initialize policy parameters θ_0

Initialize value function and MERL^h functions parameters ϕ_0

for $k = 0, 1, 2, \dots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ with length T by running policy π_{θ_k}

Compute MERL^h estimates

Compute advantage estimates A_t based on the current value function V_{ϕ_k}

Compute sum of future rewards \hat{R}_t

Gradient Update

$$\theta_{k+1} = \operatorname{argmax}_{\theta} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\varepsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right) \quad (4.8)$$

$$\phi_{k+1} = \operatorname{argmin}_{\phi} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi_k}(s_t) - \hat{R}_t \right)^2 + \boxed{\sum_{h=0}^H c_h L^{\text{MERL}^h}} \quad (4.9)$$

4.5 Experimental Study

4.5.1 Methodology

We evaluate MERL in multiple high-dimensional environments, ranging from *MuJoCo* (Todorov, Erez, and Tassa, 2012) to the *Atari 2600* games (Bellemare, Naddaf, Veness, et al., 2013) (we describe these environments in detail in Appendix A.4). The experiments in *MuJoCo* allow us to evaluate the performance of MERL on a large number of different continuous control problems. It is worthy to note that the universal characteristics of the auxiliary quantities we design ensure that MERL is directly applicable to any task. Other popular auxiliary task methods (Mirowski, Pascanu, Viola, et al., 2016; Jaderberg, Mnih, Czarnecki, et al., 2017; Burda, Edwards, Pathak, et al., 2018) are not applicable out-of-the-box to continuous control tasks like *MuJoCo*. Therefore we compare the performance of our method with PPO (Schulman, Wolski, Dhariwal, et al., 2017) where MERL heads are not used. Later, we also experiment with MERL on the *Atari 2600* games to study the transfer learning abilities of our method on a set of diverse tasks.

Implementation. For the continuous control *MuJoCo* tasks, the agents have learned using separated policy and value networks. In this case, we augment the value network’s structure to

incorporate the heads. On the contrary, when playing *Atari 2600* games from pixels, the agents learn using a CNN (Krizhevsky, Sutskever, and Hinton, 2012) shared between the policy and the value function. In that case, MERL^h heads are embedded to the last embedding layer of the shared network. In both configurations, the outputs of MERL^h heads are the same size as the quantity they predict: for instance, MERL^{VE} is a scalar whereas MERL^{FS} is a state. We provide further implementation details in Appendix A.2. The code for our method is released and open-source: github.com/yfletberliac/merl.

Hyperparameters. We used the same hyperparameters as in the main text of the corresponding paper. We made this choice within a clear and objective protocol of demonstrating the benefits of using MERL. Hence, its reported performance is not necessarily the best that can be obtained, but it still exceeds the baseline. Using MERL adds as many hyperparameters as there are heads in the multi-head layer and it is worth noting that MERL hyperparameters are the same for all tasks. We report all hyperparameters in Tables 4.1 and 4.2.

Table 4.1 – Hyperparameters used in PPO+MERL

Hyperparameter	Value
Horizon (T)	2048 (<i>MuJoCo</i>), 128 (<i>Atari 2600</i>)
Adam stepsize	$3 \cdot 10^{-4}$ (<i>MuJoCo</i>), $2.5 \cdot 10^{-4}$ (<i>Atari 2600</i>)
Nb. epochs	10 (<i>MuJoCo</i>), 3 (<i>Atari 2600</i>)
Minibatch size	64 (<i>MuJoCo</i>), 32 (<i>Atari 2600</i>)
Number of actors	1 (<i>MuJoCo</i>), 4 (<i>Atari 2600</i>)
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ε)	0.2 (<i>MuJoCo</i>), 0.1 (<i>Atari 2600</i>)
Value function coef	0.5

Table 4.2 – MERL hyperparameters

Hyperparameter	Value
MERL^{VE} coef c_{VE}	0.5
MERL^{FS} coef c_{FS}	0.01

Performance Measures. We examine the performance across a large number of trials (with different seeds for each task). Standard deviation of returns, and average return are generally considered to be the most stable measures used to compare the performance of the algorithms being studied (Islam, Henderson, Gomrokchi, et al., 2017). Thereby, in the rest of this work, we use those metrics to establish the performance of our framework quantitatively.

4.5.2 Single-Task Learning: Continuous Control

We apply MERL to PPO in several continuous control tasks, where using auxiliary tasks has not been explored in detail in the literature. Specifically, we use 9 *MuJoCo* environments for which we show 6 graphs from varied tasks in Figure 4.2. The complete set of 9 tasks is reported in Table 4.3, and the rest of the figures in Appendix A.3.1.

Table 4.3 – Average total reward of the last 100 episodes over 7 runs on the 9 *MuJoCo* environments. **Boldface** $mean \pm std$ indicate statistically better performance.

Task	PPO	PPO+MERL
Ant	1728 \pm 64	2157 \pm 212
HalfCheetah	1557 \pm 21	2117 \pm 370
Hopper	2263 \pm 125	2105 \pm 200
Humanoid	577 \pm 10	603 \pm 8
InvertedDoublePendulum	5965 \pm 108	6604 \pm 130
InvertedPendulum	474 \pm 14	497 \pm 12
Reacher	-7.84 \pm 0.7	-7.78 \pm 0.8
Swimmer	93.2 \pm 8.7	124.6 \pm 5.6
Walker2d	2309 \pm 332	2347 \pm 353

The results show that using MERL can lead to better performance on a variety of continuous control tasks. Moreover, learning seems to be more sample efficient for some tasks. This suggests that MERL can take advantage of its heads to learn relevant quantities from the beginning of learning, when the reward signals may be sparse. Interestingly, by looking at the performance across all 9 tasks, we observed better results by predicting only the next state and not the subsequent ones.

4.5.3 Transfer Learning: Atari 2600 Domain

Because of training resource constraints, we consider a transfer learning setting where, after the first 10^6 training steps, the agent switches to a new task for another 10^6 steps. The agent is not aware of the task switch. *Atari 2600* has been a challenging testbed for many years due to its high-dimensional video input (210×160) and the discrepancy of tasks between games. To investigate the advantages of MERL in transfer learning, we choose a set of 6 *Atari 2600* games with an action space of 9, which is the average size of the action space in the *Atari 2600* domain. This experimental choice is beneficial in that the 6 games provide a diverse range of game-play while sticking to the same size of action space.

Figure 4.3 demonstrates that our method can better adapt to different tasks. This can suggest that MERL heads learn and help represent information that is more generally relevant for other tasks, such as self-performance assessment or accurate expectations. In addition to adding a regularization term to the objective function with problem knowledge signals, those auxiliary

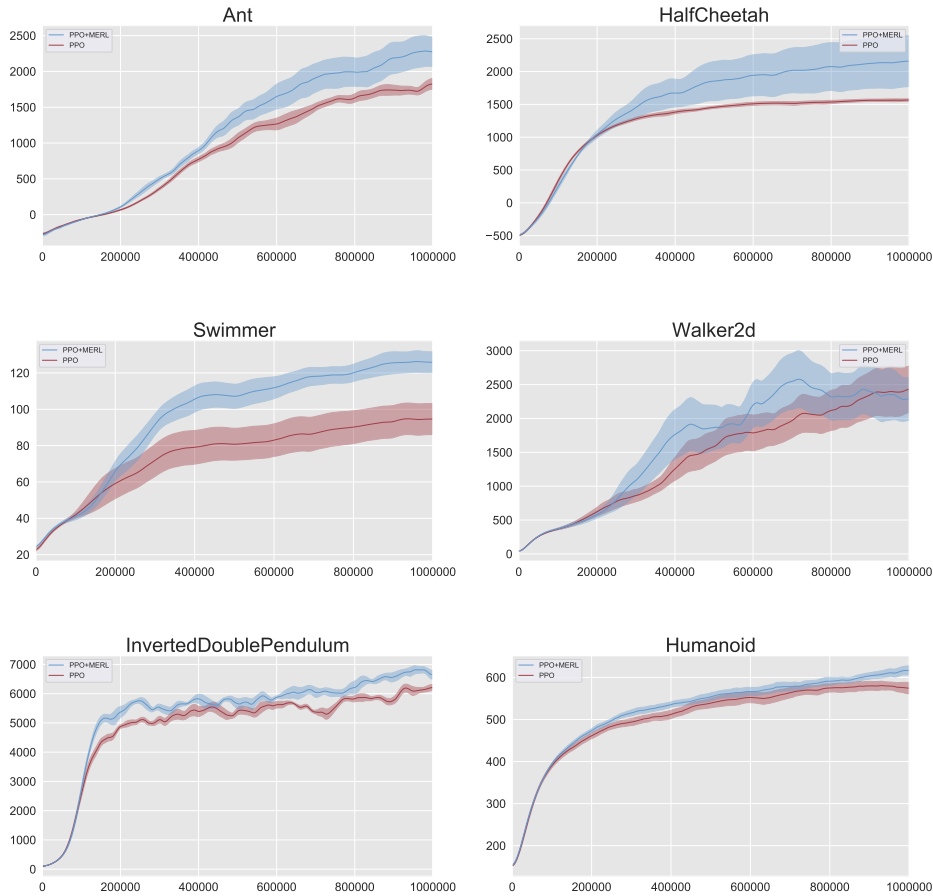


Figure 4.2 – Experiments on 6 *MuJoCo* environments (10^6 timesteps, 7 seeds) with PPO+MERL. Red is the baseline, blue is with our method. The line is the average performance, while the shaded area represents its standard deviation.

quantities make the neural network optimize for task-agnostic meta-objectives. The full set of results can be found in Appendix A.3.2.

4.5.4 Ablation Study

We conduct an ablation to evaluate the separate and combined contributions of the two heads. Figure 4.4 shows the comparative results in HalfCheetah, Walker2d, and Swimmer. Notably, with HalfCheetah, using only the MERL^{VE} head degrades the performance, but when it is combined with the MERL^{FS} head, it outperforms PPO+FS. Results of the complete ablation analysis demonstrate that each head is potentially valuable for enhancing learning and that their combination can produce remarkable results. In addition, it may be intuited that finding a variety of complementary MERL heads to cover the scope of the problem in a holistic perspective can significantly improve learning.

Use Variance in the Value Function estimates as an auxiliary loss

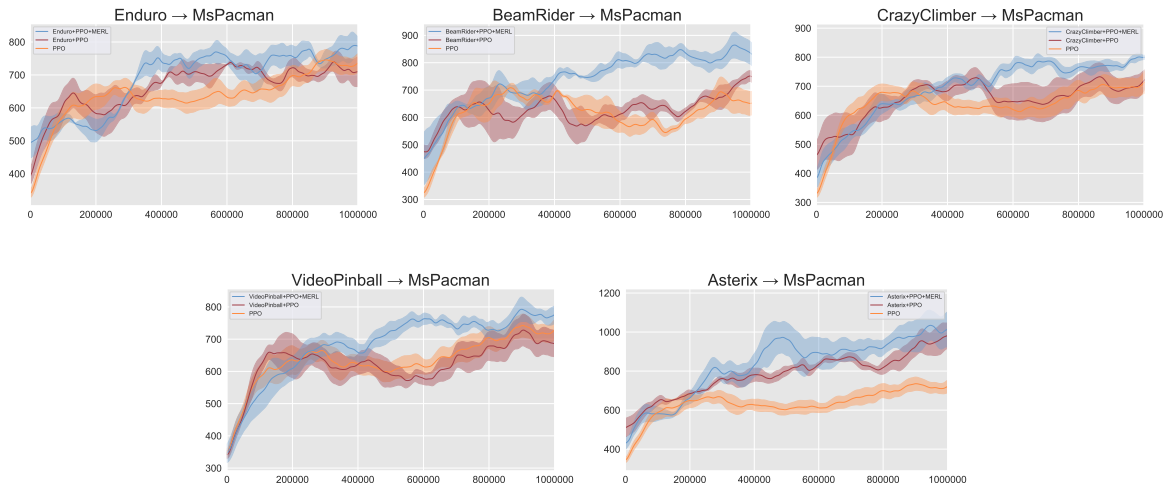


Figure 4.3 – Transfer learning tasks from 5 *Atari 2600* games to Ms. Pacman (2×10^6 timesteps, 6 seeds). Performance on the second task. Orange is PPO solely trained on Ms. Pacman, red and blue are respectively PPO and our method transferring the learning. The line is the average performance, while the shaded area represents its standard deviation.

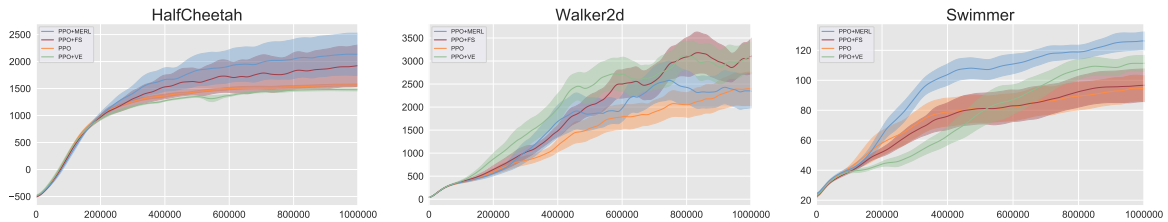


Figure 4.4 – Ablation experiments with only one MERL head (FS or VE) (10^6 timesteps, 6 seeds). Blue is MERL with the two heads, red with the FS head, green with the VE head and orange with no MERL head. The line is the average performance, the shaded area represents its standard deviation.

4.5.5 Discussion

The experiments suggest that MERL successfully optimizes the policy according to complementary quantities seeking for good performance and safe realization of tasks, *i.e.* it does not only maximize a reward but instead ensures the control problem is appropriately addressed. Moreover, we show that MERL is directly applicable to policy gradient methods while adding a negligible computation cost. Indeed, for the *MuJoCo* and *Atari 2600* tasks, the computational cost overhead is respectively 5% and 7% with our training infrastructure. All of these factors result in a generally applicable algorithm that more robustly solves difficult problems in a variety of environments with continuous action spaces or by using only raw pixels for observations. Thanks to a consistent choice of complementary quantities injected in the optimization process, MERL is able to better align an agent’s objectives with higher-level insights into how to solve a control problem. Besides, since many current methods involve that successful learning depends on the agent’s chance to reach the goal by chance in the first place, correctly predicting

MERL heads gives the agent an opportunity to learn from useful signals while improving in a given task.

Chapter conclusion

In this chapter, we propose \mathcal{V}^{ex} as a new auxiliary loss to measure the discrepancy between the value function and the returns. Our results suggest that this quantity helps learn more efficiently by successfully assessing the agent’s performance. We also proposed MERL, a generally applicable deep RL framework for learning problem-focused representations using encoder sharing, which we demonstrated the effectiveness with a combination of two auxiliary tasks. We established that injecting problem knowledge signals directly in the policy gradient optimization allows for a better state representation that is generalizable to many tasks. \mathcal{V}^{ex} provides a more problem-focused state representation to the agent, which is, therefore, not only reward-centric. MERL can be labeled as being a hybrid model-free and meta-learning framework, formed with lightweight embedded representations of self-performance assessment and accurate expectations. MERL heads introduce a regularization term to the function approximation while addressing the problem of reward sparsity through auxiliary task learning. Those features nourish a framework technically applicable to any policy gradient algorithm or environment; it does not need to be redesigned for different problems and could be extended with other relevant problem-solving quantities, comparable to \mathcal{V}^{ex} , such as prediction of time left until the end of a trajectory, echoing the work in Pardo, Tavakoli, Levдик, et al. (2018).

Chapter 5

Use Variance in the Value Function estimates to filter information

Information is the resolution of uncertainty.

Claude Shannon (1948).

A key mechanism gradient-based methods employ to optimize stochastic policies is to repeatedly compute a noisy estimate of the gradient of performance to and insert it into a stochastic gradient descent algorithm. In this chapter, we study the simple but effective idea that an RL agent will learn more effectively from some transitions than others. We employ the statistics of self-performance assessment introduced in the previous chapter to develop a modification to policy gradient algorithms where samples are filtered out when estimating the policy gradient¹.

Contents

5.1	Motivation	54
5.2	Preliminaries	55
5.4	SAUNA: Dynamic Transition Filtering	58
5.5	Experimental Study	59
5.6	Discussion	64

¹This chapter is based on an article published in the proceedings of the *29th International Joint Conference on Artificial Intelligence (IJCAI)* (Flet-Berliac and Preux, 2020).

5.1 Motivation

Part I addressed the challenge of learning to control agents in simulated environments and the recent research efforts led in this direction (Silver, Huang, Maddison, et al., 2016; Espeholt, Soyer, Munos, et al., 2018; Ha and Schmidhuber, 2018), notably in policy gradient methods (Silver, Lever, Heess, et al., 2014; Schulman, Moritz, Levine, et al., 2016; Haarnoja, Zhou, Abbeel, et al., 2018). Despite the undeniable progress, policy gradient algorithms still heavily suffer from sample inefficiency (Kakade, 2003; Wang, Bapst, Heess, et al., 2017; Wu, Mansimov, Grosse, et al., 2017). In particular, many of those methods are subject to use as much experience as possible in the most efficient way. However, quantity is not quality: the quality of the sampling procedure also determines the learning curve of the agent and its final performance. Hence, in this chapter, we hypothesize that *not all experiences are worth using* in the gradient update. Indeed, some transitions may add noise to the gradient update, diluting relevant signals, and hindering learning. The central idea of SAUNA is to reject the transitions that are not informative.

It is reasonable to assume that the use of non-informative or misinformative transitions can only mislead the learning process and waste computational time. In fact, Amari’s natural gradient (Amari, 1998) concept concerns the geometry of the search space related to the “value of information”: this has been studied for long in RL since (Kakade, 2002). In this chapter, we focus our work on a different notion of value of information, and treat it differently: we evaluate whether a transition conveys useful information and use it only if it is considered beneficial to learning. For this purpose, we use a measure of the discrepancy between the estimated state value and the observed returns. This discrepancy is formalized with the notion of the fraction of variance explained \mathcal{V}^{ex} (Kvålseth, 1985), which we defined in Chapter 4 and that we impregnate here with the idea presented. Transitions for which \mathcal{V}^{ex} is close to zero are those for which the correlation between the value function V and the observed returns is also close to zero. SAUNA keeps transitions where there is either a strong correlation or a lack of fit between V and the returns while avoiding the dilution of useful information by removing useless samples. We consider on-policy methods for their unbiasedness and stability compared to off-policy algorithms (Nachum, Norouzi, Xu, et al., 2017). However, this method can be applied to off-policy methods as well, and we leave this investigation open for future work.

In summary, the work presented in this chapter:

1. Proposes to move from a traditional policy-based sampling procedure to a refined sample selection driven by \mathcal{V}^{ex} . We explore how transition filtering simplifies the underlying state space and affects performance;
2. Hypothesizes that not all samples are useful for learning and that disturbing samples should be rejected to avoid performance loss. We provide experimental evidence corroborating this claim;

3. Finally, by combining (1) and (2), we obtain a learning algorithm that is empirically effective in learning neural network policies for challenging control tasks. Our results significantly improve the state of the art in using RL for high-dimensional continuous control.

Section 5.2 recalls basic notions of policy gradient methods in RL and the notion of “fraction of variance explained” drawn from the statistics literature. Section 5.2.2 sets the contribution of this chapter within the RL domain. Section 5.4 introduces SAUNA. Section 5.5 provides experimental evidence of the benefit of using the presented method, and also investigates various experimental aspects of SAUNA. Section 5.6 further discusses the method. Finally, Section 5.6.3 concludes and draws some lines of future research.

5.2 Preliminaries

5.2.1 Notations

Recalling the setting used in Chapter 4, we consider a Markov Decision Process (MDP) with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition distribution $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and reward function $r_t \sim \mathcal{R}(s_t, a_t)$. Let $\pi(a|s)$ denote a stochastic policy and let the objective function be the expected sum of discounted rewards:

$$J(\pi) \triangleq \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (5.1)$$

where $\gamma \in [0, 1)$ is a discount factor (Puterman, 1994) and $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ is a trajectory sampled from the environment while the agent is following a given policy π . Let us remind the notions of the value of a state in the MDP framework. The value $V^\pi(s)$ of a state s while following a policy π starting in state s is defined by: $V^\pi(s) \triangleq \mathbb{E}_{\tau \sim \pi} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s]$. Closely related is the value of a state-action pair: the quality $Q^\pi(s, a)$ of performing action a in state s and then following policy π is defined by: $Q^\pi(s, a) \triangleq \mathbb{E}_{\tau \sim \pi} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$. Finally, the advantage function quantifies how an action a is better than the average action in state s following policy π : $A^\pi(s, a) \triangleq Q^\pi(s, a) - V^\pi(s)$. MDP theory asserts that there exists an optimal policy π^* that maximizes J : we denote its value function V^* . In practice, value functions are unknown; we denote V , Q , and A their current estimates.

Policy Gradient Methods. Here, we recall that policy gradient methods aim at optimizing the policy directly (Williams, 1992). The policy π is often implemented with a function parameterized by θ : learning a policy boils down to finding the best parameters. In the sequel, we use θ to denote the parameters as well as the policy. In this chapter, we still consider the on-policy

Use Variance in the Value Function estimates to filter information

policy gradient method PPO (Schulman, Wolski, Dhariwal, et al., 2017) achieving state of the art performance on a suite of benchmark tasks despite a relatively simple implementation. Very interestingly, PPO is an evolution of TRPO that builds on the notion of natural gradient, hence Amari’s notion of “value of information” mentioned above. PPO has been shown to outperform TRPO experimentally. By building on PPO, this chapter combines two different ideas related to the notion of the value of information. Recalling previous definitions, at each episode, PPO collects (s_t, a_t, r_t) samples using its current policy θ_k . After some episodes, using these collected transitions, PPO updates its policy and gets a new one θ_{k+1} :

$$\theta_{k+1} \leftarrow \operatorname{argmax}_{\theta} \mathbb{E}_{s_t, a_t \sim \pi_{\theta_k}} [\mathcal{L}_{\text{PPO}}(s_t, a_t, \theta_k, \theta)]. \quad (5.2)$$

We use the clipped version of PPO:

$$\mathcal{L}_{\text{PPO}}(s_t, a_t, \theta_k, \theta) = \operatorname{Clip}(A^{\pi_{\theta_k}}(s_t, a_t), \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, \delta), \quad (5.3)$$

where

$$\operatorname{Clip}(A, \alpha, \delta) = \begin{cases} \min(\alpha A, (1 + \delta)A), & A \geq 0 \\ \min(\alpha A, (1 - \delta)A), & A < 0. \end{cases}$$

A is the advantage function introduced above. Clipping makes the training updates more stable: it ensures that the gradient steps do not lead the policy outside of the region of parameter space where the samples collected are informative.

5.2.2 \mathcal{V}^{ex} : Fraction of Variance Explained

Now we recall the key notion introduced in the previous chapter, namely the fraction of variance explained, still denoted by \mathcal{V}^{ex} . As shown previously, a yet elementary use of this concept improves the performance of policy gradient algorithms. In the following, we present a completely different use of this quantity. In general terms, \mathcal{V}^{ex} gives some information about the goodness of fit of a model. In statistics, it is also denoted R^2 , which is a poor notation since this quantity can be negative for non-linear models (Kvålseth, 1985) (also, in the context of RL, R usually refers to the return). This quantity is also known as the coefficient of determination. In a regression setting, assume a model \hat{y} aims at predicting y from x , given a set of N couples (x_i, y_i) , \mathcal{V}^{ex} is defined by:

$$\mathcal{V}^{ex} \triangleq 1 - \frac{\text{MSE}}{\text{VAR}} \quad (5.4)$$

where MSE is the mean squared error of the model measured on these N couples (MSE = $\frac{1}{N} \sum_i (y_i - \hat{y}(x_i))^2$), and VAR is the variance of the observed targets y_i . $\mathcal{V}^{ex} \leq 1$ and:

- $\mathcal{V}^{ex} = 1$ means that the model perfectly predicts the data (MSE = 0).
- $\mathcal{V}^{ex} = 0$ means that the model performs as always predicting the average (MSE = VAR).
- $\mathcal{V}^{ex} < 0$ means that the model performs worse than merely predicting the mean value (MSE > VAR).

5.3 Related Work

SAUNA integrates three key ideas: (i) function approximation with a neural network combining or separating the actor and the critic with an on-policy setting, (ii) transition filtering reducing information/signal dilution in the gradient update while simplifying the underlying MDP, and (iii) using \mathcal{V}^{ex} as a measure of correlation between the value function and the returns to allow better sampling and more efficient learning. Below, we consider previous work building on some of these approaches.

Actor-critic algorithms essentially use the value function to alternate between policy evaluation and policy improvement (Barto, Sutton, and Anderson, 1983; Sutton, 1984). In order to update the actor, many methods adopt the on-policy formulation (Peters and Schaal, 2008b; Mnih, Badia, Mirza, et al., 2016; Schulman, Wolski, Dhariwal, et al., 2017). However, despite their important successes, these methods suffer from sample complexity.

In the literature, research has also been conducted in prioritization sampling. While (Schaul, Quan, Antonoglou, et al., 2015) makes the learning from experience replay more efficient by using the TD error as a measure of these priorities in an off-policy setting, the presented method directly selects the samples on-policy. (Schmidhuber, 1991) is related to SAUNA in that it calculates the expected improvement in prediction error, but with the objective to maximize the intrinsic reward through artificial curiosity. Instead, the method presented in this chapter estimates the expected fraction of variance explained and filters out some of the samples to improve the learning efficiency.

\mathcal{V}^{ex} has already been used in (Flet-Berliac and Preux, 2019b) as one of the auxiliary tasks for self-assessment of performance. Finally, motion control in physics-based environments is a long-standing and active research field. In particular, there are many prior work on continuous action spaces (Levine and Abbeel, 2014; Heess, Wayne, Silver, et al., 2015; Lillicrap, Hunt,

Pritzel, et al., 2016; Schulman, Moritz, Levine, et al., 2016) that demonstrate how locomotion behavior and other skilled movements can emerge as the outcome of optimization problems.

5.4 SAUNA: Dynamic Transition Filtering

We introduce a general method to filter transitions that contain useful information for policy gradient updates. In this chapter, we detail how to couple SAUNA with PPO, an on-policy gradient algorithm achieving state of the art performance. We refer to this combination as PPO+SAUNA. SAUNA can be coupled with other algorithms, especially with off-policy methods such as DQN: we leave this for future work. Below we detail how to adapt the notion of \mathcal{V}^{ex} to RL.

5.4.1 \mathcal{V}^{ex} applied to RL

The fraction of variance that the current estimate of the value function explains about the observed returns corresponds to the proportion of the variance in the dependent variable V that is predictable from s_t . We define \mathcal{V}_τ^{ex} as the fraction of variance explained for a trajectory τ :

$$\mathcal{V}_\tau^{ex} \triangleq 1 - \frac{\sum_{t \in \tau} (R_t - V(s_t))^2}{\sum_{t \in \tau} (R_t - \langle R \rangle_\tau)^2}, \quad (5.5)$$

where $R_t = \sum_{k \geq 0} \gamma^k r_{t+k}$, r_t is the immediate reward collected at timestep t , $V(s_t)$ is the current estimate of the value of state s_t , and $\langle R \rangle_\tau$ is the average of the R_t in trajectory τ . This definition can be extended from a trajectory τ to a batch \mathcal{B} of sampled transitions $\mathcal{V}_\mathcal{B}^{ex}$. In the RL context, the interpretation of $\mathcal{V}_\mathcal{B}^{ex}$ is:

- $\mathcal{V}_\mathcal{B}^{ex} = 1$: V perfectly explains the observed returns.
- $\mathcal{V}_\mathcal{B}^{ex} = 0$: V corresponds to a simple average prediction.
- $\mathcal{V}_\mathcal{B}^{ex} < 0$: V provides a worse prediction than the average of the returns.

The intuition is that $\mathcal{V}_\mathcal{B}^{ex}$ close to 1 corresponds to well-predicted returns. $\mathcal{V}_\mathcal{B}^{ex} < 0$ corresponds to a rather large prediction error of the value function, meaning that these samples are useful because the agent has something to learn from. On the other hand, $\mathcal{V}_\mathcal{B}^{ex}$ close to 0 means that the samples do not provide any valuable information to improve the value estimates. We will demonstrate that \mathcal{V}^{ex} is indeed a relevant indicator for assessing self-performance in RL.

5.4.2 Estimating \mathcal{V}^{ex}

While sampling the environment, SAUNA rejects transitions for which $V(s_t)$ is not correlated with returns that have followed s_t . Therefore, $\mathcal{V}_\mathcal{B}^{ex}$ should be estimated at each timestep and we

define $\mathcal{V}_\theta^{ex}(s_t)$ as the prediction of \mathcal{V}_B^{ex} with parameters θ at state $s_t \in \mathcal{B}$. In addition, for shared parameters configurations, an error term on the value estimation is added to the objective. The final objective function becomes:

$$\mathcal{L}_{\text{SAUNA}}(s_t, a_t, \theta_{old}, \theta) = \mathcal{L}_{\text{PPO}}(s_t, a_t, \theta_{old}, \theta) - \quad (5.6)$$

$$c_1 (V_\theta(s_t) - R_t)^2 - \quad (5.7)$$

$$c_2 (\mathcal{V}_\theta^{ex}(s_t) - \mathcal{V}_B^{ex})^2, \quad (5.8)$$

where c_1 and c_2 are the coefficients for the squared-error losses of respectively the value function and the fraction of variance explained function. Note that only the term (5.8) is specific to SAUNA. (5.6) and (5.7) come from PPO. When the network is not shared between the policy and the value function, SAUNA embeds \mathcal{V}_B^{ex} to the value function network using a single hidden layer. The rest of the network is unchanged, making SAUNA very easy to use without significantly increasing the complexity of the underlying algorithm.

5.4.3 SAUNA Algorithm

Algorithm 4 shows the pseudo-code of SAUNA when coupled with PPO. Overall, the resulting algorithm visits a set of trajectories along which it collects useful samples in the sense explained above, assessed with regards to \mathcal{V}^{ex} . The mechanism may be viewed as analogous to the method of dropout in deep learning (Srivastava, Hinton, Krizhevsky, et al., 2014; Freeman, Metz, and Ha, 2019) although here dropout happens in the state space of the underlying MDP and is directed by \mathcal{V}^{ex} . Once a batch \mathcal{B} of T such useful samples is collected, SAUNA performs the usual gradient update following the PPO template.

The gradient update concerns the three quantities estimated by SAUNA: the policy parameters θ , line 12, the value estimation parameters ϕ , line 13, and the \mathcal{V}^{ex} estimation parameters ψ , line 14. The *if* statement filters the useful samples: $\widetilde{\mathcal{V}}_{\psi_k}^{ex}(s_{0:t-1})$ denotes the median of $\mathcal{V}_{\psi_k}^{ex}$ between timesteps 0 and $t-1$, ε_0 is a Laplace estimator (set to 10^{-8}), and ρ is the filtering threshold. One may legitimately ask why not use directly $|\mathcal{V}_{\psi_k}^{ex}(s_t)|$ in the predicate. The rationale is practical: the ratio is a standardized measure as the agent learns, stabilized by the median, more robust to outliers than the mean. For better legibility, Algorithm 4 does not share parameters between the π , V and \mathcal{V}^{ex} networks. A version where these parameters would be partially shared is straightforward.

5.5 Experimental Study

We have forked the *stable-baselines* repository (Hill, Raffin, Ernestus, et al., 2018) and minimally modified the code to incorporate our method. The code for our method is released and open-

Use Variance in the Value Function estimates to filter information

Algorithm 4 SAUNA coupled with PPO.

```
1: Initialize policy parameters  $\theta_0$ , value function parameters  $\phi_0$  and  $\mathcal{V}^{ex}$  function parameters  $\psi_0$ 
2: for  $k = 0, 1, 2, \dots$  do
3:    $s_0 \leftarrow$  initial state
4:   batch  $\mathcal{B} \leftarrow \emptyset$ 
5:   while  $\text{size}(\mathcal{B}) \leq T$  do
6:      $a_t \sim \pi_{\theta_k}(s_t)$ 
7:     execute action  $a_t$  and observe  $r_t$  and  $s_{t+1}$ 
8:     if  $\frac{|\mathcal{V}_{\psi_k}^{ex}(s_t)|}{|\widehat{\mathcal{V}}_{\psi_k}^{ex}(s_{0:t-1})| + \epsilon_0} \geq \rho$  then
9:       add  $(s_t, a_t, r_t, V_{\phi_k}(s_t), s_{t+1}, \mathcal{V}_{\psi_k}^{ex}(s_t))$  to  $\mathcal{B}$ 
10:    if  $s_{t+1}$  is a final state then
11:       $s_{t+1} \leftarrow$  initial state
12:     $\theta_{k+1} \leftarrow \operatorname{argmax}_{\theta} \sum_{t \in \mathcal{B}} \mathcal{L}_{\text{PPO}}(s_t, a_t, \theta_k, \theta)$ 
13:     $\phi_{k+1} \leftarrow \operatorname{argmin}_{\phi} \sum_{t \in \mathcal{B}} (V_{\phi}(s_t) - R_t)^2$ 
14:     $\psi_{k+1} \leftarrow \operatorname{argmin}_{\psi} \sum_{t \in \mathcal{B}} (\mathcal{V}_{\psi}^{ex}(s_t) - \mathcal{V}_{\mathcal{B}}^{ex})^2$ 
```

source: github.com/yfletberliac/sauna. Unless otherwise stated, the policy network used for all tasks is a fully-connected multi-layer perceptron with 2 hidden layers of 64 units. Moreover, the architecture for the \mathcal{V}^{ex} function head is the same as for the value function head.

5.5.1 SAUNA in the Continuous Domain

To assess SAUNA, we compare PPO+SAUNA against its natural baseline PPO. We use six simulated robotic deterministic tasks from OpenAI Gym (Brockman, Cheung, Pettersson, et al., 2016) using *MuJoCo* (Todorov, Erez, and Tassa, 2012). The two hyperparameters required by our method ($\rho = 0.3$ from Equation 5.5 and $c_2 = 0.5$ from Equation 5.8) and all the others (identical to those in (Schulman, Wolski, Dhariwal, et al., 2017)) are exactly the same for all tasks.

We made the choice of not tuning the hyperparameters for each algorithm and for each task to have a tougher assessment of SAUNA: only SAUNA-specific hyperparameters ρ and c_2 have been tuned by grid-search. Hence, the performance we report for SAUNA is not necessarily the best that could be obtained with parameter tuning. The graphs reported in Figure 5.1 show that our method outperforms PPO on all considered continuous control tasks. We also report in Appendix B.1 a comparison of A2C, a synchronous variant of Mnih, Badia, Mirza, et al., 2016 and less strong version of PPO, with A2C+SAUNA.

We then experiment with the more difficult, high-dimensional continuous domain environment of *Roboschool* (Klimov and Schulman, 2017) with various neural network sizes. In

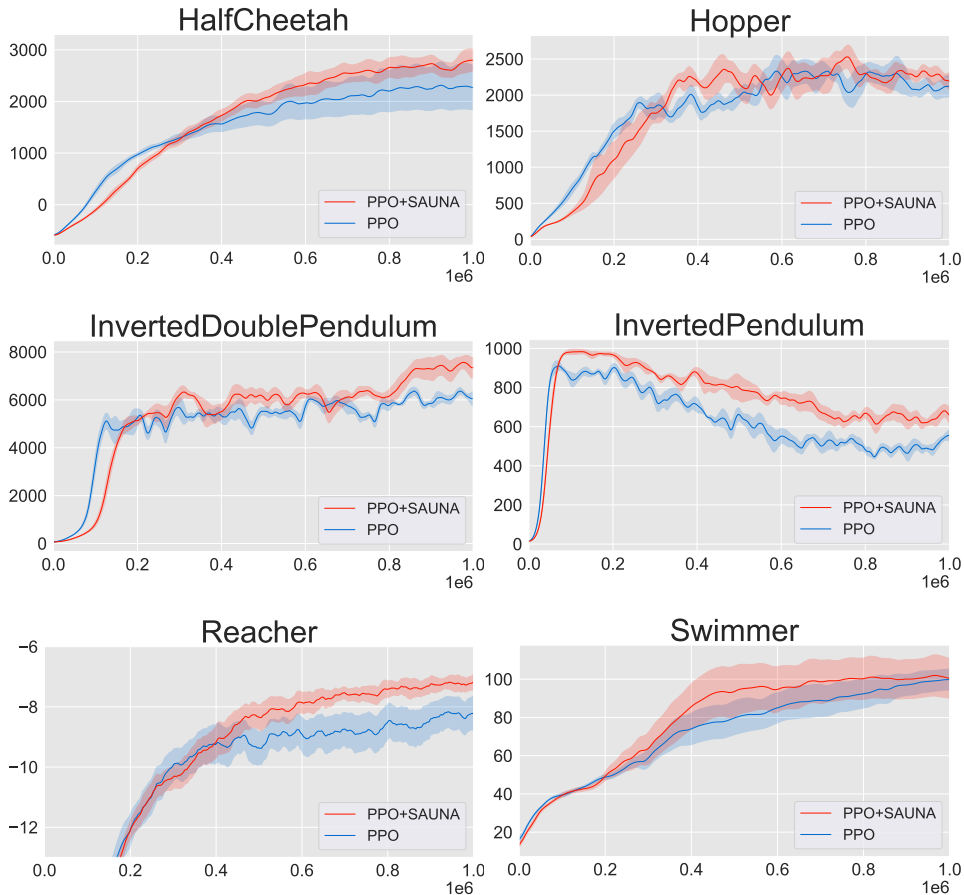


Figure 5.1 – Performance of PPO+SAUNA (red) relative to PPO (blue) on 6 *MuJoCo* environments averaged across 6 seeds. X-axis: number of environment steps. Y-axis: total undiscounted return. Shaded areas: standard deviation.

Figure 5.2a, the same fully-connected network as for the previous *MuJoCo* experiments (2 hidden layers each with 64 neurons) is used. In Figure 5.2b, the network is composed of a deeper and wider 3 hidden layers with 512, 256 and 128 neurons. We trained those agents with 32 parallel actors. In both experiments, PPO+SAUNA performs better and learns faster at the beginning. The gap closes with a larger network and our method does as well as PPO. As resources are limited in terms of the number of parameters and models become less complex, it seems natural that filtering samples according to their expected informational value helps to reduce noise in the gradient update and to speed up learning.

5.5.2 Learning with SAUNA

The Advantages of Filtering. We further study the impact of filtering out noisy samples by conducting additional experiments in predicting \mathcal{V}^{ex} while omitting the filtering step: the *if* statement (Line 8 of Algorithm 4) is removed and all transitions are kept in the batch \mathcal{B} . Indeed,

Use Variance in the Value Function estimates to filter information

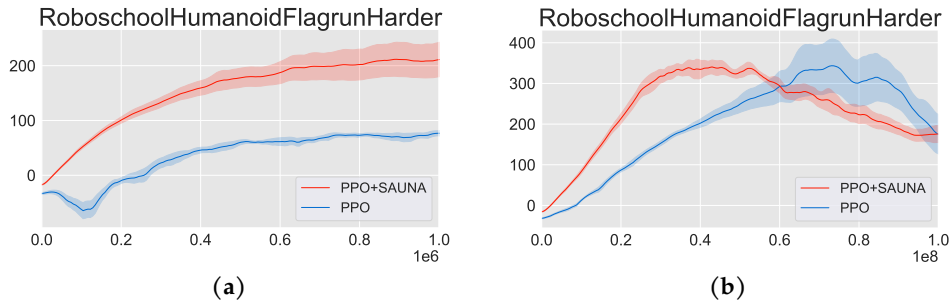


Figure 5.2 – Performance of PPO+SAUNA (red) relative to PPO (blue) on the *Roboschool* environment averaged across 6 seeds. X-axis: number of environment steps. Y-axis: total undiscounted return. Shaded areas: standard deviation.

