Informatics

Weighting Intrinsic and Extrinsic Rewards in Reinforcement Learning

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Rafael Sterzinger
Matrikelnummer 11778282

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Grechenig
Mitwirkung: Dipl.-Ing. Michael Ressmann

Wien, 18. August 2020

Unterschrift Verfasser | Unterschrift Betreuung

Informatics

Weighting Intrinsic and Extrinsic Rewards in Reinforcement Learning

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Business Informatics

by

Rafael Sterzinger
Registration Number 11778282

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Grechenig
Assistance: Dipl.-Ing. Michael Ressmann

Vienna, 18th August, 2020

Signature Author

Signature Advisor

Weighting Intrinsic and Extrinsic Rewards in Reinforcement Learning

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Rafael Sterzinger
Matrikelnummer 11778282

ausgeführt am
Institut für Information Systems Engineering
Forschungsbereich Business Informatics
Forschungsgruppe Industrielle Software
der Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thomas Grechenig

Wien, 18. August 2020

Erklärung zur Verfassung der Arbeit

Rafael Sterzinger

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18. August 2020

Rafael Sterzinger

Danksagung

Ich möchte diese Arbeit meinen beiden Eltern widmen, welche mir mein Studium überhaupt erst ermöglicht haben und mich immer in meinem Bestreben ermutigten. Ein besonderer Dank kommt hierbei meinem Vater zu, welcher mir während der Verfassung dieser Bachelorarbeit immer wieder die unerlässlichen kleinen Dinge des Alltags abgenommen hat.

Ebenfalls ein großes Dankeschön möchte ich an meinen Betreuer Michael Ressmann aussprechen, welcher mir während der Planung und Durchführung der Arbeit den nötigen Freiraum gegeben hat, zeitgleich aber darauf geachtet hat, dass ich mich in meinem Vorhaben nicht übernehme. Außerdem möchte ich mich bei ihm für sein gutes Feedback, die schnelle und ausführliche Beantwortungen meiner Fragen sowie für die zügige Durchsicht meiner Bachelorarbeit bedanken.

Zu guter Letzt richte ich auch ein Dankeschön an all meine Freunde und Verwandte, welche mich während der Anfertigung dieser Arbeit auf unterschiedlichste Art und Weise unterstützten.

Acknowledgements

I would like to dedicate this work to both of my parents, who made my studies possible in the first place and always encouraged me in my aspirations. A special thanks goes to my father, who repeatedly relieved me of the unavoidable chores of everyday life while writing this Bachelor's thesis.

Additionally, I would also like to express a big thank you to my supervisor Michael Ressmann, who gave me the necessary freedom during the planning and execution of this thesis, but at the same time ensured that I did not take on too much during my project. I would also like to thank him for his good feedback, the prompt responses to my questions, and the quick review of my Bachelor's thesis.

Last but not least, I would also like to say thank you to all my friends and relatives who supported me in various ways during the realization.

Kurzfassung

Ein wichtiger Bestandteil von Algorithmen in Reinforcement Learning sind Belohnungen, welche Agenten nutzen, um ihre Entscheidungen zu bewerten. Durch gesammelte Erfahrung versucht der Agent diese zu maximieren, wobei die Anzahl an möglichen Ausgangssituationen eine große Rolle spielt. Dies basiert darauf, dass der Agent die Konsequenzen seiner Entscheidungen nur begreifen kann, falls er sich mehrmals in derselben Situation befindet. Um bei einer großen Anzahl an möglichen Situationen dennoch in der Lage zu sein, ein Verständnis für die Auswirkungen zu schaffen, werden neuronale Netze eingesetzt. Durch deren Verwendung konnten im letzten Jahrzehnt mehrere erstaunliche Ergebnisse erzielte werden.

Diese Ergebnisse basieren auf der Tatsache, dass Belohnungen normalerweise eher häufig vergeben werden. Sollten jedoch Situationen auftreten, bei denen der Agent Belohnungen nur selten erhält, so ist dieser nicht in der Lage die Konsequenzen zu begreifen. Dieses Problem wird üblicherweise durch die Miteinbeziehung von intrinsischen Anreizen gelöst. Solch ein Anreiz kann beispielsweise Neugierde sein, welche als die Fähigkeit eines Agenten beschrieben wird, Konsequenzen seiner Entscheidungen richtig vorherzusagen. Diese Art von intrinsischer Motivation wurde intensiv von Burda et al. (2018a) erforscht. Hierbei war es ihnen möglich zu zeigen, dass selbst ein rein intrinsisch motivierter Agent in der Lage ist extrinsische Belohnungen zu sammeln. Jedoch veranschaulichten sie auch, dass Agenten durch die Zuhilfenahme von zufälligem Rauschen abgelenkt werden können, was deren Fähigkeit, Belohnungen einzusammeln, unterbindet.

Basierend auf diesen Erkenntnissen ist es das Ziel dieser Arbeit, Fragen bezüglich der Kombination von extrinsischen und intrinsischen Belohnungen zu beantworten. Gewonnene Ergebnisse legen hierbei nahe, dass optimal kombinierte Belohnungen schlechter abschneiden als bestehende Ansätze, jedoch besser sind als rein intrinsische. Abgesehen davon birgt eine optimale Kombination vorteilhafte Eigenschaften, wie z.B. schnellere und stabilere Konvergenz zu einer guten Performanz des Agenten. Es scheint, als ob extrinsische Belohnungen tendenziell stärker gewichtet werden und ihre Gewichtung mit der Häufigkeit von extrinsischen Belohnungen korreliert. Zu guter Letzt werden die Auswirkungen von zufälligem Rauschen erforscht, wobei sich die Frage stellt, ob durch Miteinbeziehung von extrinsischer Motivation diese minimiert werden können. Hierbei wurde beobachtet, dass sobald die extrinsische Gewichtung höher ist als die intrinsische, die Ablenkung nicht mehr signifikant ist.

Abstract

An important part of algorithms in reinforcement learning are rewards that a so-called agent uses to evaluate its made decisions. The agent tries to maximize these rewards by gaining experience, whereby the number of possible initial situations plays a major role. This is based on the fact that the agent can only understand the consequences of its decisions if he is in the same situation several times. In order to still be able to create an understanding of the consequences in cases where the number of possible situations is too big, neural networks are employed. By using them, several astonishing results have been achieved in the last decade.

Common for these results is the fact that rewards are awarded densely whereas if rewards are sparse the agent is unable to pick up the desired behaviour at all. This issue is usually tackled by incorporating further rewards by means of intrinsic motivation. An approach to implement intrinsic motivation is via curiosity which was profoundly explored by Burda et al. (2018a) where they expressed curiosity as an error reflecting the agent's ability to predict the consequences of its decisions. They showed that even a purely intrinsically motivated agent is able to obtain extrinsic rewards, and pointed out the limitations of such an approach by presenting the so-called noisy-TV problem where agents get distracted by random white-noise and thus are unable to collect rewards.

Based on these findings, this thesis aims to solve multiple questions concerning the combination of extrinsic and intrinsic rewards. Obtained results during the evaluation process suggest that optimally combined rewards do not improve existing approaches, however, they certainly outperform pure intrinsic ones. Additionally, other beneficial properties have been discovered such as a faster and more stable convergence. There also seems to be a pattern concerning optimally combined rewards as extrinsic reward signals tended to be weighted more heavily and their heaviness was also correlated with the sparseness of the extrinsic rewards. Finally, the impact of the noisy-TV problem has been explored with the posed question on whether or not it is possible to combat its effects via extrinsic motivation. It was observed that as soon as the extrinsic reward signal outweighs the intrinsic one the impact of of the noisy-TV is not significant anymore.

Contents

Kurzfassung **xiii**

Abstract **xv**

Contents **xvii**

1 Introduction **1**
1.1 Problem Description 1
1.2 Expected Results 2
1.3 Methodological Approach 3

2 Reinforcement Learning Background **5**
2.1 Reinforcement Learning Problem 5
2.2 Markov Decision Processes 6
2.3 Functions to Improve the Policy 8
2.4 Dynamic Programming 10
2.5 Temporal Difference Methods 10
2.6 Policy Gradient Methods 13

3 Deep Reinforcement Learning **17**
3.1 The Basic Architecture of Neural Networks 17
3.2 Multi-Layer Neural Networks 21
3.3 Deep Learning Approach in Reinforcement Learning 23
3.4 Proximal Policy Optimization 24

4 Intrinsic Motivation in Reinforcement Learning **29**
4.1 Natural Human Motivation 29
4.2 The Role of Intrinsic Motivation in Reinforcement Learning 30
4.3 Challenges Tackled with Intrinsic Motivation 31
4.4 Classification of Intrinsic Motivation 34
4.5 Embedding Intrinsic Motivation into Reinforcement Learning 35

5 Exploration **39**
5.1 Prediction Error 40

5.2 State Novelty 42
5.3 Information Gain 43
5.4 Current State of the Art Method 45

6 Implementation **49**
6.1 Proximal Policy Optimization and Auxiliary Tasks 50
6.2 Hyperparameter Tuning 52
6.3 Noisy-TV Problem 53

7 Results **55**
7.1 Evalutation 55
7.2 Montezuma's Revenge 58
7.3 Freeway 59
7.4 Breakout 62

8 Discussion, Further Work and Conclusion **65**

List of Figures **69**

List of Tables **71**

List of Algorithms **73**

Acronyms **75**

Bibliography **77**

Non-Scientific Bibliography **81**

Appendix **84**

CHAPTER 1

Introduction

1.1 Problem Description

In Reinforcement Learning (RL), algorithms mainly depend on carefully designed extrinsic reward functions which are comparable with feedback to an agent's behaviour. In order to teach an agent a certain desired behaviour, it has to optimize this reward function by trial and error. This traditional approach yielded multiple astonishing results over the last decade. Tasks which award rewards densely, i.e. almost after every performed action, are common for these results. In opposite to dense rewards, sparse rewards are scattered in the environment which poses the problem that they occur too rarely for an agent to pick up the needed skills. An example for an environment with sparse rewards where tasks depend heavily on exploration would be, e.g. *Montezumas's Revenge*.

An approach to tackle these problems is curiosity, an organism's Intrinsic Motivation (IM) to spontaneously explore its environment. IM allows an agent to incrementally learn valuable skills independently of its given task by an intrinsic and therefore more general reward function. In general, the concept of intrinsic and extrinsic deals with the question of why an agent performed a certain action in a given state. Since IM became an important part of RL, the following challenges were specified by (Aubret et al., 2019, see p. 6):

- **Sparse rewards:** The agent never reaches a reward signal in case of sparse rewards.
- **State representation:** The agent does not manage to learn a representation of its observations with independent features or meaningful distance metrics.
- **Building option:** The agent is unable to learn abstract high-level decisions independently from the task.

- **Learning a curriculum:** The agent hardly defines a curriculum among its available goals without expert knowledge.

Besides a general introduction to IM and its challenges in RL, this Bachelor's thesis has its focus on knowledge acquisition through exploration. Knowledge acquisition is described as the agent's motivation to find new knowledge about its environment, meaning that it is interested in things it can or cannot control, the function of the world, discovering new areas, or understanding the proximity. Concerning exploration, one approach is error prediction which is the agent's difficulty to predict the state following a state-action tuple. This idea is heavily explored by Burda et al. (2018a) and the prior work of Pathak et al. (2017), both of which are core references of this thesis.

Building upon the findings of Burda et al. (2018a), this Bachelor's thesis eyes on answering the question on how to optimally combine intrinsic and extrinsic rewards in order to maximize an agents score, primarily in sparse but also in dense environments. Furthermore, it will evaluate if a combined reward tackles the noisy-TV problem which was initially proposed by Schmidhuber (2010). The noisy-TV problem, also known as the white-noise problem, is an algorithm's inability to handle the local stochasticity of the environment: "*...random noise in a 3D environment attracts the agent; it will passively watch the noise since it will not be able to predict the next state.*"(Aubret et al., 2019, see p. 10)

The domain of the given problem description is RL with a focus on the model of knowledge acquisition via exploration.