SAUNA may improve the agent’s performance by simply training the shared network to optimize the \mathcal{V}^{ex} head as an auxiliary task. Figure 5.3 demonstrates the positive effects of filtering out the samples. In addition, we studied the number of filtered out samples per task and its evolution along the training. On average, SAUNA rejects 5-10% of samples at the beginning of training, 2-6% near the end.

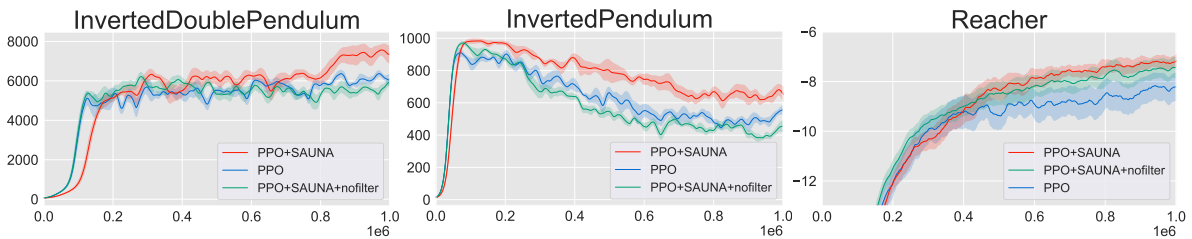


Figure 5.3 – Performance of PPO+SAUNA (red) relative to PPO (blue) and PPO with the prediction of \mathcal{V}^{ex} but without the filtering out of noisy samples (green) on 3 *MuJoCo* environments averaged across 6 seeds. X-axis: number of environment steps. Y-axis: total undiscounted return. Shaded areas: standard deviation.

The Impact of SAUNA on the mini-batch Gradients. Prior to the gradient update, SAUNA removes the useless transitions. By so doing, we hypothesized that information signals from samples with large \mathcal{V}^{ex} would be less diluted by filtering out samples. Figure 5.4 shows that SAUNA filtering leads to higher magnitude and relatively more stable gradients. As a result, policy updates make bigger steps, which ultimately seems to translate into better performance. It is questionable why performance is not negatively affected, since larger gradients could hinder learning. Experience shows that gradients contain more useful information: as the relevant signals are less diluted, the gradients are more qualitative and have been partially denoised.

HalfCheetah: Qualitative Study. In *HalfCheetah*, a well-known behavior (Lapan, 2018) is that, for multiple seeds, a PPO agent gets stuck in a local minimum in which the agent moves on

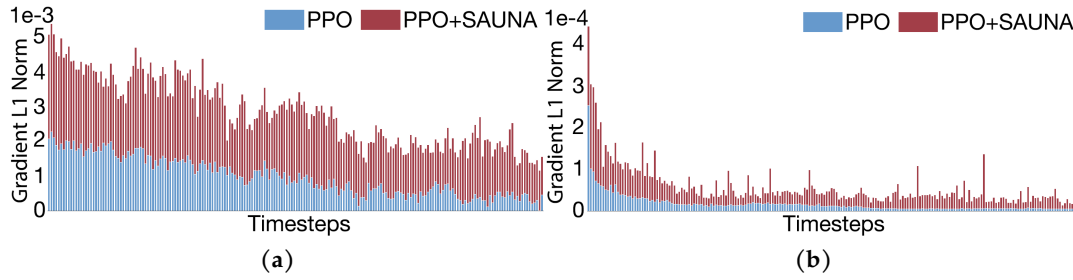


Figure 5.4 – Gradients L1-norm from the (a) first layer and (b) last layer of the shared parameters network for PPO and PPO coupled with SAUNA. Task: *HalfCheetah-v2*.

its back. However, we observed that SAUNA made it possible to leave from, or at least to avoid these local minima. This is illustrated in Figure 5.5a where we see still frames of two agents

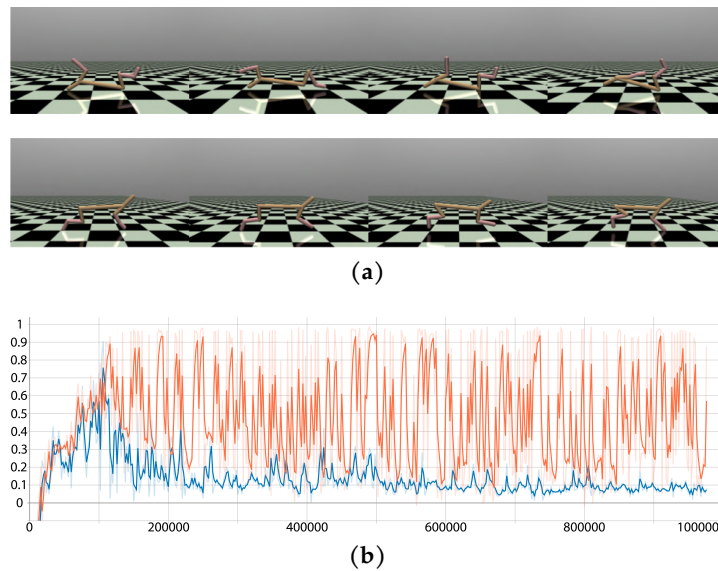


Figure 5.5 – (a) Example of PPO getting trapped in a local minimum (top row) while PPO+SAUNA reaches a better optimum (bottom row). (b) \mathcal{V}^{ex} score for PPO (blue) and PPO+SAUNA (orange).

trained with PPO+SAUNA for 10^6 timesteps on identically seeded environments. Their behavior is entirely different. Looking at \mathcal{V}^{ex} in Figure 5.5b, we can see that the graphs differ quite interestingly. The PPO agent seems to find very quickly a local minimum on its back while the blue agent’s \mathcal{V}^{ex} varies much more. This seems to allow the latter to explore more states than the former and finally to find a better optimum. Supported by the previous study, we can infer that agents trained with SAUNA are better able to explore interesting states while exploiting with confidence the value given to the states observed so far.

5.6 Discussion

Intuitively, during the policy update, the method presented in this chapter will only use qualitative samples that provide the agent with (i) reliable and exercised behavior (high \mathcal{V}^{ex}), and (ii) challenging states from the point of view of correctly predicting their value (low \mathcal{V}^{ex}). SAUNA algorithm keeps samples with high learning impact, rejecting other noisy samples from the gradient update.

5.6.1 Filtering Policy Gradient Updates and the Policy Gradient Theorem

Policy gradient algorithms are backed by the policy gradient theorem (Sutton, McAllester, Singh, et al., 2000). As long as the asymptotic stationary regime is not reached, it is not reasonable to assume the sampled states to be independent and identically distributed. Therefore, it seems intuitively better to ignore some of the samples for a certain period, to allow the most efficient use of information. In addition although this method introduces bias, we think it is partially corrected by the importance-sampling used in PPO. One can understand SAUNA as making gradient updates more robust through filtering, especially when the update is low and the noise can be dominant. Besides, filtering out disturbing samples reduces the bias in the state distribution.

5.6.2 Learning \mathcal{V}^{ex} and the Shared Network Parameters

SAUNA network predicts \mathcal{V}^{ex} in conjunction with the value function and the policy. Therefore, as its parameters are updated through gradient ascent, they converge to one of the objective function minima (hopefully, a global minimum). This parameter configuration integrates \mathcal{V}^{ex} , predicting how much the value function has fitted the observed samples, or informally speaking how well the value function is doing for state s_t . This new objective tends to lead the network to adjust predicting a quantity relevant for the task. Instead of using domain knowledge for the task, the method rather introduces problem knowledge by constraining the parameters directly.

5.6.3 Additional Experimental Results

In this section, we discuss additional experimental results which we think contribute interestingly to the study.

Mean of \mathcal{V}^{ex} . Although $\widetilde{\mathcal{V}^{ex}}$, the median of \mathcal{V}^{ex} , is more expensive to calculate, we observe that it gives better results than if we use its mean in the *if* statement of Algorithm 4. Using

the median helps (Kvålseth, 1985) because the distribution of \mathcal{V}^{ex} is not normal and includes outliers that will potentially produce misleading results.

Non-empirical \mathcal{V}^{ex} . We also experimented with using the empirical values of \mathcal{V}^{ex} in Line 8 of Algorithm 4 when calculating $\widetilde{\mathcal{V}^{ex}}$, instead of the predicted ones. This has yielded less positive results, and it is likely that this is due to the difference between the predicted and actual values at the beginning of learning, which has the effect of distorting the ratio in the *if* statement.

Adjusting state count. In order to stay in line with the policy gradient theorem (Sutton, McAllester, Singh, et al., 2000), we have worked to adjust the distribution of states d^π to what it truly is, since some states visited by the agent are not included in the batch. We adjusted it using the ratio between the number of states visited and the actual number of transitions used in the gradient update, but this did not improve the learning, and instead, we observed a decrease in performance.

Adjusted \mathcal{V}^{ex} . The definition of \mathcal{V}^{ex} is biased. An unbiased estimator does exist (known in statistics as the adjusted R^2). We performed the same set of experiments using such an adjusted \mathcal{V}^{ex} : it did not change the experimental performance significantly.

Random filtering. We experimented with dropping out at random, and before each gradient update, a number of samples corresponding to the same average number of samples that SAUNA drops. This resulted in a decrease in performance compared to PPO, as one can expect.

Atari 2600 domain. We evaluated SAUNA on the *Atari 2600* domain (Bellemare, Naddaf, Veness, et al., 2013) without observing any improvement in learning: some of the tasks were best performed by one method and others by the other.

Chapter conclusion

Policy gradient methods optimize the policy directly through gradient ascent. In Chapter 5, we have introduced a lightweight and agnostic method technically applicable to any policy gradient algorithm. We have performed some experiments to evaluate the usefulness of \mathcal{V}^{ex} as a measure to filter out samples that are perturbing the policy update: some transitions might be more or less surprising or redundant. The filtered non-informative or misinformative samples are ignored by SAUNA with a mechanism controlled by an estimation of the fraction of variance explained by the value function at each state. The relevant signals being less diluted, the mechanism improves sampling with a denoising effect on the gradients, ultimately leading to improved performance. We demonstrated the effectiveness of SAUNA when applied to PPO, a commonly used state of the art policy gradient method, on a set of benchmark high-dimensional environments. We also established that samples can be removed from the gradient update without hindering learning but, on the opposite, can improve it, and we further studied

Use Variance in the Value Function estimates to filter information

the positive impacts that such a modification in the sampling procedure has on learning, by showing that SAUNA's filtered sampling can keep the mini-batch gradient at a higher and more stable magnitude throughout training.

Chapter 6

Use Variance in the Value Function estimates as an objective function

*Choose that arrangement
which shall tend to reduce to a minimum
the time necessary
for completing the calculation.*

Ada Lovelace (1843).

In light of recent studies indicating that traditional actor-critic algorithms do not succeed in fitting the true value function, calling for the need to identify a better objective for the critic, we introduce a method to improve the learning of the critic in the actor-critic framework. In this chapter, we explore an alternative loss function for fitting critics in actor-critic RL algorithms. Instead of using the standard mean squared loss between critic predictions and value estimates, we propose to change the value function objectives to use a loss function based on the variance of the value function estimation errors. We first provide a meticulously articulated motivation and provide experiments to support the proposal which we evaluate on standard benchmarks for continuous control using popular RL algorithms¹.

Contents

6.1 Motivation	68
6.3 Preliminaries	70
6.4 AVEC: Actor with Variance Estimated Critic	71
6.5 Experimental Study	74

¹This chapter is based on an article published in the proceedings of the *9th International Conference on Learning Representations (ICLR)* (Flet-Berliac, Ouhamma, Maillard, et al., 2021). It is joint work with my colleague and friend Reda Ouhamma.

6.1 Motivation

Model-free deep reinforcement learning (RL) has been successfully used in a wide range of problem domains, ranging from teaching computers to control robots to playing sophisticated strategy games (Silver, Lever, Heess, et al., 2014; Lillicrap, Hunt, Pritzel, et al., 2016; Mnih, Badia, Mirza, et al., 2016; Schulman, Moritz, Levine, et al., 2016). State-of-the-art policy gradient algorithms currently combine ingenious learning schemes with neural networks as function approximators in the so-called actor-critic framework (Sutton, McAllester, Singh, et al., 2000; Schulman, Wolski, Dhariwal, et al., 2017; Haarnoja, Zhou, Abbeel, et al., 2018). While such methods demonstrate great performance in continuous control tasks, several discrepancies persist between what motivates the conceptual framework of these algorithms and what is implemented in practice to obtain maximum gains.

For instance, research aimed at improving the learning of value functions often restricts the class of function approximators through different assumptions, then propose a critic formulation that allows for a more stable policy gradient. However, new studies (Tucker, Bhupatiraju, Gu, et al., 2018; Ilyas, Engstrom, Santurkar, et al., 2020) indicate that state-of-the-art policy gradient methods (Schulman, Levine, Abbeel, et al., 2015; Schulman, Wolski, Dhariwal, et al., 2017) fail to fit the true value function and that recently proposed state-action-dependent baselines (Gu, Lillicrap, Sutskever, et al., 2016; Liu, Feng, Mao, et al., 2018; Wu, Rajeswaran, Duan, et al., 2018) do not reduce gradient variance more than state-dependent ones.

These findings leave the reader skeptical about actor-critic algorithms, suggesting that recent research tends to improve performance by introducing a bias rather than stabilizing the learning. Consequently, attempting to find a better baseline is questionable, as critics would typically fail to fit it (Ilyas, Engstrom, Santurkar, et al., 2020). In Tucker, Bhupatiraju, Gu, et al. (2018), the authors argue that “much larger gains could be achieved by instead improving the accuracy of the value function”. Following this line of thought, we are interested in ways to better approximate the value function. One approach addressing this issue is to put more focus on relative state-action values, an idea introduced in the literature on advantage reinforcement learning (Harmon and Baird III, n.d.) followed by works on dueling (Wang, Schaul, Hessel, et al., 2016) neural networks. More recent work (Lin and Zhou, 2020) also suggests that considering the *relative action values*, or more precisely the ranking of actions in a state leads to better policies. The main argument behind this intuition is that it suffices to identify the optimal actions to solve a task. We extend this principle of relative action value with respect to the mean value to cover both state and state-action-value functions with a new objective for the critic: minimizing the variance of residual errors.

In essence, this modified loss function puts more focus on the values of states (resp. state-actions) relative to their mean value rather than their absolute values, with the intuition that

solving a task corresponds to identifying the optimal action(s) rather than estimating the exact value of each state. In summary, this paper:

- Introduces Actor with Variance Estimated Critic (AVEC), an actor-critic method providing a new training objective for the critic based on the residual variance.
- Provides evidence for the improvement of the value function approximation as well as theoretical consistency of the modified gradient estimator.
- Demonstrates experimentally that AVEC, when coupled with state-of-the-art policy gradient algorithms, yields a significant performance boost on a set of challenging tasks, including environments with sparse rewards.
- Provides empirical evidence supporting a better fit of the true value function and a substantial stabilization of the gradient.

6.2 Related Work

Our approach builds on three lines of research, of which we give a quick overview: policy gradient algorithms, regularization in policy gradient methods, and exploration in RL.

Policy gradient methods use stochastic gradient ascent to compute a policy gradient estimator. This was originally formulated as the REINFORCE algorithm (Williams, 1992). Kakade and Langford (2002) later created conservative policy iteration and provided lower bounds for the minimum objective improvement. Peters, Mulling, and Altun (2010) replaced regularization by a trust region constraint to stabilize training. In addition, extensive research investigated methods to improve the stability of gradient updates, and although it is possible to obtain an unbiased estimate of the policy gradient from empirical trajectories, the corresponding variance can be extremely high. To improve stability, Weaver and Tao (2001) show that subtracting a baseline (Williams, 1992) from the value function in the policy gradient can be very beneficial in reducing variance without damaging the bias. However, in practice, these modifications on the actor-critic framework usually result in improved performance without a significant variance reduction (Tucker, Bhupatiraju, Gu, et al., 2018; Ilyas, Engstrom, Santurkar, et al., 2020). Currently, one of the most dominant on-policy methods are proximal policy optimization (PPO) (Schulman, Wolski, Dhariwal, et al., 2017) and trust region policy optimization (TRPO) (Schulman, Levine, Abbeel, et al., 2015), both of which require new samples to be collected for each gradient step. Another direction of research that overcomes this limitation is off-policy algorithms, which therefore benefit from all sample transitions; soft actor-critic (SAC) (Haarnoja, Zhou, Abbeel, et al., 2018) is one such approach achieving state-of-the-art performance.

Several works also investigate regularization effects on the policy gradient (Jaderberg, Mnih, Czarnecki, et al., 2017; Namkoong and Duchi, 2017; Flet-Berliac and Preux, 2019b; Kartal, Hernandez-Leal, and Taylor, 2019; Flet-Berliac and Preux, 2020); it is often used to shift the bias-variance trade-off towards reducing the variance while introducing a small bias. In RL, regularization is often used to encourage exploration and takes the form of an entropy term (Williams and Peng, 1991; Schulman, Wolski, Dhariwal, et al., 2017). Moreover, while regularization in machine learning generally consists in smoothing over the observation space, in the RL setting, Thodoroff, Durand, Pineau, et al. (2018) show that it is possible to smooth over the temporal dimension as well. Furthermore, Zhao, Niu, Xie, et al. (2016) analyze the effects of a regularization using the variance of the policy gradient (the idea is reminiscent of SVRG descent (Johnson and Zhang, 2013)) which proves to provide more consistent policy improvements at the expense of reduced performance. In contrast, as we will see later, AVEC does not change the policy network optimization procedure nor involves any additional computational cost.

Exploration has been studied under different angles in RL, one common strategy is ε -greedy, where the agent explores with probability ε by taking a random action. This method, just like entropy regularization, enforces uniform exploration and has achieved recent success in game playing environments (Mnih, Kavukcuoglu, Silver, et al., 2013; Hasselt, Guez, and Silver, 2016; Mnih, Badia, Mirza, et al., 2016). On the other hand, for most policy-based RL, exploration is a natural component of any algorithm following a stochastic policy, choosing sub-optimal actions with non-zero probability. Furthermore, policy gradient literature contains exploration methods based on uncertainty estimates of values (Kaelbling, 1993; Tokic, 2010), and algorithms which provide intrinsic exploration or curiosity bonus to encourage exploration (Schmidhuber, 2006; Bellemare, Srinivasan, Ostrovski, et al., 2016; Flet-Berliac, Ferret, Pietquin, et al., 2021).

While existing research may share some motivations with our method, no previous work in RL applies the variance of residual errors as an objective loss function. In the context of linear regression, Brown (1947) considers a median-unbiased estimator minimizing the risk with respect to the absolute-deviation loss function (Pham-Gia and Hung, 2001) (similar in spirit to the variance of residual errors), their motivation is nonetheless different to ours. Indeed, they seek to be robust to outliers whereas, when considering noiseless RL problems, one usually seeks to capture those (sometimes rare) signals corresponding to the rewards.

6.3 Preliminaries

6.3.1 Background and Notations

We consider an infinite-horizon Markov Decision Problem (MDP) with continuous states $s \in \mathcal{S}$, continuous actions $a \in \mathcal{A}$, transition distribution $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and reward function

$r_t \sim \mathcal{R}(s_t, a_t)$. Let $\pi_\theta(a|s)$ denote a stochastic policy with parameter θ , we restrict policies to being Gaussian distributions. In the following, π and π_θ denote the same object. The agent repeatedly interacts with the environment by sampling action $a_t \sim \pi(\cdot|s_t)$, receives reward r_t and transitions to a new state s_{t+1} . The objective is to maximize the expected sum of discounted rewards:

$$J(\pi) \triangleq \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (6.1)$$

where $\gamma \in [0, 1)$ is a discount factor (Puterman, 1994), and $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ is a trajectory sampled from the environment using policy π . We denote the value of a state s in the MDP framework while following a policy π by $V^\pi(s) \triangleq \mathbb{E}_{\tau \sim \pi} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s]$ and the value of a state-action pair of performing action a in state s and then following policy π by $Q^\pi(s, a) \triangleq \mathbb{E}_{\tau \sim \pi} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$. Finally, the advantage function which quantifies how an action a is better than the average action in state s is denoted $A^\pi(s, a) \triangleq Q^\pi(s, a) - V^\pi(s)$.

6.3.2 Critics in Deep Policy Gradients

In this section, we consider the case where the value functions are learned using function estimators and then used in an approximation of the gradient. Without loss of generality, we consider the algorithms that approximate the state-value function V . The analysis holds for algorithms that approximate the state-action-value function Q . Let $f_\phi : \mathcal{S} \rightarrow \mathbb{R}$ be an estimator of \hat{V}^π with ϕ its parameter. f_ϕ is traditionally learned through minimizing the mean squared error (MSE) against \hat{V}^π . At iteration k , the critic minimizes:

$$\mathcal{L}_{AC} = \mathbb{E}_s \left[(f_\phi(s) - \hat{V}^{\pi_{\theta_k}}(s))^2 \right], \quad (6.2)$$

where the states s are collected under policy π_{θ_k} , and $\hat{V}^{\pi_{\theta_k}}(s)$ is an empirical estimate of V (see Section 6.4.3 for details). Similarly, using $f_\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ instead, one can fit an empirical target \hat{Q}^π .

6.4 AVEC: Actor with Variance Estimated Critic

In this section, we introduce AVEC and discuss its correctness, motivations and implementation.

6.4.1 Defining an Alternative Critic

Recent work (Ilyas, Engstrom, Santurkar, et al., 2020) empirically demonstrates that while the value network succeeds in the supervised learning task of fitting \hat{V}^π (resp. \hat{Q}^π), it does not

Use Variance in the Value Function estimates as an objective function

fit V^π (resp. Q^π). We address this deficiency in the estimation of the critic by introducing an alternative value network loss. Following empirical evidence indicating that the problem is the approximation error and not the estimator *per se*, AVEC adopts a loss that can provide a better approximation error, and yields better estimators of the value function (as will be shown in Section 6.5.3). At update k :

$$\mathcal{L}_{\text{AVEC}} = \mathbb{E}_s \left[\left((f_\phi(s) - \hat{V}^{\pi_{\theta_k}}(s)) - \mathbb{E}_s [f_\phi(s) - \hat{V}^{\pi_{\theta_k}}(s)] \right)^2 \right], \quad (6.3)$$

with states s collected using π_{θ_k} . Note that the gradient flows in f_ϕ twice using Equation 6.3. Then, we define our bias-corrected estimator: $g_\phi : \mathcal{S} \rightarrow \mathbb{R}$ such that $g_\phi(s) = f_\phi(s) + \mathbb{E}_s[\hat{V}^{\pi_{\theta_k}}(s) - f_\phi(s)]$. Analogously to Equation 6.3, we define an alternative critic for the estimation of Q^π by replacing \hat{V}^π by \hat{Q}^π and $f_\phi(s)$ by $f_\phi(s, a)$.

Lemma 6.1 (AVEC Policy Gradient). *If $f_\phi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ satisfies the parameterization assumption (Sutton, McAllester, Singh, et al., 2000) then g_ϕ provides an unbiased policy gradient:*

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{(s,a) \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(s, a)) g_\phi(s, a)].$$

Proof. See Appendix C.1. This result also holds for the estimation of V^{π_θ} with $f_\phi : \mathcal{S} \rightarrow \mathbb{R}$.

6.4.2 Building Motivation

Here, we present the intuition behind using AVEC for actor-critic algorithms. Tucker, Bhupatiraju, Gu, et al. (2018) and Ilyas, Engstrom, Santurkar, et al. (2020) indicate that the approximation error $\|\hat{V}^\pi - V^\pi\|$ is problematic, suggesting that the variance of the empirical targets $\hat{V}^\pi(s_t)$ is high. Using $\mathcal{L}_{\text{AVEC}}$, our approach reduces the variance term of the MSE (or distance to V^π) but mechanistically also increases the bias. Our intuition is that since the bias is already quite substantial (Ilyas, Engstrom, Santurkar, et al., 2020), it may be possible to reduce the variance enough so that even though the bias increases, the total MSE reduces.

State-value function estimation. In this case, optimizing the critic with $\mathcal{L}_{\text{AVEC}}$ can be interpreted as fitting $\hat{V}'^\pi(s) = \hat{V}^\pi(s) - \mathbb{E}_{s'}[\hat{V}^\pi(s')]$ using the MSE. We show that the targets \hat{V}'^π are better estimations of $V'^\pi(s) = V^\pi(s) - \mathbb{E}_{s'}[V^\pi(s')]$ than \hat{V}^π are of V^π . To illustrate this, consider T independent random variables $(X_i)_{i \in \{1, \dots, T\}}$. We denote $X'_i = X_i - \frac{1}{T} \sum_{j=1}^T X_j$ and $\mathbb{V}(X)$ the variance of X . Then, $\mathbb{V}(X'_i) = \mathbb{V}(X_i) - \frac{2}{T} \mathbb{V}(X_i) + \frac{1}{T^2} \sum_{j=1}^T \mathbb{V}(X_j)$ and $\mathbb{V}(X'_i) < \mathbb{V}(X_i)$ as long as $\forall i \frac{1}{T} \sum_{j=1}^T \mathbb{V}(X_j) < 2\mathbb{V}(X_i)$, or more generally when state-values are not strongly nega-

tively correlated² and not very discordant. This entails that \hat{V}^π has a more compact span, and is consequently easier to fit. This analysis shows that the variance term of the MSE is reduced compared to traditional actor-critic algorithms, but does not guarantee it counterbalances the bias increase. Nevertheless, in practice, the bias is so high that the difference due to learning with AVEC is only marginal and the total MSE decreases. We empirically demonstrate this claim in Section 6.5.3.

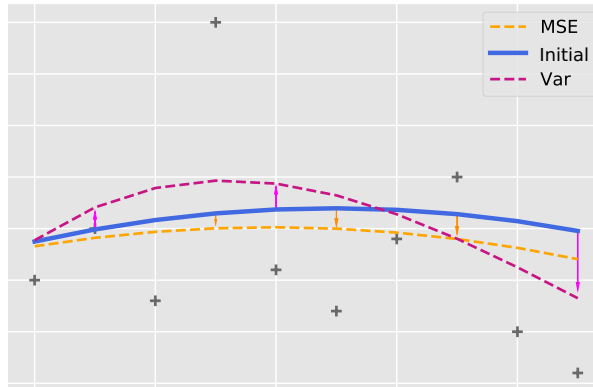


Figure 6.1 – Comparison of simple models derived when $\mathcal{L}_{\text{AVEC}}$ is used instead of the MSE.

State-action-value function estimation. In this case, Equation 6.3 translates into replacing $\hat{V}^\pi(s)$ by $\hat{Q}^\pi(s, a)$ and $f_\phi(s)$ by $f_\phi(s, a)$ and the rationale for optimizing the residual variance of the value function instead of the full MSE becomes more straightforward: the practical use of the Q-function is to disentangle the relative values of actions for each state (Sutton, McAllester, Singh, et al., 2000). AVEC’s effect on relative values is illustrated in a didactic regression with one variable example in Figure 6.1 where grey markers are observations and the blue line is our current estimation. Minimizing the MSE, the line is expected to move towards the orange one in order to reduce errors uniformly. Minimizing the residual variance, it is expected to move near the red one. In fact, $\mathcal{L}_{\text{AVEC}}$ tends to further penalize observations that are far away from the mean, implying that AVEC allows a better recovery of the “shape” of the target near extrema. In particular, we see in the figure that the maximum and minimum observation values are quickly identified. Would the approximators be linear and the target state-values independent, the two losses become equivalent since ordinary least squares would provide minimum-variance mean-unbiased estimation.

It should be noted that, as in all the works related to ours, we consider noiseless tasks, *i.e.* the transition matrix is deterministic. As such, there are no outliers and extreme state-action values correspond to learning signals. In this context, high estimation errors indicate where (in the state or action-state space) the training of the value function should be improved.

²Greensmith, Bartlett, and Baxter (2004) analyze the dependent case: in general, weakly dependent variables tend to concentrate more than independent ones.

6.4.3 Implementation

We apply this new formulation to three of the most dominant deep policy gradient methods to study whether it results in a better estimation of the value function. A better estimation of the value function implies better policy improvements. We now describe how AVEC incorporates its residual variance objective into the critics of PPO (Schulman, Wolski, Dhariwal, et al., 2017), TRPO (Schulman, Levine, Abbeel, et al., 2015) and SAC (Haarnoja, Zhou, Abbeel, et al., 2018). Let \mathcal{B} be a batch of transitions. In PPO and TRPO, AVEC modifies the learning of V_ϕ (line 12 of Algorithm 5) using:

$$\mathcal{L}_{\text{AVEC}}^1(\phi) = \mathbb{E}_{s \sim \mathcal{B}} \left[(f_\phi(s) - \hat{V}^\pi(s)) - \mathbb{E}_{s \sim \mathcal{B}} [f_\phi(s) - \hat{V}^\pi(s)] \right]^2,$$

then $V_\phi = f_\phi(s) + \mathbb{E}_{s \sim \mathcal{B}} [\hat{V}^\pi(s) - f_\phi(s)]$, where $\hat{V}^\pi(s_t) = f_{\phi_{\text{old}}}(s_t) + A_t$ such that $f_{\phi_{\text{old}}}(s_t)$ are the estimates given by the last value function and A_t is the advantage of the policy, *i.e.* the returns minus the expected values (A_t is often estimated using generalized advantage estimation (Schulman, Moritz, Levine, et al., 2016)). In SAC, AVEC modifies the objective function of $(Q_{\phi_i})_{i=1,2}$ (line 13 of Algorithm 8 in Appendix C.4.2) using:

$$\mathcal{L}_{\text{AVEC}}^2(\phi_i) = \mathbb{E}_{(s,a) \sim \mathcal{B}} \left[(f_{\phi_i}(s,a) - \hat{Q}^\pi(s,a)) - \mathbb{E}_{(s,a) \sim \mathcal{B}} [f_{\phi_i}(s,a) - \hat{Q}^\pi(s,a)] \right]^2,$$

then $Q_{\phi_i} = f_{\phi_i}(s,a) + \mathbb{E}_{(s,a) \sim \mathcal{B}} [\hat{Q}^\pi(s,a) - f_{\phi_i}(s,a)]$, where $\hat{Q}^\pi(s,a)$ is estimated using temporal difference (see Haarnoja, Zhou, Abbeel, et al. (2018)): $\hat{Q}^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \pi} [V_{\bar{\psi}}(s_{t+1})]$ with $\bar{\psi}$ the value function parameter (see Algorithm 8). The reader may have noticed that $\mathcal{L}_{\text{AVEC}}^1$ and $\mathcal{L}_{\text{AVEC}}^2$ slightly differ from Equation 6.3. The residual variance of the value function ($\mathcal{L}_{\text{AVEC}}$) is not tractable since *a priori* state-values are dependent and their joint law is unknown. Consequently, in practice, we use the empirical variance proxy assuming independence (*cf.* Appendix C.4.1). Greensmith, Bartlett, and Baxter (2004) provide some support for this approximation by showing that weakly dependent variables tend to concentrate more than independent ones. Finally, notice that AVEC does not modify any other part of the considered algorithms whatsoever, which makes its implementation straightforward and keeps the same computational complexity.

6.5 Experimental Study

In this section, we conduct experiments along four orthogonal directions: (i) we validate the superiority of AVEC compared to the traditional actor-critic training, (ii) we evaluate AVEC in environments with sparse rewards, (iii) we clarify the practical implications of using AVEC by

examining the bias in both the empirical and true value function estimations as well as the variance in the empirical gradient, and (iv) we provide an ablation analysis and study the bias-variance trade-off in the critic by considering two continuous control tasks.

We point out that a comparison to variance-reduction methods is not considered in this chapter: Tucker, Bhupatiraju, Gu, et al. (2018) demonstrated that their implementations diverge from the unbiased methods presented in the respective papers and unveiled that not only do they fail to reduce the variance of the gradient, but that their unbiased versions do not improve performance either. Note that in all experiments we choose the hyperparameters providing the best performance for the considered methods which can only penalize AVEC (cf. Appendix C.3). The code for our method is released and open-source: github.com/yfletberliac/avec. In all the figures hereafter (except Figure 6.3c and 6.3d), lines are average performances and shaded areas represent one standard deviation.

Algorithm 5 AVEC coupled with PPO or TRPO. J^{ALGO} denotes the policy loss of either algorithm (described in Schulman, Levine, Abbeel, et al. (2015) and Schulman, Wolski, Dhariwal, et al. (2017)).

```

1: Input parameters:  $\lambda_\pi \geq 0, \lambda_V \geq 0$ 
2: Initialize policy parameter  $\theta$  and value function parameter  $\phi$ 
3: for each update step do
4:   batch  $\mathcal{B} \leftarrow \emptyset$ 
5:   for each environment step do
6:      $a_t \sim \pi_\theta(s_t)$ 
7:      $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ 
8:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ 
9:   for each gradient step do
10:     $\theta \leftarrow \theta - \lambda_\pi \hat{\nabla}_\theta J^{\text{ALGO}}(\pi_\theta)$ 
11:     $\phi \leftarrow \phi - \lambda_V \hat{\nabla}_\phi \mathcal{L}_{\text{AVEC}}^1(\phi)$ 

```

6.5.1 Continuous Control

For ease of comparison with other methods, we evaluate AVEC on the MuJoCo (Todorov, Erez, and Tassa, 2012) and the PyBullet (Coumans and Bai, 2016) continuous control benchmarks (see Appendix C.5 for details) using OpenAI Gym (Brockman, Cheung, Pettersson, et al., 2016). Note that the PyBullet versions of the locomotion tasks are harder than the MuJoCo equivalents³. We choose a representative set of tasks for the experimental evaluation; their action and observation space dimensions are reported in Appendix C.5. We assess the benefits of AVEC when coupled with the most prominent policy gradient algorithms, currently state-of-

³Bullet Physics SDK [GitHub Issue](#).

Use Variance in the Value Function estimates as an objective function

Table 6.1 – Average total reward of the last 100 episodes over 6 runs of 10^6 timesteps. Comparative evaluation of AVEC with SAC and PPO. \pm corresponds to a single standard deviation over trials and (.%) is the change in performance due to AVEC.

Task	SAC	AVEC-SAC	PPO	AVEC-PPO
Ant	3084	3650 \pm 127 (+18%)	972	1202 \pm 148 (+24%)
AntBullet	1193	2252 \pm 82 (+89%)	1174	2216 \pm 99 (+89%)
HalfCheetah	10028	11018 \pm 102 (+10%)	1068	1403 \pm 37 (+31%)
HalfCheetahBullet	1255	1331 \pm 184 (+6%)	1329	2223 \pm 62 (+67%)
Humanoid	4084	4472 \pm 424 (+10%)	391	415 \pm 4.6 (+6%)
Reacher	-6.0	-5.0 \pm 0.1 (+20%)	-7.4	-5.9 \pm 0.3 (+25%)
Walker2d	3452	4334 \pm 128 (+26%)	2193	2923 \pm 151 (+33%)

the-art methods: PPO (Schulman, Wolski, Dhariwal, et al., 2017) and TRPO (Schulman, Levine, Abbeel, et al., 2015), both on-policy methods, and SAC (Haarnoja, Zhou, Abbeel, et al., 2018), an off-policy maximum entropy deep RL algorithm. We provide the list of hyperparameters and further implementation details in Appendix C.4.1 and C.3.

Table 6.1 reports the results while Figure 6.2 and C.2 show the total average return for SAC and PPO. TRPO results are provided in Appendix C.2.1 for readability. When coupled with SAC and PPO, AVEC brings very significant improvement (on average +26% for SAC and +39% for PPO) in the performance of the policy gradient algorithms, improvement which is consistent across tasks. As for TRPO, while the improvement in performance is less striking, AVEC still manages to be more efficient in terms of sampling in all tasks. Overall, AVEC improves TRPO, PPO and SAC in terms of performance and efficiency. This does not imply that our method would also improve other policy gradient methods that use the traditional actor-critic framework, but since we evaluate our method coupled with three of the best performing on- and off-policy algorithms, we believe that these experiments are sufficient to prove the relevance of AVEC. Furthermore, in our experiments we do not seek the best hyperparameters for the AVEC variants, we simply adopt the parameters allowing us to optimally reproduce the baselines. Alternatively, if one seeks to evaluate AVEC independently of a considered baseline, further hyperparameter tuning should produce better results. Notice that since no additional calculations are needed in AVEC’s implementation, computational complexity remains unchanged.

6.5.2 Sparse Reward Signals

Domains with sparse rewards are challenging to solve with uniform exploration as agents receive no feedback on their actions before starting to collect rewards. In such conditions AVEC performs better, suggesting that the *shape* of the value function is better approximated, encouraging exploration.

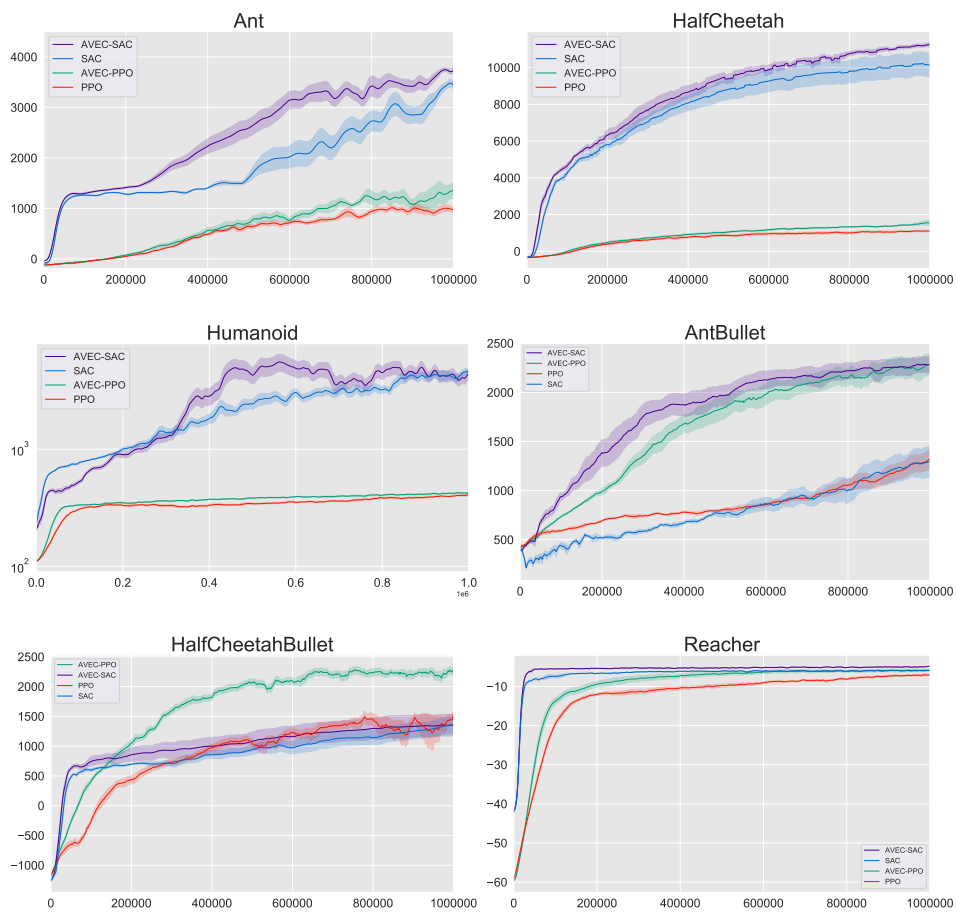


Figure 6.2 – Comparative evaluation (6 seeds) of AVEC with SAC and PPO on PyBullet (“TaskBullet”) and MuJoCo (“Task”) tasks. X-axis: number of timesteps. Y-axis: average total reward.

Use Variance in the Value Function estimates as an objective function

The relative value estimate of an unseen state is more accurate: in Section 6.4.2, AVEC identifies extreme state-values (*e.g.*, non-zero rewards in tasks with sparse rewards) faster. In Figure 6.3a and 6.3b, we report the performance of AVEC in the Acrobot and MountainCar environments: both have sparse rewards. AVEC enhances TRPO and PPO in both experiments. When PPO and AVEC-PPO both reach the best possible performance, AVEC-PPO exhibits better sample efficiency. Figure 6.3c and 6.3d illustrate how the agent improves its exploration strategy in MountainCar: while the PPO agent remains stuck at the bottom of the hill (red), the graph suggest that AVEC-PPO learns the difficult locomotion principles in the absence of rewards and visits a much larger part of the state space (green).

This improved performance in sparse environments can be explained by the fact that AVEC is able to pick up on experienced positive reward more easily. Moreover, the reconstructed shape of the value function is more accurate around such rewarding states, which pushes the agent to explore further around experienced states with high values.

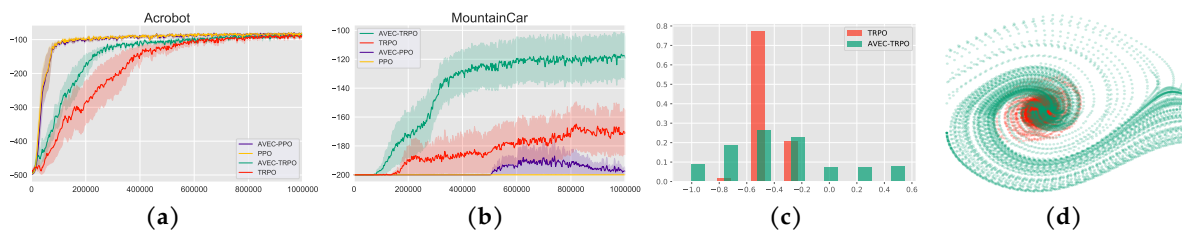


Figure 6.3 – (a,b): Comparative evaluation (6 seeds) of AVEC in sparse reward tasks. X-axis: number of timesteps. Y-axis: average total reward. (c,d): Respectively state visitation frequency and phase portrait of visited states of AVEC-TRPO (green) and TRPO (red) in MountainCar.

6.5.3 Analysis of the Variance Estimated Critic

In order to further validate AVEC, we evaluate the performance of the value network in more details: we examine (i) the estimation error (distance to the empirical target), (ii) the approximation error (distance to the true target), and (iii) the empirical variance of the gradient. (i,ii) should be put into perspective with the conclusions of Ilyas, Engstrom, Santurkar, et al. (2020) where it is found that the critic only fits the empirical value function but not the true one. (iii) should be placed in light of Tucker, Bhupatiraju, Gu, et al. (2018) highlighting a failure of recently proposed state-action-dependent baselines to reduce the variance.

Learning the Empirical Target. In Figure 6.4, we report the quality of fit (MSE) of the empirical target \hat{V}^π in the methods PPO and AVEC-PPO in the AntBullet and HalfCheetahBullet tasks. We observe that PPO better fits the empirical target than when equipped with AVEC, which is to be expected since vanilla PPO optimizes the MSE directly. This result put aside

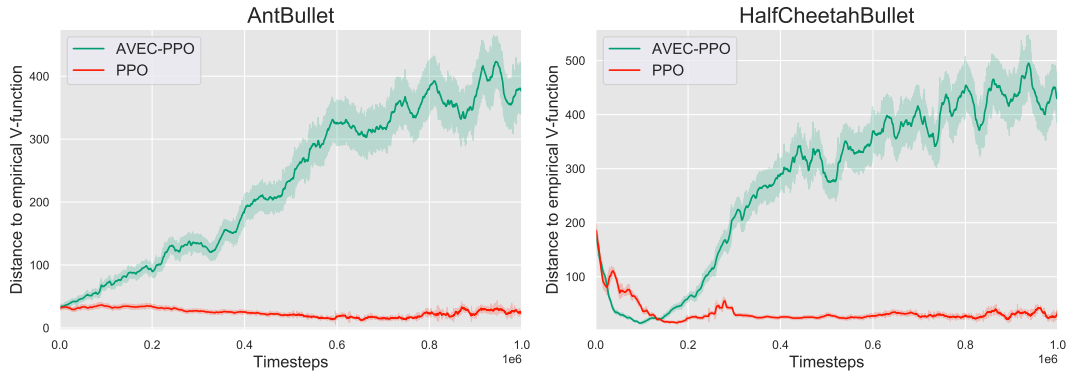


Figure 6.4 – L_2 distance to \hat{V}^π .

the remarkable improvement in the performance of AVEC-PPO (Figure 6.2) suggests that AVEC might be a better estimator of the true value function. We examine this claim below because if true, it would indicate that it is indeed possible to simultaneously improve the performance of the agents and the stability of the method.

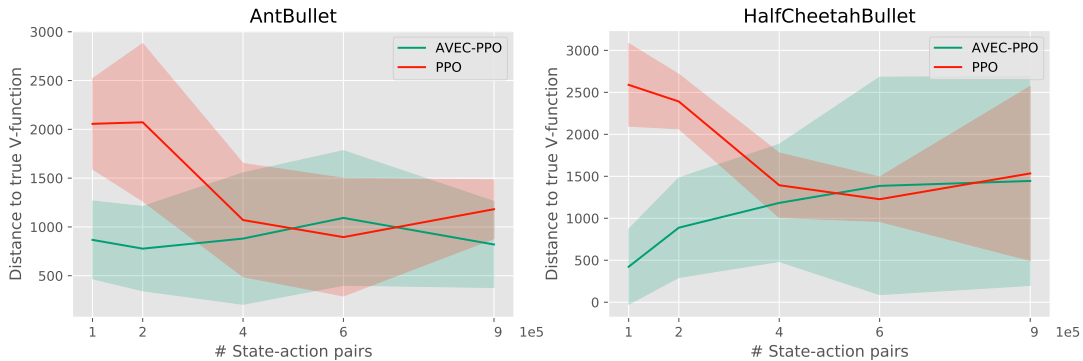


Figure 6.5 – L_2 distance to V^π . X-axis: we run PPO and AVEC-PPO and $\forall t \in \{1, 2, 4, 6, 9\} \cdot 10^5$ we stop training, use the current policy to collect $3 \cdot 10^5$ transitions and estimate V^π .

Learning the True Target. A fundamental premise of policy gradient methods is that optimizing the objective based on an empirical estimation of the value function leads to a better policy. Which is why we investigate the quality of fit of the true target. To approximate the true value function, we fit the returns sampled from the current policy using a large number of transitions ($3 \cdot 10^5$). Figure 6.5 shows that g_ϕ is far closer to the true value function half of the time (horizon is 10^6) than the estimator obtained with MSE, then as close to it. Comparing Figure 6.5 with Figure 6.4, we see that the distance to the true target is close to the estimation error for AVEC-PPO, while for PPO, it is at least two orders of magnitude higher at all times. We further investigate these results in Figure C.3 in Appendix C.2.3 where we study the variation of the squared bias and variance components of the MSE to the true target ($\text{MSE} = \text{Var} + \text{Bias}^2$). We find, as expected, that using AVEC reduces the variance term significantly while slightly increasing the

bias term, which Figure 6.5 confirms is negligible since the total MSE is substantially reduced ($\|g_\phi(\text{AVEC}) - V^\pi\|_2 \leq \|V_\phi(\text{PPO}) - V^\pi\|_2$) where $V_\phi(\text{PPO})$ is the value function estimator in PPO. For completeness, we also analyze the distance to the true target for the Q-function estimator in SAC and AVEC-SAC in AntBullet and HalfCheetahBullet in Appendix C.2.4, with similar results and interpretation. We conclude that AVEC improves the value function approximation and we expect that the gradient is more stable.

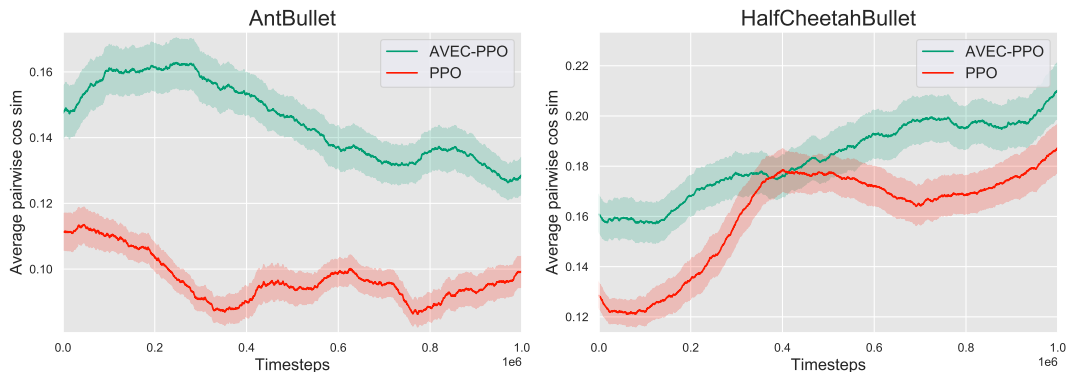


Figure 6.6 – Average gradient cosine-similarity.

Empirical Variance Reduction. We choose to study the gradient variance using the average pairwise cosine similarity metric as it allows a comparison with Ilyas, Engstrom, Santurkar, et al. (2020), with which we share the same experimental setup and scales. Figure 6.6 shows that AVEC yields a higher average (10 batches per iteration) pairwise cosine similarity, which means closer batch-estimates of the gradient and, in turn, indicates smaller gradient variance. Further analysis with additional tasks is included in Appendix C.2.5. The variance reduction effect observed in several environments suggests that AVEC is the first method since the introduction of the value function baseline to further reduce the variance of the gradient and improve performance.

6.5.4 Ablation Study

In this section, we examine how changing the relative importance of the bias and the residual variance in the loss of the value network affects learning. For this study, we choose difficult tasks of PyBullet and use PPO because it is more efficient than TRPO and requires less computations than SAC. For an estimator \hat{y}_n of $(y_i)_{i \in \{1, \dots, n\}}$, we write $\text{Bias} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)$ and $\text{Var} = \frac{1}{n-1} \sum_{i=1}^n (\hat{y}_i - y_i - \text{Bias})^2$. Consequently: $\text{MSE} = \text{Var} + \text{Bias}^2$. We denote $\mathcal{L}_\alpha = \text{Var} + \alpha \text{Bias}^2$, with $\alpha \in \mathbb{R}$. In Figure 6.7, *Bias- α* means that we use \mathcal{L}_α and *Var- α* means that we use $\mathcal{L}_{\frac{1}{\alpha}}$. We observe that while no consistent order on the choices of α is identified, AVEC seems to outperform all other weightings. Note that, for readability purposes, the graphs have been split

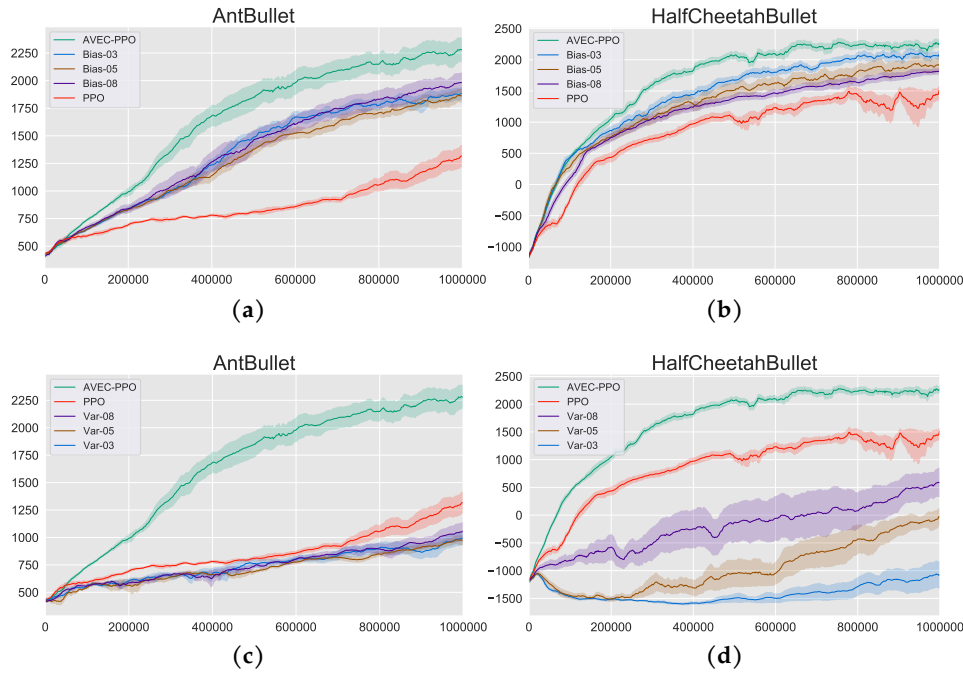


Figure 6.7 – Sensitivity analysis (6 seeds) of AVEC-PPO with respect to (a,b): the bias; (c,d): the variance. X-axis: number of timesteps. Y-axis: average total reward.

and the curves of AVEC-PPO and PPO are the same in Figure 6.7a and 6.7c, and in Figure 6.7b and 6.7d. A more extensive hyperparameter study with more α values might provide even higher performances, nevertheless we believe that the stability of an algorithm is crucial for a reliable performance. As such, the tuning of hyperparameters to achieve good results should remain mild.

Chapter conclusion

In this work, we introduce a new training objective for the critic in actor-critic algorithms to better approximate the true value function. In addition to being well-motivated by recent studies on the behaviour of deep policy gradient algorithms, we demonstrate that this modification is both theoretically sound and intuitively supported by the need to improve the approximation error of the critic. The application of **Actor with Variance Estimated Critic (AVEC)** to state-of-the-art policy gradient methods produces considerable gains in performance (on average +26% for SAC and +39% for PPO) over the standard actor-critic training, without any additional hyperparameter tuning.

First, for SAC-like algorithms where the critic learns a state-action-value function, our results strongly suggest that state-actions with extreme values are identified more quickly. Second, for

PPO-like methods where the critic learns the state-values, we show that the variance of the gradient is reduced and empirically demonstrate that this is due to a better approximation of the state-values. In sparse reward environments, the theoretical intuition behind a variance estimated critic is more explicit and is also supported by empirical evidence. In addition to corroborating the results in Ilyas, Engstrom, Santurkar, et al. (2020) proving that the value estimator fails to fit V^π , we propose a method that succeeds in improving both the sample complexity and the stability of prominent actor-critic algorithms. Furthermore, AVEC benefits from its simplicity of implementation since no further assumptions are required (such as horizon awareness Tucker, Bhupatiraju, Gu, et al. (2018) to remedy the deficiency of existing variance-reduction methods) and the modification of current algorithms represents only a few lines of code.

In this chapter, we have demonstrated the benefits of a more thorough analysis of the critic objective in policy gradient methods. Despite our strongly favourable results, we do not claim that the residual variance is the optimal loss for the state-value or the state-action-value functions, and we note that the design of comparably superior estimators for critics in deep policy gradient methods merits further study. In future work, further analysis of the bias-variance trade-off and extension of the results to stochastic environments is anticipated; we consider the problem of noise separation in the latter, as this is the first obstacle to accessing the variance and distinguishing extreme values from outliers.

Part conclusion

In the following, we review the problems and questions opened up in Chapter 1 that we have specified to address. We have proposed an alternative statistics of self-performance assessment and accurate expectation as additional performance metrics to evaluate an agent’s learning and improve its sample efficiency and performance (**Chapter 4**). We have developed the idea that an RL agent learns more effectively from some transitions than others by filtering transitions using the variance explained in the value function estimates (**Chapter 5**). In light of recent studies indicating that popular variance reduction methods do not actually reduce the gradient variance and fail to learn value function estimates, we propose an alternative, more efficient and more robust objective function for estimating the critic (**Chapter 6**). In addition, the proposed method of using residual variance to estimate the value function is more sensitive to extreme values and captures the signals corresponding to the rewards more efficiently (**Chapter 6**). Furthermore, for the writing of this thesis, we experimented further by combining AVEC and SAUNA with PPO as the base algorithm. Unfortunately, the improvements are not as clear as the individual effects of each method, and we suspect that filtering transitions using SAUNA negatively affects the learning of a critic with residual variance as its objective function.

Part III

Diversity in the Policy candidates

Chapter 7

Hard-Exploration and Real-World Features Problems

*Cujus rei demonstrationem mirabilem sane detexi.
Hanc marginis exiguitas non caperet.*

—

*I have discovered a quite remarkable demonstration of this.
But my margin is too narrow to contain it.*

Grand théorème de Fermat, Pierre de Fermat (1637).

In Part II, we have addressed the problem of learning in high-dimensional continuous state space with continuous actions by leveraging the variance in the value function estimates (variance explained or residual variance). Part III examines the problem of learning in environments with more real-world features such as safety constraints or where efficient exploration is a bottleneck. In this part, we center our attention on variance in the policy candidates instead of the variance in the value function estimates, by introducing a third protagonist to the actor-critic framework which will serve as an adversarial prior from which the policy distribution should be repulsed. Chapter 8 develops a method where the adversarial prior is a mixture of previous policies and Chapter 9 employs the formulation of safe RL and learns a risk-seeking adversary. In this chapter, we introduce the problem of learning in environments with hard-exploration characteristics and safety constraints and motivate the use of an actor-critic framework augmented with an adversary.

Contents

7.1	Hard-Exploration Problems	88
7.2	The Real-World RL Challenge	90
7.3	Adversarial Prior in the Actor-Critic Framework	91

7.1 Hard-Exploration Problems

Chapter 8 is devoted to tasks where efficient exploration is a bottleneck. The default approach to stochastic exploration which traditional actor-critic methods would propose will exhibit random walk behavior in such environments, but the likelihood of visiting states with rewards will be rapidly decreasing as the agent moves away from the starting point. In Chapter 8, we propose a new approach to make learning progress in these tasks, by reaching a new level of performance, with a more sample-efficient method.

7.1.1 MiniGrid

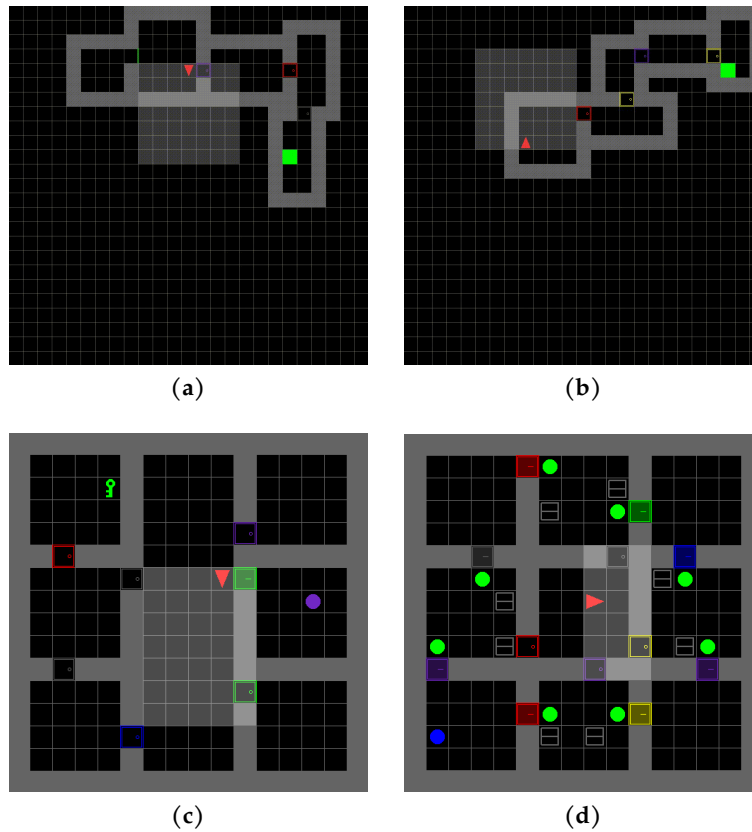


Figure 7.1 – Frames from (a,b) MultiRoom. (c) KeyCorridor. (d) ObstructedMaze.

The MiniGrid environments (Chevalier-Boisvert, Willems, and Pal, 2018) are a set of challenging sparse-reward gridworlds. There are several different MiniGrid scenarios that we consider. *MultiRoom* corresponds to a set of navigation tasks, where the goal is to go from a starting state to a goal state. In order to go from one room to another, the agent must perform a specific action to open a door. In *KeyCorridor*, the agent also has to pick up a key, since the goal state is behind a door that only lets it in with the key. In *ObstructedMaze*, keys are hidden in

boxes, and doors are obstructed by balls the agent has to get out of its way. *ObstructedMaze-Full* is the hardest configuration for this scenario, since it has the maximal number of keys, balls to move, and doors possible. In each scenario, the agent has access to a partial view of the environment, a 7×7 square that includes itself and points in the direction of its previous movement, as illustrated in Figure 7.1. Importantly, because the layouts of the environments are procedurally-generated, the map is constructed differently at each new episode, making memorization impossible due to the huge size of the state space. Incidentally, the agent must learn to generalize across the different layouts of the environments. We give more context about procedurally-generated tasks in the next section.

7.1.2 Procedurally-generated tasks

Procedurally-generated environments (*e.g.* MiniGrid (Chevalier-Boisvert, Willems, and Pal, 2018), OpenAI Procgen (Cobbe, Hesse, Hilton, et al., 2020)) are now established benchmarks for testing systematic generalization of RL where agents cannot rely on the determinism and low size of the observation space of non-procedurally generated MDPs. They seem to be considered as important problems to be considered when evaluating new RL approaches. For instance, procedurally-generated tasks have been used in a number of publications (some concurrent to our work) on generalization (Igl, Ciosek, Li, et al., 2019; Laskin, Lee, Stooke, et al., 2020; Igl, Farquhar, Luketina, et al., 2021) and exploration (Goyal, Islam, Strouse, et al., 2019; Raileanu and Rocktäschel, 2019; Campero, Raileanu, Küttler, et al., 2021).

7.1.3 Vizdoom



Figure 7.2 – Frames from the 3-D navigation task VizdoomMyWayHome.

In VizDoom (Kempka, Wydmuch, Runc, et al., 2016), we use the scenario *Find My Way Home* where the agent must learn to move along corridors and through rooms without any reward feedback from the 3-D environment. It also features first-person perspective and high-dimensional pixel observations, with each of the 9 rooms in the environment composed of different wall textures, as illustrated in Figure 7.2.

Cart-Pole Variables: x, θ

Type	Constraint
Static	Limit range: $x_l < x < x_r$
Kinematic	Limit velocity near goal: $ \theta > \theta_L \vee \dot{\theta} < \dot{\theta}_V$
Dynamic	Limit cart acceleration: $\ddot{x} < A_{\max}$

Walker Variables: θ, F

Type	Constraint
Static	Limit joint angles: $\theta_L < \theta < \theta_U$
Kinematic	Limit joint velocities: $\max_i \dot{\theta}_i < L_{\dot{\theta}}$
Dynamic	Limit foot contact forces: $F_{\text{Foot}} < F_{\max}$

Quadruped Variables: θ, u, F

Type	Constraint
Static	Limit joint angles: $\theta_{L,i} < \theta_i < \theta_{U,i}$ Enforce upright position: $0 < u_z$
Kinematic	Limit joint velocities: $\max_i \dot{\theta}_i < L_{\dot{\theta}}$
Dynamic	Limit foot contact forces: $F_{\text{Foot}} < F_{\max}$

Table 7.1 – Some of the safety constraints on Cart-Pole, Walker and Quadruped.

7.2 The Real-World RL Challenge

Many of the achievements in RL research remains difficult to implement in real-world systems as a result of a series of assumptions that are rarely met in practice. For instance, in the physical world, safety constraint can be strong and we need to integrate at least a proxy of them during training, even at the expense of performance, which can be less important than safety compliance. The real-world RL suite (Dulac-Arnold, Levine, Mankowitz, et al., 2020) provides continuous control tasks with safety constraints in high-dimensional continuous state and action spaces that aim to capture the aspects of real-world situations that commonly cause RL algorithms to fail. Some of the safety constraint for each available domain are given in Table 7.1 where x denotes a position, θ a joint angle, F a contact force and u some structural direction. A visual example of the kinematic constraint on Cart-Pole Swing-Up is also illustrated in Figure 7.3.

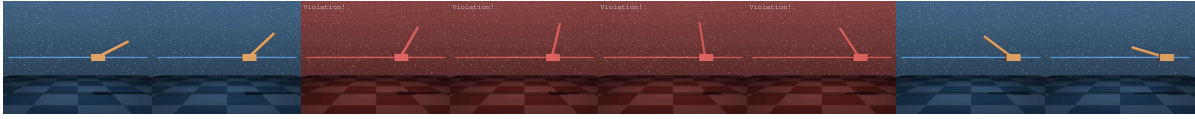


Figure 7.3 – Frames from the Cart-Pole Swing-Up task constrained on the pole angular velocity to be below a certain threshold when arriving near the top.

7.3 Adversarial Prior in the Actor-Critic Framework

While previous frameworks use entropy maximization to obtain diverse base policies, those methods come short when the dimensionality of the state space increases. We observe that the actor-critic framework would benefit from an additional component which would maintain a moving average or mixture of previous policies. This formulation will result in an adversary forcing the agent to move away from the previous policies while remaining close to the current one with the formation of a trust region. We further develop this idea in Chapter 8 on the type of tasks described in Section 7.1.1 and 7.1.2.

The adversarial prior is trained using the batch of trajectories samples by the actor. By imagining an alternative way of maintaining such adversary, one may attempt to train it to break some constraints. This would technically result into an adversarial prior representing a probabilistic unsafe region which the agent should avoid. The idea of an adversarial prior in the actor-critic framework is therefore further developed in the type of tasks described in Section 7.2.

Chapter 8

Use Repulsive Priors to motivate conservatively diversified policies

*We can only see a short distance ahead,
but we can see plenty there
that needs to be done.*

A Quarterly Review of Psychology and Philosophy, Alan Turing (1860).

Actor-critic methods consider a policy (actor) and a value function (critic) whose respective losses are built using different motivations and approaches. In this chapter, we introduce a third protagonist: the adversary. While the adversary mimics the actor by minimizing the KL-divergence between their respective action distributions, the actor, in addition to learning to solve the task, tries to differentiate itself from the adversary predictions. This novel objective stimulates the actor to follow strategies that could not have been correctly predicted from previous trajectories, making its behavior innovative in tasks where the reward is extremely rare. Our experimental analysis shows that the resulting algorithm leads to more exhaustive exploration. Notably, our method extends the state-of-the-art on a set of various hard-exploration and procedurally-generated tasks¹.

Contents

8.1 Motivation	94
8.3 Preliminaries	96
8.4 AGAC: Adversarially Guided Actor-Critic	97
8.5 Experimental Study	100

¹This chapter is based on an article published in the proceedings of the *9th International Conference on Learning Representations (ICLR)* (Flet-Berliac, Ferret, Pietquin, et al., 2021). It is joint work with my colleague and friend Johan Ferret.

8.1 Motivation

Research in deep reinforcement learning (RL) has proven to be successful across a wide range of problems (Silver, Lever, Heess, et al., 2014; Lillicrap, Hunt, Pritzel, et al., 2016; Mnih, Badia, Mirza, et al., 2016; Schulman, Moritz, Levine, et al., 2016). Nevertheless, generalization and exploration in RL still represent key challenges that leave most current methods ineffective. First, a battery of recent studies (Farebrother, Machado, and Bowling, 2018; Zhang, Ballas, and Pineau, 2018; Cobbe, Hesse, Hilton, et al., 2020; Song, Jiang, Du, et al., 2020) indicates that current RL methods fail to generalize correctly even when agents have been trained in a diverse set of environments. Second, exploration has been extensively studied in RL; however, most hard-exploration problems use the same environment for training and evaluation. Hence, since a well-designed exploration strategy should maximize the information received from a trajectory about an environment, the exploration capabilities may not be appropriately assessed if that information is memorized. In this line of research, we choose to study the exploration capabilities of our method and its ability to generalize to new scenarios. Our evaluation domains will, therefore, be tasks with sparse reward in procedurally-generated environments.

In this chapter’s work, we propose Adversarially Guided Actor-Critic (AGAC), which reconsiders the actor-critic framework by introducing a third protagonist: the adversary. Its role is to predict the actor’s actions correctly. Meanwhile, the actor must not only find the optimal actions to maximize the sum of expected returns, but also counteract the predictions of the adversary. This formulation is lightly inspired by adversarial methods, specifically generative adversarial networks (GANs) (Goodfellow, Pouget-Abadie, Mirza, et al., 2014). Such a link between GANs and actor-critic methods has been formalized by Pfau and Vinyals (2016); however, in the context of a third protagonist, we draw a different analogy. The adversary can be interpreted as playing the role of a discriminator that must predict the actions of the actor, and the actor can be considered as playing the role of a generator that behaves to deceive the predictions of the adversary. This approach has the advantage, as with GANs, that the optimization procedure generates a diversity of meaningful data, corresponding to sequences of actions in AGAC.

This chapter analyses and explores how AGAC explicitly drives diversity in the behaviors of the agent while remaining reward-focused, and to which extent this approach allows to adapt to the evolving state space of procedurally-generated environments where the map is constructed differently with each new episode. Moreover, because stability is a legitimate concern since specific instances of adversarial networks were shown to be prone to hyperparameter sensitivity issues (Arjovsky and Bottou, 2017), we also examine this aspect in our experiments.

The contributions of this work are as follows: (i) we propose a novel actor-critic formulation inspired from adversarial learning (AGAC), (ii) we analyse empirically AGAC on key reinforcement learning aspects such as diversity, exploration and stability, (iii) we demonstrate

significant gains in performance on several sparse-reward hard-exploration tasks including procedurally-generated tasks.

8.2 Related Work

Actor-critic methods (Barto, Sutton, and Anderson, 1983; Sutton, 1984) have been extended to the deep learning setting by Mnih, Badia, Mirza, et al. (2016), who combined deep neural networks and multiple distributed actors with an actor-critic setting, with strong results on Atari. Since then, many additions have been proposed, be it architectural improvements (Vinyals, Babuschkin, Czarnecki, et al., 2019), better advantage or value estimation (Schulman, Moritz, Levine, et al., 2016; Flet-Berliac, Ouhamma, Maillard, et al., 2021), or the incorporation of off-policy elements (Wang, Bapst, Heess, et al., 2017; Oh, Guo, Singh, et al., 2018; Flet-Berliac and Preux, 2020). Regularization was shown to improve actor-critic methods, either by enforcing trust regions (Schulman, Levine, Abbeel, et al., 2015; Schulman, Wolski, Dhariwal, et al., 2017; Wu, Mansimov, Grosse, et al., 2017), or by correcting for off-policiness (Munos, Stepleton, Harutyunyan, et al., 2016; Gruslys, Dabney, Azar, et al., 2018); and recent works analyzed its impact from a theoretical standpoint (Ahmed, Le Roux, Norouzi, et al., 2019; Geist, Scherrer, and Pietquin, 2019; Vieillard, Kozuno, Scherrer, et al., 2020; Vieillard, Pietquin, and Geist, 2020). Related to our work, Han and Sung (2020) use the entropy of the mixture between the policy induced from a replay buffer and the current policy as a regularizer. To the best of our knowledge, none of these methods explored the use of an adversarial objective to drive exploration.

While introduced in supervised learning, adversarial learning (Goodfellow, Shlens, and Szegedy, 2015; Miyato, Maeda, Koyama, et al., 2016; Kurakin, Goodfellow, and Bengio, 2017) was leveraged in several RL works. Ho and Ermon (2016) propose an imitation learning method that uses a discriminator whose task is to distinguish between expert trajectories and those of the agent while the agent tries to match expert behavior to fool the discriminator. Bahdanau, Hill, Leike, et al. (2019) use a discriminator to distinguish goal states from non-goal states based on a textual instruction, and use the resulting model as a reward function. Florensa, Held, Geng, et al. (2018) use a GAN to produce sub-goals at the right level of difficulty for the current agent, inducing a form of curriculum. Additionally, Pfau and Vinyals (2016) provide a parallel between GANs and the actor-critic framework.

While exploration is driven in part by the core RL algorithms (Fortunato, Azar, Piot, et al., 2018; Han and Sung, 2020; Ferret, Pietquin, and Geist, 2021), it is often necessary to resort to exploration-specific techniques. For instance, intrinsic motivation encourages exploratory behavior from the agent. Some works use state-visitation counts or pseudo-counts to promote exhaustive exploration (Bellemare, Srinivasan, Ostrovski, et al., 2016), while others use curiosity

rewards, expressed in the magnitude of prediction error from the agent, to push it towards unfamiliar areas of the state space (Burda, Edwards, Storkey, et al., 2018). Ecoffet, Huizinga, Lehman, et al. (2019) propose a technique akin to tree traversal to explore while learning to come back to promising areas. Eysenbach, Gupta, Ibarz, et al. (2018) show that encouraging diversity helps with exploration, even in the absence of reward.

Last but not least, generalization is a key challenge in RL. Zhang, Vinyals, Munos, et al. (2018) showed that, even when the environment is not deterministic, agents can overfit to their training distribution and that it is difficult to distinguish agents likely to generalize to new environments from those that will not. In the same vein, recent work has advocated using procedurally-generated environments, in which a new instance of the environment is sampled when a new episode starts, to assess generalization capabilities better (Justesen, Rodriguez Torrado, Bontrager, et al., 2018; Cobbe, Hesse, Hilton, et al., 2020). Finally, methods based on network randomization (Igl, Ciosek, Li, et al., 2019), noise injection (Lee, Lee, Shin, et al., 2020), and credit assignment (Ferret, Marinier, Geist, et al., 2020) have been proposed to reduce the generalization gap for RL agents.

8.3 Preliminaries

We place ourselves in the Markov Decision Processes (Puterman, 1994) framework. A Markov Decision Process (MDP) is a tuple $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma\}$, where \mathcal{S} is the state space, \mathcal{A} is the action space, \mathcal{P} is the transition kernel, R is the bounded reward function and $\gamma \in [0, 1)$ is the discount factor. Let π denote a stochastic policy mapping states to distributions over actions. We place ourselves in the infinite-horizon setting, *i.e.*, we seek a policy that optimizes $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. The value of a state is the quantity $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s]$ and the value of a state-action pair $Q^\pi(s, a)$ of performing action a in state s and then following policy π is defined as: $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$. The advantage function, which quantifies how an action a is better than the average action in state s , is $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. Finally, the entropy \mathcal{H}^π of a policy is calculated as: $\mathcal{H}^\pi(s) = \mathbb{E}_{\pi(\cdot|s)}[-\log \pi(\cdot|s)]$.

Actor-Critic and Deep Policy Gradients. An actor-critic algorithm is composed of two main components: a policy and a value predictor. In deep RL, both the policy and the value function are obtained via parametric estimators; we denote θ and ϕ their respective parameters. The policy is updated via policy gradient, while the value is usually updated via temporal difference or Monte Carlo rollouts. In practice, for a sequence of transitions $\{s_t, a_t, r_t, s_{t+1}\}_{t \in [0, N]}$, we use the following policy gradient loss (including the commonly used entropic penalty):

$$\mathcal{L}_{PG} = -\frac{1}{N} \sum_{t'=t}^{t+N} (A_{t'} \log \pi(a_{t'} | s_{t'}, \theta) + \alpha \mathcal{H}^\pi(s_{t'}, \theta)),$$

where α is the entropy coefficient and A_t is the generalized advantage estimator (Schulman, Moritz, Levine, et al., 2016) defined as: $A_t = \sum_{t'=t}^{t+N} (\gamma\lambda)^{t'-t} (r_{t'} + \gamma V_{\phi_{\text{old}}}(s_{t'+1}) - V_{\phi_{\text{old}}}(s_{t'}))$, with λ a fixed hyperparameter and $V_{\phi_{\text{old}}}$ the value function estimator at the previous optimization iteration. To estimate the value function, we solve the non-linear regression problem $\text{minimize}_{\phi} \sum_{t'=t}^{t+N} (V_{\phi}(s_{t'}) - \hat{V}_{t'})^2$ where $\hat{V}_{t'} = A_t + V_{\phi_{\text{old}}}(s_{t'})$.

8.4 AGAC: Adversarially Guided Actor-Critic

To foster diversified behavior in its trajectories, AGAC introduces a third protagonist to the actor-critic framework: the adversary. The role of the adversary is to accurately predict the actor’s actions, by *minimizing* the discrepancy between its action distribution π_{adv} and the distribution induced by the policy π . Meanwhile, in addition to finding the optimal actions to *maximize* the sum of expected returns, the actor must also counteract the adversary’s predictions by *maximizing* the discrepancy between π and π_{adv} (see Appendix D.2 for an illustration). This discrepancy, used as a form of exploration bonus, is defined as the difference of action log-probabilities (see Equation (8.1)), whose expectation is the Kullback–Leibler divergence:

$$D_{\text{KL}}(\pi(\cdot|s) \parallel \pi_{\text{adv}}(\cdot|s)) = \mathbb{E}_{\pi(\cdot|s)} [\log \pi(\cdot|s) - \log \pi_{\text{adv}}(\cdot|s)].$$

Formally, for each state-action pair (s_t, a_t) in a trajectory, an action-dependent bonus $\log \pi(a_t|s_t) - \log \pi_{\text{adv}}(a_t|s_t)$ is added to the advantage. In addition, the value target of the critic is modified to include the action-independent equivalent, which is the KL-divergence $D_{\text{KL}}(\pi(\cdot|s_t) \parallel \pi_{\text{adv}}(\cdot|s_t))$. We discuss the role of these mirrored terms below, and the implications of AGAC’s modified objective from a more theoretical standpoint in the next section. In addition to the parameters θ (resp. θ_{old} the parameter of the policy at the previous iteration) and ϕ defined above (resp. ϕ_{old} that of the critic), we denote ψ (resp. ψ_{old}) that of the adversary.

AGAC minimizes the following loss:

$$\mathcal{L}_{\text{AGAC}} = \mathcal{L}_{\text{PG}} + \beta_V \mathcal{L}_V + \beta_{\text{adv}} \mathcal{L}_{\text{adv}}.$$

In the new objective $\mathcal{L}_{\text{PG}} = -\frac{1}{N} \sum_{t=0}^N (A_t^{\text{AGAC}} \log \pi(a_t|s_t, \theta) + \alpha \mathcal{H}^{\pi}(s_t, \theta))$, AGAC modifies A_t as:

$$A_t^{\text{AGAC}} = A_t + c \left(\log \pi(a_t|s_t, \theta_{\text{old}}) - \log \pi_{\text{adv}}(a_t|s_t, \psi_{\text{old}}) \right), \quad (8.1)$$

with c a varying hyperparameter that controls the dependence on the action log-probability difference. To encourage exploration without preventing asymptotic stability, c is linearly

annealed during the course of training. \mathcal{L}_V is the objective function of the critic defined as:

$$\mathcal{L}_V = \frac{1}{N} \sum_{t=0}^N \left(V_{\phi}(s_t) - \left(\hat{V}_t + c D_{\text{KL}}(\pi(\cdot|s_t, \theta_{\text{old}}) \| \pi_{\text{adv}}(\cdot|s_t, \psi_{\text{old}})) \right) \right)^2. \quad (8.2)$$

Finally, \mathcal{L}_{adv} is the objective function of the adversary:

$$\mathcal{L}_{\text{adv}} = \frac{1}{N} \sum_{t=0}^N D_{\text{KL}}(\pi(\cdot|s_t, \theta_{\text{old}}) \| \pi_{\text{adv}}(\cdot|s_t, \psi)). \quad (8.3)$$

Eqs. (8.1), (8.2) and (8.3) are the three equations that our method modifies (we color in blue the specific parts) in the traditional actor-critic framework. The terms β_V and β_{adv} are fixed hyperparameters.