1.2 Expected Results

As aforementioned, this Bachelor's thesis builds upon the results of Burda et al. (2018a) and pursues their posed question on how to optimally combine intrinsic (r_{int}) and extrinsic (r_{ext}) rewards. In order to do so, the coefficients α and β have to be selected accordingly.

$$r = \alpha r_{int} + \beta r_{ext}$$

From that, an optimal solution to this equation maximizes the agent's overall reward when testing.

After reproducing their test setup, the benchmarks are expected to at least match the mean rewards mentioned in the *Additional Results* section of their paper. In this section, the authors explored the performance of combined rewards on five different Atari 2600 games and already showed that the combination of rewards yield a higher mean reward. Therefore, an exhaustive hyperparameter tuning should improve these results.

However, this also poses the question of how algorithms with combined rewards perform in densely rewarding environments. Therefore, it would be interesting to observe the performance in environments with dense rewards since there might be an improvement to be seen as well.

In the *Discussion* section, Burda et al. (2018a) mention the limitations of prediction-error based curiosity which was earlier introduced with the so-called noisy-TV problem. Given this problem, the question is raised on whether or not it is possible to tackle the agent's distraction by combining curiosity with Extrinsic Motivation (EM). This question will be pursued during the evaluation process with the posed hypothesis that it is possible to overcome an agent's distraction given that it obtains enough EM. Lastly, this further raises the question of how much EM might be needed.

With the given prospect to beat state-of-the-art approaches to environments with sparse rewards and to evaluate the amount of extrinsic reward needed to put a distracted agent back on track, it is the opportunity to pursue important constructs, that reflect the natural human propensity to learn and assimilate, in the domain of computer science that motivated this Bachelor's thesis.

1.3 Methodological Approach

Before the methodological approach is described, a summary of the just posed research questions is provided: (a) Does the combination of extrinsic and intrinsic rewards obtain better results? (b) Are there any patterns concerning optimally combined rewards? (c) Is it possible to further improve the performance of agents in densely rewarding environments by utilizing IM and EM simultaneously? (d) Can local stochasticity, also known as the white-noise problem, be tackled by employing EM?

In order to pursue these questions, a benchmarking environment must be crafted where adopted and tuned RL algorithms can be evaluated. For this intent, there are several existing solutions such as the generally established *OpenAI Gym* which has been solely created for the purpose of developing and comparing RL algorithms. However, as this thesis is primarily concerned with both IM and EM based approaches, the benchmarking environment will be built on the released source code by Burda et al. (2018a) as it is better-suited. With their implementation, they conducted a large-scale study across 54 different standard benchmarks including several Atari 2600 games which proves its capability to conduct the necessary experiments. Further implementation details can be taken from the work by Pathak et al. (2017).

Before needed experiments can be carried out, several adaptation to the existing codebase must be performed. These adaptations primarily include the enhancement of the existing reward function in order to be able to award intrinsic and extrinsic rewards simultaneously as well as the incorporation of random white-noise to enable testing capabilities regarding the mentioned noisy-TV problem. Afterwards, the evaluation process can be started where a special focus is laid on ensuring validity, reproducibility as well as comparability of the obtained results and hence, multiple runs with different seeds will be carried out on the same environment. This procedure is based on the best practices proposed by Francois-Lavet et al.. In summary, the main goal of these experiments is to find optimal weightings of the two reward signals in regards to maximizing the respective performance metric.

Finally, the obtained data is evaluated by comparing different statistical key figures such as the mean, the standard deviation, and the standard error as well as the minimum and maximum. Potential differences between these key figures will be verified utilizing two-sided t-tests. Furthermore, quantitative results will be visualized with the *Python* library *Matplotlib* as a visual representation of data usually facilitates the identification of interesting patterns. The interpretation of the findings will be based on knowledge gathered during the realization of this thesis.

Therefore, this Bachelor's thesis follows a programming approach where different weightings of the rewards are tested on the implemented benchmarking environment as well as a quantitative approach since data must be evaluated using statistics. Additionally, the theoretical aspect of this thesis will be covered by an exhaustive literature research with an emphasize on directly comparable approaches, i.e. prediction-error based curiosity. For further information regarding the theory, the bibliography is recommended.

CHAPTER 2

Reinforcement Learning Background

The following chapter begins with a description of the Reinforcement Learning (RL) problem and its affiliation to the broader spectrum of Machine Learning (ML). Here, also the distinctions between the different ML disciplines are pointed out. Furthermore, the described problem is then formally notated under the utilization of Markov Decision Processs (MDPs). In Functions to Improve the Policy, the policy as well as the corresponding state-value and action-value function are introduced which act as measurements on how favourable a state or an action might be. This is followed by several tabular solution methods such as Dynamic Programming (DP) and Temporal-Difference (TD) learning where a clear focus is laid upon the latter. During these sections, Monte-Carlo (MC) methods are broached as well. Finally, the chapter is concluded by considering Policy Gradient (PG) methods for approximate solutions which are very different to tabular ones.

2.1 Reinforcement Learning Problem

RL, the science of decision making, is about learning the consequences of performed actions and using this knowledge to maximize a numeric reward signal which should direct the search for a given objective (Sutton and Barto, 2018, see p. 1f). RL problems are tackled by a learner or rather an agent that performs actions in an environment in a trial-and-error paradigm. The agent has no prior knowledge of the consequences of the possible actions but has to explore which action to take in a certain situation to maximize the number of obtainable rewards. There is another layer of difficulty added on top, if the reward for an action is delayed because the agent has to understand which of the executed actions leads to desirable rewards it received far ahead in the future. Lastly, a proper weighting between exploration and exploitation has to be developed as

the agent must exploit actions with known consequences and good rewards to perform well but also explore new ones in order to find even higher rewarding actions (Kaelbling et al., 1996).

In addition to the aforementioned environment, agent, and reward signal, RL consists of further subelements since it would be desirable e.g. not only to know the immediate reward but also to gain measurement for all possible future rewards from a given state onward. For this purpose, a value function is used. It acts as a long-term indicator that describes the attractiveness of a state an agent might end up at after interacting with its environment. This measurement is interesting because there could be an action which leads to a high immediate reward but results in an undesirable state with no future rewards later on. Furthermore, the agent should pick up a policy that consists of state-action tuples which should be refined and adjusted depending on the received rewards. Lastly, the agent may want to build a model which estimates the environment's behaviour, in order to plan its actions. This final addition divides the RL problem spectrum into two parts: model-free and model-based approaches (Sutton and Barto, 2018, 6f).

As opposed to other known ML disciplines such as supervised learning, there is no supervisor in RL that explicitly tells the agent which action would be correct in a given state. Because of that, the agent has to come up with its own policy based on the repeated feedback it received from the immediate reward and the following state (Kaelbling et al., 1996). This spares the necessity of creating a training dataset with corresponding state-action tuples beforehand and thus, RL is the preferred choice in cases where an agent interacts with and influences its environment. Furthermore, teaching based on supervision would also mean that an agent will be limited to the labelling ability of its supervisor which is another reason why it must be able to learn from its own experiences (Silver, c). Since RL is quite distinct from supervised learning and does not rely on labelled examples, it is sometimes misclassified as unsupervised learning which has the purpose to find structure in unlabelled data. However, this is not applicable for RL as the agent has to maximize a reward function and not to find structure in data. Therefore, Sutton and Barto (2018, see p. 2) classified RL as the third subfield of ML.

2.2 Markov Decision Processes

To formalize the RL problem, the idea and notation of an MDP is used to notate the sequential interactions between an agent and its environment. At every discrete time-step, $t = 0, 1, 2, 3, \ldots \in \mathbb{N}$, the agent observes the current state, $S_t \in \mathcal{S}$, of the environment and selects an available action in a given state, $A_t \in \mathcal{A}(s)$. Sometimes this notation is simplified to $A_t \in \mathcal{A}$ assuming that the action set $\mathcal{A}$ does not depend on the state (Sutton and Barto, 2018, see p. 48). Once the agent has selected an action, it obtains a scalar reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and depending on the action, the environment transforms into a subsequent state S_{t+1} (Figure 2.1). By means of continuous interactions between the agent and the environment, a trajectory of the form $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$

is created.

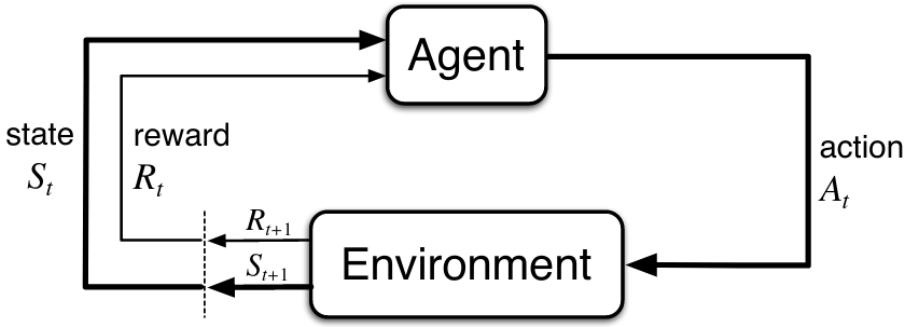


Figure 2.1: The RL framework formalized as an MDP[1]

In a finite horizon, the agent has to plan only a fixed number of time-steps ahead and the sets $\mathcal{S}, \mathcal{A}$, and $\mathcal{R}$ are limited (Sutton and Barto, 2018; Kaelbling et al., 1996, see p. 47f). Furthermore, the possibility that the values $s' \in \mathcal{S}$ and $r \in \mathcal{R}$ may occur at a time-step $t+1$, can be described with a probability (Equation 2.1) solely depending on the action $a \in \mathcal{A}(s)$ taken in a preceding state $s \in \mathcal{S}$.

$$p(s', r|s, a) = P\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{2.1}$$

Since p describes a probability, the sum over all possible combinations of s and a must be equal to 1. Moreover, the individual probabilities of p describe the dynamics of an MDP. Observing this notation shows that the state s must contain all the necessary information and must not depend on states preceding s. If this is the case, the state is called to have the Markov property (Equation 2.2) (Francois-Lavet et al., 2018).

$$P\{S_{t+1}|S_t, A_t\} = P\{S_{t+1}|S_1, A_1, \ldots, S_t, A_t\} \tag{2.2}$$

With the notation of p, properties of the MDP like the state-transition probabilities (Equation 2.3) or the expected value of rewards (Equation 2.4) can be computed given state-action tuples.

$$p(s'|s, a) = P\{S_{t+1} = s'|S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{2.3}$$

$$r(s', s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) \tag{2.4}$$

As mentioned in the Reinforcement Learning Problem, rewards at each time-step R_t are used to formalize a goal which the agent should pursue. This is possible due to the reward

[1] (Sutton and Barto, 2018, see p. 48)

hypothesis which states that every goal one can possibly think of, can be formalized through a numeric reward signal and be achieved by maximizing the cumulative long-term rewards (Sutton and Barto, 2018, see p. 53).

Mathematically, the overall reward is denoted as the sum over all rewards (Equation 2.5), with T as the final time-step.

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T = \sum_{t=1}^{T} R_t \tag{2.5}$$

However, this notation is only applicable if an environment has an identifiable terminal state. When this state is reached, it breaks, for instance, a game sequence into multiple ones where each one is called an episode. No matter how the agent reached the final state, the environment will reset and start over with a default initial state.

Sometimes, this separation of episodes might not be as clear as the limit of the sequence might be infinite. This is called a continuing task, in contrast to episodic tasks where T is finite. Therefore, if the agent acts in an environment with a continuing task, the sum in Equation 2.5 might not converge. This issue is solved by introducing a discounting rate γ, $0 \leq \gamma \leq 1$, which avoids infinite rewards (Sutton and Barto, 2018, see p. 54f). Given this discounting rate, the agent now tries to maximize the discounted sum of cumulative long-term rewards (Equation 2.6).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = R_{t+1} + \gamma G_{t+1} = \sum_{k=1}^{\infty} \gamma^k R_{t+k} \tag{2.6}$$

With γ as a newly introduced hyperparameter, the weighting of the rewards can be adjusted according to the given task. A low γ means that the agent values immediate rewards more and ones in the near future. On the other hand, a high γ forces the agent to consider long-term future rewards as well, thus leading to more far-sighted agents.