Under the proposed actor-critic formulation, the probability of sampling an action is increased if the modified advantage is positive, *i.e.* (i) the corresponding return is larger than the predicted value and/or (ii) the action log-probability difference is large. More precisely, our method favors transitions whose actions were less accurately predicted than the average action, *i.e.* $\log \pi(a|s) - \log \pi_{\text{adv}}(a|s) \geq D_{\text{KL}}(\pi(\cdot|s) \| \pi_{\text{adv}}(\cdot|s))$. This is particularly visible for $\lambda \rightarrow 1$, in which case the generalized advantage is $A_t = G_t - V_{\phi_{\text{old}}}(s_t)$, resulting in the appearance of both aforementioned mirrored terms in the modified advantage:

$$A_t^{\text{AGAC}} = G_t - \hat{V}_t^{\phi_{\text{old}}} + c \left(\log \pi(a_t|s_t) - \log \pi_{\text{adv}}(a_t|s_t) - \hat{D}_{\text{KL}}^{\phi_{\text{old}}}(\pi(\cdot|s_t) \| \pi_{\text{adv}}(\cdot|s_t)) \right),$$

with G_t the observed return, $\hat{V}_t^{\phi_{\text{old}}}$ the estimated return and $\hat{D}_{\text{KL}}^{\phi_{\text{old}}}(\pi(\cdot|s_t) \| \pi_{\text{adv}}(\cdot|s_t))$ the estimated KL-divergence (estimated components of $V_{\phi_{\text{old}}}(s_t)$ from Equation 8.2).

To avoid instability, in practice the adversary is a separate estimator, updated with a smaller learning rate than the actor. This way, it represents a delayed and more steady version of the actor’s policy, which prevents the agent from having to constantly adapt or focus solely on fooling the adversary.

8.4.1 Building Motivation

In the following, we provide an interpretation of AGAC by studying the dynamics of attraction and repulsion between the actor and the adversary. To simplify, we study the equivalent of AGAC in a policy iteration (PI) scheme. PI being the dynamic programming scheme underlying the standard actor-critic, we have reasons to think that some of our findings translate to the original AGAC algorithm. In PI, the quantity of interest is the action-value, which AGAC would modify as:

$$Q_{\pi_k}^{\text{AGAC}} = Q_{\pi_k} + c (\log \pi_k - \log \pi_{\text{adv}}),$$

with π_k the policy at iteration k . Incorporating the entropic penalty, the new policy π_{k+1} verifies:

$$\pi_{k+1} = \arg \max_{\pi} \mathcal{J}_{\text{PI}}(\pi) = \arg \max_{\pi} \mathbb{E}_s \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\pi_k}^{\text{AGAC}}(s, a) - \alpha \log \pi(a|s)].$$

We can rewrite this objective:

$$\begin{aligned} \mathcal{J}_{\text{PI}}(\pi) &= \mathbb{E}_s \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\pi_k}^{\text{AGAC}}(s, a) - \alpha \log \pi(a|s)] \\ &= \mathbb{E}_s \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\pi_k}(s, a) + c (\log \pi_k(a|s) - \log \pi_{\text{adv}}(a|s)) - \alpha \log \pi(a|s)] \\ &= \mathbb{E}_s \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\pi_k}(s, a) + c (\log \pi_k(a|s) - \log \pi(a|s) + \log \pi(a|s) - \log \pi_{\text{adv}}(a|s)) - \alpha \log \pi(a|s)] \\ &= \mathbb{E}_s \left[\underbrace{\mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\pi_k}(s, a)]}_{\pi_k \text{ is attractive}} - \underbrace{c D_{\text{KL}}(\pi(\cdot|s) || \pi_k(\cdot|s))}_{\pi_{\text{adv}} \text{ is repulsive}} + \underbrace{c D_{\text{KL}}(\pi(\cdot|s) || \pi_{\text{adv}}(\cdot|s)) + \alpha \mathcal{H}(\pi(\cdot|s))}_{\text{enforces stochastic policies}} \right]. \end{aligned}$$

Thus, in the PI scheme, AGAC finds a policy that maximizes Q -values, while at the same time remaining close to the current policy and far from a mixture of the previous policies (*i.e.*, π_{k-1} , π_{k-2} , π_{k-3} , ...). Note that we experimentally observe (see Section 8.5.3) that our method performs better with a smaller learning rate for the adversarial network than that of the other networks, which could imply that a stable repulsive term is beneficial.

This optimization problem is strongly concave in π (thanks to the entropy term), and is state-wise a Legendre-Fenchel transform. Its solution is given by (see Appendix D.5 for the full derivation):

$$\pi_{k+1} \propto \left(\frac{\pi_k}{\pi_{\text{adv}}} \right)^{\frac{c}{\alpha}} \exp \frac{Q_{\pi_k}}{\alpha}.$$

This result gives us some insight into the behavior of the objective function. Notably, in our example, if π_{adv} is fixed and $c = \alpha$, we recover a KL-regularized PI scheme (Geist, Scherrer, and Pietquin, 2019) with the modified reward $r - c \log \pi_{\text{adv}}$.

8.4.2 Implementation

In all of the experiments, we use PPO (Schulman, Wolski, Dhariwal, et al., 2017) as the base algorithm and build on it to incorporate our method. Hence,

$$\mathcal{L}_{\text{PG}} = -\frac{1}{N} \sum_{t'=t}^{t+N} \min \left(\frac{\pi(a_{t'}|s_{t'}, \theta)}{\pi(a_{t'}|s_{t'}, \theta_{\text{old}})} A_{t'}^{\text{AGAC}}, \text{clip} \left(\frac{\pi(a_{t'}|s_{t'}, \theta)}{\pi(a_{t'}|s_{t'}, \theta_{\text{old}})}, 1 - \varepsilon, 1 + \varepsilon \right) A_{t'}^{\text{AGAC}} \right),$$

with $A_{t'}^{\text{AGAC}}$ given in Equation (8.1), N the temporal length considered for one update of parameters and ε the clipping parameter. Similar to RIDE (Raileanu and Rocktäschel, 2019), we also discount PPO by episodic state visitation counts, except for VizDoom (*cf.* Section 8.5.1). The actor, critic and adversary use the convolutional architecture of the Nature paper of DQN (Mnih, Kavukcuoglu, Silver, et al., 2015) with different hidden sizes (see Appendix D.4 for architecture

details). The three neural networks are optimized using Adam (Kingma and Ba, 2015). Our method does not use RNNs in its architecture; instead, in all our experiments, we use frame stacking. Indeed, Hausknecht and Stone (2015) interestingly demonstrate that although recurrence is a reliable method for processing state observation, it does not confer any systematic advantage over stacking observations in the input layer of a CNN. Note that the parameters are not shared between the policy, the critic and the adversary and that we did not observe any noticeable difference in computational complexity when using AGAC compared to PPO. We direct the reader to Appendix D.3 for a list of hyperparameters. In particular, the c coefficient of the adversarial bonus is linearly annealed.

At each training step, we perform a stochastic optimization step to minimize \mathcal{L}_{AGAC} using stop-gradient:

$$\theta \leftarrow \text{Adam}(\theta, \nabla_{\theta} \mathcal{L}_{PG}, \eta_1), \quad \phi \leftarrow \text{Adam}(\phi, \nabla_{\phi} \mathcal{L}_V, \eta_1), \quad \psi \leftarrow \text{Adam}(\psi, \nabla_{\psi} \mathcal{L}_{adv}, \eta_2).$$

8.5 Experimental Study

In this section, we describe our experimental study in which we investigate: (i) whether the adversarial bonus alone (*e.g.* without episodic state visitation count) is sufficient to outperform other methods in VizDoom, a sparse-reward task with high-dimensional observations, (ii) whether AGAC succeeds in partially-observable and procedurally-generated environments with high sparsity in the rewards, compared to other methods, (iii) how well AGAC is capable of exploring in environments without extrinsic reward, (iv) the training stability of our method. In all of the experiments, lines are average performances and shaded areas represent one standard deviation. The code for our method is released and open-source: github.com/yfletberliac/agac.

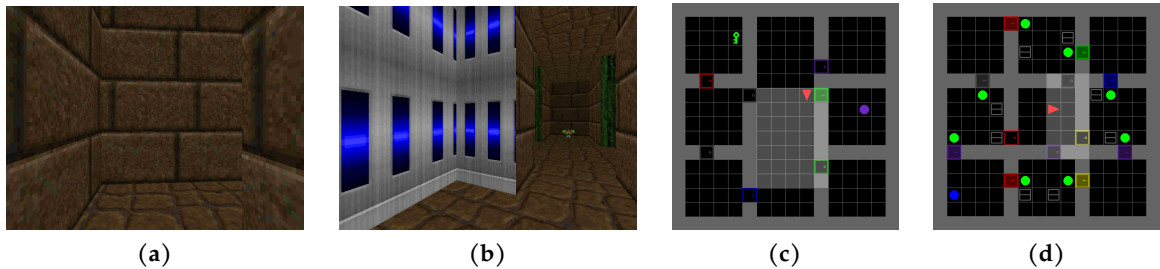


Figure 8.1 – Frames from (a,b) the 3-D navigation task VizdoomMyWayHome. (c) MiniGrid-KeyCorridorS6R3. (d) MiniGrid-ObstructedMazeFull.

Environments. To carefully evaluate the performance of our method, its ability to develop robust exploration strategies and its generalization to unseen states, we choose tasks that have been used in prior work, which are tasks with high-dimensional observations, sparse reward

and procedurally-generated environments. In **VizDoom** (Kempka, Wydmuch, Runc, et al., 2016), the agent must learn to move along corridors and through rooms without any reward feedback from the 3-D environment. The **MiniGrid** environments (Chevalier-Boisvert, Willems, and Pal, 2018) are a set of challenging partially-observable and sparse-reward gridworlds. In this type of procedurally-generated environments, memorization is impossible due to the huge size of the state space, so the agent must learn to generalize across the different layouts of the environment. Each gridworld has different characteristics: in the MultiRoom tasks, the agent is placed in the first room and should reach a goal placed in the most distant room. In the KeyCorridor tasks, the agent must navigate to pick up an object placed in a room locked by a door whose key is in another room. Finally, in the ObstructedMaze tasks, the agent must pick up a box that is placed in a corner of a 3x3 maze in which the doors are also locked, the keys are hidden in boxes and balls obstruct the doors. All considered environments (see Figure 8.1 for some examples) are available as part of OpenAI Gym (Brockman, Cheung, Pettersson, et al., 2016).

Baselines. For a fair assessment of our method, we compare to some of the most prominent methods specialized in hard-exploration tasks: **RIDE** (Raileanu and Rocktäschel, 2019), based on an intrinsic reward associated with the magnitude of change between two consecutive state representations and state visitation, **Count** as Count-Based Exploration (Bellemare, Srinivasan, Ostrovski, et al., 2016), which we couple with IMPALA (Espeholt, Soyer, Munos, et al., 2018), **RND** (Burda, Edwards, Storkey, et al., 2018) in which an exploration bonus is positively correlated to the error of predicting features from the observations and **ICM** (Pathak, Agrawal, Efros, et al., 2017), where a module only predicts the changes in the environment that are produced by the actions of the agent. Finally, we compare to most the recent and best performing method at the time of introducing the method in procedurally-generated environments: **AMIGo** (Campero, Raileanu, Küttler, et al., 2021) in which a goal-generating teacher provides count-based intrinsic goals.

8.5.1 Adversarially-based Exploration (No Episodic Count)

Table 8.1 – Average return in VizDoom at different timesteps.

Nb. of Timesteps	2M	4M	6M	8M	10M
AGAC	0.74 ± 0.05	0.96 ± 0.001	0.96 ± 0.001	0.97 ± 0.001	0.97 ± 0.001
RIDE	0.	0.	0.95 ± 0.001	0.97 ± 0.001	0.97 ± 0.001
ICM	0.	0.	0.95 ± 0.001	0.97 ± 0.001	0.97 ± 0.001
AMIGo	0.	0.	0.	0.	0.
RND	0.	0.	0.	0.	0.
Count	0.	0.	0.	0.	0.

In this section, we assess the benefits of using an adversarially-based exploration bonus and examine how AGAC performs without the help of count-based exploration. In order to provide a comparison to state-of-the-art methods, we choose VizDoom, a hard-exploration problem used in prior work. In this game, the map consists of 9 rooms connected by corridors where 270 steps separate the initial position of the agent and the goal under an optimal policy. Episodes are terminated either when the agent finds the goal or if the episode exceeds 2100 timesteps. Importantly, while other algorithms (Raileanu and Rocktäschel, 2019; Campero, Raileanu, Küttler, et al., 2021) benefit from count-based exploration, this study has been conducted with our method not benefiting from episodic count whatsoever. Results in Table 8.1 indicate that AGAC clearly outperforms other methods in sample-efficiency. Only the methods ICM and RIDE succeed in matching the score of AGAC, and with about twice as much transitions ($\sim 3M$ vs. $6M$). Interestingly, AMIGo performs similarly to Count and RND. We find this result surprising because AMIGo has proven to perform well in the MiniGrid environments. Nevertheless, it appears that concurrent works to ours experienced similar issues with the accompanying implementation². The results of AGAC support the capabilities of the adversarial bonus and show that it can, on its own, achieve significant gains in performance. However, the VizDoom task is not procedurally-generated; hence we have not evaluated the generalization to new states yet. In the following section, we use MiniGrid to investigate this.

8.5.2 Hard-Exploration Tasks with Partially-Observable Environments

We now evaluate our method on multiple hard-exploration procedurally-generated tasks from MiniGrid. Details about MiniGrid can be found in Appendix D.3.1. Figure 8.2 indicates that AGAC significantly outperforms other methods on these tasks in sample-efficiency and performance. AGAC also outperforms the current state-of-the-art method, AMIGo, despite the fact that it uses the fully-observable version of MiniGrid. Note that we find the same poor performance results when training AMIGo in MiniGrid, similar to Vizdoom results. For completeness, we also report in Table D.1 of Appendix D.1.1 the performance results with the scores reported in the original papers Raileanu and Rocktäschel (2019) and Campero, Raileanu, Küttler, et al. (2021). We draw similar conclusions: AGAC clearly outperforms the state-of-the-art RIDE, AMIGo, Count, RND and ICM.

In all the considered tasks, the agent must learn to generalize across a very large state space because the layouts are generated procedurally. We consider three main arguments to explain why our method is successful: (i) our method makes use of partial observations: in this context, the adversary has a harder time predicting the actor’s actions; nevertheless, the mistakes of the former benefit the latter in the form of an exploration bonus, which pushes the agent to explore further in order to deceive the adversary, (ii) the exploration bonus (*i.e.* intrinsic reward) does

²AMIGo implementation [GitHub Issue](#).

not dissipate compared to most other methods, as observed in Figure D.3 in Appendix D.1.4, (iii) our method does not make assumptions about the environment dynamics (*e.g.*, changes in the environment produced by an action as in Raileanu and Rocktäschel (2019)) since this can hinder learning when the space of state changes induced by an action is too large (such as the action of moving a block in ObstructedMaze).

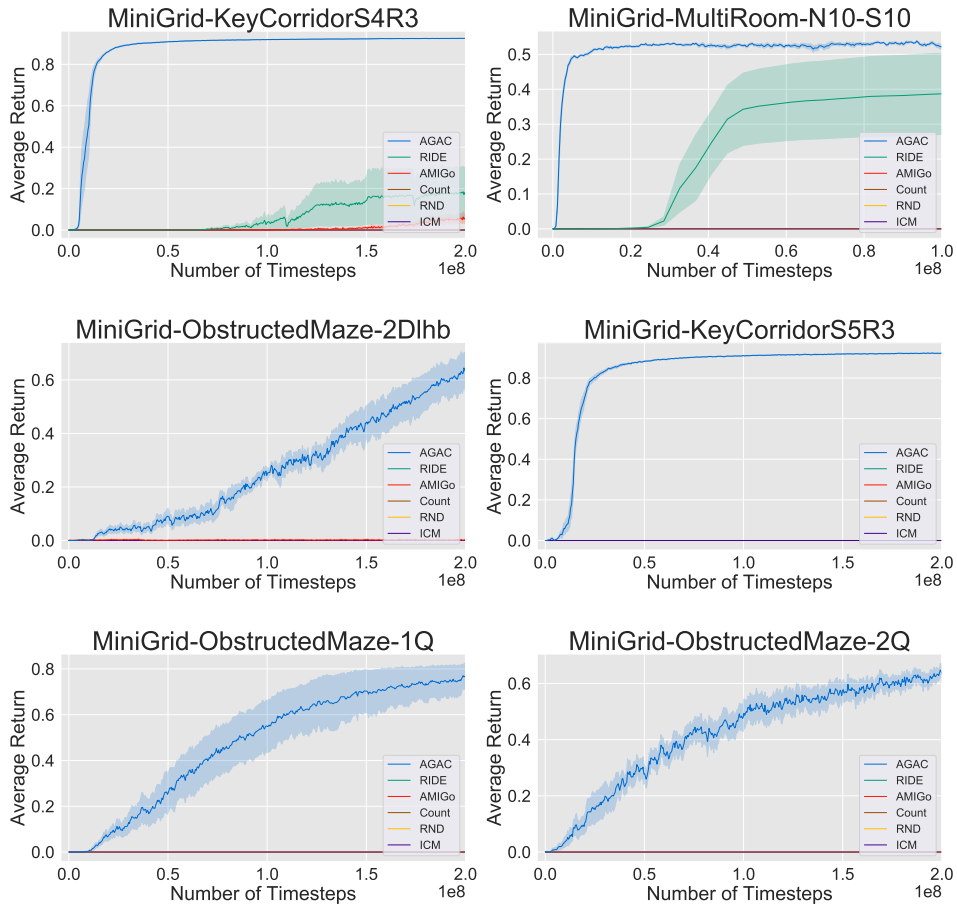


Figure 8.2 – Performance evaluation of AGAC.

In Appendix D.1.3, we also include experiments in two environments with extremely sparse reward signals: KeyCorridorS8R3 and ObstructedMazeFull. Despite the challenge, AGAC still manages to find rewards and can perform well by taking advantage of the diversified behaviour induced by our method. To the best of our knowledge, no other method ever succeeded to perform well (> 0 average return) in those tasks. We think that given more computing time, AGAC’s score could go higher.

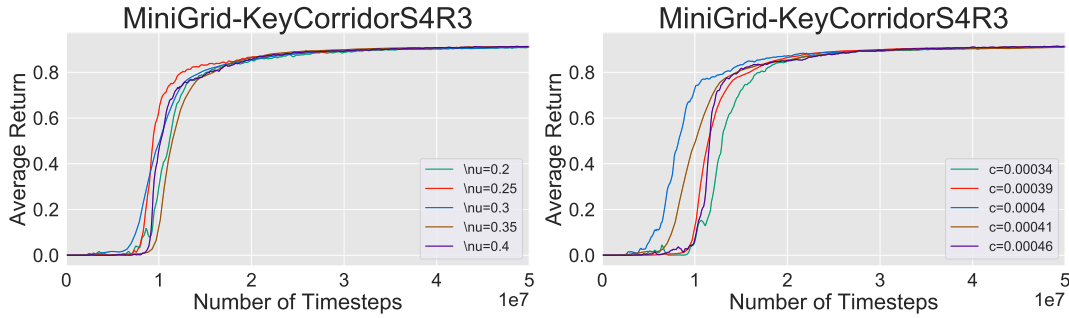


Figure 8.3 – Sensitivity analysis of AGAC in KeyCorridorS4R3.

8.5.3 Training Stability

Here we want to analyse the stability of the method when changing hyperparameters. The most important parameters in AGAC are c , the coefficient for the adversarial bonus, and the learning rates ratio $\nu = \frac{\eta_2}{\eta_1}$. We choose KeyCorridorS4R3 as the evaluation task because among all the tasks considered, its difficulty is at a medium level. Figure 8.3 shows the learning curves. For readability, we plot the average return only; the standard deviation is sensibly the same for all curves. We observe that deviating from the hyperparameter values found using grid search results in a slower training. Moreover, although reasonable, c appears to have more sensitivity than ν .

8.5.4 Exploration in Reward-free Environment

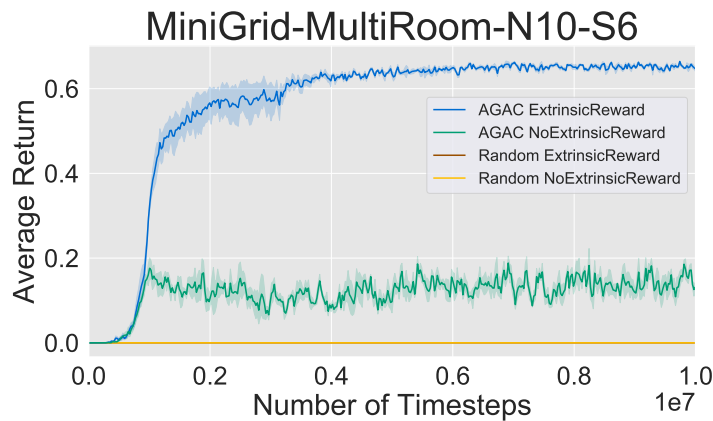


Figure 8.4 – Average return on N10S6 with and without extrinsic reward.

To better understand the effectiveness of our method and inspect how the agent collects rewards that would not otherwise be achievable by simple exploration heuristics or other methods, we analyze the performance of AGAC in another (procedurally-generated) challenging environment, MultiRoomN10S6, when there is no reward signal, *i.e.* no extrinsic reward.

Beyond the good performance of our method when extrinsic rewards are given to the agent, Figure 8.4 indicates that the exploration induced by our method makes the agent succeed in a significant proportion of the episodes: in the configuration “NoExtrinsicReward” the reward signal is not given (the goal is invisible to the agent) and the performance of AGAC stabilizes around an average return of ~ 0.15 . Since the return of an episode is either 0 or 1 (depending on whether the agent reached the goal state or not), and because this value is aggregated across several episodes, the results indicate that reward-free AGAC succeeds in $\sim 15\%$ of the tasks. Comparatively, random agents have a zero average return. This poor performance is in accordance with the results in Raileanu and Rocktäschel (2019) and reflects the complexity of the task: in order to go from one room to another, an agent must perform a specific action to open a door and cross it within the time limit of 200 timesteps. In the following, we visually investigate how different methods explore the environments.

8.5.5 Visualizing Coverage and Diversity



Figure 8.5 – State visitation heatmaps for RND, Count, a random uniform policy, RIDE, and AGAC trained in a singleton environment (top row) and procedurally-generated environments (bottom row) without extrinsic reward for 10M timesteps in the MultiRoomN10S6 task.

In this section, we first investigate how different methods explore environments without being guided by extrinsic rewards (the green goal is invisible to the agent) on both *procedurally-generated* and *singleton* environments. In singleton environments, an agent has to solve the same task in the same environment/maze in every episode. Figure 8.5 shows the state visitation heatmaps (darker areas correspond to more visits) after a training of 10M timesteps. We observe that most of the methods explore inefficiently in a singleton environment and that only RIDE succeeds in reaching the fifth room while AGAC reaches the last (tenth) room. After

training the agents in procedurally-generated environments, the methods explore even less efficiently while AGAC succeeds in exploring all rooms.

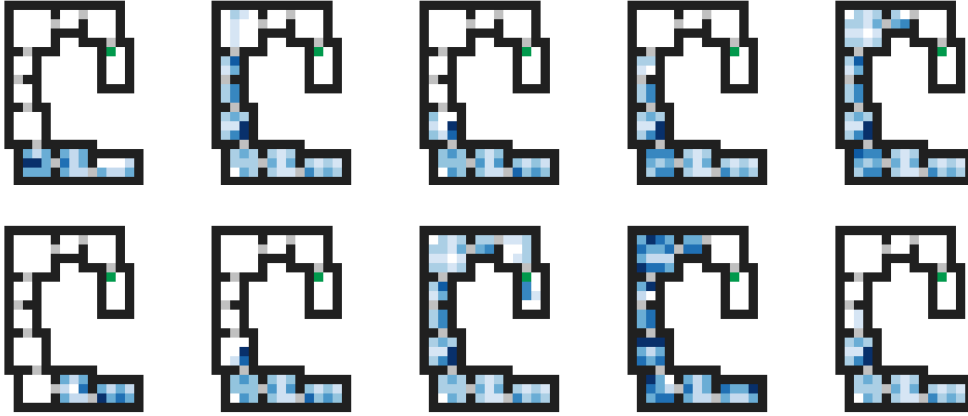


Figure 8.6 – State visitation heatmaps of the last ten episodes of an agent trained in *procedurally-generated* environments without extrinsic reward for 10M timesteps in the MultiRoomN10S6 task. The agent is continuously engaging in new strategies.

We now qualitatively study the diversity of an agent’s behavior when trained with AGAC. Figure 8.6 presents the state visitation heatmaps of the last ten episodes for an agent trained in *procedurally-generated* environments in the MultiRoomN10S6 task without extrinsic reward. The heatmaps correspond to the behavior of the resulting policy, which is still learning from the AGAC objective. Looking at the figure, we can see that the strategies vary at each update with, for example, back-and-forth and back-to-start behaviors. Although there are no extrinsic reward, the strategies seem to diversify from one update to the next. Finally, Figure D.1 in Appendix D.1.2 shows the state visitation heatmaps in a different configuration: when the agent has been trained on a *singleton* environment in the MultiRoomN10S6 task without extrinsic reward. Same as previously, the agent is updated between each episode. Looking at the figure, we can make essentially the same observations as previously, with a noteworthy behavior in the fourth heatmap of the bottom row where it appears the agent went to the fourth room to remain inside it. Those episodes indicate that, although the agent sees the same environment repeatedly, the successive adversarial updates force it to continuously adapt its behavior and try new strategies.

Chapter conclusion

This chapter introduced AGAC, a modification to the traditional actor-critic framework: an adversary network is added as a third protagonist. The mechanics of AGAC have been discussed from a policy iteration point of view, and we provided theoretical insight into the inner workings

of the proposed algorithm: the adversary forces the agent to remain close to the current policy while moving away from the previous ones. In a nutshell, the influence of the adversary makes the actor *conservatively diversified*.

In the experimental study, we have evaluated the adversarially-based bonus in VizDoom and empirically demonstrated its effectiveness and superiority compared to other relevant methods (some benefiting from count-based exploration). Then, we have conducted several performance experiments using AGAC and have shown a significant performance improvement over some of the most popular exploration methods (RIDE, AMIGo, Count, RND and ICM) on a set of various challenging tasks from MiniGrid. These procedurally-generated environments have served another purpose which is to validate the capacity of our method to generalize to unseen scenarios. In addition, the training stability of our method has been studied, showing a greater but acceptable sensitivity for c , the adversarial bonus coefficient. Finally, we have investigated the exploration capabilities of AGAC in a reward-free setting where the agent demonstrated exhaustive exploration through various strategic choices, confirming that the adversary successfully drives diversity in the behavior of the actor.

Chapter 9

Use Repulsive Priors to motivate risk-sensitive policies

It is a good thing to have two ways of looking at a subject, and to admit that there are two ways of looking at it.

The genesis of Maxwell's equations, James Clerk Maxwell (1860).

Although we have witnessed policy-based methods being effective for a number of sequential decision-making problems under uncertainty, they still stumble to thrive in real-world systems where risk or safety is a binding constraint. Using what we have learned in Chapter 8, in this chapter, we formulate the RL problem with safety constraints as a non-zero-sum game where an adversary tries to break the safety constraint while the RL agent tries to maximize the constrained value function given the adversary's policy. We propose a method which can address different safety criteria such as safe exploration, mean-variance risk sensitivity, and CVaR-like risk sensitivity. In each of these variations, we show the agent differentiates itself from the adversary's unsafe actions (and satisfies various safety constraints) in addition to learning to solve the task¹.

Contents

9.1 Motivation	110
9.2 Preliminaries	111
9.4 SAAC: Safe Adversarial Soft Actor-Critics	115
9.5 Experimental Study	120

¹This chapter is based on a collaboration with Debabrota Basu.

9.1 Motivation

As discussed in Chapter 1 and 2, designing an RL algorithm requires both efficient quantification of uncertainty regarding the incomplete information and the probabilistic decision making policy, and effective design of a policy that can leverage these quantifications to achieve optimal performance. Recent success of RL in structured games, like Chess and Go (Silver, Schrittwieser, Simonyan, et al., 2017), and simulated environments, like continuous control using simulators (Lillicrap, Hunt, Pritzel, et al., 2016; Degraeve, Hermans, Dambre, et al., 2019), have drawn significant amount of interest.

Still, real-world deployment of RL in *e.g.* industrial processes, unmanned vehicles, or robotics does not only require efficiency in terms of performance but also being sensitive to risks involved in decisions (Pan, You, Wang, et al., 2017; Dulac-Arnold, Levine, Mankowitz, et al., 2020; Thananjeyan, Balakrishna, Nair, et al., 2021). In this chapter, we are interested in works quantifying risks in RL and designing risk-sensitive (or robust, or safe) RL algorithms (Garcia and Fernández, 2015; Pinto, Davidson, Sukthankar, et al., 2017; Ray, Achiam, and Amodei, 2019; Wachi and Sui, 2020; Eriksson, Basu, Alibeigi, et al., 2021; Eysenbach and Levine, 2021).

Risk-sensitive RL. In risk-sensitive RL, the perception of risk-sensitivity or safety is embedded mainly using two approaches. The first approach is constraining the RL algorithm to converge in a restricted, “safe” region of the state space (Geibel and Wysotzki, 2005; Koller, Berkenkamp, Turchetta, et al., 2018; Ray, Achiam, and Amodei, 2019; Thananjeyan, Balakrishna, Nair, et al., 2021). Here, the “safe” region is the part of the state space that obeys some external risk-based constraints, such as the non-slippery part of the floor for a walker. RL algorithms developed using this approach either try to construct policies that generate trajectories which stay in this safe region with high probability (Geibel and Wysotzki, 2005), or to start with a conservative “safe” policy and then to incrementally estimate the maximal safe region (Berkenkamp, Moriconi, Schoellig, et al., 2016).

The other approach is to define a risk-measure on the long-term cumulative return of a policy for a fixed environment, and then to minimize the corresponding total risk (Howard and Matheson, 1972; Garcia and Fernández, 2015; Prashanth and Fu, 2018). A risk-measure is a statistics computed on the cumulative return which quantifies either the spread of the return distribution around its mean value or the heaviness of this distribution’s tails (Szegő, 2004). Example of such risk measures are variance, conditional value-at-risk (CVaR) (Rockafellar and Uryasev, 2000), exponential utility (Howard and Matheson, 1972), variance (Prashanth and Ghavamzadeh, 2016), etc. These risk-measures are extensively used real-world applications like dynamic pricing (Lim and Shanthikumar, 2007), financial decision making (Artzner, Delbaen, Eber, et al., 1999), robust control (Chen, Aravena, and Zhou, 2005), and other decision making problems where risk has consequential effects.

Contributions of this chapter. This chapter unifies both of these approaches as a constrained RL problem, and further derives an equivalent non-zero sum (NZS) stochastic game formulation (Sorin, 1986) of it. In this NZS game formulation inspired from the AGAC framework developed in Chapter 8, risk-sensitive RL reduces to *a game between an agent and an adversary*: the adversary tries to break the *safety constraints*, i.e. either to move out of the “safe” region or to increase the risk measures corresponding to a given policy. In contrast, the agent tries to construct a policy that maximizes its expected return given the adversarial feedback, which is a statistics computed on the adversary’s constraint breaking. Note that in Chapter 8, the adversarial guidance is deployed for a trust-region policy gradient in order to enhance exploration where the adversary is oblivious to risk and maintains only a policy.

We propose a generic actor-critic framework where any two compatible actor-critic RL algorithms are employed to enact as the agent and the adversary to ensure risk-sensitive performance. In order to instantiate our approach, we propose in Section 9.4 a specific algorithm, *Safe Adversarially guided Actor-Critic* (SAAC), that deploys two Soft Actor-Critics (SAC) (Haarnoja, Zhou, Abbeel, et al., 2018) as the agent and the adversary. We further derive the policy gradients for the SACs corresponding to the agent and the adversary, which shows that the risk-sensitivity of the agent is ensured by a term repulsing it from the adversary in the policy space. Interestingly, this term can also be used to seek risk and explore more.

In Section 9.5, we experimentally verify the risk-sensitive performance of SAAC under safe region, CVaR, and variance constraints for continuous control tasks from the real-world RL suite (Dulac-Arnold, Levine, Mankowitz, et al., 2020). In these tasks, we show that SAAC is not only risk-sensitive but also outperforms the state-of-the-art risk-sensitive RL and distributional RL algorithms.

9.2 Preliminaries

In this section, we elaborate the details of the three main components of our work: Markov Decision Process (MDP), MaxEnt RL, and risk-sensitive RL.

9.2.1 Markov Decision Processes

We continue to consider RL problems that can be modelled as a *Markov Decision Process* (MDP) (Puterman, 1994) defined as a tuple $\mathcal{M} \triangleq (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$. $\mathcal{S} \subseteq \mathbb{R}^d$ is the *state space*. \mathcal{A} is the admissible *action space*. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the *reward function* that quantifies the goodness or badness of a state-action pair (s, a) . $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$ is the *transition kernel* that dictates the probability to go to a next state given the present state and action. The goal of the agent is to compute a *policy* $\pi : \mathcal{S} \rightarrow \Delta_{\mathcal{A}}$ that maximizes the expected value of cumulative

rewards obtained by a time horizon $T \in \mathbb{N}$. For a given policy π , the *value function* or the expected value of discounted cumulative rewards is

$$\begin{aligned} V_\pi(s) &\triangleq \mathbb{E}_{\substack{a_t \sim \pi(s_t) \\ s_t \sim \mathcal{T}(s_{t-1}, a_{t-1})}} \left[\sum_{t=0}^T \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s \right] \\ &\triangleq \mathbb{E}_{\pi, \mathcal{M}}[Z_\pi^T(s)], \end{aligned}$$

where $\gamma \in [0, 1)$ is the *discount factor* that quantifies the effect of the reward at present step to the next one. We refer to $Z_\pi^T(s)$ as the *return* of policy π up to time T .

9.2.2 Maximum-Entropy RL

In this chapter, we adopt the Maximum-Entropy RL (MaxEnt RL) framework (Eysenbach and Levine, 2019, 2021), also known as entropy-regularized RL (Neu, Jonsson, and Gómez, 2017; Geist, Scherrer, and Pietquin, 2019). MaxEnt RL aims to maximize the sum of value function and the conditional action entropy, $\mathcal{H}_\pi(a|s)$, for a policy π :

$$\begin{aligned} &\arg \max_{\pi} V_\pi(s) + \mathcal{H}_\pi(a|s) \\ &= \arg \max_{\pi} \mathbb{E}_{\substack{a_t \sim \pi(s_t) \\ s_t \sim \mathcal{T}(s_{t-1}, a_{t-1})}} \left[Z_\pi^T(s) - \log \pi(a_t|s_t) \mid s_0 = s \right]. \end{aligned}$$

Unlike the classical value function maximizing RL that always has a deterministic policy as a solution (Puterman, 1994), MaxEnt RL tries to learn stochastic policies such that states with multiple near-optimal actions has higher entropy and states with single optimal action has lower entropy. Interestingly, solving MaxEnt RL is equivalent to computing a policy π that has minimum KL-divergence from a target distribution $\mathcal{T} \circ \mathcal{R}$ given a trajectory $\tau = \{s_0, a_0, \dots, s_T, a_T\}$:

$$\arg \max_{\pi} V_\pi(s) + \mathcal{H}_\pi(a|s) = \arg \min_{\pi} D_{\text{KL}}(\pi(\tau) \parallel \mathcal{T} \circ \mathcal{R}(\tau)). \quad (9.1)$$

The target distribution $\mathcal{T} \circ \mathcal{R}$ is a softmax or Boltzmann distribution on the cumulative rewards given trajectory τ : $\mathcal{T} \circ \mathcal{R}(\tau) \propto p_0(s) \prod_{t=0}^T \mathcal{T}(s_{t+1}|s_t, a_t) \exp[Z_\pi^T(s)]$, the policy distribution is the distribution of generating trajectory τ given the policy π and MDP \mathcal{M} : $\pi(\tau) \propto p_0(s) \prod_{t=0}^T \mathcal{T}(s_{t+1}|s_t, a_t) \pi(a_t|s_t)$. Thus, in MaxEnt RL, the optimal policy is a softmax or Boltzmann distribution over the expected future return of state-action pairs.

This perspective of MaxEnt RL allows us to design SAAC which transforms the robust RL into an adversarial game in the softmax policy space. MaxEnt RL is widely used in solving complex RL problems as: it enhances exploration (Haarnoja, Zhou, Abbeel, et al., 2018), it

transforms the optimal control problem in RL into a probabilistic inference problem (Todorov, 2007; Toussaint, 2009), and it modifies the optimization problem by smoothing the value function landscape (Williams and Peng, 1991; Ahmed, Le Roux, Norouzi, et al., 2019).

Soft Actor-Critic (SAC) (Haarnoja, Zhou, Abbeel, et al., 2018). Specifically, we use the SAC framework to solve the MaxEnt RL problem. Following the actor-critic methodology, SAC uses two components, an actor and a critic, to iteratively maximize $V_\pi(s) + \mathcal{H}_\pi(a|s)$. Given the collection of transitions in a set \mathcal{D} , the critic minimizes the soft Bellman residual with a function approximation Q_ϕ :

$$J(Q_\phi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\phi(s_t, a_t) - \left(\mathcal{R}(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_t, a_t)} [V_{\bar{\phi}}(s_{t+1})] \right) \right)^2 \right], \quad (9.2)$$

where $V_{\bar{\phi}}(s_t) \triangleq \mathbb{E}_{a_t \sim \pi} [Q_{\bar{\phi}}(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$. Equation (9.2) makes use of a target soft Q-function with parameters $\bar{\phi}$ obtained using an exponentially moving average of the soft Q-function parameters ϕ . Mnih, Kavukcuoglu, Silver, et al. (2015) has demonstrated this technique stabilizes training. Given the Q_ϕ , the actor learns the policy parameters θ by minimizing $J(\pi_\theta)$:

$$J(\pi_\theta) = \mathbb{E}_{s_t \sim \mathcal{D}} [\mathbb{E}_{a_t \sim \pi_\theta} [\alpha \log(\pi_\theta(a_t|s_t)) - Q_\phi(s_t, a_t)]] . \quad (9.3)$$

Here, α is called entropy temperature, and regulates the relative importance of the entropy term versus the reward and produces better results. We use the version of SAC with an automatic temperature tuning scheme for α (Haarnoja, Zhou, Hartikainen, et al., 2018).