2.3 Functions to Improve the Policy

As mentioned in section 2.1, the policy forms an important component of the RL framework and is usually denoted as π. By following a policy, the agent's probability to select a specific action at time-step t is $\pi(a|s) = P\{A_t = a|S_t = s\}$ and therefore π describes a probability distribution over actions given states (Silver, b). Additionally, the aforementioned value function component is dependent on the policy, since π defines which action-state tuples are considered from the current state onwards. Formally, the state-value function for an MDP is denoted by $v_\pi(s)$ which stands for the expected future rewards when following the policy, starting in state s and "$\mathbb{E}_\pi[\cdot]$ *denotes the expected value of a random variable given that the agent follows policy* π" (Sutton and Barto, 2018, see p. 58).

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k} \middle| S_t = s\right] \tag{2.7}$$

In a similar way, the action-value function $q_\pi(s, a)$ of an MDP can be defined. This calculates the expected future reward when the agent takes action a in state s and then follows the policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k} \middle| S_t = s|A_t = a\right] \tag{2.8}$$

The policy is improved by adjusting the value functions v_π and q_π through the agent's obtained experience. Since RL aims to solve a given objective by maximizing rewards, it implicitly goes along with optimizing the agent's policy. If a policy π achieves a higher value than a different policy π', which also implies $v_\pi(s) \geq v_{\pi'}(s), \forall s \in \mathcal{S}$, then the two policies can be ordered partially, with $\pi \geq \pi'$. There can also be multiple policies performing equally well (Sutton and Barto, 2018, see p. 62f) and for any MDP the following theorem holds true (Silver, b, see p. 43):

- There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$
- All optimal policies achieve the optimal [state-]value function, $v_{\pi_*}(s) = v_*(s)$
- All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s, a) = q_*(s, a)$

A very simple optimal policy for a given state can be derived through the action-value function $q_*(s, a)$, simply by selecting the action with the maximum value.

$$\pi_*(a|s) = \begin{cases} 1 \text{ if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s, a) \\ 0 \text{ otherwise} \end{cases} \tag{2.9}$$

The optimal value functions are given by the following equations:

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in \mathcal{S} \tag{2.10}$$

$$q_*(s, a) = \max_\pi q_\pi(s, a), \forall s \in \mathcal{S} \wedge \forall a \in \mathcal{A} \tag{2.11}$$

2.4 Dynamic Programming

The state-value function, introduced in section 2.3, has a similar recursive property as the equation for the discounted expected reward (Equation 2.6). Therefore, the state-value function can be expressed in an equal manner by decomposing it into the discounted descendant states $\gamma v_\pi(s')$ and the immediate reward r:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s'] \Big] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \forall s \in \mathcal{S} \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s] \end{aligned} \tag{2.12}$$

The above-mentioned equation is known as the Bellman equation for v_π (Sutton and Barto, 2018, see p. 59) which can be directly solved due to its linearity. However, this calculation is only feasible for small MDPs since its complexity is $O(n^3)$ for n states (Silver, b).

For larger MDPs multiple iterative solutions exist:

- DP
- MC Evaluation
- TD Learning

The general idea of DP, which was developed by Bellman, is to utilize the property of recursion to simplify the complexity of calculations by storing interim results. In that sense, it is a method that aims at solving complex problems by breaking them down into manageable subproblems.

All of the three above-mentioned approaches are mainly used for two aspects in RL: predicting the value function v_π (planning) and calculating the optimal value function v_{π_*} (controlling). As for DP, an iterative process is used. This process is an interplay between the Bellman equation (Equation 2.12), used to calculate the value function for a given policy as well as Equation 2.10 and Equation 2.11 which update the policy (Silver, a).

2.5 Temporal Difference Methods

In the following subsections the idea of TD learning, one of the more central and novel ideas of RL, is introduced which combines DP as well as MC evaluation. Throughout

the theory of RL, the ideas and relationships of these three methods are commonly reoccurring.

2.5.1 Temporal Difference Prediction

Besides DP and MC evaluation, TD prediction is among one of the three most popular techniques. Furthermore, TD combines the advantages of the other two in order to predict the value function. Firstly, it uses the benefit of bootstrapping offered by DP which uses the value function predictions as the target during the learning process (Szepesvári, 2010, see p. 18). Secondly, TD prediction directly learns from raw experiences in a similar way to MC evaluation. Hence, it is also model-free. However, it only awaits the reward of one time-step to form a target for the estimated value function V instead of the every-visit MC method which awaits the fully known return for the target before approximation (Sutton and Barto, 2018, see p. 120).

$$\begin{aligned} \text{MC method: } & V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \\ \text{TD method: } & V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \end{aligned} \tag{2.13}$$

The parameter α is a constant which defines the step-size of each adjustment to the value function. The two equations express the above-mentioned differences certainly well:

> "Whereas Monte Carlo methods must wait until the end of the episode to determine the increment to $V(S_t)$ (only then is G_t known), TD methods need to wait only until the next time-step. At time $t+1$, they immediately form a target and make a useful update using the observed reward R_{t+1} and the estimate $V(S_{t+1})$."
>
> (Sutton and Barto, 2018, see p. 120)

The TD method is also known as TD(0) that refers to the general case TD(λ), a method that unifies TD and MC.

2.5.2 SARSA (On-Policy)

SARSA builds upon the idea of TD but aims to find an optimal value function and thus solve the control problem. In a similar manner to TD, SARSA aims to estimate the action-value function $q_\pi(s, a)$, given a current policy π. Since it only focuses on one policy at a time, it is also known as an on-policy algorithm. The action-value function is optimized by means of the following calculation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2.14}$$

At each transition between the state-action tuple, the values $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$ are included in the optimization process, eponymous for SARSA (Sutton and Barto, 2018, see

p. 129). Under the assumption that every state-action tuple is visited infinite times and that new policies are created greedily with respect to the current action-value function, SARSA is guaranteed to converge to the optimal action-value value function and thus to the optimal policy.

Algorithm 2.1: SARSA for estimating $Q \approx q_*$[2]

Input: Step size $\alpha \in (0, 1]$, small $\epsilon > 0$

initialize $Q(s, a)$, *for all* $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, *arbitrarily except that* $Q(terminal, \cdot) = 0$;

forall *episodes* **do**
  initialize S;
  choose A *from* S *using policy derived from* Q *(e.g.* ϵ*-greedy)*;
  while S *is non-terminal* **do**
    take *action* A;
    observe R, S';
    choose A' *from* S' *using policy derived from* Q *(e.g.* ϵ*-greedy)*;
    update $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$;
    update $S \leftarrow S'$, $A \leftarrow A'$;
  end
end

2.5.3 Q-Learning (Off-Policy)

Another control algorithm which is built upon the idea of TD is Q-learning which aims to directly estimate the optimal action-value function q_*. As denoted by Watkins and Dayan, the following calculation is used to update the action-value function Q.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{2.15}$$

As it does not optimize for the current policy but directly for the optimal one, Q-learning is considered an off-policy method and therefore the policies must no be interchanged during the optimization (Szepesvári, 2010, see p. 57).

[2] (Sutton and Barto, 2018, see p. 130)

Algorithm 2.2: Q-learning for estimating $\pi \approx \pi_*$[3]

Input: Step size $\alpha \in (0, 1]$, small $\epsilon > 0$

initialize $Q(s, a)$, *for all* $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, *arbitrarily except that* $Q(terminal, \cdot) = 0$;

forall *episodes* **do**
  initialize S;
  while S *is non-terminal* **do**
    choose A *from* S *using policy derived from* Q *(e.g.* ϵ*-greedy)*;
    take *action* A;
    observe R, S';
    update $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$;
    update $S \leftarrow S'$;
  end
end

2.6 Policy Gradient Methods

The algorithms introduced until now all had one thing in common: they were all action-value based methods. This means that values for given actions must be estimated in advance before a policy, based on the best achievable value in each state, can be derived. Therefore, a policy would not even exist without these estimations (Sutton and Barto, 2018, see p. 321).

In contrast, this section takes methods into account that directly learn a parameterized policy which is not based on a previously learned value function. However, there are also methods which consider the best of both worlds similarly to action-value based approaches, but the policy is not implicitly derived and the value function is not required for the selection of an action. Moreover, the value function is rather used to facilitate the learning process of the policy's parameters. These methods are so-called actor-critic methods (Silver, d) where the actor stands for the learned policy and the critic is a reference to the approximated value function (Sutton and Barto, 2018, see p. 321).

The parameters of a policy π are normally denoted in vector notation by $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and the policy is rewritten as $\pi(a|s, \boldsymbol{\theta}) = P\{A_t = a|S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$. More precisely, at time-step t, the probability of an action a is given by the current state s of the environment and the given parameters $\boldsymbol{\theta}$. In order to learn a policy directly, the parameters are adapted based on the derivative with respect to $\boldsymbol{\theta}$ of a numeric performance measure $J(\boldsymbol{\theta})$. As

[3](Sutton and Barto, 2018, see p. 131)

opposed to algorithms discussed in chapter 3, this derivative/gradient is used to maximize the policy's performance. With the usage of gradient ascent in J, the parameters are updated by the following equation:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta_t})} \tag{2.16}$$

The expectation of $\widehat{\nabla J(\boldsymbol{\theta_t})} \in R^{d'}$ is an estimation of the gradient of the performance measure $J(\boldsymbol{\theta_t})$ and α specifies a step-size parameter (Silver, d). Methods which use this gradient ascent to approach the parameters $\boldsymbol{\theta_t}$ for the policy π are so-called PG methods.

In a discrete action space with few choices, state-action pairs are usually parameterized by using a scalar preference function $h(s, a, \boldsymbol{\theta})$, e.g. a linear combination of features, $h(s, a, \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{x}(s, a)$ (Sutton and Barto, 2018, see p. 321), (Silver, d). Moreover, policies are required to be stochastic and thus, the most preferred action in a given state obtains the highest probability to be selected, e.g. by a Softmax distribution (Equation 2.17). For additional information, see also subsection 3.1.2.

$$\pi(a|s, \boldsymbol{\theta}) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}} = \frac{e^{\boldsymbol{\theta}^T \boldsymbol{x}(s,a)}}{\sum_b e^{\boldsymbol{\theta}^T \boldsymbol{x}(s,b)}} \tag{2.17}$$

In summary, the utilization of policy-gradient based methods gain four advantages compared to action-value based methods (Sutton and Barto, 2018, see p. 322f):

1. The parametrization of the policy under the usage of Softmax for defining action preferences allows for the approximation of deterministic policies. Action-value based methods which select actions with an ϵ-greedy approach, on the other hand, must always leave the option open for a random action to be selected, with probability ϵ. Therefore, these methods cannot approach a deterministic policy. Even if the selection of actions with respect to action-value tuples is based on the Softmax distribution, the policy would not approach a deterministic one. Rather, it would converge to the true values of the action-value tuples which results in values different from the probabilities 0 and 1.

2. In comparison to the previous advantage, it furthermore allows for arbitrary action selection which may be useful in certain types of tasks. For instance, playing the game of rock, paper, scissors with a deterministic policy (e.g. always selecting rock) the strategy would be easily exploitable. In this case, a stochastic uniform random policy, which does not allow for predictions, is much better-suited (Silver, d).

3. In certain situations, it might be more efficient to parameterize the policy instead of the action-value parametrization as the approximation of policies could be less complex. For instance, figuring out precisely how much score an agent can obtain in the Atari 2600 game *Breakout* (Figure 7.1c) from one point onward when moving

left might be very difficult. However, creating a policy that always moves left if the ball comes from a certain direction will be a lot easier to approximate (Silver, d). In these situations, policy based methods will achieve a superior policy and converge faster(Sutton and Barto, 2018, see p. 323).

4. Lastly, the initial configuration of a policy's parameters enables the possibility to supply an agent with prior knowledge about an optimal policy which is usually the main reason why policy based methods are taken into consideration.

CHAPTER 3

Deep Reinforcement Learning

The preceding chapter has dealt with the Reinforcement Learning Background while presenting multiple solutions to the Reinforcement Learning (RL) problem. However, approaches such as SARSA (subsection 2.5.2) or Q-learning (subsection 2.5.3) only perform well if each state is visited countless times in order to estimate the optimal policy. In the case of video games where states are often represented as the current frame of the game, the dimension of the state space is intangible which is also known as the curse of dimensionality (Goodfellow et al., 2016, see p. 151). That is because each frame consists of thousands of pixels and each of these pixels can accept various values ranging from 0 to 255. To solve this issue, Deep Learning (DL) techniques which emerged from Neural Networks (NNs) are applied to RL due to their capability of estimating very complex functions such as the value function for a high dimensional state space. By combining these techniques the field of Deep Reinforcement Learning (DRL) emerged which enables superhuman-like problem-solving skills (Francois-Lavet et al., 2018).

Instead of presenting right away adapted, traditional RL approaches, this chapter begins with an introduction to NNs and DL. Afterwards, it provides an overview of traditional DRL approaches which is finally concluded with one specific optimization method, the so-called Proximal Policy Optimization (PPO). A focus is laid on this approach because PPO represents the algorithm which is used in the work of Burda et al. (2018a), the core literature of this thesis.

3.1 The Basic Architecture of Neural Networks

As many ideas presented in this thesis, NNs are also inspired by nature as its structure is a resemblance of the human brain. The name already indicates that a NN is composed of neurons which are connected with one another by dendrites and axons. Additional connections between axons and dendrites, called synapses, define how important a transferred signal between them is. Similar to a human synapse, the strength of an artificial one is

changed via a learning process which refines the weighting of the signal (Aggarwal, 2018, see p. 1f). This idea of representation was introduced by McCulloch and Pitts who started with a simple model which was later on adapted and improved by Rosenblatt (Awad and Khanna, 2015, see p. 167).

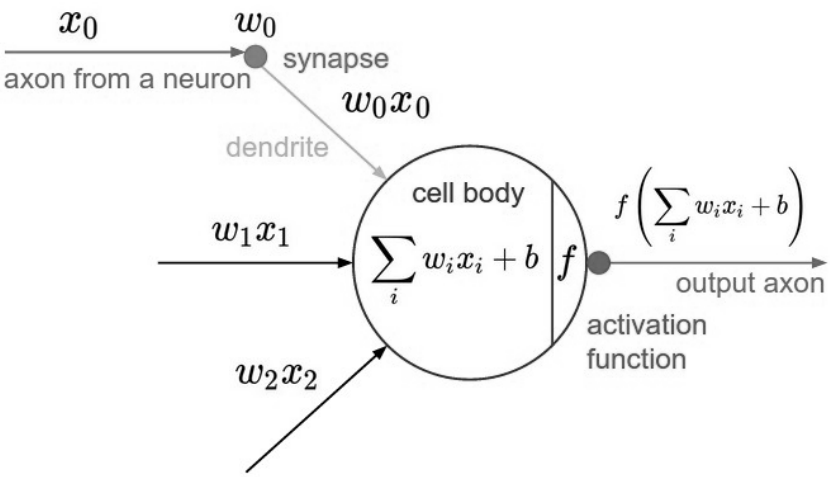


Figure 3.1: From a biological neuron to a mathematical model in its simplest form: an artificial NN[1]

In its simplest form, this NN is called a perceptron or feed-forward network, with one input layer and one output node. In this form, the NN allows for a generalized variation of a linear function. A general perceptron is of the form $(\bar{\boldsymbol{X}}, y)$, where $\bar{\boldsymbol{X}} = [x_0, x_1, \ldots, x_d]$ are the input variables in vector notation and $y \in \mathbb{R}$ represents the observed value. The observed value stands for a numerical scalar used for a regression problem, e.g. predicting the price of a house given some numerical input variables such as the number of rooms and square meters. In the case of classification, this perceptron allows for binary classification where the observed value is $y \in [-1, +1]$. For instance, this can be used to predict whether or not a credit-card transaction was fraudulent given the amount and frequency of the transaction. Given the input values $\bar{\boldsymbol{X}}$ and the weights $\bar{\boldsymbol{W}} = [w_0, w_1, \ldots, w_d]$, a result of a linear function is calculated by the following prediction:

$$\bar{\boldsymbol{X}} \cdot \bar{\boldsymbol{W}} = \sum_{j=0}^{d} x_j * w_j \tag{3.1}$$

Finally, a prediction $\hat{y}$ for y, based on Equation 3.1, is performed by activating the result making usage of an activation function. In the case of classification, the sign activation function is used to map the prediction $\hat{y}$ to the previously specified range of $[-1, 1]$. After the activation of the output value, an error between the prediction and the actual outcome can be calculated, $E(\bar{\boldsymbol{X}}) = y - \hat{y}$ (Aggarwal, 2018, see p. 5ff). If the error

[1] https://cs231n.github.io/neural-networks-1/

$E(\bar{\boldsymbol{X}})$ is non-zero, the weights must be adapted in order to perform a closer prediction next time. By optimizing the error function or loss function, the classification error is reduced. The optimization process is also known as gradient descent which updates the weights according to the negative direction of the function's gradient with respect to each input (Aggarwal, 2018, see p. 7) and is formally described by the following notation:

$$\bar{\boldsymbol{W}} \Leftarrow \bar{\boldsymbol{W}} + \alpha E(\bar{\boldsymbol{X}})\bar{\boldsymbol{X}} \tag{3.2}$$

The parameter α is the learning rate which defines how large the updates to the weights should be. On the one hand, a large value of α allows for faster convergence to a local optimum with the accompanying risk of never reaching one at all. On the other hand, a small value steadily approaches the optima but takes longer.

As the perceptron represents a linear function, it must not only learn the slope but also the intercept which is also known as the invariant part. The intercept value is also called bias and is especially necessary in scenarios where all the features are centred around the mean and the values of the target class are not centred around the origin (Aggarwal, 2018, see p. 6). To solve this issue, the bias variable is introduced to the perceptron as an additional neuron and an extra weight which is adjusted during the error optimization process. This additional neuron solely has the purpose to feed the scalar 1 to the output node and by refining the weight in between, the intercept value is learned. By incorporating the bias neuron, the following equation emerges:

$$\bar{\boldsymbol{X}} \cdot \bar{\boldsymbol{W}} + b = \sum_{j=0}^{d} x_j * w_j + b \tag{3.3}$$

3.1.1 Activation Functions

In general, activation functions play an important role in NNs as they define the output values of neurons. However, they are especially significant for Multi-Layer Perceptrons (MLPs) as they determine the modelling power of the network. By using non-linear functions, more sophisticated compositions can be created as they are not reducible to simple perceptrons (Aggarwal, 2018, see p. 13).

Apart from the previously noted sign function, other activation functions exist as well and a selection of them can be observed in Figure 3.2. As mentioned in section 3.1, the sign function allows for binary classification of class labels. Furthermore, to enable the evaluation of the certitude of a NN, the Sigmoid activation function is employed instead. Using this function allows for a mapping of the pre-activated value to a range between $[0, 1]$, enabling an estimation of the network's certainty. Therefore, $\hat{y}$ indicates the likeliness of an input being a certain target class. In contrast to this, the identity activation function is used to predict real numbers. By using the identity function, the NN is equivalent to the least-squares regression algorithm. An important property that all activation functions have in common is monotonicity and besides the identity function,

they also "*saturate at large absolute values at which increasing further does not change the activation much.*"(Aggarwal, 2018, see p. 13)

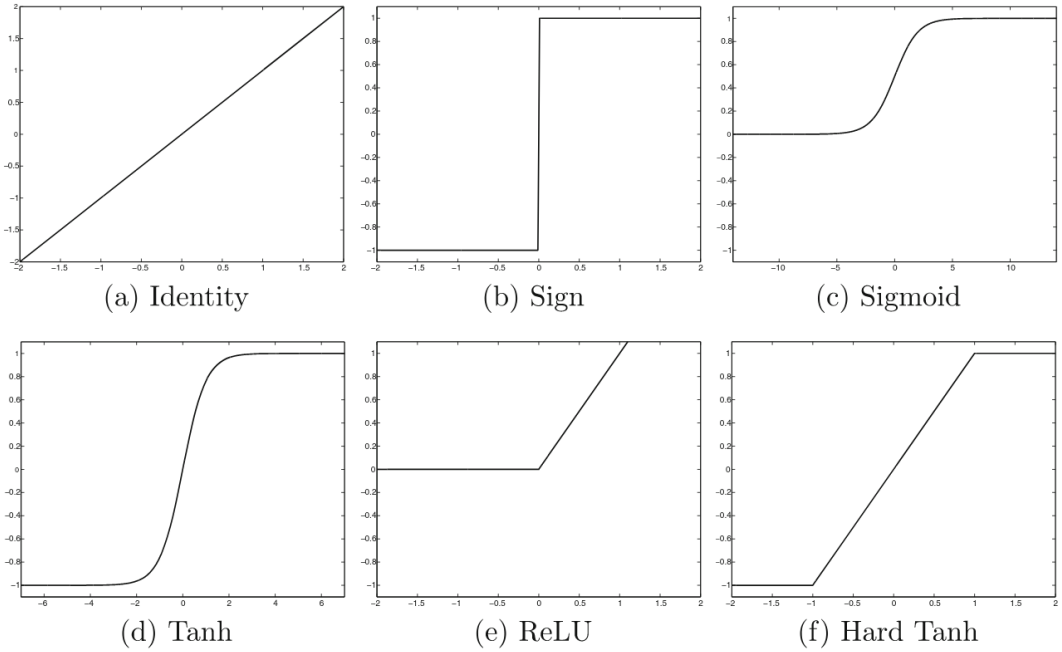


Figure 3.2: Various activation functions and their relationships between the domain and the codomain[2]

Activation functions are applied on the pre-activated value and therefore, a formal notation of $\hat{y}$ is denoted by $\hat{y} = \phi(\bar{\boldsymbol{W}} \cdot \bar{\boldsymbol{X}})$. Over the course of the advancement of NN, initial activation functions such as the Sigmoid and the Tangens-Hyperbolicus activation function have been replaced by piecewise linear functions such as the Rectified Linear Unit (ReLU) activation function.

$$\text{Sigmoid:} \qquad \phi(v) = \frac{1}{1 + e^{-v}}$$

$$\text{Tangens-Hyperbolicus:} \qquad \phi(v) = \frac{e^{2v} - 1}{e^{2v} + 1}$$

$$\text{ReLU:} \qquad \phi(v) = \max\{v, 0\}$$

This is due to the fact that it facilitates the training process of MLPs as they are less expensive to compute.

[2](Aggarwal, 2018, see p. 13)

3.1.2 Loss Functions

In section 3.1, a generic loss function $E(\bar{\boldsymbol{X}})$ was introduced. However, depending on the target and the prediction $\hat{y}$, another loss function may be better-suited as certain loss functions are more sensitive to errors for classification than for predicting real numbers (Aggarwal, 2018, see p. 15). For instance, regression tasks modelled by NNs with an identity activation function commonly use the squared loss $(y - \hat{y})^2$ as an error measurement. In the case of classification tasks with multiple class labels, output values are normally activated employing the Softmax activation function (Goodfellow et al., 2016, see p. 78), with $\boldsymbol{v}$ being the n pre-activated output values.

$$\text{Softmax: } \sigma(\boldsymbol{v})_i = \frac{e^{v_i}}{\sum_{j=0}^{n} e^{v_j}}$$

This function allows for a probabilistic output which means that it maps the pre-activated values to a probability distribution. Since the output of the Softmax function is a vector of probabilities instead of a scalar value, a different loss function is needed as well. Additionally, classification tasks can have binary or categorical/multiple targets which further define and differentiate which loss function will be used.