9.2.3 Safe RL

Risk Measure for Safety. Safe or risk-sensitive RL with MDPs is first considered in Howard and Matheson (1972), where they aim to maximize an exponential utility function over the cumulative reward: $V_\pi(s|\lambda) = \lambda^{-1} \log \mathbb{E}[\exp(\lambda Z_\pi^T(s))]$. This is equivalent to maximizing $V_\pi(s) + \lambda \mathbb{V}[Z_\pi^T(s)]$, such that the high variance in return is penalized for $\lambda < 0$ and encouraged for $\lambda > 0$. Though this approach of using exponential utility in risk-sensitive discrete MDPs dominates the initial phase of safe RL research (Marcus, Fernández-Gaucheraud, Hernández-Hernandez, et al., 1997; Coraluppi and Marcus, 1999; Garcia and Fernández, 2015), with the invent of coherent risks (Artzner, Delbaen, Eber, et al., 1999)², researchers have looked into other risk measures, such as Conditional Value-at-Risk (CVaR)³ (Chow, Tamar, Mannor, et al., 2015). Also, application of RL to large scale problems (Chow and Ghavamzadeh, 2014; Chow, Ghavamzadeh, Janson, et al., 2017), tried to make the algorithms scalable and to extend to the

²Variance is not a coherent risk but standard deviation is.

³CVaR $_\lambda$ quantifies expectation of the lowest $\lambda\%$ of a probability distribution (Rockafellar and Uryasev, 2000).

continuous MDPs (Ray, Achiam, and Amodei, 2019). Our approach is flexible to consider all these risk measures and both discrete and continuous MDP settings.

Safe Exploration. Another approach is to consider a part of the state-space to be “safe” and constrain the RL algorithm to explore inside it with high probability. Geibel and Wysotzki (2005) considered a subset of terminal states as “error” states $\mathcal{E} \subseteq \mathcal{S}$ and developed a constrained MDP problem to avoid reaching it:

$$\arg \max_{\pi} V_{\pi}(s) \text{ s.t. } \forall s \in \mathcal{S} \setminus \mathcal{E}, \rho_{\pi}(s) \leq \delta. \quad (9.4)$$

Here, $\rho_{\pi}(s)$ is the total number of times the agent goes to the terminal error states \mathcal{E} and $\delta > 0$ is a certain threshold. Due to the existence of these error states, even a policy with low variance can produce large risks (*e.g.* falls or accidents) (Ray, Achiam, and Amodei, 2019). As mentioned in Prashanth and Fu (2018) and Chow and Ghavamzadeh (2014), safety constraints can be adopted to develop a constrained MDP (Altman, 1999) formulation of risk-sensitive RL.

9.3 Problem Formulation: Safe RL as a Non-Zero Sum Game

Safe RL as Constrained MDP (CMDP). All of the aforementioned methods to safe RL can be expressed as a CMDP problem that aims to maximize the value function V_{π} of a policy π while constraining the total risk ρ_{π} below a certain threshold δ :

$$\arg \max_{\pi} V_{\pi}(s) \text{ s.t. } \rho_{\pi}(s) \leq \delta \text{ for } \delta > 0. \quad (9.5)$$

- If Mean-Standard Deviation (MSD) (Prashanth and Ghavamzadeh, 2016) is the risk measure, $\rho_{\pi}(s) \triangleq \mathbb{E} \left[Z_{\pi}^T(s) | \pi, s_0 = s \right] + \lambda \sqrt{\mathbb{V} \left[Z_{\pi}^T(s) | \pi, s_0 = s \right]}$ ($\lambda < 0$).
- If CVaR is the risk measure, $\rho_{\pi}(s) \triangleq \text{CVaR}_{\lambda} \left[Z_{\pi}^T(s) | \pi, s_0 = s \right]$ for $\lambda \in [0, 1)$.
- For the safe exploration constraint of staying in the “non-error” states $\mathcal{S} \setminus \mathcal{E}$, we choose $\rho_{\pi}(s) \triangleq \mathbb{E} \left[\sum_{t=0}^T \mathbb{1}(s_{t+1} \in \mathcal{E}) | \pi, s_0 = s \in \mathcal{S} \setminus \mathcal{E} \right] = \sum_{t=0}^T \mathbb{P}_{\pi} [s_{t+1} \in \mathcal{E}]$ such that $s_0 = s$ is a non-error state.

CMDP as a Non-Zero Sum (NZS) Game. The most common technique to address the constraint optimization in Equation (9.5) is formulating its Lagrangian.

$$\mathcal{L}(\pi, \beta_0) \triangleq V_{\pi}(s) - \beta_0 \rho_{\pi}(s), \text{ for } \beta_0 \geq 0. \quad (9.6)$$

For $\beta_0 = 0$, this reduces to its risk-neutral counterpart. Instead, as $\beta_0 \rightarrow \infty$, this reduces to the unconstrained risk-sensitive approach. Thus, the choice of β_0 is important. We automatically tune it as described in Section 9.4.3.

Now, the important question is to estimate the risk function $\rho_\pi(s)$. Researchers have either solved an explicit optimization problem to estimate the parameter or subspace corresponding to the risk measure, or used a stochastic estimator of the risk gradients. These approaches are poorly scalable and lead to high variance estimates as there is no provably convergent CVaR estimator in RL settings. In order to circumvent these issues, we deploy *an adversary* that aims to maximize the cumulative risk $\rho_\pi(s)$ given the same initial state s and trajectory τ as *the agent* maximizing Equation (9.6) and use it as a proxy for the risk constraint:

$$\begin{aligned} \theta^* &\triangleq \arg \max_{\theta} \mathcal{L}(\theta, \beta_0) = V_{\pi_\theta}(s) - \beta_0 V_{\pi_\omega}(s), \\ \omega^* &\triangleq \arg \max_{\omega} V_{\pi_\omega}(s). \end{aligned} \tag{9.7}$$

Here, we consider that the policies of the agent and the adversary are parameterized by θ and ω respectively. The value function of the adversary $V_{\pi_\omega}(s)$ is designed to estimate the corresponding risk $\rho_\pi(s)$. This is a non-zero sum game (NZS) as the objectives of the adversary and the agent are not the same and does not sum up to 0. Following this formulation, any safe RL problem expressed as a CMDP (Equation (9.5)), can be reduced to a corresponding agent-adversary non-zero sum game (Equation (9.7)). The adversary tries to maximize the risk, and thus to shrink the feasibility region of the agent’s value function. The agent tries to maximize the regularized Lagrangian objective in the shrunken feasibility region. We refer to this duelling game as *Risk-sensitive Non-zero Sum (RNS)* game.

Given this RNS formulation of Safe RL problems, we derive a MaxEnt RL equivalent in the next section, which naturally leads to a dueling soft actor-critic algorithm (SAAC) for executing safe RL tasks.

9.4 SAAC: Safe Adversarial Soft Actor-Critics

In this section, we first derive a MaxEnt RL formulation of the Risk-sensitive Non-zero Sum (RNS) game. We show that this naturally leads to a duel between the adversary and the agent in the policy space. Following that, we elaborate the generic architecture of SAAC, and the details of designing the risk-seeking adversary for different risk constraints. We conclude the section with a note on automatic adjustment of regularization parameters.

9.4.1 Risk-sensitive Non-zero Sum (RNS) Game with MaxEnt RL

In order to perform the RNS game with MaxEnt RL, we substitute the value functions in Equation (9.7) with corresponding soft Q-functions. Thus, the adversary's objective is maximizing:

$$\mathbb{E}_{\pi_\omega}[Q_\omega(s, \cdot)] + \alpha_0 \mathcal{H}_{\pi_\omega}(\pi_\omega(\cdot|s))$$

for $\pi_\omega \in \Pi_\omega$, and the agent's objective is maximizing:

$$\mathbb{E}_{\pi_\theta}[Q_\theta(s, \cdot)] + \alpha_0 \mathcal{H}_{\pi_\theta}(\pi_\theta(\cdot|s)) - \beta_0 (\mathbb{E}_{\pi_\theta}[Q_\omega(s, \cdot)] + \alpha_0 \mathcal{H}_{\pi_\omega}(\pi_\omega(\cdot|s))) \quad (9.8)$$

for $\pi_\theta \in \Pi_\theta$. Following the equivalent KL-divergence formulation in policy space, the adversary's objective is:

$$\omega^* = \arg \min_{\omega} D_{\text{KL}} \left(\pi_\omega(\cdot|s) \parallel \exp(\alpha_0^{-1} Q_\omega(s, \cdot)) / Z_\omega(s) \right). \quad (9.9)$$

Similarly, the agent's objective is:

$$\begin{aligned} \theta^* &= \arg \max_{\theta} \mathbb{E}_{\pi_\theta}[Q_\theta(s, \cdot)] + \alpha_0(1 + \beta_0) \mathcal{H}_{\pi_\theta}(\pi_\theta(\cdot|s)) \\ &\quad + \alpha_0 \beta_0 \mathbb{E}_{\pi_\theta}[\log(\pi_\omega(\cdot|s)) - \log \exp[\alpha_0^{-1} Q_\omega(s, \cdot)]] + \alpha_0 \beta_0 D_{\text{KL}}(\pi_\theta(\cdot|s) \parallel \pi_\omega(\cdot|s)) \\ &= \arg \min_{\theta} D_{\text{KL}} \left(\pi_\theta(\cdot|s) \parallel \exp((\alpha_0(1 + \beta_0))^{-1} Q_\theta(s, \cdot)) / Z_\theta(s) \right) \\ &\quad - \alpha_0 \beta_0 \mathbb{E}_{\pi_\theta}[\log(\pi_\omega(\cdot|s)) - \log \exp[\alpha_0^{-1} Q_\omega(s, \cdot)]] - \alpha_0 \beta_0 D_{\text{KL}}(\pi_\theta(\cdot|s) \parallel \pi_\omega(\cdot|s)) \\ &= \arg \min_{\theta} D_{\text{KL}} \left(\pi_\theta(\cdot|s) \parallel \exp(\alpha^{-1} Q_\theta(s, \cdot)) / Z_\theta(s) \right) - \beta D_{\text{KL}}(\pi_\theta(\cdot|s) \parallel \pi_{\omega^*}(\cdot|s)). \end{aligned} \quad (9.10)$$

Here, $\alpha = \alpha_0(1 + \beta_0)$ and $\beta = \alpha_0 \beta_0$. The last equality holds true as the adversary's optimal policy $\pi_{\omega^*}(\cdot|s) = \exp(\alpha_0^{-1} Q_{\omega^*}(s, \cdot)) / Z_{\omega^*}(s)$, and since the optimization is over θ , adding $\log Z_\omega(s)$ does not make a change.

Following this reduction, we observe that performing the RNS game with MaxEnt RL is equivalent to performing the traditional MaxEnt RL for adversary with a risk-seeking Q-function Q_ω , and a modified MaxEnt RL for the agent that includes the usual soft Q-function and a KL-divergence term repulsing the agent's policy π_θ from the adversary's policy π_ω . This behaviour of RNS game in policy space allows to propose a duelling soft actor-critic algorithm, namely SAAC, to solve risk-sensitive RL problems.

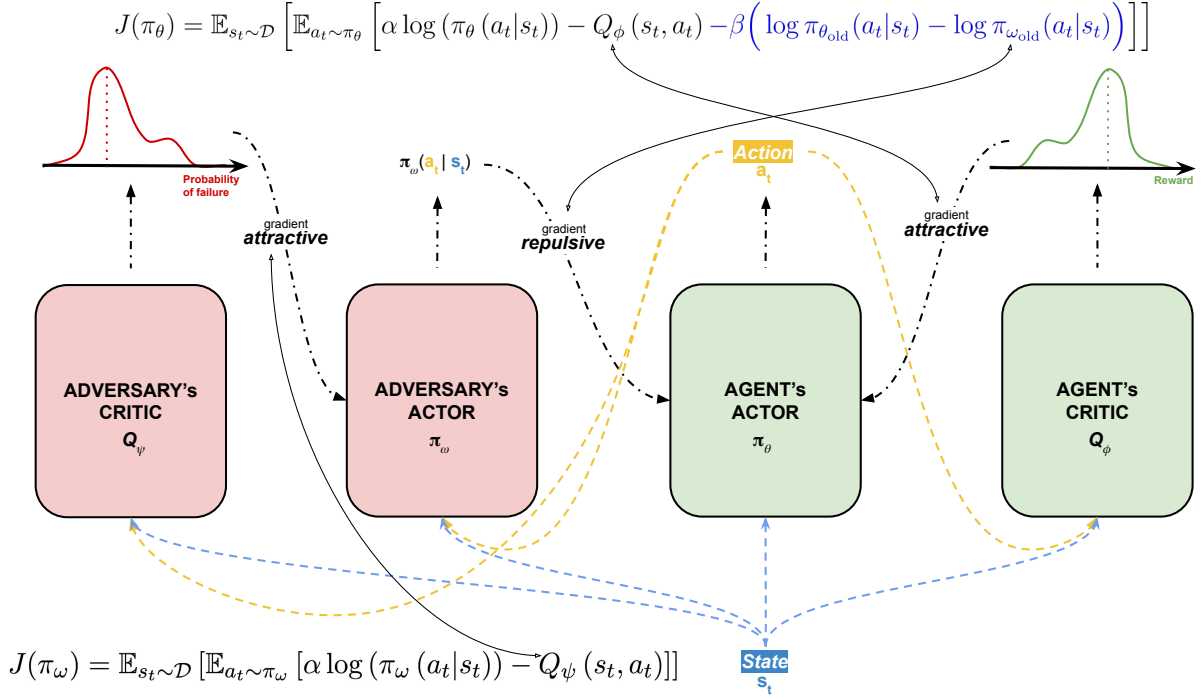


Figure 9.1 – Schematic overview of the Safe Adversarially guided Actor-Critics (SAAC) algorithm.

9.4.2 The SAAC Algorithm

We propose an algorithm SAAC to solve the objective of the agent (Equation (9.10)) and of the adversary (Equation (9.9)). In SAAC, we deploy two soft actor-critics (SACs) to enact the agent and the adversary respectively. We illustrate the schematic of SAAC in Figure 9.1.

As a building block for SAAC, we deploy the recent version of SAC (Haarnoja, Zhou, Abbeel, et al., 2018) that uses two soft Q-functions to mitigate positive bias in the policy improvement step in Equation (9.3), which was encountered in van Hasselt (2010) and Fujimoto, Hoof, and Meger (2018). In the design of SAAC, we introduce two new ideas: an off-policy deep actor-critic algorithm within the MaxEnt RL framework and a Risk-sensitive Non-zero Sum (RNS) game. SAAC engages the agent in safer strategies while finding the optimal actions to *maximize* the expected returns. The role of the adversary is to find a policy that maximizes the probability of breaking the constraints given by the environment. The adversary is trained online with off-policy data given by the agent. We denote the parameter of the adversary policy using ω (resp. ω_{old} the parameter at the previous iteration). For each sequence of transition from the replay buffer \mathcal{D} , the adversary should find actions that minimize the following loss:

$$J(\pi_\omega) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\omega} \left[\alpha \log(\pi_\omega(a_t|s_t)) - Q_\psi(s_t, a_t) \right] \right].$$

Use Repulsive Priors to motivate risk-sensitive policies

Finally, leveraging the RNS-based reduced objective, SAAC makes the agent’s actor minimize $J(\pi_\theta)$:

$$J(\pi_\theta) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\theta} \left[\alpha \log(\pi_\theta(a_t|s_t)) - Q_\phi(s_t, a_t) - \beta \left(\log \pi_{\theta_{\text{old}}}(a_t|s_t) - \log \pi_{\omega_{\text{old}}}(a_t|s_t) \right) \right] \right].$$

In blue is the repulsion term introduced by SAAC. The method alternates between collecting samples from the environment with the current agent’s policy and updating the function approximators, namely the adversary’s critic Q_ψ , the adversary’s policy π_ω , the agent’s critic Q_ϕ and the agent’s policy π_θ . It performs stochastic gradient descent on the corresponding loss functions with batches sampled from the replay buffer \mathcal{D} . We provide a generic description of SAAC in Algorithm 6.

Algorithm 6 SAAC.

- 1: **Input parameters:** $\tau, \lambda_Q, \lambda_\pi, \lambda_\alpha, \lambda_\beta$
 - 2: **Initialize** adversary’s and agent’s policies and Q-functions parameters ω, ψ, θ and ϕ
 - 3: **Initialize** temperature parameters α and β
 - 4: $\mathcal{D} \leftarrow \emptyset$
 - 5: **for** each iteration **do**
 - 6: **for** each step **do**
 - 7: $a_t \sim \pi_\theta(a_t|s_t)$
 - 8: $s_{t+1} \sim \mathcal{P}(s_t, a_t)$
 - 9: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
 - 10: **for** each gradient step **do**
 - 11: sample batch \mathcal{B} from \mathcal{D}
 - 12: $\psi \leftarrow \psi - \lambda_Q \hat{\nabla}_\psi J_Q(\psi)$ } Update Adversary
 - 13: $\omega \leftarrow \omega - \lambda_\pi \hat{\nabla}_\omega J(\pi_\omega)$ }
 - 14: $\beta \leftarrow \beta - \lambda_\beta \hat{\nabla}_\beta J(\beta)$ }
 - 15: $\bar{\psi} \leftarrow \tau \psi + (1 - \tau) \bar{\psi}$ }
 - 16: $\phi \leftarrow \phi - \lambda_Q \hat{\nabla}_\phi J_Q(\phi)$ } Update Agent
 - 17: $\theta \leftarrow \theta - \lambda_\pi \hat{\nabla}_\theta J(\pi_\theta)$ }
 - 18: $\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha J(\alpha)$ }
 - 19: $\bar{\phi} \leftarrow \tau \phi + (1 - \tau) \bar{\phi}$ }
-

9.4.3 Automating Adversarial Adjustment

Similar to the solution introduced in Haarnoja, Zhou, Abbeel, et al. (2018), the adversary temperature β and the entropy temperature α are automatically adjusted. Since the adversary bonus can differ across tasks and during training, a fixed coefficient would be a poor solution. We use \bar{A} to denote the adversary’s bonus target, which is a hyperparameter in SAAC. By formulating a constrained optimization problem where the KL-divergence between the agent

and the adversary is constrained, β is learned by gradient descent with respect to:

$$J(\beta) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\log \beta \cdot \left(D_{\text{KL}}(\pi_\theta(\cdot|s_t) || \pi_\omega(\cdot|s_t)) - \bar{\mathcal{A}} \right) \right].$$

In addition, we use $\bar{\mathcal{H}}$ as the target entropy (a hyperparameter needed in SAC) and learn the entropy temperature α by taking a gradient step with respect to the loss:

$$J(\alpha) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\log \alpha \cdot \left(-\log \pi_\theta(a_t|s_t) - \bar{\mathcal{H}} \right) \right].$$

9.4.4 Designing the Risk-Seeking Adversary

9.4.4.1 Subspace Constraints

At every step, the environment signals whether the constraints have been satisfied or not. We construct a reward signal based on this information. This constraint reward, denoted as r_c , is 1 if all the constraints have been broken, and 0 otherwise. $J(Q_\psi)$ is the soft Bellman residual for the critic responsible with constraint satisfaction:

$$J(Q_\psi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\psi(s_t, a_t) - (r_c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \rho} [V_{\bar{\psi}}(s_{t+1})]) \right)^2 \right], \quad (9.11)$$

with $V_{\bar{\psi}}(s_t) = \mathbb{E}_{a_t \sim \pi_\omega} [Q_{\bar{\psi}}(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$. Equation 9.11 also makes use of a target soft Q-function with parameters $\bar{\psi}$ obtained using an exponentially moving average of the soft Q-function parameters ψ . The loss functions for the agent's critic (Equation 9.2) and the adversary's critic (Equation 9.11) are similar in essence. They aim to approximate Q-values obtained from the extrinsic rewards (the task's rewards) and the constraint rewards (whether the constraints have been violated or not) respectively. We denote SAAC-Cons as the method incorporating constraints in SAAC.

9.4.4.2 Coherent Risk Measures

We also design adversary critics to consider two coherent risk measures: Mean-Standard Deviation (MSD) and CVaR.

Mean-Standard Deviation (MSD). In this case, we consider optimizing a Mean-Standard Deviation risk (Prashanth and Ghavamzadeh, 2016), which we estimate using:

$$Q_\psi(s, a) = Q_\phi(s, a) + \lambda \sqrt{\mathbb{V}[Q_\phi(s, a)]}.$$

Use Repulsive Priors to motivate risk-sensitive policies

In the equation above, $\lambda < 0$ is a hyperparameter that dictates the lower $\lambda - \text{SD}$ considered to represent the lower tail. In the experiments, we use $\lambda = -1$. In practice, we approximate the variance $\mathbb{V}[Q_\phi(s, a)]$ using the state-action pairs in the current batch of samples. We refer to the associated method as SAAC-MSD.

Conditional Value-at-Risk (CVaR). In this case, when given a state-action pair (s, a) , the Q-value distribution is approximated by a number of quantile values at quantile fractions (Eriksson, Basu, Alibeigi, et al., 2021). Let $\{\tau_i\}_{i=0, \dots, N}$ denote an ensemble of quantile fractions, which satisfy $\tau_0 = 0, \tau_N = 1, \tau_i < \tau_j \forall i < j, \tau_i \in [0, 1] \forall i = 0, \dots, N$, and $\hat{\tau}_i = (\tau_i + \tau_{i+1})/2$. If $Z^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{Z}$ denotes the soft action-value of policy π ,

$$Q_\psi(s, a) = - \sum_{i=0}^{N-1} (\tau_{i+1} - \tau_i) g'(\hat{\tau}_i) Z_{\hat{\tau}_i}^{\pi_\theta}(s, a; \phi),$$

with $g(\tau) = \min\{\tau/\lambda, 1\}$, where $\lambda \in (0, 1)$. In the experiments, we consider $\lambda = 0.25$, *i.e.* we truncate the right tail of the distribution approximation by dropping 75% of the topmost atoms. We call the associated method SAAC-CVaR.

9.5 Experimental Study

Experimental Setup. First, we compare some possible variants of our method. Indeed, as presented in Section 9.4.4, the adversary has different quantifications of risk to fulfill the objective of finding actions with high probability of breaking the constraints: SAAC-Cons, SAAC-CVaR, and SAAC-MSD. Then, we compare our method with best performing competitors in continuous control problems: SAC (Haarnoja, Zhou, Abbeel, et al., 2018) and TQC (Kuznetsov, Shvechikov, Grishin, et al., 2020). TQC builds on top of C51 (Bellemare, Dabney, and Munos, 2017) and QR-DQN (Dabney, Rowland, Bellemare, et al., 2018), and adapt the distributional RL methods for continuous control. Further, they apply truncation for the approximated distributions to control their overestimation and use ensembling on the approximators for additional performance improvement. Finally, we qualitatively compare the behavior of our risk-averse method with that of SAC, using state vectors collected during validation in test environments. Note that for all the experiments, the agents are trained for 1M timesteps and their performance is evaluated at every 1000-th step.

Similar to TQC, we implement SAAC on top of SAC and choose to automatically tune the adversary temperature β (Section 9.4.3) and the entropy temperature α . Last but not least, using SAAC on top of SAC introduces only one hyperparameter: the learning rate for the automatic tuning of β . All the other hyperparameters are the same as for SAC and are available for

Table 9.1 – Comparison of SAAC variants.

Method	Efficiency (xSAC)	# Failures $\pm\sigma$
SAC	$\times 1$	65.88 ± 17.25
SAAC-Cons	$\times 1.33$	48.66 ± 15.99
SAAC-CVaR	$\times 2.02$	54.39 ± 15.37
SAAC-MSD	$\times \mathbf{2.21}$	$\mathbf{19.31 \pm 3.02}$

Table 9.2 – In *quadruped-upright-walk*.

Method	Efficiency (xSAC)	# Failures $\pm\sigma$
SAC	$\times 1$	8443.93 ± 696.47
TQC	$\times 0.97$	8297.63 ± 697.88
TQC-CVaR	$\times 1.03$	6298.33 ± 1078.50
SAAC-MSD	$\times \mathbf{1.19}$	$\mathbf{4632.80 \pm 657.35}$

Table 9.3 – In *quadruped-joint-walk*.

Method	Efficiency (xSAC)	# Failures $\pm\sigma$
SAC	$\times 1$	12583.43 ± 997.29
TQC	$\times 1.07$	11738.57 ± 995.62
TQC-CVaR	$\times 1.05$	9015.82 ± 1011.31
SAAC-MSD	$\times \mathbf{1.27}$	$\mathbf{8069.45 \pm 803.42}$

consultation in Haarnoja, Zhou, Abbeel, et al. (2018, Appendix D). For TQC, we employ the same hyperparameters as reported in Kuznetsov, Shvechikov, Grishin, et al. (2020).

Description of Environments. To validate the framework of RNS Game with MaxEnt RL, we conduct a set of experiments in the DM control suite (Tassa, Doron, Muldal, et al., 2018). More specifically, we use the real-world RL challenge⁴ (Dulac-Arnold, Levine, Mankowitz, et al., 2020), which introduces a set of real-world inspired challenges. As was developed at the beginning of this chapter, we are particularly interested in tasks where a set of constraints are imposed on existing control domains. In the following, we give a short description of the tasks and safety constraints used in the experiments, with their respective observation space (\mathcal{S}) and action space (\mathcal{A}) dimensions. First, *realworldrl-walker-walk* ($\mathcal{S} \times \mathcal{A} = 18 \times 6$) corresponds to the dm-control suite *walker* task with (i) joint-specific constrains on the joint angles to be within a range, and (ii) a constrain on the joint velocities to be within a range. Next, *realworldrl-quadruped-joint-walk* ($\mathcal{S} \times \mathcal{A} = 78 \times 12$) corresponds to the dm-control suite *quadruped* task with the same set of constraints as just described. *realworldrl-quadruped-upright-walk* has a constrain on the quadruped’s torso’s z-axis to be oriented upwards, and *realworldrl-quadruped-force-walk* limits foot contact forces when touching the ground.

9.5.1 Comparison between Risk Quantifiers of SAAC

First, we compare different variants of SAAC in the *realworldrl-walker-walk-returns* task. From Table 9.1 and Figure 9.2a (lines are average performances and shaded areas represent one standard deviation) we evaluate how our method affects the performance and risk aversion of agents.

⁴The code can be found here: github.com/google-research/realworldrl_suite

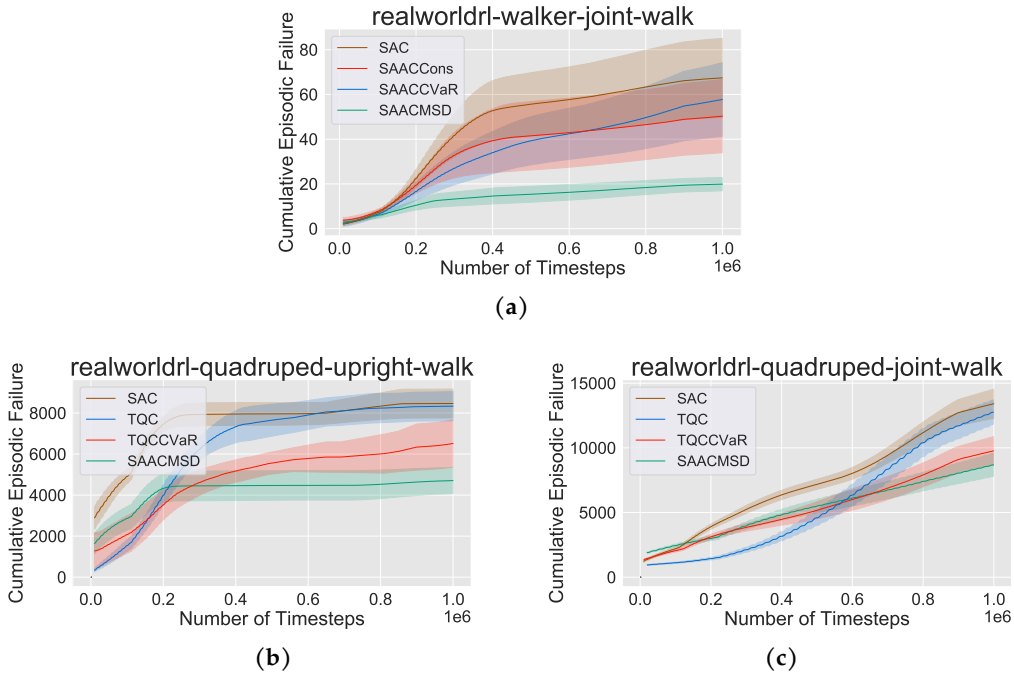


Figure 9.2 – Constraints: SAAC vs. baselines.

In addition to the rate at which the maximum average return is reached by each of the methods compared to SAC, we compare the cumulative number of failures of the agents (the lower the better). As expected, risk-sensitive agents such as SAAC decrease the probability of breaking safety constraints. Concurrently, they achieve the maximum average return with much higher sample efficiency, SAAC-MSD ahead. Henceforth, we use the SAAC-MSD version of our method to compare with the baselines.

9.5.2 Comparison of SAAC to Baselines

Now, we compare the best performing SAAC variant SAAC-MSD with SAC (Haarnoja, Zhou, Abbeel, et al., 2018), TQC (Kuznetsov, Shvechikov, Grishin, et al., 2020) and TQC-CVaR, *i.e.* an extension of TQC with 16% of the topmost atoms dropped (*cf.* (Kuznetsov, Shvechikov, Grishin, et al., 2020, Appendix B Table 6)) of all Q-function atoms. In Table 9.2 and Figure 9.2b, we evaluate SAAC-MSD in *realworldrl-quadruped-upright-walk*. In Table 9.3 and Figure 9.2c, we report the results for *realworldrl-quadruped-joint-walk*.

Table 9.3 shows that SAAC-MSD performs better than all other baselines both in terms of final performance and in terms of finding risk-averse policies. Moreover, although TQC-CVaR exhibits fewer number of failures over the course of learning, it performs slightly worse than its non-truncated counterpart TQC. Table 9.2 confirms the advantage of using SAAC-MSD as a risk-averse MaxEnt RL method over the baselines: overall using SAAC allows the agents to

achieve faster convergence using safer policies during training. Interestingly, TQC achieves the maximum score of the task a bit later than the SAC agent. Nevertheless, TQC-CVaR, its CVaR variant, opens the door for better sample efficiency score with much safer policies.

9.5.3 Visualization of Safer State Space Visitation

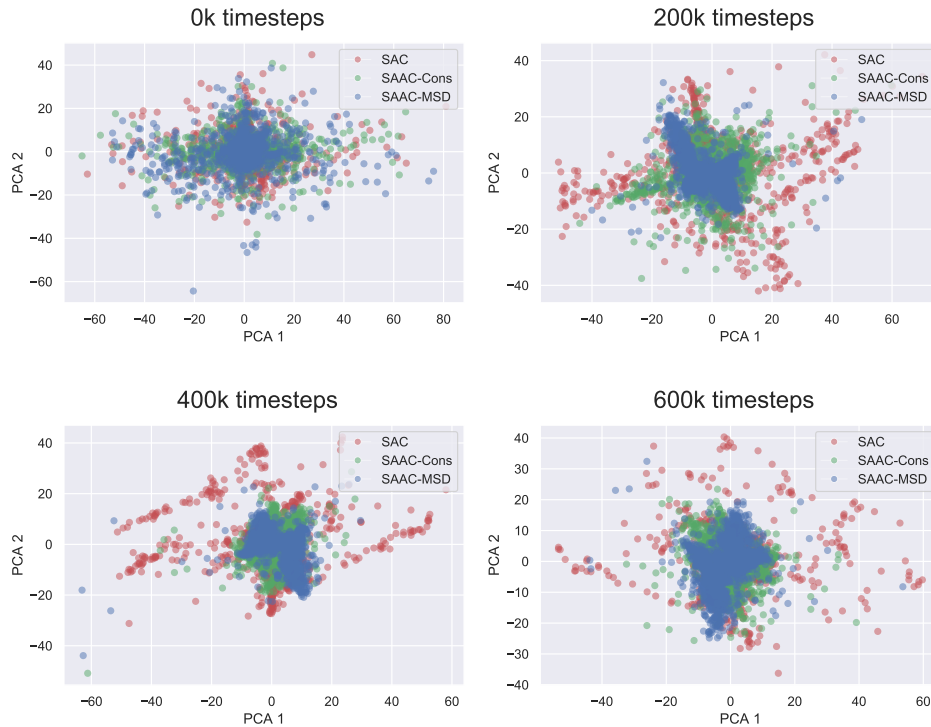


Figure 9.3 – Visualization of visited state space at different stages of learning in the *realworldrl-walker-walk* task.

In this experiment, we choose SAC, SAAC-Cons and SAAC-MSD to train a relatively wide spectrum of agents using the same experimental protocol as in Section 9.5.1 and 9.5.2 on the *realworldrl-walker-walk* task. We collect batches of states visited during the evaluation phase in a test environment at different stages of the training. The state vectors are projected from a 18-D space to a 2-D space using PCA. We present the results in Figure 9.3. At the beginning of training, there is no clear distinction in terms of explored state regions, as the learning has not begun yet. On the contrary, during the 200k-600k timesteps range, there is a significant difference in terms of state space visitation. In light of the cumulative number of failures shown in Figure 9.2a, the results suggest that SAC engages in actions leading to more unsafe states. Conversely, SAAC seems to successfully constraint the agent to safe regions.

9.6 Related Work

Safe RL. Performing risk-sensitive or safe RL requires considering either the worst-case outcomes (Heger, 1994; Nilim and El Ghaoui, 2005; Tamar, Xu, and Mannor, 2013) or risk measures, such as CVaR (Chow and Ghavamzadeh, 2014) and variance (Prashanth and Ghavamzadeh, 2016), computed on $Z_{\pi}^T(s)$. In this chapter, we are only interested in the stochastic setup, and thus in the risk measures than the worst-case scenarios. (Tamar, Xu, and Mannor, 2013; Chow, Ghavamzadeh, Janson, et al., 2017) have tried to solve expensive optimization problems to design safe RL algorithms with such risk measure-based constraints. Since these methods are slow and often limited to discrete MDPs, Borkar (2002), Garcia and Fernández (2015), Kuznetsov, Shvechikov, Grishin, et al. (2020), and Eriksson, Basu, Alibeigi, et al. (2021) have tried to estimate the risk by approximating the whole return distribution. With the advent of distributional RL (Bellemare, Dabney, and Munos, 2017; Dabney, Rowland, Bellemare, et al., 2018), this approach has propelled but accurately estimating the return distribution often requires ensembles of deep Q-networks or quantile regressors. This can make the computation heavy and expensive. Alternatively, escaping such issues requires parametric assumptions on the return distribution (Tang and Kucukelbir, 2017). We avoid this issue by using an adversarial soft actor-critic to seek risk and to estimate it. Additionally, there are other families of method that aim to avoid “unsafe” parts of the state space. They use constraints to avoid exploring such “unsafe” states. The RL algorithms developed to do so are typically different in functionality than the risk measure-based safe RL algorithms. SAAC uses a unified RNS game based formulation to address both types of safety constraints.

Minimax Games for Safe RL. Game theoretic frameworks have been studied in different formulations of safe RL. Specifically, Pinto, Davidson, Sukthankar, et al. (2017) proposed the Robust Adversarial RL (RARL) framework where the reward and the transition depends not only on the agent’s actions but also on the adversary’s. Thus, the adversary essentially perturbs the rewards and transitions either at every step (Mandlekar, Zhu, Garg, et al., 2017; Pattanaik, Tang, Liu, et al., 2017) or at every m steps (Pan, Seita, Gao, et al., 2019) that leads to zero sum Markov game formulation for safe RL analogous to the robust control (Keel, Bhattacharyya, and Howze, 1988). This setting is fundamentally different than the risk measure and subspace constraint based settings that we consider. Recently, Zhang, Yang, and Wang (2021) extended the zero sum Markov game based formulation to exponential utility based risk-sensitive RL. Their work is specific to linear-quadratic control, whereas it is known that solving such zero sum minimax games with dynamic programming has exponential complexity in the number of actions (Perolat, Scherrer, Piot, et al., 2015). In this chapter, we differ from the traditional zero sum game formulation, which is restrictive to include the risk measures and subspace constraints. In our knowledge, SAAC is the first method to propose a non-zero sum game

formulation of risk-sensitive RL that provides a flexibility to design and update the risk-seeking adversary.

MaxEnt RL for Safety. We adopt the MaxEnt RL framework to reduce RNS games to a duelling SAC game between an adversary and an agent with repulsive policies, and thus to develop the SAAC algorithm. Recently, Eysenbach and Levine (2021) have first shown that MaxEnt RL can be used to enhance robustness against adversarial perturbations in rewards and transitions. But in best of our knowledge, we are the first to develop MaxEnt RL algorithm to address the safe RL problem under risk measures and subspace constraints.

Chapter conclusion

In this chapter, we touched on the problem of risk-sensitive RL under safety constraints and coherent risk measures. We proposed that maximizing the value function under risk or safety constraints is equivalent to playing a risk-sensitive non-zero sum (RNS) game. In the RNS game, an adversary tries to maximize the risk of a decision trajectory while the agent tries to maximize a weighted sum of its value function given the adversary’s feedback. Specifically, under the MaxEnt RL framework, this RNS game reduces to deploying two soft-actor critics for the agent and the adversary while accounting for a repulsion term between their policies. This allowed us to formulate a duelling SAC-based algorithm, called SAAC. We instantiated our method for subspace, mean-standard deviation, and CVaR constraints, and also experimentally tested it on various continuous control tasks. This algorithm translates into better risk-sensitive performance than SAC and the risk-sensitive distributional RL baselines in all these environments.

Part conclusion

In this part conclusion, we review again the problems and questions opened up in Chapter 1 that we have proposed to address. In environments where rewards are non-zero at most timesteps and where the layouts are procedurally-generated at each episode, noise-based exploration strategies and to some extent count-based exploration tend to be insufficient and inadequate. We have developed a general framework for discrete and continuous action spaces to maintain an adversarial prior of a mixture of previous policies from which the agent should distance itself from. This method successfully helps the RL agent engage in *conservatively* diversified policies and extends the state-of-the-art in tasks where efficient-exploration is a bottleneck (Chapter 8). Finally, instead of reasoning on the need to explore states that have not yet been discovered, we employ the framework developed in Chapter 8 from the perspective of the requirement to remain in a safe region of the state space. To that end, the adversary learns how to break safety constraints and results in the representation of a probabilistic unsafe region which the agent should avoid (Chapter 9).

Part IV

Conclusion

Chapter 10

General Conclusion and Perspectives

*Experience is not what happens to you;
it's what you do with what happens to you.*

Aldous Huxley (1932).

10.1 Epilogue

In this thesis, we proposed a range of solutions for building more sample-efficient RL methods that push the boundaries in problems with continuous states and action spaces, including in a setting with safety compliance needs, and extended the scope of capabilities of RL agents in environments of hard-exploration in which agents must have the ability to generalize to unseen scenarios. These methods have been built on two tracks that come together under the umbrella of the actor-critic framework and the derivation of variance. The first track, in Part II, concerns the use of variance in the value function estimates while the second track, in Part III, regards the distributional distance between the agent's policy and some adversarial distribution representing a mixture of either previous policies or risk-seeking policies. The following is a reminder of the issues we have highlighted and a review of the solutions proposed to address them.

Alternative statistics of performance. The optimization and evaluation of RL methods is generally based solely on the sum of future rewards. In Chapter 4, we developed a general framework theoretically applicable to any actor-critic algorithm or environment, which relies on a set of auxiliary losses to be integrated into the learning process. In addition to metrics of accurate expectation (future states prediction), we introduced the fraction of variance explained by the value function, equivalent to the coefficient of determination. This variance-based metric

indicates the performance of the supervised learning task that the value estimator is trying to solve.

Filter misinformative data. It is reasonable to assume that the use of non-informative or misinformative transitions can only mislead the learning process and waste computational time. Indeed, an RL agent may learn more effectively from some transitions than others. In Chapter 5, we evaluated the simple yet effective idea of filtering some of the transitions that would otherwise be used in estimating the policy gradient. As a criterion for this transition filtering, we use again the same (explained) variance in the value function estimates metric defined in Chapter 4, and filter out the variance-neutral samples.