For binary targets, different combinations of activation and loss functions can be utilized to obtain the same result for prediction. For example, binary targets are classified by means of logistic regression which makes usage of the identity activation function to predict $\hat{y}$. A loss function for this prediction is defined by the following equation:

$$L = \log(1 + e^{-y \cdot \hat{y}}) \tag{3.4}$$

A further option is to simply use the Sigmoid activation function which maps the output $\hat{y}$ between 0 and 1. Under the assumption that the observed value y is normalized between -1 and 1, the negative logarithm of $|\frac{y}{2} - \frac{1}{2} + \hat{y}|$ is deployed as a loss function (Aggarwal, 2018, see p. 15). This loss function provides a measurement on how likely it is that the target has correctly been classified.

Concerning categorical targets, post-activated values $\hat{y}_0, \ldots, \hat{y}_n$ are examined one by one. The loss for the ith prediction is given by the following equation:

$$L = -\log(\hat{y}_i) \tag{3.5}$$

This definition is known as the cross-entropy loss and represents the extension of logistic regression (Aggarwal, 2018, see p. 15).

3.2 Multi-Layer Neural Networks

Comparing a simple perceptron to an MLP, the major difference is the employment of multiple layers, the so-called hidden layers. Instead of a perceptron where all computations

are visible, hidden layers in an MLP hide the computations (Aggarwal, 2018, see p. 17). However, the MLP still is a feed-forward network and the perceptron as well as the MLP can both be denoted as a directed acyclic graph. The additional layers in an MLP can be described as a composition of multiple functions as opposed to a perceptron where the input is directly transmitted to the output layer. By composing the additional layers via chain-like structures, the input is fed through multiple layers before it allows a prediction of $\hat{y}$. For instance, an MLP with three layers approximates the function $f(\bar{\boldsymbol{X}})$ by chaining three functions $f^{(3)}(f^{(2)}(f^{(1)}(\bar{\boldsymbol{X}})))$ together for a prediction (Goodfellow et al., 2016, see p. 163f).

The amount of layers or rather the length of the chain is known as the depth of a NN/an MLP which also established the name Deep Learning. As aforementioned in the introduction to chapter 3, only MLPs with their ability to chain multiple functions enable the approximation of a complex value function in a high dimensional state space.

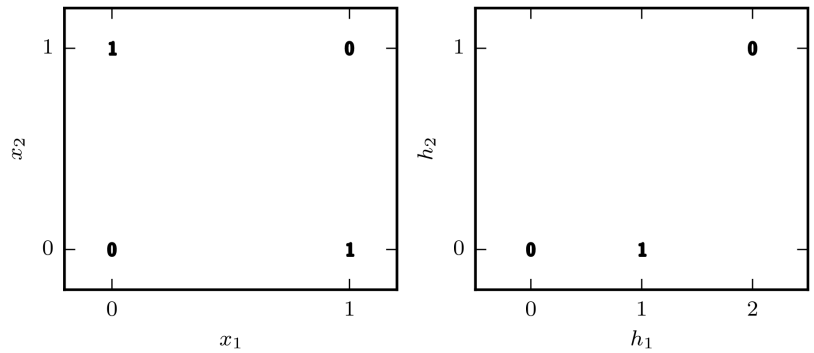


Figure 3.3: The XOR problem with the left plot being the original space using a perceptron and the right plot being the learned space using an MLP[3]

The reason why MLPs are the preferred choice instead of a perceptron is illustrated by an example of an NN which should estimate a model for the logical operator XOR, the "exclusive or". In detail, the XOR function takes two binary values x_0, x_1 and returns 1 if $x_1 \neq x_2$ and 0 otherwise. This described behaviour is also known as the target function $f^*(\bar{\boldsymbol{X}})$ of the XOR function, the NN should approximate. Observing Figure 3.3, the learned linear function of a perceptron would no be able to correctly implement the XOR function. Applying the model learned by the perceptron to the left, the original space, an approximated linear function cannot separate these values and thus would not be able to return the correct y. This is also known under the term of linear separability. However, after transforming the original space with the help of a hidden layer and a non-linear activation function (e.g. ReLU) to the space on the right side, the XOR function can be solved as the space is now linear separable (Goodfellow et al., 2016; Aggarwal, 2018, see p. 166ff; see p. 32ff). This shows that an MLP allows for much more complex function

[3](Goodfellow et al., 2016, see p. 168)

approximation. With the newly introduced capability, the learning process mentioned in Equation 3.2 in section 3.1 is not applicable anymore. Instead, a new algorithm named backpropagation was introduced by Rumelhart et al. which can be looked up either in Aggarwal (2018, see p. 21) or in a more formal way in Goodfellow et al. (2016, see p. 197).

3.3 Deep Learning Approach in Reinforcement Learning

The advancements of DL over the last couple of years, starting around 2010, allowed the domain of RL to evolve as well. Especially through the previously mentioned combination of these two fields and the new creation of the DRL domain, many astonishing accomplishments were achieved (Francois-Lavet et al., 2018). A summary of four different achievements which were selected by Aggarwal (2018, see p. 374) are listed in the following enumeration:

- Video games are a classic example where DRL excelled over the course of the last few years. Especially, games from the Atari 2600 gaming console are famous environments where DRL algorithms are applied, receiving only the raw pixels as an input. Based on the pixels, a state representation is formed on which the agent takes actions, makes mistakes, gains experience and finally improves its decision making. This procedure is equivalent to the traditional RL approach known from chapter 2. However, it is now applied to a much richer state space and thanks to DL, the agent can surpass humans in these games (Mnih et al., 2013; Schulman et al., 2015). Furthermore, the reason why video games are so popular even in modern RL is that they function as simulations of real-life microcosms that are perfectly suitable for testing algorithms before being applied in the real world.

- Another very famous breakthrough occurred in 2016 when an algorithm by the name *AlphaGo* (Silver et al., 2017) defeated the top-ranked players in the board game *Go*. The reason why this event was so remarkable is that *Go* is a very complex game and being good at it takes a lot of human-like intuition. Additionally, the state space is very large when compared to other board games such as e.g. *Chess*. However, it was the unconventional learning process which made *AlphaGo* well-known as it improved itself by obtaining experience by playing against another version of itself.

- DRL is also considered a viable option for self-driving cars. Although a more common approach to this problem is the utilization of supervised learning, the usage of various car sensors for decision making allows these automated cars to have a lower error rate when compared to humans.

- Another important domain where DRL plays a key factor is the creation of self-learning robots. Here, the difficulty lies in teaching a robot basic human-like motions such as walking. Fortunately, the task of teaching a robot to walk can be

described in the RL framework as the robot's goal is to reach a given position as fast as possible. The robot's state is a description of its available limbs and motors in a continuous space. Therefore, DL techniques must be applied as well to reach good performance which does not only allow the robot to learn how to walk but also how to roll and crawl.

All of these accomplishments are based on different DRL approaches. However, an exhaustive explanation of each of these algorithms would go beyond the scope of this thesis. Therefore, solely an overview is provided by Figure 3.4 and solely PPO is introduced. For further information, the paper by Francois-Lavet et al. is recommended.

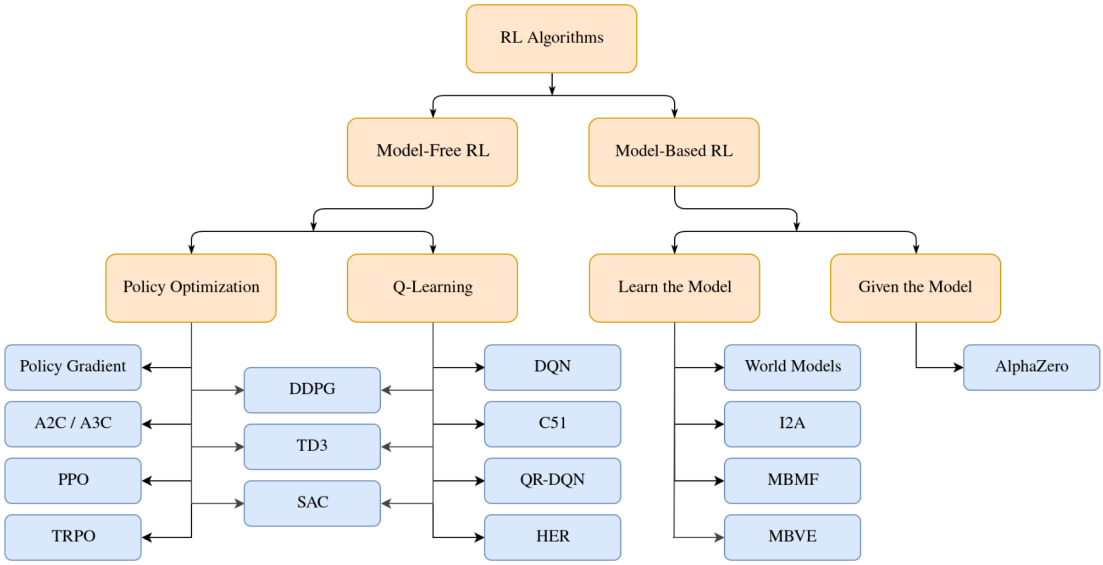


Figure 3.4: A non-exhaustive taxonomy of DRL algorithms[4]

3.4 Proximal Policy Optimization

PPO belongs to the family of Policy Gradient (PG) (section 2.6), model-free based methods that switches between gathering data by interacting with the environment and a stochastic gradient ascent optimization process of a surrogate target function. In comparison to Equation 2.16, Schulman et al. (2017) denote the estimator for the gradient slightly different:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{3.6}$$

[4] https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Here, $\hat{A}_t$ estimates an advantage function for a time-step t where the advantage function describes how good an action a might be, given the expected return if the policy π is followed directly. In general, it is denoted by $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ for a given policy, according to Francois-Lavet et al. Furthermore, $\hat{\mathbb{E}}_t[\cdot]$ is the empirical mean over some batch samples of interactions with the environment.

In software, the estimation of the gradient $\hat{g}$ is obtained by creating a similar objective function that acts as the estimator. This is done by differentiating the following loss function:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\Big[\log\pi(a_t|s_t)\hat{A}_t\Big] \tag{3.7}$$

Additionally, PPO is based on an idea named Trust Region Policy Optimization (TRPO) where objective function (Equation 3.6) is replaced by a surrogate objective (Equation 3.8) with an additional constraint (Equation 3.9) on the update size of the policy (Schulman et al., 2015).

$$\hat{\mathbb{E}}_t\Big[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\Big] \tag{3.8}$$

$$\hat{\mathbb{E}}_t\Big[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]\Big] \leq \delta \tag{3.9}$$

Instead of the constraint, a penalty is sometimes integrated with a corresponding weighting as well, combining Equation 3.8 and 3.9. Furthermore, KL in Equation 3.9 stands for the Kullback-Leibler divergence.

With the ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t$ the main objective for PPO can now be constructed:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big] \tag{3.10}$$

This objective differs mainly from the surrogate objective of TRPO (Equation 3.8) as it does not allow for excessively large changes to the policy. This is done by clipping the initial objective if the probability ratio r_t moves too far away from 1 and whether or not the advantage $\hat{A}_t$ is negative or positive. Thus, a restriction is placed upon the incentive to move outside of $[1-\epsilon, 1+\epsilon]$, with ϵ being a hyperparameter. Moreover, the minimum of the Equation 3.10 functions as a lower bound for the objective (Equation 3.8).

Referring to the results of Francois-Lavet et al. where the authors tried different surrogate objectives on seven environments with three different seeds, clipping the objective (Equation 3.10) with an $\epsilon = 0.2$ resulted in the best normalized averaged score of 0.82. By using different $\epsilon \in [0.1, 0.3]$, the score varies between $[0.70, 0.82]$ whereas KL-penalized objectives achieved scores between $[0.62, 0.72]$ and a non-clipped

and penalized objective (Equation 3.7) performed the worst with an average score of -0.39 (Francois-Lavet et al., 2018, see Table 1).

When comparing PPO algorithms to similar approaches such as TRPO, Francois-Lavet et al. list the following advantages:

> "[T]hey are much simpler to implement, more general and have better sample complexity (empirically). [Furthermore,] ... PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time."
>
> (Francois-Lavet et al., 2018, see p. 1)

In DRL applications where NN share parameters between a value function and a policy, they both must be combined to form one uniform loss function. Additionally, to construct the objective which PPO tries to maximize, a bonus in the form of an entropy is added to ensure enough exploration, resulting in a final target that is given by the following equation:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \Big[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \Big] \tag{3.11}$$

In this equation, S is the bonus in form of an entropy, L_t^{VF} denotes the squared-error loss $(V_\theta(s_t) - V_t^{\text{targ}})^2$ and c_1, c_2 are coefficients which control the influence of these two terms. Implementing and optimizing the objective (Equation 3.11) is carried out by following the policy and collecting samples of interactions with the environment for T time-steps. The gathered data is then used to apply an update. Additionally, the advantage function needed for Equation 3.10 is approximated by an estimator which only considers T time-steps with an index $t \in [0, T]$ as well and is denoted as follows:

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{3.12}$$

The PPO algorithm (see Algorithm 3.1) puts theory into practice where in each iteration, N workers gather data for T time-steps, calculate the loss with these NT time-steps and apply the just calculated loss in minibatches for K epochs (Francois-Lavet et al., 2018).

Algorithm 3.1: PPO, Actor-Critic Style[5]

```
for iteration = 1, 2, ... do
    for actor = 1, 2, ..., N do
        run policy π_{θ_old} in environment for T time-steps;
        compute advantage estimates Â_1, Â_2, ..., Â_T;
    end
    optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT;
    update θ_old ← θ;
end
```

[5](Schulman et al., 2017, see p. 5)

CHAPTER 4

Intrinsic Motivation in Reinforcement Learning

The idea of including Intrinsic Motivation (IM) into the Reinforcement Learning (RL) framework has been investigated since early 1990. As in realistic environments extrinsic rewards occur rather rarely, thus quick progress by an agent cannot be expected. With its roots deeply anchored in psychology, IM was initially used due to its ability of discovering surprising or novel patterns which enables an agent to be creative in the sense that it learns non-trivial, new behaviour (Schmidhuber, 2010). This natural phenomenon is based on the observation that animals engage in curiosity-driven, exploratory, and playful behaviour despite the fact that a reward or a reinforcement would not be granted to them (White, 1959).

This chapter begins with an introduction to the psychological aspects of IM as well as the term motivation in general and is continued with the role of IM in the RL framework. Afterwards, current challenges of RL which are tackled by IM are pointed out (see Listing 1.1 for a summary), and the embedding of IM in RL is presented. Finally, this chapter is concluded with the classification of IM in RL: knowledge acquisition and skill learning where a subdomain of knowledge acquisition is more deeply investigated in chapter 5 as it represents the domain on which the implementation (chapter 6) is based on.

4.1 Natural Human Motivation

In a digression to psychology, IM is described as the natural propensity of humans to gather knowledge and assimilate themselves. Activities which are intrinsically motivated are usually carried out because they are enjoyable and interesting. Therefore, the reward in IM is based on the activity itself. In opposition to that, Extrinsic Motivation (EM)

reflects external control sources or true self-regulation which may achieve separable outcomes, e.g. food or money. Motivation in general, is very particular to different people (Ryan and Deci, 2000). Not only can it vary in the amount/level of motivation a person can have, but the orientation of motivation can be different as well. The question about the orientation of a person's motivation is very significant since e.g. pupils may be very motivated to excel in a subject out of curiosity (IM) or rather out of the possibility of achieving good grades (EM) which might yield further positive benefits for them or avoid sanctions.

Moreover, research pointed out that the quality of performance and experience depends on the kind of motivation propelling behaviour. For instance, it is shown that IM allows for an improved learning experience and more creativity. On the other hand, extrinsic motivated actions might be executed with disinterest, resistance or optionally with an inner acceptance as the action poses an identifiable value (Ryan and Deci, 2000). Continuing for instance with the previous example of the pupils' motivation, it is the teacher's job to promote inherently uninteresting actions by means of extrinsic motivation with a focus on volitional and active forms instead of controlling and passive ones. By doing this, the pupils are more likely to identify the action's value so that they internalize the behaviour which allows for an improved learning experience.

Finally, Ryan and Deci add a third type of motivation to the taxonomy of natural human motivation, the so-called amotivation which describes activities that are perceived as irrelevant. Amotivation has different origins such as a feeling of incompetence to carry out a certain task, an action not yielding any value, or any desired outcome. Therefore, the taxonomy of human motivation consists of three branches: amotivation, extrinsic motivation, and intrinsic motivation, ranging "*from amotivation or unwillingness, to passive compliance, to active personal commitment. With increasing internalization (and its associated sense of personal commitment) come[s] greater persistence, more positive self-perceptions, and better quality of engagement.*" (Ryan and Deci, 2000, see p. 60f)

4.2 The Role of Intrinsic Motivation in Reinforcement Learning

With the help of IM, existing challenges of RL such as abstracting actions or the exploration of the environment should be addressed which were not able to be solved by the promising capabilities of Neural Network (NN). As mentioned in the Problem Description, in traditional RL approaches, the agent is unable to explore the environment in settings where rewards are sparse. Furthermore, even in dense environments, the agent's learned behaviour is rather not reusable in similar as well as in completely different tasks. This is due to the fact that an agent finds it very challenging to generalize obtained skills and thus is incapable of high-level decision making. For instance, Todorov et al. indicate that such a high-level abstraction could be the trajectory of a robot to open a door which consists of low-level actions/movements in four directions.

	With *feedback*	Without *feedback*
Active	Reinforcement	Intrinsic motivation
Passive	Supervised	Unsupervised

Table 4.1: A taxonomy on the different types of learning[1]

As described earlier, the idea for IM in RL is derived from nature, especially by the animals' urge for exploration. However, a probably better fitting example would be a baby's task to solve existential problems as it best reflects an RL-agent in an unknown environment. For instance, a baby who avoids suffering from hunger and thirst has to learn the consequences of its interactions. Although immediate needs might be satisfied for some time, the baby keeps on exploring by conducting various experiments. Questions on certain movements of, e.g. eyes, fingers, or the tongue and their expected sensory feedback are continually answered. Ultimately, this facilitates the baby's ability of prediction which allows for easier planning of actions. Furthermore, the baby constantly strives to discover new effects as it eventually gets bored by the things it has already grasped, thus allowing to learn very complex behaviours by building on previously gathered knowledge. By following this simple algorithm of maximizing the internal joy of discovering or creating new patterns, eventually, the baby might become a computer scientist (Schmidhuber, 2010). Concerning RL, this process allows an agent to autonomously gather new skills and knowledge which then facilitates the mastering of a novel task.

Besides the improved exploration, IM generates other benefits as well. For instance, an agent can learn skills incrementally and independently while not being dependant on its given main task. Furthermore, the agent can decide on which skills might be adequate for a certain task and thus improve them selectively. This is all possible due to the broader intrinsic reward function and the fact that this spares the necessity of an expert or supervisor. Hence, without a supervisor, learning becomes more flexible and RL algorithms generalize more across different environments (Aubret et al., 2019).

Additionally to the classification of subfields in Machine Learning (ML) in section 2.1, Aubret et al. add a fourth field to the spectrum of learning types, differentiating between reinforced and intrinsically motivated learning (see Table 4.1). Therefore, a distinction is drawn depending on whether or not expert supervision, which gives feedback, is involved.

4.3 Challenges Tackled with Intrinsic Motivation

In the following subsection the four main challenges of Deep Reinforcement Learning (DRL) are introduced where for each of these explained problems, IM provides a viable solution.

[1](Aubret et al., 2019, see p. 4)

4.3.1 Sparse Rewards

As noticed in section 1.1, traditional RL algorithms perform best in environments which reward the agent densely. By using naive exploration approaches, e.g. ϵ-greedy, good results can be accomplished with an additional option, for instance, to add Gaussian noise to the action selection process which further improves exploration (Lillicrap et al., 2019). However, even these improved exploration methods are incapable of achieving a good performance in sparsely rewarding settings such as *Montezuma's Revenge* which is a typical benchmark for such environments. In order to receive a reward in this game, the agent has to collect different items such as keys and use them to open doors. These events occur rather rarely and hence, the agent does not receive immediate feedback concerning the impact of its performed actions, e.g. moving into a certain direction. Therefore, it is not uncommon that an agent never reaches such a reward, thus preventing the creation of a good policy (Aubret et al., 2019).

This issue can be tackled by shaping a task-specific intermediary function which rewards densely. Hence, additional rewards are created which should point the agent in the right direction. Although this approach might work out well, it is usually impractical as in order to craft this function, very high expert knowledge is needed to carefully design this rewarding system. Additionally, such functions commonly pose many side effects resulting in unexpected errors as the agent might pick up an undesired behaviour by exploiting the reward system.

4.3.2 Building a Good State Representation

According to Böhmer et al., a good state representation allows for good generalization, is low-dimensional, should represent values of a policy truthfully, and fulfil the Markov property.

By employing a good representation, the agent's learning process can be considerably accelerated. For instance, considering a navigation task, it causes a significant difference whether an agent has to find out its location and the location of its target by accessing non-linear transformed raw pixels or by obtaining the positions directly, creating solely the need to check the distance between the target and itself.

Normally, a state representation can be constructed using expert knowledge of a specific domain. However, not task-specific, hence, generic priors can be used as well with the benefit that these priors can be learned by representation-learning algorithms, creating disentangled, minimal feature spaces (Bengio et al., 2014). In traditional RL, the problem of learning a good state representation is exacerbated as in order to improve, the agent depends on optimizing the reward signal via backpropagation, thus depending on receiving rewards densely. With the additional possible obstacle of noise in the raw state representation, the agent will learn nothing from the interactions made in a sparse rewarding environment, although they might be rich in information. Even if the agent manages to learn a state representation based on a reward signal, the representation will not generalize across different tasks. Hence, IM is a key component as

state representations learned through IM are independent of the actual task and therefore generalizable (Aubret et al., 2019).

4.3.3 Temporal Abstraction of Actions

In temporal abstraction of actions, a sequence of low-level actions is summarized with a so-called intra-option policy that is referenced to a high-level action. Therefore, a high-level action or so-called option consists of multiple low-level actions/other options. By choosing an option in a certain state, the intra-option policy defines which low-level actions must be carried out in each subsequent state until the high-level action is completely executed. Furthermore, the sequence length of these actions is usually fixed (Aubret et al., 2019). By utilizing these abstractions, the learning process can be sped up significantly. An additional consequence of allowing these abstractions is that the problem of delayed rewards, as described in section 2.1, is tackled since the option can be credited directly. Aubret et al. describe this with the following example:

> "[L]et us assume that a robot is trying to reach a cake on a table which is far from the robot. If the robot has an option `get to the table` and follows it, the robot will then only have to take the cake to be rewarded. Then it will be easy to associate the acquisition of the cake (the reward) to the option `get to the table`. In contrast, if the robot has to learn to handle each of its joints (low-level or primitives actions), it will be difficult to determine which action is responsible for the acquisition of the cake, among all executed actions."
>
> (Aubret et al., 2019, see p. 5f)

Another benefit of temporal abstractions of actions is that they allow for better exploration. Similarly to the previously made example, an exploration task can easily be combined to an option, thus enabling it to immediately receive a reward and therefore "making a sparse reward function dense". In order to obtain such an intra-option policy, there are two options: define it manually by using expert knowledge or teach it using a reward function. However, the former has the problem that the learned options are not generalizable across multiple tasks and hence are useless for exploration tasks (Aubret et al., 2019). Similar to the problem of Building a Good State Representation, by using IM, this challenge can be tackled.

4.3.4 Building a Curriculum

The challenge of building a curriculum deals with structuring tasks in a multi-task RL scenario, a scenario where an agent has to solve multiple tasks at once. In such scenarios, a so-called curriculum is employed that acts as a schedule, defining the order in which different tasks should be solved. This idea comes from the observation that

incrementally building upon gathered knowledge from similar but less complex tasks facilitates mastering more difficult ones (Aubret et al., 2019).