Critics optimize the residual variance. We have seen that previous studies empirically demonstrate that while the value network succeeds in the supervised learning task of fitting the empirical value of a state or state-action pair, it does not fit the actual state or state-action value. In addition, the variance reduction methods studied do not actually reduce the variance. With the objective of proposing a more efficient and robust objective function for estimating the critic, or at least to opening the door to additional work on the subject, Chapter 6 develops a technique to learn value functions in actor-critic methods using the residual variance as an objective function. In addition, we have studied the better estimations of the value function in case of continuous control sparse reward tasks where a greater sensitivity to extreme values and rare signals corresponding to rewards is beneficial.

Actors are repulsed by their past policies. In the challenging environments introduced in Chapter 7 in which agents are confronted with partially-observable and procedurally-generated tasks, simple exploration induced by stochastic policies or count-based techniques is not sufficient to discover and learn from highly-sparse rewards. In Chapter 8, we propose a method with dynamics of attraction and repulsion between the actor and an adversary. The approach leads the actor's policy to increase its distance from a mixture of previous policies, which incidentally adds to the variance of successive candidate policies.

Actors are repulsed by their risk-seeking alter-ego. In continuous control tasks whose safety features inspired by the real-world constraints, it may be challenging to restrict the agent to a safe region or to quantify a risk-measure to be calculated on the cumulative return relevant to the task. For instance, the safe region may not be clear to delineate. In Chapter 9, we combined the two ideas of safe region and risk-measure by representing the unsafe region using an adversary derived from the method in Chapter 8. Instead of learning a mixture of previous trajectories, the adversary learns to break the safety constraints and successfully represents a probabilistic unsafe region that the agent must avoid. The actor maximizes its divergence from

the adversary’s distribution, resulting in increased sample efficiency, better performance and safer trajectories.

10.2 Frontiers

Recent work in the field of deep reinforcement learning, including this thesis, address many challenges, namely, how to develop reinforcement learning algorithms that are sample efficient, scalable, and reliable. In the following, we take more liberty in imagining what would be the points of difficulty to overcome in order to go even further in the practicality and reusability of the methods developed and their application in even more complex environments, with the real world as focal point.

Reusing representations. Each time a new agent is trained to accomplish a certain task on a given environment, RL practitioners usually start from scratch and use randomly initialized function approximation at the beginning of training. In image classification, generation, or natural language processing tasks, it is common practice to warm-up a particular predictive application with general-purpose pre-trained models¹. The predictive nature of deep RL approaches is an important reason for their success, but research on the topic of improving, sharing and reusing representations has yet to flourish despite the use of identical benchmarks by the community. Nevertheless, on a similar topic, inspiring work (Ha and Schmidhuber, 2018) propose the use of a pre-trained Variational Auto-Encoder (VAE) to encode each frame of a pixel-based environment into low dimensional latent vectors by following a random policy. Such “world models” have initiated several other work (Oh, Guo, Lee, et al., 2015; Hafner, Lillicrap, Fischer, et al., 2019; Kaiser, Babaeizadeh, Mišos, et al., 2020). We also see efforts to improve access to data collected from a diverse range of tasks (*e.g.* the *Atari* benchmark and the real-world RL challenge) with recent releases of offline data to help research on offline RL, increase reproducibility of experiments and attempt to address problems related to limited computational budget (Gulcehre, Wang, Novikov, et al., 2020).

Dynamically adjusting hyperparameters. The choice of adjusting a hyperparameter online during training is a research direction that is attracting some interest in deep RL. In Xu, Hasselt, and Silver (2018), it is used to adjust the discount factor and the length of bootstrapping intervals. In Haarnoja, Zhou, Hartikainen, et al. (2018), it is used to automatically adjust the entropy coefficient. Such technique allows to adaptively adjust critical parameters during training making hyperparameter search unnecessary, save time and computational resources, and allowing to propose more general methods that does not need to be tuned for each task.

¹Dozens of pre-trained models are freely available online for [computer vision](#) or [NLP applications](#).

General Conclusion and Perspectives

Admittedly, methods with adaptively calibrated hyperparameters often require an additional hyperparameter to learn that calibrated parameter, but instead of setting a fixed value, the surrogate parameter can be a relaxed lower bound for some metric (*e.g.* a lower-bound on the entropy term (Haarnoja, Zhou, Hartikainen, et al., 2018)). In the case of the AVEC method introduced in Chapter 6, we can for instance anticipate further work on a method to adaptively control bias and variance in the value estimate.

Closing the gap with reality. RL practitioners already have at their disposal a variety of games and environments (Todorov, Erez, and Tassa, 2012; Bellemare, Naddaf, Veness, et al., 2013; Coumans and Bai, 2016; Chevalier-Boisvert, Willems, and Pal, 2018; Dulac-Arnold, Levine, Mankowitz, et al., 2020), some of which we have used intensively in this thesis, that require learning a variety of behaviors, in high dimensionality, sometimes with delayed rewards, stochasticity, partial observability or non-stationary dynamics. Nevertheless, it seems that there is still a lot of work to be done in order to integrate the characteristics of our real systems into our simulations, which calls for the need to use and develop more benchmarks addressing more facets of the risk specifications of an AI model deployed in the real world, such as standardized criteria for evaluating the safety characteristics of models and risk assessment procedures.

Learning from models. The majority of current deep learning techniques are applicable in relatively complex environments only on narrow tasks such as continuous control. In order to attempt to solve tasks that require more flexibility, echoing the richness of the real world, some form of model of the world may be required. Moreover, the difficulties associated with lifelong learning reinforce the idea that memory-based or model-based RL is a solution that could catalyse the adaptability of current algorithms. As support for this point, transfer learning problems already benefit from a learned model (Grzes and Kudenko, 2009; Zhang, Satija, and Pineau, 2018) which can facilitate the acquisition of conceptual knowledge abstracted from the perceptual details from which a competence was learned. We also link model learning to learning goal-directed policies (Pong, Gu, Dalal, et al., 2018; Nasiriany, Pong, Lin, et al., 2019) and hierarchical RL approaches (Li, Wang, Tang, et al., 2019; Nachum, Tang, Lu, et al., 2019) since those methods maintain a model of supervision where a top-level supervisory control system uses RL to approximate utility based on experience. Now, the line between model-free and model-based techniques becomes blurred. The question then of whether or not to learn a model of the world could be circumvented by asking how to learn a model of the world without the risk of introducing bias due to a faulty model and without losing the advantages of model-free RL which rely entirely on online and historical ground-truth data?

All these technical questions are definitely part of the subjects on which I intend to concentrate on in my future research. I am also increasingly interested in the less technical but equally important short-term societal issues of the (often predictive) models found in production today.

Bibliography

- Ahmed, Zafarali, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans (Sept. 2019). Understanding the Impact of Entropy on Policy Optimization. **In** *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 151–160.
- Altman, Eitan (1999). *Constrained Markov decision processes*. Vol. 7. CRC Press.
- Amari, Shun-Ichi (1998). Natural Gradient Works Efficiently in Learning. *Neural Computation* 10.2, pp. 251–276.
- Andrychowicz, Marcin, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39.1, pp. 3–20.
- Antol, Stanislaw, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh (2015). Vqa: Visual question answering. **In** *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2425–2433.
- Arjovsky, Martin and Léon Bottou (2017). Towards principled methods for training generative adversarial networks. **In** *International Conference on Learning Representation*.
- Artzner, Philippe, Freddy Delbaen, Jean-Marc Eber, and David Heath (1999). Coherent measures of risk. *Mathematical finance* 9.3, pp. 203–228.
- Bahdanau, Dzmitry, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joëlle Pineau, Aaron Courville, and Yoshua Bengio (2017). An actor-critic algorithm for sequence prediction. **In** *International Conference on Learning Representations*.
- Bahdanau, Dzmitry, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette (2019). Learning to Understand Goal Specifications by Modelling Reward. **In** *International Conference on Learning Representations*.
- Baker, Monya (May 2016). 1,500 scientists lift the lid on reproducibility. *Nature* 533.7604, pp. 452–454.
- Barto, Andrew and Sridhar Mahadevan (Jan. 2003). Recent Advances in Hierarchical Reinforcement Learning. **en.** *Discrete Event Dynamic Systems* 13.1, pp. 41–77.
- Barto, Andrew, Richard Sutton, and Charles Anderson (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* 5, pp. 834–846.

Bibliography

- Beck, Amir and Marc Teboulle (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters* 31.3, pp. 167–175.
- Bellemare, Marc, Will Dabney, and Rémi Munos (June 2017). A Distributional Perspective on Reinforcement Learning. **In** *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 449–458.
- Bellemare, Marc, Yavar Naddaf, Joel Veness, and Michael Bowling (June 2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47, pp. 253–279.
- Bellemare, Marc, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos (2016). Unifying count-based exploration and intrinsic motivation. **In** *Advances in Neural Information Processing Systems*, pp. 1471–1479.
- Bellman, Richard and Robert Kalaba (1957). Dynamic programming and statistical communication theory. *National Academy of Sciences of the United States of America* 43.8, p. 749.
- Berkenkamp, Felix, Riccardo Moriconi, Angela Schoellig, and Andreas Krause (2016). Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. **In** *IEEE Conference on Decision and Control (CDC)*, pp. 4661–4666.
- Berner, Christopher, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.
- Bertsekas, Dimitri (1997). Nonlinear programming. *Journal of the Operational Research Society* 48.3, pp. 334–334.
- Borkar, Vivek (2002). Q-Learning for Risk-Sensitive Control. *Mathematics of Operations Research* 27.2, pp. 294–311.
- Bottou, Léon, Frank Curtis, and Jorge Nocedal (2018). Optimization methods for large-scale machine learning. *Siam Review* 60.2, pp. 223–311.
- Bradtke, Steven and Andrew Barto (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning* 22.1, pp. 33–57.
- Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). OpenAI Gym.
- Brown, George (Dec. 1947). On Small-Sample Estimation. *Annals of Mathematical Statistics* 18.4, pp. 582–585.
- Brown, Tom, 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 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 (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

- Burda, Yuri, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei Efros (2018). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.
- Burda, Yuri, Harrison Edwards, Amos Storkey, and Oleg Klimov (2018). Exploration by random network distillation. **In** *International Conference on Learning Representations*.
- Campero, Andres, Roberta Raileanu, Heinrich Küttler, Joshua Tenenbaum, Tim Rocktäschel, and Edward Grefenstette (2021). Learning with AMIGo: Adversarially Motivated Intrinsic Goals. *International Conference on Learning Representations*.
- Catto, Erin (2011). *Box2d: A 2d physics engine for games*. <https://github.com/pybox2d/pybox2d>.
- Chen, Xinjia, Jorge Aravena, and Kemin Zhou (2005). Risk analysis in robust control-making the case for probabilistic robust control. **In** *Proceedings of American Control Conference*. Portland, Oregon, USA, pp. 1533–1538.
- Chevalier-Boisvert, Maxime, Lucas Willems, and Suman Pal (2018). *Minimalistic Gridworld Environment for OpenAI Gym*. <https://github.com/maximecb/gym-minigrid>.
- Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. **In** *Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734.
- Chow, Yinlam and Mohammad Ghavamzadeh (2014). Algorithms for CVaR Optimization in MDPs. **In** *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc.
- Chow, Yinlam, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone (2017). Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research* 18.1, pp. 6070–6120.
- Chow, Yinlam, Aviv Tamar, Shie Mannor, and Marco Pavone (2015). Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. **In** *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc.
- Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2016). Fast and accurate deep network learning by exponential linear units (elus). **In** *International Conference on Learning Representation*.
- Clouse, Jeffery and Paul Utgoff (1992). A teaching method for reinforcement learning. **In** *Machine Learning*. Elsevier, pp. 92–101.
- Cobbe, Karl, Chris Hesse, Jacob Hilton, and John Schulman (13–18 Jul 2020). Leveraging Procedural Generation to Benchmark Reinforcement Learning. **In** *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 2048–2056.
- Colas, Cédric, Olivier Sigaud, and Pierre-Yves Oudeyer (2018). How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*.
- Coraluppi, Stefano and Steven Marcus (1999). Risk-sensitive and minimax control of discrete-time, finite-state Markov decision processes. *Automatica* 35.2, pp. 301–309.

Bibliography

- Côté, Marc-Alexandre, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler (2019). TextWorld: A Learning Environment for Text-Based Games. *Computer Games*, pp. 41–75.
- Coumans, Erwin and Yunfei Bai (2016). *PyBullet, a Python module for physics simulation for games, robotics and machine learning*.
- Dabney, Will, Georg Ostrovski, David Silver, and Rémi Munos (Oct. 2018). Implicit Quantile Networks for Distributional Reinforcement Learning. **In** *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1096–1105.
- Dabney, Will, Mark Rowland, Marc Bellemare, and Rémi Munos (Apr. 2018). Distributional Reinforcement Learning With Quantile Regression. *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1.
- Dahl, George, Dong Yu, Li Deng, and Alex Acero (2012). Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing* 20.1, pp. 30–42.
- Dayan, Peter and Geoffrey Hinton (Sept. 2000). Feudal Reinforcement Learning. *Advances in Neural Information Processing Systems* 5.
- Pierre de Fermat (1637). Dernier théorème de Fermat. *Annotated version of Diophante’s “l’Arithmétique”*.
- de Vries, Harm, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville (2017). Guesswhat?! visual object discovery through multi-modal dialogue. **In** *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5503–5512.
- Degrave, Jonas, Michiel Hermans, Joni Dambre, and Francis Wyffels (2019). A Differentiable Physics Engine for Deep Learning in Robotics. *Frontiers in Neurorobotics* 13, p. 6.
- Deisenroth, Marc, Gerhard Neumann, and Jan Peters (2013). A survey on policy search for robotics. *Foundations and trends in Robotics* 2.1-2, pp. 388–403.
- Deisenroth, Marc and Carl Rasmussen (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. **In** *Proceedings of the 28th International Conference on Machine Learning*. ICML’11. Bellevue, Washington, USA: Omnipress, pp. 465–472.
- Demongeot, Jacques, Yannis Flet-Berliac, and Hervé Seligmann (2020). Temperature Decreases Spread Parameters of the New Covid-19 Case Dynamics. *Biology* 9.5, p. 94.
- Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). Imagenet: A large-scale hierarchical image database. **In** *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.
- Depas, Thomas and Yannis Flet-Berliac (2019). Princess of Parallelograms. *Exhibition Panorama 21 - Le Fresnoy National Studio of Contemporary Arts*.
- Dimeas, Aris and Nikos Hatzigrygiou (2007). Agent based control for microgrids. **In** *IEEE Power Engineering Society General Meeting*, pp. 1–5.
- Domingues, Omar Darwiche, Yannis Flet-Berliac, Edouard Leurent, Pierre Ménard, Xuedong Shang, and Michal Valko (2021). *rlberry - A Reinforcement Learning Library for Research and Education*. <https://github.com/rlberry-py/rlberry>.

- Dosovitskiy, Alexey and Vladlen Koltun (2016). Learning to act by predicting the future. *In International Conference on Learning Representations*.
- Du, Yunshu, Wojciech Czarnecki, Siddhant Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan (2018). Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*.
- Duan, Yan, Xi Chen, Rein Houthoofd, John Schulman, and Pieter Abbeel (20–22 Jun 2016). Benchmarking Deep Reinforcement Learning for Continuous Control. *In Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1329–1338.
- Duchi, John, Elad Hazan, and Yoram Singer (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12.61, pp. 2121–2159.
- Dulac-Arnold, Gabriel, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester (2020). An empirical investigation of the challenges of real-world reinforcement learning.
- Ecoffet, Adrien, Joost Huizinga, Joel Lehman, Kenneth Stanley, and Jeff Clune (2019). Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*.
- Eriksson, Hannes, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis (2021). SENTINEL: Taming Uncertainty with Ensemble-based Distributional Reinforcement Learning. *arXiv preprint arXiv:2102.11075*.
- Espeholt, Lasse, Hubert Soyer, Rémi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu (Oct. 2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *In Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1407–1416.
- Eysenbach, Benjamin, Abhishek Gupta, Julian Ibarz, and Sergey Levine (2018). Diversity is All You Need: Learning Skills without a Reward Function. *In International Conference on Learning Representations*.
- Eysenbach, Benjamin and Sergey Levine (2019). If MaxEnt RL is the answer, what is the question? *arXiv preprint arXiv:1910.01913*.
- Eysenbach, Benjamin and Sergey Levine (2021). Maximum Entropy RL (Provably) Solves Some Robust RL Problems. *arXiv preprint arXiv:2103.06257*.
- Farebrother, Jesse, Marlos Machado, and Michael Bowling (2018). Generalization and regularization in DQN. *arXiv preprint arXiv:1810.00123*.
- Ferret, Johan, Raphael Marinier, Matthieu Geist, and Olivier Pietquin (July 2020). Self-Attentional Credit Assignment for Transfer in Reinforcement Learning. *In Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, pp. 2655–2661.
- Ferret, Johan, Olivier Pietquin, and Matthieu Geist (2021). Self-Imitation Advantage Learning. *In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*.

Bibliography

- Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 501–509.
- Flet-Berliac, Yannis (2019). The Promise of Hierarchical Reinforcement Learning. *The Gradient - Stanford AI Lab*.
- Flet-Berliac, Yannis and Debabrota Basu (2021). SAAC: Safe Reinforcement Learning as an Adversarial Game of Actor-Critics. *Preprint*.
- Flet-Berliac, Yannis, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist (2021). Adversarially Guided Actor-Critic. **In** *International Conference on Learning Representations*.
- Flet-Berliac, Yannis, Reda Ouhamma, Odalric-Ambrym Maillard, and Philippe Preux (2021). Learning Value Functions in Deep Policy Gradients using Residual Variance. **In** *International Conference on Learning Representations*.
- Flet-Berliac, Yannis and Philippe Preux (2019a). High-Dimensional Control Using Generalized Auxiliary Tasks. *Tech. rep. hal-02295705*.
- Flet-Berliac, Yannis and Philippe Preux (2019b). MERL: Multi-Head Reinforcement Learning. **In** *Deep Reinforcement Learning Workshop of the 33rd conference on advances in Neural Information Processing Systems*.
- Flet-Berliac, Yannis and Philippe Preux (2019c). Samples Are Useful? Not Always: denoising policy gradient updates using variance explained. *arXiv preprint arXiv:1904.04025*.
- Flet-Berliac, Yannis and Philippe Preux (July 2020). Only Relevant Information Matters: Filtering Out Noisy Samples To Boost RL. **In** *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, pp. 2711–2717.
- Florensa, Carlos, David Held, Xinyang Geng, and Pieter Abbeel (Oct. 2018). Automatic Goal Generation for Reinforcement Learning Agents. **In** *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1515–1528.
- Fortunato, Meire, Mohammad Azar, Bilal Piot, Jacob Menick, Ian Osband, Alexander Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg (2018). Noisy Networks for Exploration. **In** *International Conference on Learning Representation*.
- François-Lavet, Vincent, Peter Henderson, Riashat Islam, Marc Bellemare, and Joëlle Pineau (2018). An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning* 11.3-4, pp. 219–354.
- Freeman, Daniel, Luke Metz, and David Ha (2019). Learning to Predict Without Looking Ahead: World Models Without Forward Prediction. **In** *Advances in Neural Information Processing Systems*.
- Fujimoto, Scott, Herke van Hoof, and David Meger (Oct. 2018). Addressing Function Approximation Error in Actor-Critic Methods. **In** *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1587–1596.
- Gabillon, Victor, Mohammad Ghavamzadeh, and Bruno Scherrer (2013). Approximate Dynamic Programming Finally Performs Well in the Game of Tetris. **In** *Advances in Neural*

-
- Information Processing Systems*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Vol. 26. Curran Associates, Inc.
- Garcia, Javier and Fernando Fernández (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16.1, pp. 1437–1480.
- Garipov, Timur, Pavel Izmailov, Dmitrii Podoprikin, Dmitry Vetrov, and Andrew Wilson (2018). Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. **In** *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.
- Geibel, Peter and Fritz Wyszotzki (2005). Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research* 24, pp. 81–108.
- Geist, Matthieu, Bruno Scherrer, and Olivier Pietquin (Sept. 2019). A Theory of Regularized Markov Decision Processes. **In** *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2160–2169.
- Ghadirzadeh, Ali, Atsuto Maki, Danica Kragic, and Mårten Björkman (2017). Deep predictive policy training using reinforcement learning. **In** *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2351–2358.
- Glorot, Xavier and Yoshua Bengio (13–15 May 2010). Understanding the difficulty of training deep feedforward neural networks. **In** *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterton. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pp. 249–256.
- Goldberg, David and John Henry Holland (1988). Genetic algorithms and machine learning.
- Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio (2016). *Deep learning*. Vol. 1. 2. MIT press Cambridge.
- Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). Generative adversarial nets. **In** *Advances in Neural Information Processing Systems*, pp. 2672–2680.
- Goodfellow, Ian, Jonathon Shlens, and Christian Szegedy (2015). Explaining and harnessing adversarial examples. **In** *International Conference on Learning Representations*.
- Goyal, Anirudh, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio (2019). Transfer and Exploration via the Information Bottleneck. **In** *International Conference on Learning Representations*.
- Greensmith, Evan, Peter Bartlett, and Jonathan Baxter (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5, pp. 1471–1530.
- Grinsztajn, Nathan, Olivier Beaumont, Emmanuel Jeannot, and Philippe Preux (2020). Geometric deep reinforcement learning for dynamic DAG scheduling. **In** *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 258–265.
- Gruslys, Audrunas, Will Dabney, Mohammad Azar, Bilal Piot, Marc Bellemare, and Rémi Munos (2018). The Reactor: A fast and sample-efficient Actor-Critic agent for Reinforcement Learning. **In** *International Conference on Learning Representations*.

Bibliography

- Grzes, Marek and Daniel Kudenko (2009). Learning shaping rewards in model-based reinforcement learning. *In AAMAS 2009 Workshop on Adaptive Learning Agents*. Vol. 115. Citeseer, p. 30.
- Gu, Shixiang, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine (20–22 Jun 2016). Continuous Deep Q-Learning with Model-based Acceleration. *In Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 2829–2838.
- Gulcehre, Caglar, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matthew Hoffman, Nicolas Heess, and Nando de Freitas (2020). RL Unplugged: A Suite of Benchmarks for Offline Reinforcement Learning. *In Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 7248–7259.
- Gullapalli, Vijaykumar (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks* 3.6, pp. 671–692.
- Ha, David and Jürgen Schmidhuber (2018). Recurrent world models facilitate policy evolution. *In Advances in Neural Information Processing Systems*, pp. 2450–2462.
- Haarnoja, Tuomas, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine (Oct. 2018). Latent Space Policies for Hierarchical Reinforcement Learning. *In Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1851–1860.
- Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (Oct. 2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *In Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1861–1870.
- Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.
- Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (Sept. 2019). Learning Latent Dynamics for Planning from Pixels. *In Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2555–2565.
- Han, Seungyul and Youngchul Sung (2020). Diversity Actor-Critic: Sample-Aware Entropy Regularization for Sample-Efficient Exploration. *arXiv preprint arXiv:2006.01419*.
- Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Sathesh, Shubho Sengupta, Adam Coates, and Andrew Ng (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.
- Harmon, Mance and Leemon Baird III (n.d.). Multi-player residual advantage learning with general function approximation ().
- Hasselt, Hado van, Arthur Guez, and David Silver (Mar. 2016). Deep Reinforcement Learning with Double Q-Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1.

- Hausknecht, Matthew and Peter Stone (2015). Deep Recurrent Q-Learning for Partially Observable MDPs. *In AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1026–1034.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). Deep residual learning for image recognition. *In IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.
- Heess, Nicolas, Dhruva Tirumala, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, and David Silver (2017). Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*.
- Heess, Nicolas, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez (2015). Learning continuous control policies by stochastic value gradients. *In Advances in Neural Information Processing Systems*, pp. 2944–2952.
- Heger, Matthias (1994). Consideration of risk in reinforcement learning. *In Machine Learning Proceedings 1994*. Elsevier, pp. 105–111.
- Henderson, Peter, Riashat Islam, Philip Bachman, Joëlle Pineau, Doina Precup, and David Meger (Apr. 2018). Deep Reinforcement Learning That Matters. *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1.
- Hessel, Matteo, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver (Apr. 2018). Rainbow: Combining Improvements in Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1.
- Hill, Ashley, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu (2018). *Stable Baselines*. <https://github.com/hill-a/stable-baselines>.
- Ho, Jonathan and Stefano Ermon (2016). Generative adversarial imitation learning. *In Advances in Neural Information Processing Systems*, pp. 4565–4573.
- Howard, Ronald and James Matheson (1972). Risk-sensitive Markov decision processes. *Management science* 18.7, pp. 356–369.
- Huang, Gao, Zhuang Liu, Laurens van der Maaten, and Kilian Weinberger (2017). Densely connected convolutional networks. *In IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708.
- Aldous Huxley (1932). *Texts and Pretexts*. Chatto and Windus, p. 5.
- Igl, Maximilian, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann (2019). Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck. *In Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.
- Igl, Maximilian, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson (2021). Transient Non-stationarity and Generalisation in Deep Reinforcement Learning. *In International Conference on Learning Representations*.

Bibliography

- Ilyas, Andrew, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry (2020). A Closer Look at Deep Policy Gradients. *In International Conference on Learning Representations*.
- Islam, Riashat, Peter Henderson, Maziar Gomrokchi, and Doina Precup (2017). Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *In Reproducibility in Machine Learning Workshop, ICML*.
- Iyer, Rahul, Yuezhong Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara (2018). Transparency and Explanation in Deep Reinforcement Learning Neural Networks. *In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. AIES '18. New Orleans, LA, USA: Association for Computing Machinery, pp. 144–150.
- Jaakkola, Tommi, Michael Jordan, and Satinder Singh (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural computation* 6.6, pp. 1185–1201.
- Jaderberg, Max, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Leibo, David Silver, and Koray Kavukcuoglu (2017). Reinforcement learning with unsupervised auxiliary tasks. *International Conference on Learning Representations*.
- Johnson, Rie and Tong Zhang (2013). Accelerating stochastic gradient descent using predictive variance reduction. *In Advances in Neural Information Processing Systems*, pp. 315–323.
- Justesen, Niels, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi (2018). Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. *In NeurIPS Workshop on Deep Reinforcement Learning*.
- Kaelbling, Leslie Pack (1993). Learning to achieve goals. *In IJCAI*. Citeseer, pp. 1094–1099.
- Kaiser, Łukasz, Mohammad Babaeizadeh, Piotr Miłoś, Błażej Osipiński, Roy Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski (2020). Model Based Reinforcement Learning for Atari. *In International Conference on Learning Representations*.
- Kakade, Sham (2002). A natural policy gradient. *In Advances in Neural Information Processing Systems*, pp. 1531–1538.
- Kakade, Sham (2003). On the sample complexity of reinforcement learning. PhD thesis. University of London.
- Kakade, Sham and John Langford (2002). Approximately Optimal Approximate Reinforcement Learning. *In Proceedings of the 19th International Conference on Machine Learning*. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 267–274.
- Kartal, Bilal, Pablo Hernandez-Leal, and Matthew Taylor (Oct. 2019). Terminal Prediction as an Auxiliary Task for Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 15.1, pp. 38–44.
- Keel, Le, Shankar Bhattacharyya, and Jo Howze (1988). Robust control with structure perturbations. *IEEE Transactions on Automatic Control* 33.1, pp. 68–78.
- Kelley, Henry (1960). Gradient theory of optimal flight paths. *Ars Journal* 30.10, pp. 947–954.
- Kempka, Michał, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning. *In IEEE Conference on Computational Intelligence and Games*, pp. 1–8.

- Kimura, Hajime (2007). Reinforcement learning in multi-dimensional state-action space using random rectangular coarse coding and gibbs sampling. *In SICE Annual Conference*, pp. 2754–2761.
- Kimura, Hajime and Shigenobu Kobayashi (1998). An Analysis of Actor/Critic Algorithms Using Eligibility Traces: Reinforcement Learning with Imperfect Value Function. *In Proceedings of the 15th International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 278–286.
- Kingma, Diederik and Jimmy Ba (2015). Adam: A method for stochastic optimization. *In International Conference on Learning Representation*.
- Klimov, Oleg and John Schulman (2017). *Roboschool*.
- Klyubin, Alexander, Daniel Polani, and Chrystopher Nehaniv (2005). Empowerment: a universal agent-centric measure of control. *In IEEE Congress on Evolutionary Computation*.
- Kober, Jens, Andrew Bagnell, and Jan Peters (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32.11, pp. 1238–1274.
- Koller, Torsten, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause (2018). Learning-based model predictive control for safe exploration. *In IEEE Conference on Decision and Control (CDC)*, pp. 6059–6066.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks. *In Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc.
- Kupcsik, Andras Gabor, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann (2013). Data-Efficient Generalization of Robot Skills with Contextual Policy Search. *In Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI'13. Bellevue, Washington: AAAI Press, pp. 1401–1407.
- Kurakin, Alexey, Ian Goodfellow, and Samy Bengio (2017). Adversarial machine learning at scale. *In International Conference on Learning Representations*.
- Kuznetsov, Arsenii, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov (13–18 Jul 2020). Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. *In Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 5556–5566.
- Kvålseth, Tarald (1985). Cautionary Note about R^2 . *The American Statistician* 39.4, pp. 279–285.
- Lapan, Maxim (2018). *Deep Reinforcement Learning Hands-On*. Packt Publishing.
- Laskin, Misha, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas (2020). Reinforcement Learning with Augmented Data. *In Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 19884–19895.
- Ursula Le Guin (1969). *The Left Hand of Darkness*. Ace Books.
- LeCun, Yann (1988). A theoretical framework for back-propagation. *In Connectionist Models Summer School*. Vol. 1, pp. 21–28.

Bibliography

- Lee, Kimin, Kibok Lee, Jinwoo Shin, and Honglak Lee (2020). Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning. *In International Conference on Learning Representations*.
- Lesort, Timothée, Natalia Díez-Rodríguez, Jean-François Goudou, and David Filliat (2018). State representation learning for control: An overview. *Neural Networks* 108, pp. 379–392.
- Levent, Tanguy, Philippe Preux, Erwan le Pennec, Jordi Badosa, Gonzague Henri, and Yvan Bonnassieux (2019). Energy Management for Microgrids: a Reinforcement Learning Approach. *In IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, pp. 1–5.
- Levine, Sergey (2017). *Lecture CS 294: Deep Reinforcement Learning*. UC Berkeley.
- Levine, Sergey and Pieter Abbeel (2014). Learning neural network policies with guided policy search under unknown dynamics. *In Advances in Neural Information Processing Systems*, pp. 1071–1079.
- Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17.1, pp. 1334–1373.
- Levine, Sergey, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 37.4-5, pp. 421–436.
- Li, Siyuan, Rui Wang, Minxue Tang, and Chongjie Zhang (2019). Hierarchical Reinforcement Learning with Advantage-Based Auxiliary Rewards. *In Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.
- Li, Xiujun, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He (2016). Recurrent reinforcement learning: a hybrid approach. *In International Conference on Learning Representations*.
- Lillicrap, Timothy, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016). Continuous control with deep reinforcement learning. *In International Conference on Learning Representations*.
- Lim, Andrew and George Shanthikumar (2007). Relative entropy, exponential utility, and robust dynamic pricing. *Operations Research* 55.2, pp. 198–214.
- Lim, Sungsu, Ajin Joseph, Lei Le, Yangchen Pan, and Martha White (2019). Actor-Expert: A Framework for using Q-learning in Continuous Action Spaces. MA thesis.
- Lin, Kaixiang and Jiayu Zhou (2020). Ranking Policy Gradient. *In International Conference on Learning Representations*.
- Lin, Long-Ji (1992a). Reinforcement learning for robots using neural networks. PhD thesis.
- Lin, Long-Ji (1992b). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8.3-4, pp. 293–321.
- Lipton, Zachary, Kamyar Azizzadenesheli, Abhishek Kumar, Lihong Li, Jianfeng Gao, and Li Deng (2016). Combating reinforcement learning’s sisyphian curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*.
- Liu, Hao, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu (2018). Action-dependent control variates for policy optimization via stein identity. *In International Conference on Learning Representations*.

- Ada Lovelace (1843). Notes. *Annotated translation of Menabrea's "Notions sur la machine analytique de Charles Babbage"*.
- Mahajan, Dhruv, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten (2018). Exploring the limits of weakly supervised pretraining. *In European Conference on Computer Vision*, pp. 181–196.
- Malinowski, Mateusz, Marcus Rohrbach, and Mario Fritz (2015). Ask your neurons: A neural-based approach to answering questions about images. *In IEEE International Conference on Computer Vision*, pp. 1–9.
- Mandlekar, Ajay, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese (2017). Adversarially robust policy learning: Active construction of physically-plausible perturbations. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939.
- Mania, Horia, Aurelia Guy, and Benjamin Recht (2018). Simple random search of static linear policies is competitive for reinforcement learning. *In Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.
- Mao, Hongzi, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula (2016). Resource management with deep reinforcement learning. *In ACM Workshop on Hot Topics in Networks*, pp. 50–56.
- Marcus, Steven, Emmanuel Fernández-Gaucherand, Daniel Hernández-Hernandez, Stefano Coraluppi, and Pedram Fard (1997). Risk sensitive Markov decision processes. *In Systems and control in the twenty-first century*. Springer, pp. 263–279.
- James Clerk Maxwell (1860). Maxwell's theory leading to his "Treatise on Electricity and Magnetism". *Correspondance*.
- Millán, José Del R, Daniele Posenato, and Eric Dedieu (2002). Continuous-action Q-learning. *Machine Learning* 49.2, pp. 247–265.
- Mirowski, Piotr, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell (2016). Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*.
- Miyato, Takeru, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii (2016). Distributional smoothing with virtual adversarial training. *In International Conference on Learning Representations*.
- Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (20–22 Jun 2016). Asynchronous Methods for Deep Reinforcement Learning. *In Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1928–1937.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Belle-mare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan

Bibliography

- Wierstra, Shane Legg, and Demis Hassabis (Feb. 2015). Human-level control through deep reinforcement learning. en. *Nature* 518.7540, pp. 529–533.
- Moldovan, Teodor Mihai, Sergey Levine, Michael Jordan, and Pieter Abbeel (2015). Optimism-driven exploration for nonlinear systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3239–3246.
- Moreno, David, Carlos Regueiro, Roberto Iglesias, and Senén Barro (2004). Using prior knowledge to improve reinforcement learning in mobile robotics. In *Towards Autonomous Robotics Systems*.
- Munos, Rémi, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare (2016). Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062.
- Nachum, Ofir, Shixiang Gu, Honglak Lee, and Sergey Levine (2018). Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.
- Nachum, Ofir, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans (2017). Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785.
- Nachum, Ofir, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine (2019). Why Does Hierarchy (Sometimes) Work So Well in Reinforcement Learning? In *Deep Reinforcement Learning Workshop, NeurIPS*.
- Namkoong, Hongseok and John Duchi (2017). Variance-based Regularization with Convex Objectives. In *Advances in Neural Information Processing Systems*, pp. 2971–2980.
- Nasiriany, Soroush, Vitchyr Pong, Steven Lin, and Sergey Levine (2019). Planning with Goal-Conditioned Policies. In *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc.
- Nelson, Barry (1990). Control variate remedies. *Operations Research* 38.6, pp. 974–992.
- Neu, Gergely, Anders Jonsson, and Vicenç Gómez (2017). A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*.
- Nguyen, Derrick and Bernard Widrow (1990). The truck backer-upper: An example of self-learning in neural networks. In *Advanced Neural Computers*, pp. 11–19.
- Nilim, Arnab and Laurent El Ghaoui (2005). Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53.5, pp. 780–798.
- Nocedal, Jorge and Stephen Wright (2006). *Numerical optimization*. Springer Science & Business Media.
- Oh, Junhyuk, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh (2015). Action-Conditional Video Prediction using Deep Networks in Atari Games. In *Advances in Neural Information Processing Systems*. Ed. by C Cortes, N D Lawrence, D D Lee, M Sugiyama, R Garnett, and R Garnett. Curran Associates, Inc., pp. 2845–2853.

- Oh, Junhyuk, Yijie Guo, Satinder Singh, and Honglak Lee (Oct. 2018). Self-Imitation Learning. *In Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3878–3887.
- Oudeyer, Pierre-Yves and Frederic Kaplan (2007). What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics* 1, p. 6.
- Packer, Charles, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song (2018). Assessing Generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:1810.12282*.
- Pan, Xinlei, Daniel Seita, Yang Gao, and John Canny (2019). Risk averse robust adversarial reinforcement learning. *In International Conference on Robotics and Automation (ICRA)*, pp. 8522–8528.
- Pan, Xinlei, Yurong You, Ziyang Wang, and Cewu Lu (2017). Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*.
- Pardo, Fabio, Arash Tavakoli, Vitaly Levnik, and Petar Kormushev (Oct. 2018). Time Limits in Reinforcement Learning. *In Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4045–4054.
- Pathak, Deepak, Pulkit Agrawal, Alexei Efros, and Trevor Darrell (June 2017). Curiosity-driven Exploration by Self-supervised Prediction. *In Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 2778–2787.
- Pattanaik, Anay, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary (2017). Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*.
- Perolat, Julien, Bruno Scherrer, Bilal Piot, and Olivier Pietquin (July 2015). Approximate Dynamic Programming for Two-Player Zero-Sum Markov Games. *In Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1321–1329.
- Peters, Jan, Katharina Mulling, and Yasemin Altun (July 2010). Relative Entropy Policy Search. *Proceedings of the AAAI Conference on Artificial Intelligence* 24.1.
- Peters, Jan and Stefan Schaal (2008a). Natural actor-critic. *Neurocomputing* 71.7-9, pp. 1180–1190.
- Peters, Jan and Stefan Schaal (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Networks* 21.4, pp. 682–697.
- Pfau, David and Oriol Vinyals (2016). Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*.
- Pham-Gia, Thu and Tran Loc Hung (2001). The mean and median absolute deviations. *Mathematical and Computer Modelling* 34.7-8, pp. 921–936.
- Pineau, Joëlle (2018). Reproducible, Reusable, and Robust Reinforcement Learning. *Advances in Neural Information Processing Systems*.
- Pineau, Joëlle, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d’Alché-Buc, Emily Fox, and Hugo Larochelle (2020). Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:2003.12206*.

Bibliography

- Pinto, Lerrel, James Davidson, Rahul Sukthankar, and Abhinav Gupta (June 2017). Robust Adversarial Reinforcement Learning. **In** *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 2817–2826.
- Pong, Vitchyr, Shixiang Gu, Murtaza Dalal, and Sergey Levine (2018). Temporal Difference Models: Model-Free Deep RL for Model-Based Control. **In** *International Conference on Learning Representations*.
- Pourchot, Aloïs and Olivier Sigaud (2019). CEM-RL: Combining evolutionary and gradient-based methods for policy search. **In** *International Conference on Learning Representations*.
- Prashanth, L A and Michael Fu (2018). Risk-sensitive reinforcement learning: A constrained optimization viewpoint. *arXiv e-prints*, arXiv–1810.
- Prashanth, L A and Mohammad Ghavamzadeh (2016). Variance-constrained actor-critic algorithms for discounted and average reward MDPs. *Machine Learning* 105.3, pp. 367–417.
- Puterman, Martin (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.
- Raghu, Maithra, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein (June 2017). On the Expressive Power of Deep Neural Networks. **In** *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 2847–2854.
- Raileanu, Roberta and Tim Rocktäschel (2019). RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. **In** *International Conference on Learning Representations*.
- Rastrigin (1963). About Convergence of Random Search Method in Extremal Control of Multi-Parameter Systems. Russian. *Automation and Remote Control* 24.10, pp. 1337–1342.
- Ray, Alex, Joshua Achiam, and Dario Amodei (2019). Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*.
- Ribeiro, Carlos (1998). Embedding a priori knowledge in reinforcement learning. *Journal of Intelligent and Robotic Systems* 21.1, pp. 51–71.
- Riedmiller, Martin, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degraeve, Tom van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg (Oct. 2018). Learning by Playing Solving Sparse Reward Tasks from Scratch. **In** *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4344–4353.
- Robbins, Herbert and Sutton Monro (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407.
- Rockafellar, Tyrrell and Stanislav Uryasev (2000). Optimization of conditional value-at-risk. *Journal of Risk* 2, pp. 21–42.
- Ruder, Sebastian (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Rumelhart, David, Geoffrey Hinton, and Ronald Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California University San Diego La Jolla Institute for Cognitive Science.