For instance, a general task for an agent could be that it has to place objects such as cubes into boxes. However, this task can be divided in subtasks such as grabbing the cube and then putting it into a box. By splitting up this task into the two subtasks, the agent can take advantage of the prior learned skill of grabbing a cube for the task of putting it into a box. Without this differentiation, the agent might be unable to ever fulfil the general task as it would take the agent a long list of subsequent actions/joint movements to succeed.

Usually, standard methods for decomposing a general task mostly rely on expert knowledge which unfortunately always goes along with the inability to scale and generalize well and thus, IM finds its application to solve this issue as it does not only speed up the learning process but it also facilitates exploration.

4.4 Classification of Intrinsic Motivation

Concerning the classification of IM in RL, the authors Aubret et al. propose a new categorization, differentiating between knowledge acquisition and skill learning. With this separation, emphasis is laid on skill abstraction and skill acquisition.

Concerning the motivation based on **knowledge acquisition**, the agent's internal goal is to gather new knowledge about its surroundings. In this scenario, knowledge is described as its ability to e.g. control the environment, understand occurring processes, explore unknown areas, or assess proximity. Based on this definition, building an agent with this motivation can greatly improve exploration in settings with sparse rewards. This is usually done by giving an agent a feeling for state novelty or information gain which poses an intrinsic reward that it tries to maximize. Another benefit presents the idea of empowerment which means that an agent is rewarded by IM whenever it approaches an area that can be controlled. Finally, this motivation might allow the agent to create a relevant state representation and thus, this branch faces the challenges described in subsection 4.3.1 and subsection 4.3.2.

The second branch, concerning motivation based on **skill learning**, is described as the ability of an agent to efficiently acquire skills that are task-independent and preferably reusable. Here, one idea deals with the agent's ability of abstracting skills in order to study them one by one or rather to create a skill representation which challenges the introduced problem of subsection 4.3.3. Finally, choosing the best-suited skills to fulfil a given task and the order in which they should be obtained is another IM that can tackle the challenge mentioned in subsection 4.3.4.

An overview of the just described classification with practical approaches can be observed in Table 4.2.

Knowledge Acquisition
Exploration
Prediction Error State Novelty Novelty as Discrepancy towards other States[3] Information Gain
Empowerment
Learning a Relevant State Representation
State Space as as Measure of Distance One Feature for one Object of Interaction

Skill Learning
Skill Abstraction
Building the Goal Space from the State Space Mutual Information between Goals and Trajectories
Curriculum Learning
Goal Sampling Multi-Armed Bandit Adversarial Training

Table 4.2: Classification of IMs in RL[2]

4.5 Embedding Intrinsic Motivation into Reinforcement Learning

In order to embed IM into RL a new framework is presented by Singh et al., splitting the reward signal into primary and secondary rewards. The primary reward signal represents the standard one whereas the secondary reward describes a local reward signal which value is similarly computed as the value function (Equation 2.8) denoted in section 2.3. Additionally, the representation of the RL framework as a Markov Decision Process (MDP) is again taken into consideration. Instead of describing the agent's surroundings traditionally, they suggested a division of the environment into two separate parts: the external part and the internal part, where the external part reflects the environment of the agent in which it has to complete its assigned task and on the other hand, the internal part which calculates the intrinsic reward based on previously made interactions (Aubret et al., 2019).

[2](Aubret et al., 2019, see p. 9)

[3]Continuing, Aubret et al. unite state novelty and novelty as discrepancy towards other states

The newly adapted RL framework can be observed in Figure 4.1. Here, the critic describes the part which computes the reward in either way and assigns the credit to the carried out actions. Additionally, the state now includes sensations optionally combined with the agent's performed actions. However, the left panel yet describes the traditional RL framework in a slightly different notation to the one presented in section 2.2 whereas the right panel includes the new refinements where all rewards derive from the internal part (Singh et al., 2010).

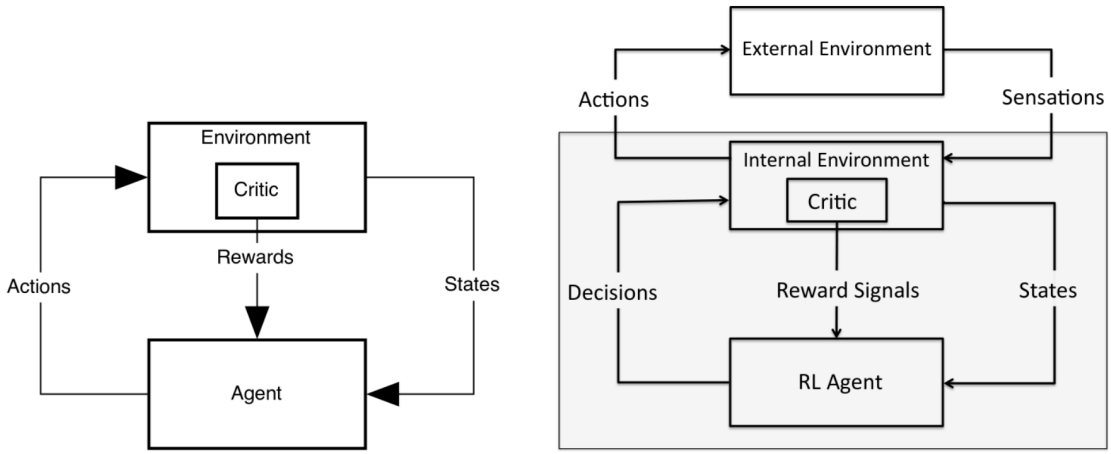


Figure 4.1: Adapting the classic RL framework to incorporate IM[4]

One of the most prominent IM is curiosity which is deployed because it acts as a meta-skill allowing the agent to easily acquire different behaviour. In general, the acquisition of task-unspecific skills by means of IM allows for more intelligent behaviour, thus increasing the efficiency of the adopted framework by serving assigned goals when compared to the traditional one.

When implementing an RL algorithm, there are multiple ways to include an intrinsic reward signal. The most common approach is to simply combine the existing extrinsic reward signal (r_{ext}) with the new intrinsic one (r_{int}) by defining coefficients α and β. These coefficients serve the purpose of weighting the two signals depending on how important either one should be to the agent. As this approach poses the core research question of this thesis, that is, how to optimally combine these two rewards, the equation $r = \alpha r_{int} + \beta r_{ext}$ has already been introduced in section 1.2. Additionally, when the intrinsic reward is dependant on the value of the extrinsic one, the sum is solely computed on the level of the value function, i.e. $V(s) = \alpha V_{int}(s) + \beta V_{ext}(s)$ as it is the case in the work of Kim et al. (2019b). This work as well as many other exploration based methods such as the paper by Burda et al. (2018b) are discussed in the following chapter. Besides that, the papers by Gregor et al., Vezhnevets et al., and Huang et al. are also worth

[4](Singh et al., 2010, see p. 2)

mentioning as they combine extrinsic and intrinsic rewards as well but on the other hand consider IM in the context of skill learning which is not a focus of this thesis.

CHAPTER 5

Exploration

In this chapter only the significant branch of knowledge acquisition through exploration is considered as this thesis builds upon a "Large-Scale Study of Curiosity-Driven Learning", a paper published by Burda et al. (2018a). Here, emphasis is placed on prediction-error based methods but for the sake of completeness other exploration methods based on state novelty or information gain are taken into account as well. However, the following list of publications poses only a selection which was chosen mainly based on the performance of these methods or was written by authors who contributed to the underlying paper of this thesis.

- Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models (Stadie et al.)
- Curiosity-Driven Exploration by Self-Supervised Prediction (Pathak et al.)
- EMI: Exploration with Mutual Information (Kim et al.)
- Unifying Count-Based Exploration and Intrinsic Motivation (Bellemare et al.)
- Exploration by Random Network Distillation (Burda et al.)
- Curiosity-Bottleneck: Exploration By Distilling Task-Specific Novelty (Kim et al.)
- Self-Supervised Exploration via Disagreement (Pathak et al.)

A more detailed list is provided by Aubret et al. (2019, see Table 3) in their section on *Knowledge acquisition*.

By not only considering the approach by Burda et al. (2018a), this chapter also represents a state-of-the-art review for exploration via IM. Finally, at the end of this chapter

"Go-Explore: A New Approach for Hard-Exploration Problems" by Ecoffet et al. (2019) is presented which is currently the best performing method to solve the exploration challenge offered by *Montezuma's Revenge* as of August 2020.

5.1 Prediction Error

The idea of prediction-error based exploration is to point the agent in a direction where a prediction of the next state, based on a state-action tuple, is rather difficult. Formally, this prediction-error based intrinsic reward[1] can be denoted as

$$r_{int} = ||\phi(s_{t+1}) - F(\phi(s_t), a_t)||_2 \tag{5.1}$$

where $||x||_2$ is the L2 distance/the Euclidean distance (Aubret et al., 2019). In this equation, ϕ stands for a generic function which encodes the raw state space into a space of features and F denotes a NN/forward dynamics network which predicts the following feature space depending on the current encoded state and the taken action.

In the core paper to this thesis, Burda et al. (2018a) deployed four different encodings for ϕ and compared the auxiliary feature spaces in terms of compactness, sufficiency, as well as stability in order to find out whether or not an agent can fulfil tasks when purely driven by IM. Concerning the properties of an encoding function ϕ, they describe compactness to be low in dimensionality, sufficiency to contain all the relevant information about the environment, and stability to be a stationary function. Instead of the previously denoted intrinsic reward, Burda et al. (2018a) used the so-called surprisal:

$$r_{int} = -\log p(\phi(s_{t+1})|s_t, a_t) \tag{5.2}$$

Regarding the actual encoding functions used for their investigations, they started with the identity function, $\phi(s) = s$, which proved to be very inefficient as it lacked compactness even though the encoding was sufficient and stable. Additionally, the prediction-error based on this encoding is probably dominated by irrelevant small details.

In contrast to that, they introduced so-called Random Features (RFs) where features are extracted using a random initialized Convolutional Neural Network (CNN)[2]. This network is not trained as its weights are fixed immediately after initialization. Furthermore, the properties of this encoding are stability as the network was fixed and possible compactness depending on the amount of RFs extracted. However, this encoding function might fail to be sufficient as only RFs are extracted and thus chances are that they do not convey all the relevant information.

A further method for feature encoding posed the idea of Variational Auto-Encoders (VAEs)[3]. In that case, a feedforward network/auto-encoder compresses the state space

[1] For uniformity reasons, the Greek letter ϕ will be used instead of g throughout this chapter
[2] For further information, the literature by Aggarwal (2018, see p. 40ff) is recommended
[3] For further information, the literature by Aggarwal (2018, see p. 207-214) is recommended

that is later used to predict the distance between the current state s_t and s_{t+1}. Moreover, the encoded part of this auto-encoder represents the function ϕ which allows for a compact and sufficient but unstable feature encoding. For this encoding method, the authors build upon the initially proposed method by Stadie et al.

Lastly, the encoding in Inverse Dynamics Features (IDFs) was explored where the task is to predict the action a_t which led from state s_t to s_{t+1}. The features of s_t and s_{t+1} learned by a predicting forward model correspond to parts of the environment which the agent is in immediate control of since it can estimate the consequences of its actions. This encoding yields the benefit of compactness as it successfully encodes the raw feature space but fails to be stable and additionally might be insufficient in containing all the significant information. As a side note, this idea is mainly based on the Intrinsic Curiosity Module (ICM), as depicted in Figure 5.1, proposed by Pathak et al. (2017).