- Rummery, Gavin and Mahesan Niranjan (1994). *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK.
- Schaefer, Andrew, Matthew Bailey, Steven Shechter, and Mark Roberts (2005). Modeling medical treatment using Markov decision processes. *In Operations research and health care*. Springer, pp. 593–612.
- Schaul, Tom, Daniel Horgan, Karol Gregor, and David Silver (July 2015). Universal Value Function Approximators. *In Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1312–1320.
- Schaul, Tom, John Quan, Ioannis Antonoglou, and David Silver (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.
- Schmidhuber, Jürgen (1991). Curious model-building control systems. *In International Joint Conference on Neural Networks*.
- Schmidhuber, Jürgen (2006). Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science* 18.2, pp. 173–187.
- Schmidhuber, Jürgen (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development* 2.3, pp. 230–247.
- Schmidhuber, Jürgen and Rudolf Huber (1991). Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems* 2.1/2, pp. 135–141.
- Schmidhuber, Jürgen and Reiner Wahnsiedler (1993). Planning simple trajectories using neural subgoal. *In International Conference on Simulation of Adaptive Behavior*. Vol. 2. MIT Press, p. 196.
- Schulman, John (2017). *Lecture CS 294: Deep Reinforcement Learning*. UC Berkeley.
- Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (July 2015). Trust Region Policy Optimization. *In Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1889–1897.
- Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel (2016). High-dimensional continuous control using generalized advantage estimation. *In International Conference on Learning Representations*.
- Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Claude Shannon (1948). A mathematical theory of communication. *The Bell System Technical Journal* 27.3, pp. 379–423.
- Shelhamer, Evan, Parsa Mahmoudieh, Max Argus, and Trevor Darrell (2016). Loss is its own reward: Self-supervision for reinforcement learning. *In International Conference on Learning Representations*.
- Shi, Tianlin, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang (2017). World of Bits: An Open-Domain Platform for Web-Based Agents. *In Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 3135–3144.

Bibliography

- Shu, Tianmin, Caiming Xiong, and Richard Socher (2018). Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning. *In International Conference on Learning Representations*.
- Sigaud, Olivier and Freck Stulp (2019). Policy search in continuous action domains: an overview. *Neural Networks* 113, pp. 28–40.
- Silver, David, Aja Huang, Chris Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (Jan. 2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529.7587, pp. 484–489.
- Silver, David, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller (22–24 Jun 2014). Deterministic Policy Gradient Algorithms. *In Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Beijing, China: PMLR, pp. 387–395.
- Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis (Oct. 2017). Mastering the game of Go without human knowledge. *Nature* 550.7676, pp. 354–359.
- Song, Xingyou, Yiding Jiang, Yilun Du, and Behnam Neyshabur (2020). Observational overfitting in reinforcement learning. *In International Conference on Learning Representations*.
- Sorin, Sylvain (1986). Asymptotic properties of a non-zero sum stochastic game. *International Journal of Game Theory* 15.2, pp. 101–107.
- Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15.1, pp. 1929–1958.
- Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.
- Suddarth, Steve and Yannick Kergosien (1990). Rule-injection hints as a means of improving network performance and learning time. *Neural Networks*, pp. 120–129.
- Sun, Yi, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber (2009). Efficient natural evolution strategies. *In Annual Conference on Genetic and Evolutionary Computation*, pp. 539–546.
- Sünderhauf, Niko, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, and Peter Corke (2018). The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research* 37.4-5, pp. 405–420.
- Sutskever, Ilya, Oriol Vinyals, and Quoc Le (2014). Sequence to Sequence Learning with Neural Networks. *In Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc.
- Sutton, Richard (1984). Temporal Credit Assignment in Reinforcement Learning. PhD thesis. University of Massachusetts Amherst.

- Sutton, Richard (1988). Learning to predict by the methods of temporal differences. *Machine Learning* 3.1, pp. 9–44.
- Sutton, Richard and Andrew Barto (1998). *Introduction to reinforcement learning*. Cambridge: MIT Press.
- Sutton, Richard, David McAllester, Satinder Singh, and Yishay Mansour (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. **In** *Advances in Neural Information Processing Systems*.
- Sutton, Richard, Doina Precup, and Satinder Singh (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112.1-2, pp. 181–211.
- Szegö, Giorgio (2004). *Risk measures for the 21st century*. Vol. 1. Wiley New York.
- Szita, István and András Lörincz (2006). Learning Tetris using the noisy cross-entropy method. *Neural computation* 18.12, pp. 2936–2941.
- Tamar, Aviv, Huan Xu, and Shie Mannor (2013). Scaling up robust MDPs by reinforcement learning. *arXiv preprint arXiv:1306.6189*.
- Tang, Yunhao and Alp Kucukelbir (2017). Variational deep q network. *arXiv preprint arXiv:1711.11225*.
- Tassa, Yuval, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. (2018). Deepmind control suite. *arXiv preprint arXiv:1801.00690*.
- Tesauro, Gerald (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM* 38.3, pp. 58–68.
- Nikola Tesla (1915). The Wonder World to Be Created by Electricity. *Manufacturer's Record*.
- Tham, Chen Khong (1994). Modular on-line function approximation for scaling up reinforcement learning. PhD thesis. University of Cambridge.
- Thananjeyan, Brijen, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minh Hwang, Joseph Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg (2021). Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters* 6.3, pp. 4915–4922.
- Thodoroff, Pierre, Audrey Durand, Joëlle Pineau, and Doina Precup (2018). Temporal Regularization for Markov Decision Process. **In** *Advances in Neural Information Processing Systems*.
- Edward Thorndike (1911). *Animal intelligence: Experimental studies*. New York: Macmillan, p. 244.
- Thrun, Sebastian and Anton Schwartz (1993). Issues in using function approximation for reinforcement learning. **In** *Connectionist Models Summer School*. Hillsdale, NJ, pp. 255–263.
- Tieleman, Tijmen and Geoffrey Hinton (2012). *Lecture 6.5: RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning.
- Todorov, Emanuel (2007). Linearly-solvable Markov decision problems. **In** *Advances in Neural Information Processing Systems*, pp. 1369–1376.

Bibliography

- Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). Mujoco: A physics engine for model-based control. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033.
- Tokic, Michel (2010). Adaptive ε -greedy exploration in reinforcement learning based on value differences. *In Annual Conference on Artificial Intelligence*. Springer, pp. 203–210.
- Toussaint, Marc (2009). Robot Trajectory Optimization Using Approximate Inference. *In Proceedings of the 26th International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, pp. 1049–1056.
- Tucker, George, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine (Oct. 2018). The Mirage of Action-Dependent Baselines in Reinforcement Learning. *In Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5015–5024.
- Alan Turing (1860). Computing Machinery and Intelligence. *MIND*.
- van den Oord, Aaron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.
- van Hasselt, Hado (2010). Double Q-learning. *In Advances in Neural Information Processing Systems*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). Attention is All you Need. *In Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc.
- Vieillard, Nino, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist (2020). Leverage the Average: an Analysis of Regularization in RL. *In Advances in Neural Information Processing Systems*.
- Vieillard, Nino, Olivier Pietquin, and Matthieu Geist (2020). Munchausen Reinforcement Learning. *In Advances in Neural Information Processing Systems*.
- Vinyals, Oriol, Igor Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, Alexander Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Paine, Caglar Gulcehre, 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 (Nov. 2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575.7782, pp. 350–354.
- Wachi, Akifumi and Yanan Sui (13–18 Jul 2020). Safe Reinforcement Learning in Constrained Markov Decision Processes. *In Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 9797–9806.

- Wampler, Kevin and Zoran Popović (2009). Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)* 28.3, pp. 1–8.
- Wang, Jack, David Fleet, and Aaron Hertzmann (2010). Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics (TOG)* 29.4, pp. 1–8.
- Wang, Ziyu, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas (2017). Sample efficient actor-critic with experience replay. *In International Conference on Learning Representations*.
- Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas (20–22 Jun 2016). Dueling Network Architectures for Deep Reinforcement Learning. *In Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1995–2003.
- Watkins, Christopher and Peter Dayan (May 1992). Q-learning. *Machine Learning* 8.3, pp. 279–292.
- Wawrzyński, Paweł (2009). Real-time reinforcement learning by sequential actor–critics and experience replay. *Neural Networks* 22.10, pp. 1484–1497.
- Weaver, Lex and Nigel Tao (2001). The Optimal Reward Baseline for Gradient-Based Reinforcement Learning. *In Advances in Neural Information Processing Systems*.
- Werbos, Paul (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. PhD thesis. Harvard University.
- Werbos, Paul (1989). Neural networks for control and system identification. *In IEEE Conference on Decision and Control (CDC)*, pp. 260–265.
- Whitehead, Steven (1991). Complexity and cooperation in Q-learning. *In Eighth International Workshop on Machine Learning*, pp. 363–367.
- Whiteson, Shimon, Brian Tanner, Matthew Taylor, and Peter Stone (2011). Protecting against evaluation overfitting in empirical reinforcement learning. *In IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 120–127.
- Wierstra, Daan, Tom Schaul, Jan Peters, and Jürgen Schmidhuber (2008). Natural evolution strategies. *In IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387.
- Williams, Ronald (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8.3-4, pp. 229–256.
- Williams, Ronald and Jing Peng (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science* 3.3, pp. 241–268.
- Wu, Cathy, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel (2018). Variance reduction for policy gradient with action-dependent factorized baselines. *In International Conference on Learning Representations*.
- Wu, Yuhuai, Elman Mansimov, Roger Grosse, Shun Liao, and Jimmy Ba (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *In Advances in Neural Information Processing Systems*, pp. 5279–5288.

Bibliography

- Xu, Zhongwen, Hado P van Hasselt, and David Silver (2018). Meta-Gradient Reinforcement Learning. *In Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.
- Yu, Chao, Jiming Liu, and Shamim Nematy (2019). Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*.
- Zhang, Amy, Nicolas Ballas, and Joëlle Pineau (2018). A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*.
- Zhang, Amy, Harsh Satija, and Joëlle Pineau (2018). Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*.
- Zhang, Chiyuan, Oriol Vinyals, Rémi Munos, and Samy Bengio (2018). A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*.
- Zhang, Marvin, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine (Sept. 2019). SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. *In Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 7444–7453.
- Zhang, Yufeng, Zhuoran Yang, and Zhaoran Wang (13–15 Apr 2021). Provably Efficient Actor-Critic for Risk-Sensitive and Robust Adversarial RL: A Linear-Quadratic Case. *In Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 2764–2772.
- Zhao, Tingting, Gang Niu, Ning Xie, Jucheng Yang, and Masashi Sugiyama (20–22 Nov 2016). Regularized Policy Gradients: Direct Variance Reduction in Policy Gradient Estimation. *In Asian Conference on Machine Learning*. Ed. by Geoffrey Holmes and Tie-Yan Liu. Vol. 45. Proceedings of Machine Learning Research. Hong Kong: PMLR, pp. 333–348.
- Zhou, Wenji, Yang Yu, Yingfeng Chen, Kai Guan, Tangjie Lv, Changjie Fan, and Zhi-Hua Zhou (July 2019). Reinforcement Learning Experience Reuse with Policy Residual Representation. *In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, pp. 4447–4453.
- Zhu, Yuke, Roozbeh Mottaghi, Eric Kolve, Joseph Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. *In IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3357–3364.

Appendices

The following appendices are complements to previous chapters. **Appendix A** complements Chapter 4 on using variance in the value function estimates as an auxiliary task, **Appendix B** complements Chapter 5 on using variance in the value function estimates to filter information, **Appendix C** complements Chapter 6 on using variance in the value function estimates to learn value functions, and **Appendix D** complements Chapter 8 on using adversarial priors to motivate conservatively diversified policies. Finally, in **Appendix E** we include examples of additional projects aimed at making a positive contribution to open research and education.

Appendix A

MERL *or* Using Variance in the Value Function estimates as an auxiliary task

A.1 Implementation of MERL coupled with DDPG

Deep Deterministic Policy Gradient (DDPG) (Lillicrap, Hunt, Pritzel, et al., 2016) is a model-free off-policy actor-critic algorithm, combining Determinist Policy Gradient (DPG) (Silver, Lever, Heess, et al., 2014) with Deep Q-Network (DQN) (Mnih, Kavukcuoglu, Silver, et al., 2013). While the original DQN works in discrete action space and stabilizes the learning of the Q-function with experience replay and a target network, DDPG extends it to continuous action space with the actor-critic framework while learning a deterministic policy.

Let P denote the distribution $P(\cdot|s, a)$ from which the next state s' is sampled. The Bellman equation describing the optimal action-value function $Q^*(s, a)$ is given by:

$$Q^*(s, a) = \mathbb{E}_{s' \sim P} \left[r(s, a) + \gamma \max_{a'} Q^*(s', a') \right]. \quad (\text{A.1})$$

Assuming the function approximator of $Q^*(s, a)$ is a neural network $Q_\phi(s, a)$ with parameters ϕ , an essential part of DDPG is that computing the maximum over actions is intractable in continuous action spaces, therefore the algorithm uses a target policy network to compute an action which approximately maximizes $Q_{\phi_{\text{target}}}$. Given the collection of transitions (s, a, r, s', d) in a set \mathcal{D} , where d denotes whether s' is terminal, we obtain the mean-squared Bellman error (MSBE) function with:

$$L^{\text{DDPG}}(\phi, \mathcal{D}) = \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} \left[\left(Q_\phi(s, a) - \left(r + \gamma(1 - d) Q_{\phi_{\text{target}}}(s', \mu_{\theta_{\text{target}}}(s')) \right) \right)^2 \right]. \quad (\text{A.2})$$

Algorithm 7 shows the pseudo-code for DDPG+MERL. In Equation A.3, the targets are computed, then in Equation A.4 and Equation A.5 respectively the Q-function and MERL^h are updated by one step of gradient descent (each MERL objective is associated with its loss coefficient c_h). In Equation A.6, the policy is updated by one step of gradient ascent. Finally, in Equation A.8, the targets networks are updated with ρ a hyperparameter between 0 and 1.

Algorithm 7 MERL coupled with DDPG.

Initialize policy parameters θ
Initialize Q-function and MERL^h functions parameters ϕ
Initialize empty replay buffer \mathcal{D}
Set target parameters: $\theta_{target} \leftarrow \theta$ and $\phi_{target} \leftarrow \phi$
while did not converge **do**
 Observe state s and select action a
 Execute action a in the environment
 Observe next state s' , reward r and done signal d to indicate whether s' is terminal
 Collect (s, a, r, s', d) in the replay buffer \mathcal{D} , if s' is terminal, reset the environment state

if time to update **then**
 for $k = 0, 1, 2, \dots$ **do**
 Randomly sample a batch $B = \{(s, a, r, s', d)\}$ of transitions from \mathcal{D} .
 Compute targets

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) \quad (\text{A.3})$$

Gradient Update

$$\phi_{k+1} \leftarrow \underset{\phi}{\operatorname{argmin}} \sum_B (Q_{\phi}(s, a) - y(r, s', d))^2 \quad (\text{A.4})$$

$$+ \underset{\phi}{\operatorname{argmin}} \sum_{h=0}^H c_h L^{\text{MERL}^h} \quad (\text{A.5})$$

$$\theta_{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \sum_B Q_{\phi}(s, \mu_{\theta}(s)) \quad (\text{A.6})$$

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi \quad (\text{A.7})$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta \quad (\text{A.8})$$

A.2 Implementation Details

The policy network used for the *MuJoCo* tasks is a fully-connected multi-layer perceptron with two hidden layers of 64 units. For *Atari 2600*, the neural network is shared between the policy

and the value function and is the same as in Mnih, Badia, Mirza, et al. (2016). Each MERL head MERL^h is composed of a fully-connected layer of 64 units and outputs the desired quantity.

A.3 Additional Results

A.3.1 Single-Task Learning: Continuous Control

Figure A.1 evaluates PPO+MERL on the complete set of 9 *MuJoCo* tasks. In Figure A.2, we evaluate MERL on 3 *MuJoCo* tasks in an off-policy setting, using DDPG (Silver, Lever, Heess, et al., 2014).

Note that, while others have reported similar issues in the open-sourced implementations we experimented with, including *baselines* from OpenAI, it is difficult to tune DDPG to reproduce results from other works even when using the reported hyperparameters. That is why in Figure A.2 we only include the tasks for which DDPG was performing well, and we evaluate MERL on those tasks. Those curves suggest that the auxiliary losses introduced by MERL allow to further improve the performance of an agent learning off-policy.

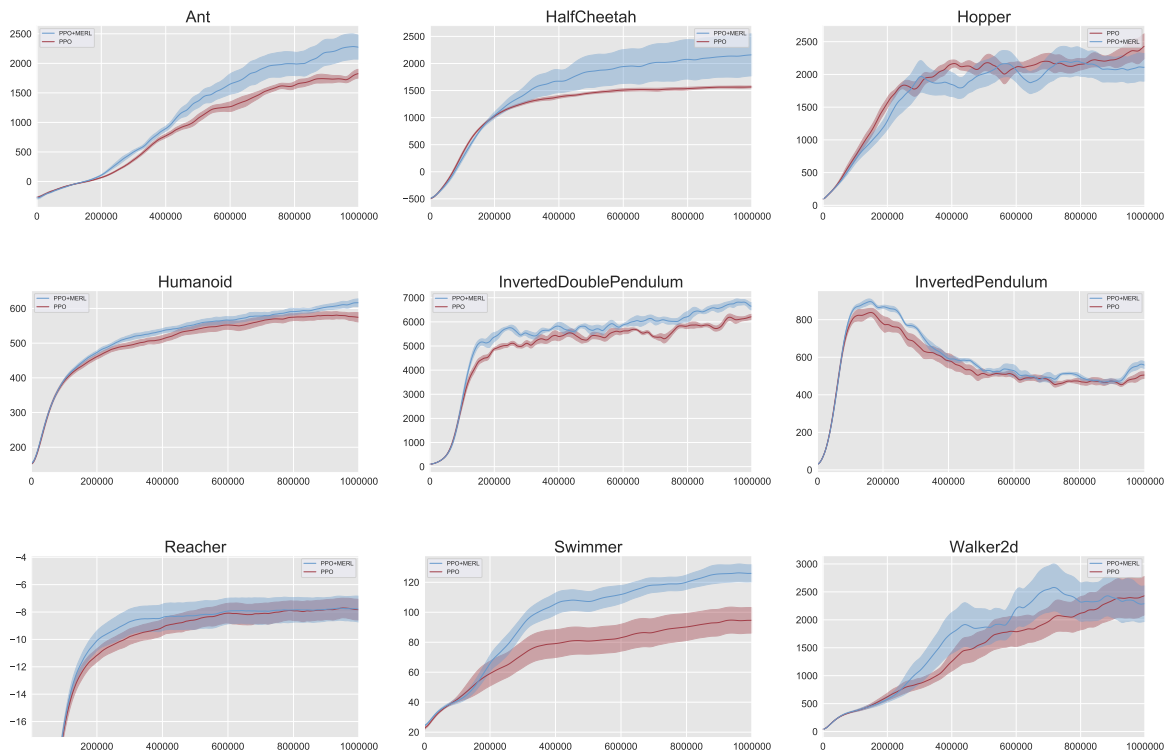


Figure A.1 – Experiments on 9 *MuJoCo* environments (10^6 timesteps, 7 seeds) with PPO+MERL. Red is the baseline, blue is with our method. The line is the average performance, while the shaded area represents its standard deviation.

MERL or Using Variance in the Value Function estimates as an auxiliary task

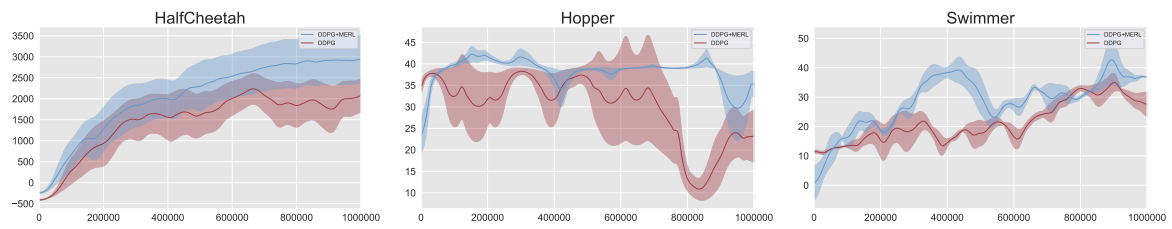


Figure A.2 – Experiments on 2 *MuJoCo* environments (10^6 timesteps, 7 seeds) with DDPG+MERL. Red is the baseline and blue our method. The line is the average performance, while the shaded area represents its standard deviation.

A.3.2 Transfer Learning: *Atari 2600* Domain

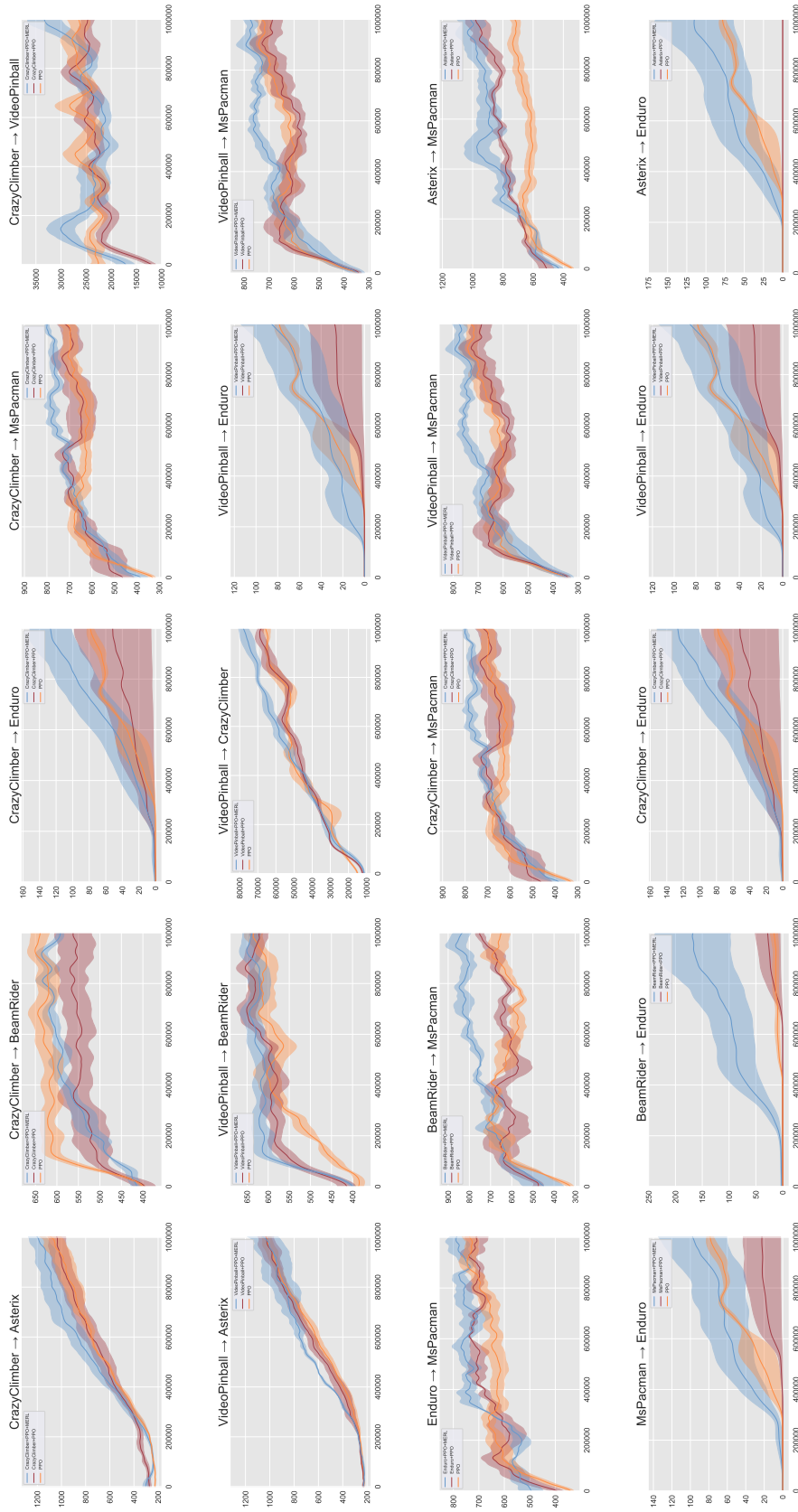


Figure A.3 – Experiments on the 20 transfer learning pairs with 6 *Atari* 2600 games (2×10^6 timesteps, 4 seeds). Orange is PPO solely trained on the subsequent 5 tasks, red is PPO transfer learning and blue is PPO+MERL transfer learning. The line is the average performance, while the shaded area represents its standard deviation.

A.4 Environment Details

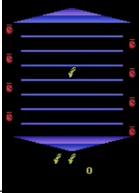

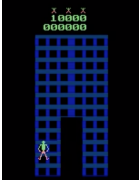

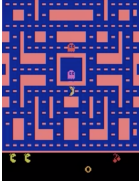
A.4.1 *MuJoCo* environments

Table A.1 – *MuJoCo* environments

Environment	Description
Ant-v2	Make a four-legged creature walk forward as fast as possible.
HalfCheetah-v2	Make a 2D cheetah robot run.
Hopper-v2	Make a two-dimensional one-legged robot hop forward as fast as possible.
Humanoid-v2	Make a three-dimensional bipedal robot walk forward as fast as possible, without falling over.
InvertedPendulum-v2	This is a <i>MuJoCo</i> version of CartPole. The agent’s goal is to balance a pole on a cart.
InvertedDoublePendulum-v2	This is a harder version of InvertedPendulum, where the pole has another pole on top of it. The agent’s goal is to balance a pole on a pole on a cart.
Reacher-v2	Make a 2D robot reach to a randomly located target.
Swimmer-v2	Make a 2D robot swim.
Walker2d-v2	Make a two-dimensional bipedal robot walk forward as fast as possible.

A.4.2 Atari 2600 games (action space size = 9)

Table A.2 – Every Atari 2600 games with action space size = 9

Environment	Screenshot	Description
AsterixNoFrameskip-v4		The agent guides Taz between the stage lines in order to eat hamburgers and avoid the dynamites.
BeamRiderNoFrameskip-v4		The agent's objective is to clear the Shield's 99 sectors of alien craft while piloting the BeamRider ship.
CrazyClimberNoFrameskip-v4		The agent assumes the role of a person attempting to climb to the top of four skyscrapers.
EnduroNoFrameskip-v4		Enduro consists of manoeuvring a race car. The objective of the race is to pass a certain number of cars each day. Doing so will allow the player to continue racing for the next day.
MsPacmanNoFrameskip-v4		The gameplay of Ms. Pac-Man is very similar to that of the original Pac-Man. The player earns points by eating pellets and avoiding ghosts.
VideoPinballNoFrameskip-v4		Video Pinball is a loose simulation of a pinball machine: ball shooter, flippers, bumpers and spinners.

Appendix B

SAUNA *or* Using Variance in the Value Function estimates to filter information

B.1 Additional Results

Table B.1 – Average total reward of the last 100 episodes over 6 runs on the 6 *MuJoCo* environments on PPO and A2C. **Boldface** $mean \pm std$ indicate better mean performance. (.%) is the change in performance due to SAUNA.

Task	PPO	PPO+SAUNA	A2C	A2C+SAUNA
HalfCheetah	2277 \pm 432	2929 \pm 169 (+29%)	1389 \pm 157	1731 \pm 147 (+25%)
Hopper	2106 \pm 133	2250 \pm 73 (+7%)	1367 \pm 110	1627 \pm 97 (+19%)
InvertedDoublePendulum	6100 \pm 143	6893 \pm 350 (+12%)	4151 \pm 67	5132 \pm 409 (+24%)
InvertedPendulum	532 \pm 19	609 \pm 24 (+14%)	686 \pm 15	684 \pm 10 (-0.3%)
Reacher	-7.5 \pm 0.8	-7.2 \pm 0.3 (+4%)	-9.2 \pm 0.8	-8.5 \pm 0.7 (+8%)
Swimmer	99.5 \pm 5.4	100.8 \pm 10.4 (+1%)	44.1 \pm 10.3	59.0 \pm 5.5 (+34%)

Appendix C

AVEC or Using Variance in the Value Function estimates to learn value functions

C.1 Proof of Section 6.4.1 results

In this section, we consider the case in which the state-action-value function of a policy π_θ is approximated. We prove that given some assumptions on this estimator function, we can use it to yield a valid gradient direction, *i.e.*, we are able to prove policy improvement when following this direction.

In this setting, the critic minimizes the following loss:

$$\mathbb{E}_{(s,a)\sim\pi} \left[(\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a) - \mathbb{E}_{(s,a)\sim\pi}[\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a)])^2 \right].$$

When a local optimum is reached, the gradient of the latter expression is zero:

$$\nabla_\phi \mathcal{L}_{\text{AVEC}} = \mathbb{E}_{(s,a)\sim\pi} \left[(\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a) - \mathbb{E}_{(s,a)\sim\pi}[\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a)]) \left(\frac{\partial f_\phi(s,a)}{\partial \phi} - \mathbb{E}_{(s,a)\sim\pi} \left[\frac{\partial f_\phi(s,a)}{\partial \phi} \right] \right) \right] = 0.$$

In the expression above, the expected value of the partial derivative disappears because the term in the first bracket is centered:

$$\mathbb{E}_{(s,a)\sim\pi} \left[(\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a) - \mathbb{E}_{(s,a)\sim\pi}[\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a)]) \mathbb{E}_{(s,a)\sim\pi} \left[\frac{\partial f_\phi(s,a)}{\partial \phi} \right] \right]$$

$$\begin{aligned}
 &= \mathbb{E}_{(s,a) \sim \pi} \left[\frac{\partial f_\phi(s,a)}{\partial \phi} \right] \mathbb{E}_{(s,a) \sim \pi} [\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a) - \mathbb{E}_{(s,a) \sim \pi} [\hat{Q}^{\pi_\theta} - f_\phi]] \\
 &= 0.
 \end{aligned}$$

Simplifying the gradient at the local optimum becomes:

$$\mathbb{E}_{(s,a) \sim \pi} \left[(\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a) - \mathbb{E}_{(s,a) \sim \pi} [\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a)]) \left(\frac{\partial f_\phi(s,a)}{\partial \phi} \right) \right] = 0. \quad (\text{C.1})$$

Then, if we denote $g_\phi = f_\phi(s,a) + \mathbb{E}_{(s,a) \sim \pi} [\hat{Q}^{\pi_\theta}(s,a) - f_\phi(s,a)]$, and use the policy parameterization assumption:

$$\frac{\partial f_\phi(s,a)}{\partial \phi} = \frac{\partial \pi_\theta(s,a)}{\partial \theta} \frac{1}{\pi_\theta(s,a)}, \quad (\text{C.2})$$

we obtain:

$$\boxed{\nabla_\theta J = \mathbb{E}_{(s,a) \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(s,a)) g_\phi(s,a)].} \quad (\text{C.3})$$

Proof. By combining the parameterization assumption in Equation C.2 with Equation C.1, we have:

$$\mathbb{E}_{(s,a) \sim \pi_\theta} \left[(\hat{Q}^{\pi_\theta}(s,a) - g_\phi(s,a)) \frac{\partial \pi_\theta(s,a)}{\partial \theta} \frac{1}{\pi_\theta(s,a)} \right] = 0. \quad (\text{C.4})$$

Since the expression above is null, we have the following:

$$\begin{aligned}
 \nabla_\theta J &= \mathbb{E}_{(s,a) \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(s,a)) \hat{Q}^{\pi_\theta}(s,a)] \\
 &= \mathbb{E}_{(s,a) \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(s,a)) \hat{Q}^{\pi_\theta}(s,a)] - \mathbb{E}_{(s,a) \sim \pi_\theta} [(\hat{Q}^{\pi_\theta}(s,a) - g_\phi(s,a)) \frac{\partial \pi_\theta(s,a)}{\partial \theta} \frac{1}{\pi_\theta(s,a)}] \\
 &= \mathbb{E}_{(s,a) \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(s,a)) g_\phi(s,a)].
 \end{aligned}$$

□

Remark C.1 (Assumption in Equation C.2). *While the proof seems more or less generic, the assumption in Equation C.2 is extremely constraining to the possible approximators. Sutton, McAllester, Singh, et al. (2000) quotes J. Tsitsiklis who believes that a linear g_ϕ in the features of the policy may be the only feasible solution for this condition. Concretely, such an assumption cannot hold since neural networks are the standard approximators used in practice. Moreover, empirical analysis (Ilyas, Engstrom, Santurkar, et al., 2020) indicates that commonly used algorithms fail to*

fit the true value function. However, this does not rule out the usefulness of the approach but rather begs for more questioning of the true effect of such biased baselines.

C.2 Additional Results

C.2.1 Comparative Evaluation of AVEC with TRPO

In order to evaluate the performance gains in using AVEC instead of the usual actor-critic framework, we produce some additional experiments with the TRPO (Schulman, Levine, Abbeel, et al., 2015) algorithm. Figure C.1 shows the learning curves while Table C.1 reports the results.

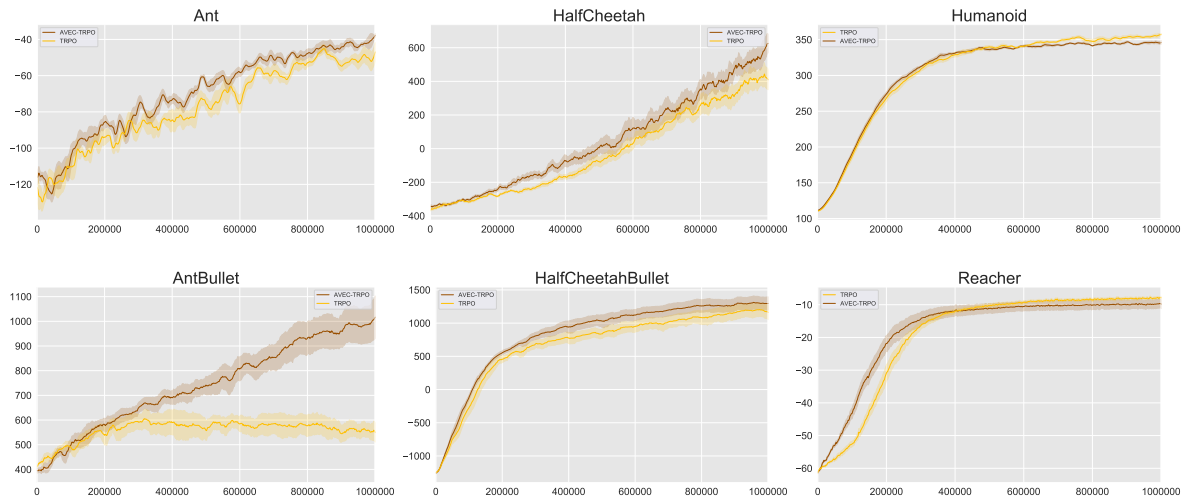


Figure C.1 – Comparative evaluation of AVEC with TRPO. We run with 6 different seeds: lines are average performances and shaded areas represent one standard deviation.

Table C.1 – Average total reward of the last 100 episodes over 6 runs of 10^6 timesteps. Comparative evaluation of AVEC with TRPO. \pm corresponds to a single standard deviation over trials and (.%) is the change in performance due to AVEC.

Task	TRPO	AVEC-TRPO
Ant	-50.5	-43.5 \pm 2.2 (+16%)
AntBullet	564	970 \pm 70 (+72%)
HCheetah	346	466 \pm 56 (+35%)
HCBullet	1154	1281 \pm 94 (+11%)
Humanoid	352	344 \pm 1.2 (-3%)
Reacher	-8.5	-9.9 \pm 1.3 (-16%)

C.2.2 Continuous Control: Walker2d

Figure C.2 shows the total average return for AVEC coupled with SAC and PPO on the Walker2d task. Similar to considered other continuous control tasks from *MuJoCo* and *PyBullet*, AVEC brings a significant performance improvement (+26% for SAC and +33% for PPO), confirming the generality of our approach.

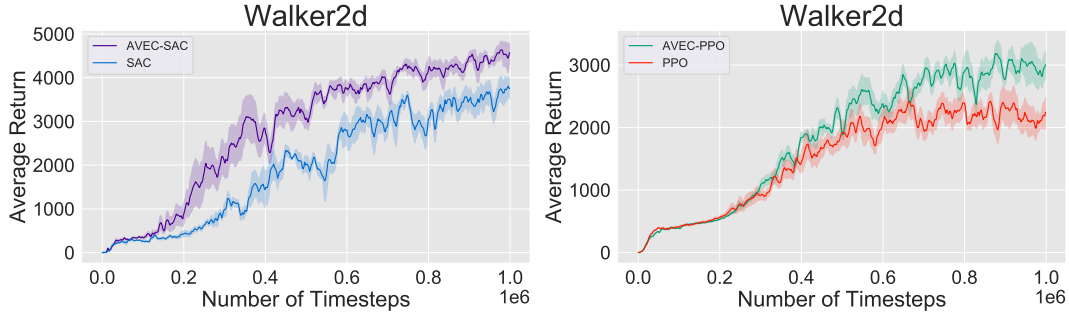


Figure C.2 – Comparative evaluation (6 seeds) of AVEC with SAC (left) and PPO (right) on the Walker2d *MuJoCo* task. Lines are average performances and shaded areas represent one standard deviation.

C.2.3 Variation of the Bias and Variance terms: AVEC +PPO

In Figure C.3, we show the variation of the bias and variance terms in the MSE between the estimators (of AVEC-PPO and PPO) and the true target: $\mathbb{E}[\|g_\phi - V^\pi\|_2^2] = \text{Bias}(\text{AVEC})^2 + \text{Var}(\text{AVEC})$ and $\mathbb{E}[\|V_\phi(\text{PPO}) - V^\pi\|_2^2] = \text{Bias}(\text{PPO})^2 + \text{Var}(\text{PPO})$ where $V_\phi(\text{PPO})$ is the value function estimator in PPO. We observe that the variance reduction is more substantial than that of the bias. Using those results and Figure 6.5 showing that the distance of the estimator to V^π is lower when using AVEC confirms that the variance reduction effect counterbalances the bias increase. Note that the % Variation of the Var term is always negative in our experiments, and that the shaded areas that suggest otherwise are merely due to a false assumption of symmetrical deviations, itself due to the assumption of Gaussianity needed to construct confidence intervals.

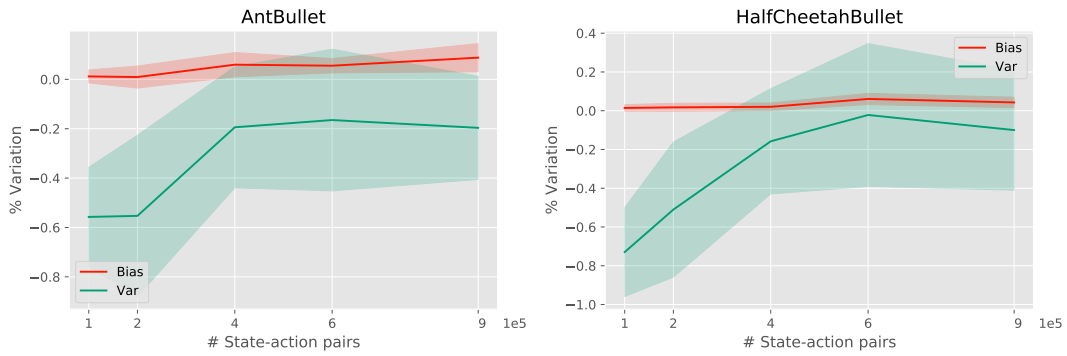


Figure C.3 – % variation of the bias and variance terms in the MSE between the estimator and the true target: $\% \text{Variation}(\text{Bias}) = \frac{\text{Bias}^2(\text{AVEC-PPO}) - \text{Bias}^2(\text{PPO})}{\text{Bias}^2(\text{PPO})}$ and $\% \text{Variation}(\text{Var}) = \frac{\text{Var}(\text{AVEC-PPO}) - \text{Var}(\text{PPO})}{\text{Var}(\text{PPO})}$. X-axis: we run PPO and AVEC-PPO and for every $t \in \{1, 2, 4, 6, 9\} \cdot 10^5$, we stop training, use the current policy to interact with the environment for $3 \cdot 10^5$ transitions, and use these transitions to estimate the true value function. Lines are average variations and shaded areas represent one standard deviation (5 seeds).

C.2.4 Learning the True Target: AVEC + SAC

In Figure C.4, we compare the error between the Q-function estimator and the true Q-function for SAC and AVEC-SAC in AntBullet and HalfCheetahBullet. We note a modest but consistent reduction in this error when using AVEC coupled with SAC, echoing the significant performance gains in Figure 6.2.

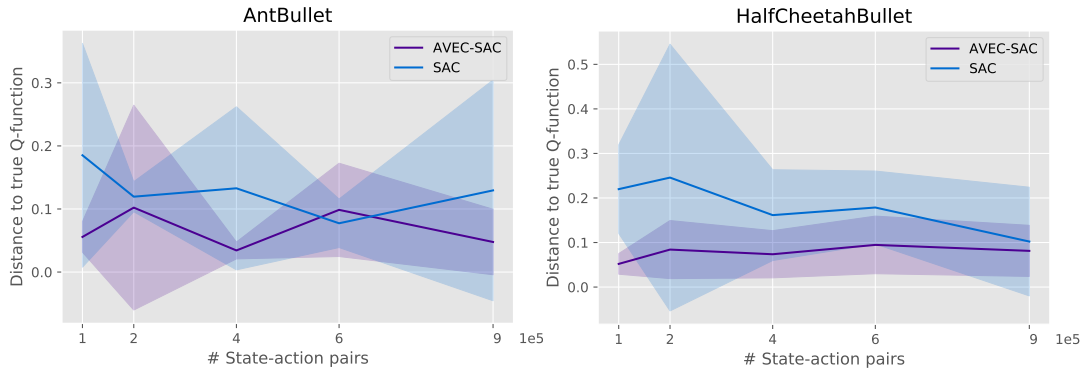


Figure C.4 – Distance to the true Q-function (SAC). X-axis: we run SAC and AVEC-SAC and for every $t \in \{1, 2, 4, 6, 9\} \cdot 10^5$ we stop training, use the current policy to interact with the environment for $3 \cdot 10^5$ transitions, and use these transitions to estimate the true value function. Lines are average performances and shaded areas represent one standard deviation.

C.2.5 Variance Reduction: AVEC + PPO

In Figure C.5, we study the empirical variance of the gradient in measuring the average pairwise cosine similarity (10 gradient measurements) in two additional tasks: HopperBullet and Walker2DBullet. We also vary the trajectory size used in the estimation of the gradient.

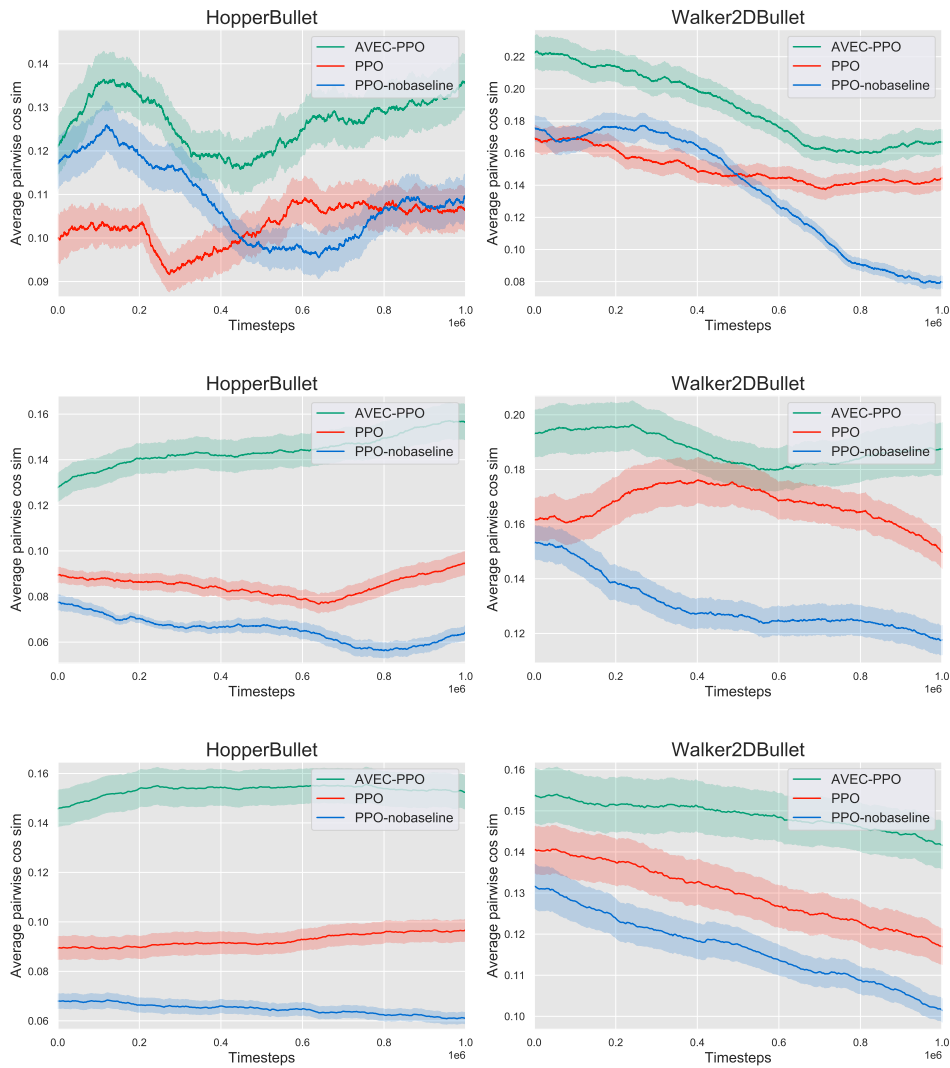


Figure C.5 – Average cosine similarity between gradient measurements. AVEC empirically reduces the variance compared to PPO or PPO without a baseline (PPO-nobaseline). Trajectory size used in estimation of the gradient variance: 3000 (upper row), 6000 (middle row), 9000 (lower row). Lines are average performances and shaded areas represent one standard deviation.

C.3 Experiment Details

In all experiments we choose to use the same hyperparameter values for all tasks as the best-performing ones reported in the literature or in their respective open source implementation documentation. We thus ensure the best performance for the conventional actor-critic framework. In other words, since we are interested in evaluating the impact of this new critic, everything else is kept as is. This experimental protocol may not benefit AVEC.

In Table C.2, C.3 and C.4, we report the list of hyperparameters common to all continuous control experiments.

Table C.2 – Hyperparameters used both in SAC and AVEC-SAC.

Parameter	Value
Adam stepsize	$3 \cdot 10^{-4}$
Discount (γ)	0.99
Replay buffer size	10^6
Batch size	256
Nb. hidden layers	2
Nb. hidden units per layer	256
Nonlinearity	ReLU
Target smoothing coefficient (τ)	0.01
Target update interval	1
Gradient steps	1

Table C.3 – Hyperparameters used both in PPO and AVEC-PPO.

Parameter	Value
Horizon (T)	2048
Adam stepsize	$2.5 \cdot 10^{-4}$
Nb. epochs	10
Nb. minibatches	32
Nb. hidden layers	2
Nb. hidden units per layer	64
Nonlinearity	tanh
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ε)	0.2

Table C.4 – Hyperparameters used both in TRPO and AVEC-TRPO.

Parameter	Value
Horizon (T)	2048
Adam stepsize	$1 \cdot 10^{-4}$
Nb. hidden layers	2
Nb. hidden units per layer	64
Nonlinearity	tanh
Discount (γ)	0.99
GAE parameter (λ)	0.95
Stepsize KL	0.01
Nb. iterations for the conjugate gradient	15

C.4 Implementation Details

C.4.1 Implementation of AVEC in practice

Theoretically, $\mathcal{L}_{\text{AVEC}}$ is defined as the residual variance of the value function (cf. Equation 6.3). However, state-values for a non-optimal policy are dependent and the variance is not tractable without access to the joint law of state-values. Consequently, to implement AVEC in practice we use the best-known proxy at hand, which is the empirical variance formula assuming independence:

$$\mathcal{L}_{\text{AVEC}} = \frac{1}{T-1} \sum_{t=1}^T \left((f_{\phi}(s_t) - \hat{V}^{\pi}(s_t)) - \frac{1}{T} \sum_{t=1}^T (f_{\phi}(s_t) - \hat{V}^{\pi}(s_t)) \right)^2,$$

where T is the size of the sampled trajectory.

C.4.2 Implementation of AVEC coupled with SAC

In Algorithm 8, J_V is the squared residual error objective to train the soft value function. See Haarnoja, Zhou, Abbeel, et al. (2018) for further details and notations about SAC, not directly relevant here.

C.5 Environment Details

Algorithm 8 AVEC coupled with SAC.

- 1: **Input parameters:** $\beta \in [0, 1], \lambda_V \geq 0, \lambda_Q \geq 0, \lambda_\pi \geq 0$
 - 2: **Initialize** policy parameter θ , value function parameter ψ and $\bar{\psi}$ and Q-functions parameters ϕ_1 and ϕ_2
 - 3: $\mathcal{D} \leftarrow \emptyset$
 - 4: **for** each iteration **do**
 - 5: **for** each step **do**
 - 6: $a_t \sim \pi_\theta(a_t|s_t)$
 - 7: $s_{t+1} \sim \mathcal{P}(s_t, a_t)$
 - 8: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
 - 9: **for** each gradient step **do**
 - 10: sample batch \mathcal{B} from \mathcal{D}
 - 11: $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
 - 12: $\phi_i \leftarrow \phi_i - \lambda_Q \hat{\nabla}_{\phi_i} \mathcal{L}_{\text{AVEC}}^2(\phi_i)$ for $i \in \{1, 2\}$
 - 13: $\theta \leftarrow \theta - \lambda_\pi \hat{\nabla}_\theta J(\pi_\theta)$
 - 14: $\bar{\psi} \leftarrow \beta\psi + (1 - \beta)\bar{\psi}$
-

Table C.5 – Environment details.

Environment	Description
Ant-v2	Make a four-legged creature walk forward as fast as possible.
AntBulletEnv-v0	Idem. Ant is heavier, encouraging it to typically have two or more legs on the ground (source: <i>PyBullet</i> Guide - url).
HalfCheetah-v2	Make a 2D cheetah robot run.
HalfCheetahBulletEnv-v0	Idem.
Humanoid-v2	Make a three-dimensional bipedal robot walk forward as fast as possible, without falling over.
Reacher-v2	Make a 2D robot reach to a randomly located target.
Walker2d-v0	Make a 2D robot walk forward as fast as possible.
Walker2DBulletEnv-v0	Idem.
HopperBulletEnv-v0	Make a two-dimensional one-legged robot hop forward as fast as possible
Acrobot-v1	Swing the end of a two-joint acrobot up to a given height.
MountainCar-v0	Get an under powered car to the top of a hill.

Table C.6 – Actions and observations dimensions.

Task	\mathcal{S}	\mathcal{A}
Ant	\mathbb{R}^{111}	\mathbb{R}^8
AntBullet	\mathbb{R}^{28}	\mathbb{R}^8
HalfCheetah	\mathbb{R}^{17}	\mathbb{R}^6
HalfCheetahBullet	\mathbb{R}^{26}	\mathbb{R}^6
Humanoid	\mathbb{R}^{376}	\mathbb{R}^{17}
Reacher	\mathbb{R}^{11}	\mathbb{R}^2
Walker2d	\mathbb{R}^{17}	\mathbb{R}^6
Walker2DBullet	\mathbb{R}^{22}	\mathbb{R}^6
HopperBullet	\mathbb{R}^{15}	\mathbb{R}^3
Acrobot	\mathbb{R}^6	3
MountainCar	\mathbb{R}^2	3

Appendix D

AGAC *or* Using Adversarial Priors to motivate conservatively diversified policies

D.1 Additional Results

D.1.1 MiniGrid Performance

In this section, we report the final performance of all methods considered in the MiniGrid experiments of Figure 8.2 with the scores reported in Raileanu and Rocktäschel (2019) and Campero, Raileanu, Küttler, et al. (2021). All methods have a budget of 200M frames.

Table D.1 – Final average performance of all methods on several MiniGrid environments.

Task	KC-S4R3	KC-S5R3	MR-N10S10	OM-2Dlhb	OM-1Q	OM-2Q
AGAC	0.95	0.93	0.52	0.64	0.78	0.63
RIDE	0.19	0.	0.40	0.	0.	0.
AMIGo	0.54	0.	0.	0.20	0.	0.
RND	0.	0.	0.	0.03	0.	0.
Count	0.	0.	0.	0.	0.	0.
ICM	0.	0.	0.	0.	0.	0.

D.1.2 State Visitation Heatmaps in Singleton Environment with No Extrinsic Reward

In this section, we provide additional state visitation heatmaps. The agent has been trained on a singleton environment from the MultiRoomN10S6 task without extrinsic reward. The last

ten episodes of the training suggest that although the agent experiences the same maze over and over again, the updates force it to change behavior and try new strategies.

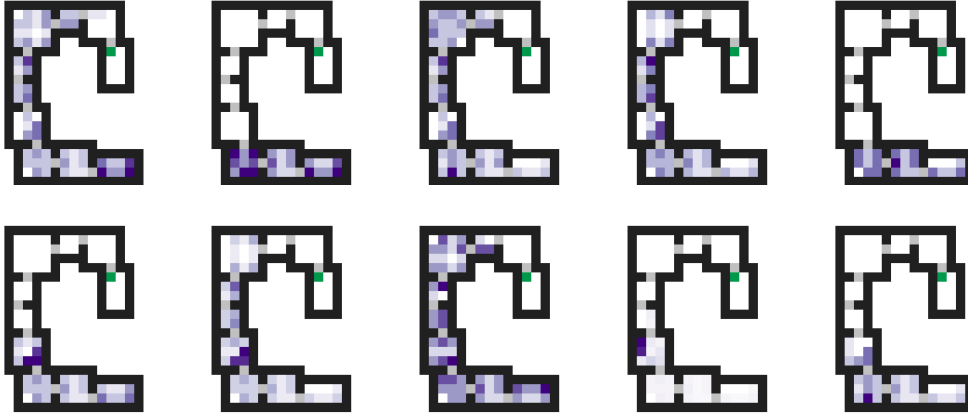


Figure D.1 – State visitation heatmaps of the last ten episodes of an agent trained in a *singleton* environment with no extrinsic reward 10M timesteps in the MultiRoomN10S6 task. The agent is continuously engaging into new strategies.

D.1.3 (Extremely) Hard-Exploration Tasks with Partially-Observable Environments

In this section, we include additional experiments on one of the hardest tasks available in MiniGrid. The first is KeyCorridorS8R3, where the size of the rooms has been increased. In it, the agent has to pick up an object which is behind a locked door: the key is hidden in another room and the agent has to explore the environment to find it. The second, ObstructedMazeFull, is similar to ObstructedMaze4Q, where the agent has to pick up a box which is placed in one of the four corners of a 3x3 maze: the doors are locked, the keys are hidden in boxes and the doors are obstructed by balls. In those difficult tasks, only our method succeeds in exploring well enough to find rewards.

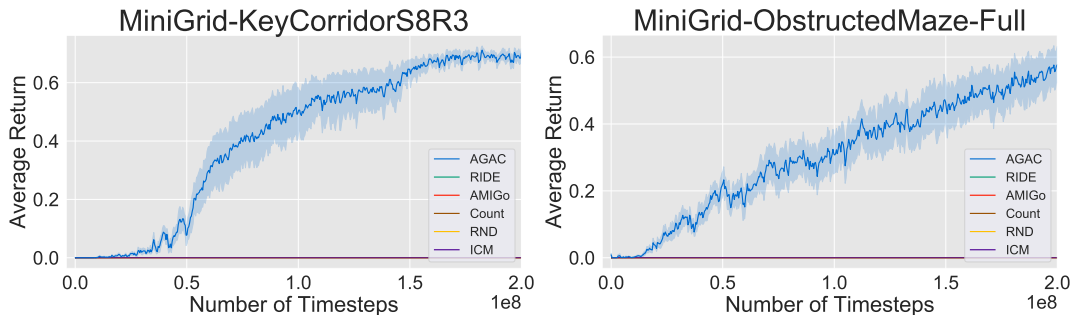


Figure D.2 – Performance evaluation of AGAC compared to RIDE, AMIGo, Count, RND and ICM on extremely hard-exploration problems.

D.1.4 Mean Intrinsic Reward

In this section, we report the mean intrinsic reward computed for an agent trained in MultiRoomN12S10 to conveniently compare our results with that of Raileanu and Rocktäschel (2019). We observe in Figure D.3 that the intrinsic reward is consistently larger for our method and that, contrary to other methods, does not converge to low values. Please note that, in all considered experiments, the adversarial bonus coefficient c in Equations 8.2 and 8.3 is linearly annealed throughout the training since it is mainly useful at the beginning of learning when the rewards have not yet been met. In the long run, this coefficient may prevent the agent from solving the task by forcing it to always favour exploration over exploitation.

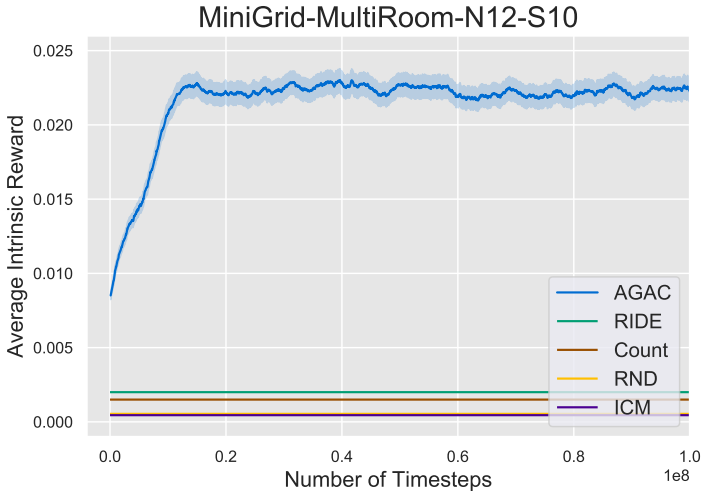


Figure D.3 – Average intrinsic reward for different methods trained in MultiRoomN12S10.

D.2 Illustration of AGAC



Figure D.4 – A simple schematic illustration of AGAC. Left: the adversary minimizes the KL-divergence with respect to the action probability distribution of the actor. Right: the actor receives a bonus when counteracting the predictions of the adversary.

D.3 Experiment Details

D.3.1 MiniGrid setup

Here, we describe in more details the experimental setup we used in our MiniGrid experiments.

There are several different MiniGrid scenarios that we consider in Chapter 8. MultiRoom corresponds to a set of navigation tasks, where the goal is to go from a starting state to a goal state. The notation MultiRoom- $N2S4$ means that there are 2 rooms in total, and that each room has a maximal side of 4. In order to go from one room to another, the agent must perform a specific action to open a door. Episodes are terminated with zero reward after a maximum of $20 \times N$ steps with N the number of rooms. In KeyCorridor, the agent also has to pick up a key, since the goal state is behind a door that only lets it in with the key. The notation KeyCorridor-S3R4 means that there are 4 side corridors, leading to rooms that have a maximal side of 3. The maximum number of steps is 270. In ObstructedMaze, keys are hidden in boxes, and doors are obstructed by balls the agent has to get out of its way. The notation ObstructedMaze-1Dl means that there are two connected rooms of maximal side 6 and 1 door (versus a 3x3 matrix and 2 doors if the leading characters are $2D$), adding h as a suffix places keys in boxes, and adding b as a suffix adds balls in front of doors. Using Q as a suffix is equivalent to using lhb (that is, both hiding keys and placing balls to be moved). The maximum number of steps is 576. ObstructedMazeFull is the hardest configuration for this scenario, since it has the maximal number of keys, balls to move, and doors possible.

In each scenario, the agent has access to a partial view of the environment, a 7x7 square that includes itself and points in the direction of its previous movement.

D.3.2 Hyperparameters

In all experiments, we train six different instances of our algorithm with different random seeds. In Table D.2, we report the list of hyperparameters.

Table D.2 – Hyperparameters used in AGAC.

Parameter	Value
Horizon T	2048
Nb. epochs	4
Nb. minibatches	8
Nb. frames stacked	4
Nonlinearity	ELU (Clevert, Unterthiner, and Hochreiter, 2016)
Discount γ	0.99
GAE parameter λ	0.95
PPO clipping parameter ε	0.2
β_V	0.5
c	$4 \cdot 10^{-4}$ ($4 \cdot 10^{-5}$ in VizDoom)
c anneal schedule	linear
β_{adv}	$4 \cdot 10^{-5}$
Adam stepsize η_1	$3 \cdot 10^{-4}$
Adam stepsize η_2	$9 \cdot 10^{-5} = 0.3 \cdot \eta_1$

D.4 Implementation Details

In Figure D.5 is depicted the architecture of our method.

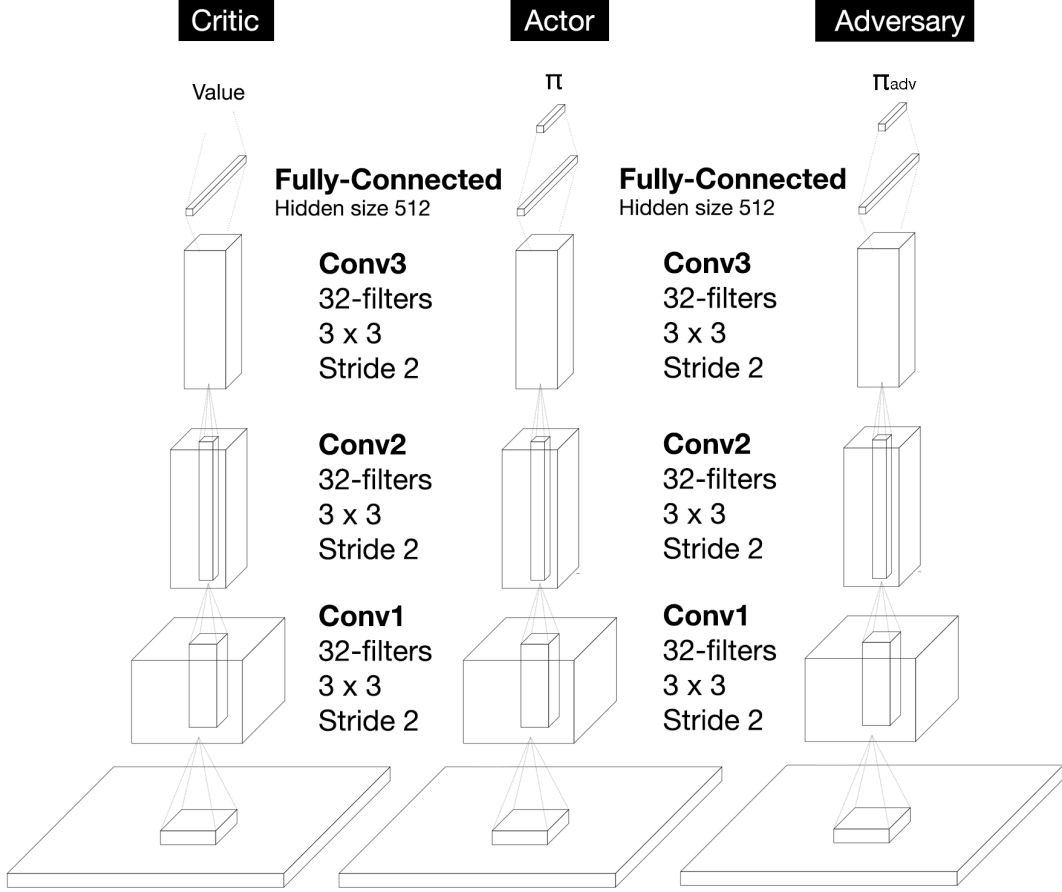


Figure D.5 – Artificial neural architecture of the critic, the actor and the adversary.

D.5 Proof of Section 8.4.1 results

In this section, we provide a short proof for the result of the optimization problem in Section 8.4.1. We recall the result here:

$$\pi_{k+1} = \arg \max_{\pi} \mathcal{J}_{PI}(\pi) \propto \left(\frac{\pi_k}{\pi_{adv}} \right)^{\frac{c}{\alpha}} \exp \frac{Q_{\pi_k}}{\alpha},$$

with the objective function:

$$\mathcal{J}_{PI}(\pi) = \mathbb{E}_s \mathbb{E}_{a \sim \pi(\cdot|s)} [Q_{\pi_k}(s, a) + c (\log \pi_k(a|s) - \log \pi_{adv}(a|s)) - \alpha \log \pi(a|s)].$$

Proof. We first consider a simpler optimization problem: $\arg \max_{\pi} \langle \pi, Q_{\pi_k} \rangle + \alpha \mathcal{H}(\pi)$, whose solution is known (Vieillard, Kozuno, Scherrer, et al., 2020, Appendix A). The expression for the maximizer is the α -scaled softmax:

$$\pi^* = \frac{\exp(\frac{Q_{\pi_k}}{\alpha})}{\langle 1, \exp(\frac{Q_{\pi_k}}{\alpha}) \rangle}.$$

We now turn towards the optimization problem of interest, which we can rewrite as:

$$\arg \max_{\pi} \langle \pi, Q_{\pi_k} + c (\log \pi_k - \log \pi_{\text{adv}}) \rangle + \alpha \mathcal{H}(\pi).$$

By the simple change of variable $\tilde{Q}_{\pi_k} = Q_{\pi_k} + c (\log \pi_k - \log \pi_{\text{adv}})$, we can reuse the previous solution (replacing Q_{π_k} by \tilde{Q}_{π_k}). With the simplification:

$$\exp \frac{Q_{\pi_k} + c (\log \pi_k - \log \pi_{\text{adv}})}{\alpha} = \left(\frac{\pi_k}{\pi_{\text{adv}}} \right)^{\frac{c}{\alpha}} \exp \frac{Q_{\pi_k}}{\alpha},$$

we obtain the result and conclude the proof. □

Appendix E

Experimentation and Learning in Deep Reinforcement Learning

E.1 Reproducible Experiments

We have published online and open-sourced the code of all the work presented in this thesis: [MERL](#), [SAUNA](#), [AVEC](#) and [AGAC](#). Besides, we believe that the publication of the code may not be sufficient to allow a good use of the method and a good understanding of the implementation, as such, my co-authors and I have done and still do our best to remain accessible when other researchers, Ph.D. students or master students contact us regarding our methods and code.

E.2 Contributions to Open Research and Education

Below, we present two Python libraries that we contributed to and which are published online under an open-source license.

E.2.1 `RLBERRY` software

`RLBERRY` (github.com/rlberry-py/rlberry) is an RL library designed for research and educational purposes (Domingues, Flet-Berliac, Leurent, et al., 2021). It has been conceived to allow the user to concentrate on the actual RL method to implement rather than on the more troublesome and error-prone elements like running agents in parallel, reliably plotting results, optimizing hyperparameters, comparing to baselines, or creating custom environments.

In a nutshell, the main features of `RLBERRY` is that it:

- Provides implementations of several RL agents to use as a starting point or as baselines;

Experimentation and Learning in Deep Reinforcement Learning

- Provides a set of benchmark environments, very useful to debug and challenge one's own algorithms;
- Handles all random seeds consolidated way, ensuring experiment reproducibility;
- Provides multiple notebooks that can be run directly from the browser through Google Colab;
- Is fully compatible with several commonly used RL libraries like OpenAI gym (Brockman, Cheung, Pettersson, et al., 2016) and Stable Baselines (Hill, Raffin, Ernestus, et al., 2018).

E.2.2 RLSS-2019 materials

In the context of the Reinforcement Learning Summer School (RLSS), that we organized as members of the Scool/SequeL team at Inria in July 2019, and which consisted of two weeks dedicated to the theory and practice of sequential decision making, a number of materials for the practical sessions have been created by numerous contributors: senior developers and researchers, Ph.D. students and M.Sc. students. RLSS-2019 (github.com/yfletberliac/rlss-2019) includes tutorials on how to set up a computer for development for participants with a potentially non-technical background, notebooks with tutorials on bandits, RL, and deep RL, and a larger project based on TextWorld (Côté, Kádár, Yuan, et al., 2019) which produces text-based quests of custom complexity and MiniWoB (Shi, Karpathy, Fan, et al., 2017) which is a set of browser-based tasks simulating human interaction with webpages. All notebooks can be run directly from the browser through Google Colab notebooks.

List of Figures

1.1	Agent-environment interaction.	4
1.2	Taxonomy of RL algorithms.	6
1.3	Outline of this thesis structured around the actor-critic components.	8
2.1	Illustration of a feed-forward multi-layer network.	18
2.2	Loss surfaces as a function of network weights in a two-dimensional subspace.	20
2.3	Illustrations of the score function gradient estimator mechanics. From Schulman (2017) and Levine (2017).	25
3.1	Negative of the empirical value function ($-\max_a \hat{Q}^\pi(s, a)$) learned on a single run by an agent in the Mountain Car task.	36
3.2	Examples of continuous control tasks.	37
4.1	Schematic overview of MERL system.	44
4.2	Experiments on 6 <i>MuJoCo</i> environments (10^6 timesteps, 7 seeds) with PPO+MERL. Red is the baseline, blue is with our method.	49
4.3	Transfer learning tasks from 5 <i>Atari 2600</i> games to Ms. Pacman (2×10^6 timesteps, 6 seeds). Performance on the second task. Orange is PPO solely trained on Ms. Pacman, red and blue are respectively PPO and our method transferring the learning.	50
4.4	Ablation experiments with only one MERL head (FS or VE) (10^6 timesteps, 6 seeds). Blue is MERL with the two heads, red with the FS head, green with the VE head and orange with no MERL head.	50
5.1	Performance of PPO+SAUNA (red) relative to PPO (blue) on 6 <i>MuJoCo</i> environments averaged across 6 seeds.	61
5.2	Performance of PPO+SAUNA (red) relative to PPO (blue) on the <i>Roboschool</i> environment averaged across 6 seeds.	62

List of Figures

5.3	Performance of PPO+SAUNA (red) relative to PPO (blue) and PPO with the prediction of \mathcal{V}^{ex} but without the filtering out of noisy samples (green) on 3 <i>MuJoCo</i> environments averaged across 6 seeds.	62
5.4	Gradients L1-norm from the first layer and last layer of the shared parameters network for PPO and PPO coupled with SAUNA. Task: <i>HalfCheetah-v2</i>	63
5.5	(a) Example of PPO getting trapped in a local minimum (top row) while PPO+SAUNA reaches a better optimum (bottom row). (b) \mathcal{V}^{ex} score for PPO (blue) and PPO+SAUNA (orange).	63
6.1	Comparison of simple models derived when \mathcal{L}_{AVEC} is used instead of the MSE.	73
6.2	Comparative evaluation (6 seeds) of AVEC with SAC and PPO on PyBullet (“TaskBullet”) and MuJoCo (“Task”) tasks.	77
6.3	(a,b): Comparative evaluation (6 seeds) of AVEC in sparse reward tasks. (c,d): Respectively state visitation frequency and phase portrait of visited states of AVEC-TRPO (green) and TRPO (red) in MountainCar.	78
6.4	L_2 distance to \hat{V}^π	79
6.5	L_2 distance to V^π . X-axis: we run PPO and AVEC-PPO and $\forall t \in \{1, 2, 4, 6, 9\} \cdot 10^5$ we stop training, use the current policy to collect $3 \cdot 10^5$ transitions and estimate V^π	79
6.6	Average gradient cosine-similarity.	80
6.7	Sensitivity analysis (6 seeds) of AVEC-PPO with respect to (a,b): the bias; (c,d): the variance.	81
7.1	Frames from (a,b) MultiRoom. (c) KeyCorridor. (d) ObstructedMaze.	88
7.2	Frames from the 3-D navigation task VizdoomMyWayHome.	89
7.3	Frames from the Cart-Pole Swing-Up task constrained on the pole angular velocity to be below a certain threshold when arriving near the top.	91
8.1	Frames from (a,b) the 3-D navigation task VizdoomMyWayHome. (c) MiniGrid-KeyCorridorS6R3. (d) MiniGrid-ObstructedMazeFull.	100
8.2	Performance evaluation of AGAC.	103
8.3	Sensitivity analysis of AGAC in KeyCorridorS4R3.	104
8.4	Average return on N10S6 with and without extrinsic reward.	104
8.5	State visitation heatmaps for RND, Count, a random uniform policy, RIDE, and AGAC trained in a singleton environment (top row) and procedurally-generated environments (bottom row) without extrinsic reward for 10M timesteps in the MultiRoomN10S6 task.	105
8.6	State visitation heatmaps of the last ten episodes of an agent trained in <i>procedurally-generated</i> environments without extrinsic reward for 10M timesteps in the MultiRoomN10S6 task. The agent is continuously engaging in new strategies.	106

9.1	Schematic overview of the Safe Adversarially guided Actor-Critics (SAAC) algorithm.	117
9.2	Constraints: SAAC vs. baselines.	122
9.3	Visualization of visited state space at different stages of learning in the <i>realworldrl-walker-walk</i> task.	123
A.1	Experiments on 9 <i>MuJoCo</i> environments (10^6 timesteps, 7 seeds) with PPO+MERL. Red is the baseline, blue is with our method.	161
A.2	Experiments on 2 <i>MuJoCo</i> environments (10^6 timesteps, 7 seeds) with DDPG+MERL. Red is the baseline and blue our method.	162
A.3	Experiments on the 20 transfer learning pairs with 6 <i>Atari 2600</i> games (2×10^6 timesteps, 4 seeds). Orange is PPO solely trained on the subsequent 5 tasks, red is PPO transfer learning and blue is PPO+MERL transfer learning.	163
C.1	Comparative evaluation of AVEC with TRPO.	167
C.2	Comparative evaluation (6 seeds) of AVEC with SAC (left) and PPO (right) on the Walker2d <i>MuJoCo</i> task.	168
C.3	% variation of the bias and variance terms in the MSE between the estimator and the true target: $\% \text{Variation}(\text{Bias}) = \frac{\text{Bias}^2(\text{AVEC-PPO}) - \text{Bias}^2(\text{PPO})}{\text{Bias}^2(\text{PPO})}$ and $\% \text{Variation}(\text{Var}) = \frac{\text{Var}(\text{AVEC-PPO}) - \text{Var}(\text{PPO})}{\text{Var}(\text{PPO})}$. X-axis: we run PPO and AVEC-PPO and for every $t \in \{1, 2, 4, 6, 9\} \cdot 10^5$, we stop training, use the current policy to interact with the environment for $3 \cdot 10^5$ transitions, and use these transitions to estimate the true value function.	169
C.4	Distance to the true Q-function (SAC). X-axis: we run SAC and AVEC-SAC and for every $t \in \{1, 2, 4, 6, 9\} \cdot 10^5$ we stop training, use the current policy to interact with the environment for $3 \cdot 10^5$ transitions, and use these transitions to estimate the true value function.	170
C.5	Average cosine similarity between gradient measurements. AVEC empirically reduces the variance compared to PPO or PPO without a baseline (PPO-nobaseline). Trajectory size used in estimation of the gradient variance: 3000 (upper row), 6000 (middle row), 9000 (lower row).	171
D.1	State visitation heatmaps of the last ten episodes of an agent trained in a <i>singleton</i> environment with no extrinsic reward 10M timesteps in the MultiRoomN10S6 task. The agent is continuously engaging into new strategies.	178
D.2	Performance evaluation of AGAC compared to RIDE, AMIGo, Count, RND and ICM on extremely hard-exploration problems.	178
D.3	Average intrinsic reward for different methods trained in MultiRoomN12S10.	179
D.4	A simple schematic illustration of AGAC. Left: the adversary minimizes the KL-divergence with respect to the action probability distribution of the actor. Right: the actor receives a bonus when counteracting the predictions of the adversary.	179

List of Figures

D.5 Artificial neural architecture of the critic, the actor and the adversary. [182](#)

List of Algorithms

- 1 Cross Entropy Method (CEM). 24
- 2 Vanilla Policy Gradient (VPG). 26
- 3 MERL coupled with PPO. 46
- 4 SAUNA coupled with PPO. 60
- 5 AVEC coupled with PPO or TRPO. 75
- 6 SAAC. 118
- 7 MERL coupled with DDPG. 160
- 8 AVEC coupled with SAC. 174

List of Tables

4.1	Hyperparameters used in PPO+MERL	47
4.2	MERL hyperparameters	47
4.3	Average total reward of the last 100 episodes over 7 runs on the 9 <i>MuJoCo</i> environments. Boldface <i>mean</i> \pm <i>std</i> indicate statistically better performance.	48
6.1	Average total reward of the last 100 episodes over 6 runs of 10^6 timesteps. Comparative evaluation of AVEC with SAC and PPO. \pm corresponds to a single standard deviation over trials and (.%) is the change in performance due to AVEC.	76
7.1	Some of the safety constraints on Cart-Pole, Walker and Quadruped.	90
8.1	Average return in VizDoom at different timesteps.	101
9.1	Comparison of SAAC variants.	121
9.2	In <i>quadruped-upright-walk</i>	121
9.3	In <i>quadruped-joint-walk</i>	121
A.1	<i>MuJoCo</i> environments	164
A.2	Every <i>Atari 2600</i> games with action space size = 9	164
B.1	Average total reward of the last 100 episodes over 6 runs on the 6 <i>MuJoCo</i> environments on PPO and A2C. Boldface <i>mean</i> \pm <i>std</i> indicate better mean performance. (.%) is the change in performance due to SAUNA.	164
C.1	Average total reward of the last 100 episodes over 6 runs of 10^6 timesteps. Comparative evaluation of AVEC with TRPO. \pm corresponds to a single standard deviation over trials and (.%) is the change in performance due to AVEC.	167
C.2	Hyperparameters used both in SAC and AVEC-SAC.	172
C.3	Hyperparameters used both in PPO and AVEC-PPO.	172
C.4	Hyperparameters used both in TRPO and AVEC-TRPO.	173

C.5 Environment details.	174
C.6 Actions and observations dimensions.	175
D.1 Final average performance of all methods on several MiniGrid environments. .	177
D.2 Hyperparameters used in AGAC.	181