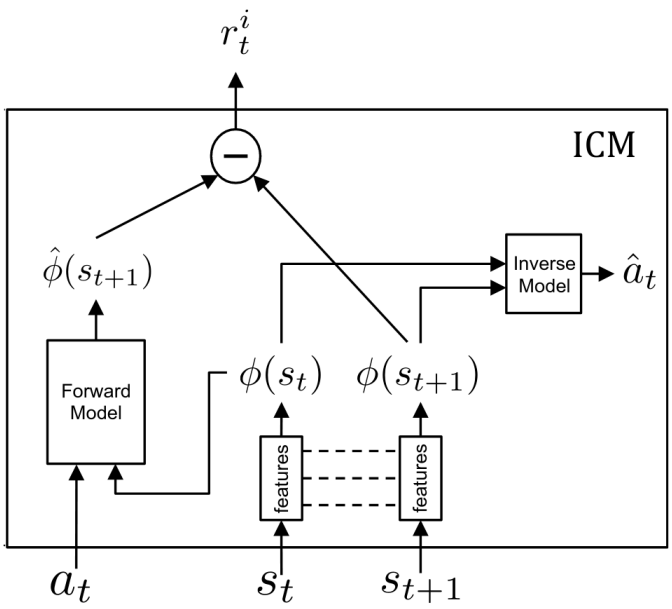


Figure 5.1: The ICM by Pathak et al. (2017) where r_t^i denotes the curiosity based intrinsic reward[4]

In summary, the encoding of the raw feature space using RFs showed that a curiosity based agent can be a very good competition to other state-of-the-art methods and are especially well-suited for simpler environments with less sophisticated visual observations. This is due to the fact that in these settings it is easier to preserve enough important information over the raw signal and thus be sufficient. A good example for this is the simulated environment of the Atari 2600 gaming console where RFs performed better in over 70% when compared to an agent that chooses its actions arbitrarily. Although RFs can perform astonishingly well, they generalize badly across different environments. Here, the encoding with IDFs performs better as it can transfer learned features from, e.g. one

[4](Pathak et al., 2017, 3)

Mario Bros level to another (Burda et al., 2018a). Additionally, the IDF-curious agent was able to achieve a better score than a random agent in over 75% of the Atari 2600 games and when compared to an RF-curious agent, it performed better 55% of the time, thus making IDFs the preferred encoding. Concerning methods based on VAEs for ϕ, they perform either worse or the same as RFs or IDFs and additionally were somewhat unstable.

Unfortunately, all of these feature encodings are unable to deal with the white-noise problem which was mentioned in section 1.1.

Another selected approach is the paper "EMI: Exploration with Mutual Information" by Kim et al. (2019a). With a similar action space as in the work by Pathak et al. (2017), they managed to significantly improve the agent's performance in several Atari 2600 games. However, this fell at the expenses of multiple additional layers of complexity. Instead of training a complex forward model that predicts the next encoded state, they transformed the complexity into a maximization problem of mutual information I between $I([s, a]; s')$ and $I([s, s']; a)$ and only used a simple linear model for F to represent the space (Aubret et al., 2019). Concerning the problem with stochasticity, Kim et al. (2019a) were able to tackle occurring white-noise by proposing an additional error model which offloads non-linearity from the model F. By doing so, screen transitions, e.g. moving from one room to another in *Montezuma's Revenge*, which are hard to predict and similar to white-noise, can be bypassed.

5.2 State Novelty

A further important approach to enable exploration by IM is the idea of state novelty where the agent earns intrinsic rewards when it enters states it has not visited before. This kind of idea can be formalized as a count-based function which decreases the agent's intrinsic reward for entering a certain state depending on how often it had already visited this state before. Formally, the count-based function is denoted as the following:

$$r_{int} = \frac{1}{N(s_t)} \tag{5.3}$$

where $N(s_t)$ stands for a counter which represents the agent's number of visits of the state s_t (Aubret et al., 2019). As one can imagine, this method is only efficient in discrete state spaces as in continuous ones the agent hardly visits states multiple times.

A solution to this has been proposed by Bellemare et al. which is based on density models that enable a pseudo-count. By utilizing this method, the number of visits in a state as well as related neighbourhood states are combined to one counter which allows for better generalization. Therefore, the previously made notation is adapted to the following equation:

$$r_{int} = \frac{1}{\hat{N}(s_t)} \tag{5.4}$$

where $\hat{N}(s_t)$ defines the pseudo-count which is denoted as follows:

$$\hat{N}(s_t) = \frac{p(s)(1 - p'(s))}{p'(s) - p(s)} \tag{5.5}$$

Here, $p(s)$ stands for a density model which calculates the probability of entering state s and $p'(s)$ is the probability to visit state s afterwards. The usage of the pseudo-count based on density models yield good results which enable exploration but add additional layers of complexity (Aubret et al., 2019).

For the next method, Burda et al. (2018b) built upon their idea of using RFs but this time utilizing the encoding to combine it with a pseudo-count based approach that enables exploration through state novelty. However, they still somewhat made usage of prediction errors as they deployed a second NN which solely fulfils the purpose of predicting the output of the random initialized one. The idea behind this is that the second NN will produce a rather large error if the agent has never visited a specific state but over time gets more certainty about its predictions the more the agent has been in that state. By doing so, not only were they able to reduce the complexity of previous proposed methods such as the one which is based on density models but they also achieved one of the best scores in *Montezuma's Revenge* (Ecoffet et al., 2019).

One more selected state-novelty based approach is the one by Kim et al. (2019b) who utilized the possibility of distilling recent states into a probability distribution. In their paper "Curiosity-Bottleneck: Exploration By Distilling Task-Specific Novelty", they propose a method which produces a latent state space with a distribution of very large entropy that is based on their prior work of maximizing mutual information. Stating Aubret et al.:

> "The intrinsic reward for a state is then the KL-divergence between a fixed diagonal Gaussian prior and the posterior of the distribution of latent variables. It results that, as long as the agent does not find any reward, it will look for rare states which have a distribution in the latent space that is different from the prior (common states fit the prior). When the agent finds the reward, the latent distribution will be different from the prior and the intrinsic reward will guide the agent towards interesting areas."
>
> (Aubret et al., 2019, see p. 12)

Referring to section 4.5, Kim et al. (2019b) also made usage of EM by utilizing an extrinsic reward signal to tackle stochasticity which originates from white-noise.

5.3 Information Gain

A final method to allow exploration is exploration via information gain. In this case, an intrinsic reward is credited depending on the agent's ability to reduce uncertainty of the

dynamics of its environment. By using information-gain based methods, the agent can be led to states it is rather uncertain about while at the same time being able to predict environment transitions. Therefore, the agent is unaffected of the white-noise problem based on stochasticity or state transitions (Aubret et al., 2019). Furthermore, under the usage of the parameter set θ which is used in a parametric model and the function U which defines uncertainty, the information gain can be formalized as follows:

$$r_{int} = U_{t+k}(\theta) - U_t(\theta) \tag{5.6}$$

A well-suited example to introduce information-gain based approaches is "Self-Supervised Exploration via Disagreement" by Pathak et al. (2019). Here, the authors use multiple NNs/forward models to predict the next state s_{t+1} based on a state-action tuple. Further, the different predictions of the subsequent state s_{t+1} are composed and their variance acts as the intrinsic reward. Depending on the number of trained networks, the more forward models are learned, the better the performance of the agent will be as the models' predictions will converge to the expected observation of state s_{t+1}. Moreover, for illustration purposes, the algorithm is visualized in Figure 5.2.

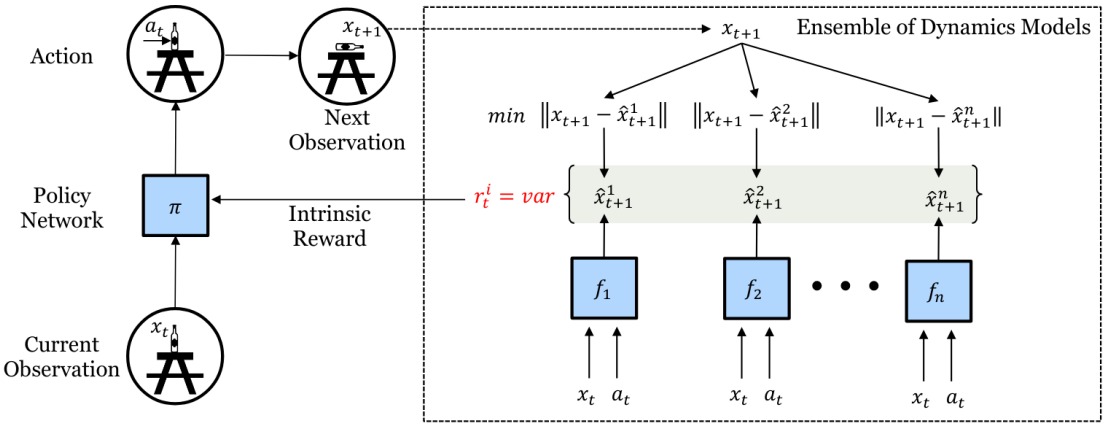


Figure 5.2: A visualisation of the algorithm introduced in "Self-Supervised Exploration via Disagreement" where the current observation is denoted as x_t and r^i_t stands for the intrinsic reward[5]

By utilizing the variance of the predictions, the agent's desired behaviour is depicted as a high variance that solely occurs if the models are not trained yet which forces the agent to explore. Concerning stochasticity, this problem is also solved by the variance since the irritating white-noise becomes irrelevant as the average of all predictions converges truthfully. Hence, this approach yields good performance when compared to other state-of-the-art methods but is very computationally expensive due to the necessity of training multiple forward models.

[5](Pathak et al., 2019, see p. 2)

5.4 Current State of the Art Method

The current record on *Montezuma's Revenge* is held by Ecoffet et al. (2019) who were able to drastically improve the state of the art with a four times higher score than previously introduced methods. Despite the fact that IM algorithms are usually the best performing solutions to encourage exploration in sparse environments, by the introduction of their paper "Go-Explore: A New Approach for Hard-Exploration Problems", the authors presented a promising solution which however is not based on IM. Surprisingly, their exploration method is based on an agent that selects actions randomly, an approach which on its own would usually not perform well. However, they added a significant prior step to exploration as they teach the agent to re-enter a previously visited promising position before it starts exploring again. This step solves a fundamental issue of intrinsically motivated exploration which can be observed in Figure 5.3. Although IM is normally the standard approach to environments with sparse rewards, they all have one fundamental problem in common where the agent fails to re-enter already visited states as the path which leads to these states is not interesting anymore and thus hinders the agent to further discover its environment.

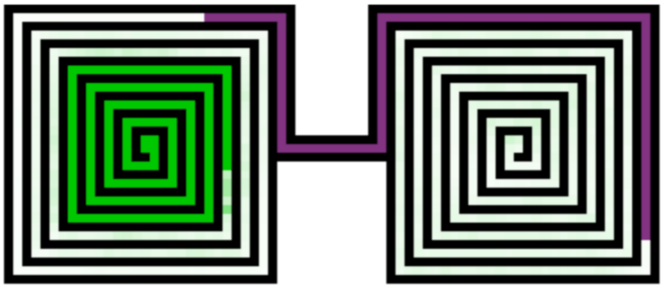


Figure 5.3: An illustration of intrinsically motivated exploration methods where an agent is unable to rediscover promising areas[6]

In Figure 5.3, this problem is illustrated where the white path denotes already discovered parts of an agent's environment, the green one stands for intrinsic rewards it has yet to obtain, and the purple area marks the path it has explored in the current episode. This example shows the problem of an agent who is uninterested in visiting known states again and thus unable to ever discover the green intrinsically rewarding area.

According to the authors, their exploration algorithm is structured as follows:

[6](Ecoffet et al., 2019, see p. 3)

> "It exploits the following principles: (1) remember states that have previously been visited, (2) first return to a promising state (without exploration), then explore from it, and (3) solve simulated environments through exploiting any available means (including by introducing determinism), then robustify (create a policy that can reliably perform the solution) via imitation learning."
>
> (Ecoffet et al., 2019, see p. 1)

A graphical illustration of this algorithm can be inspected in Figure 5.4 where states are encoded in so-called cells which are low-resolution, grey-scale images.

With this algorithm, the trained agent was able to achieve superhuman performance and thus surpass human players in the game of *Montezuma's Revenge*. Furthermore, in another difficult exploration game called *Pitfall*, their algorithm was the first one which managed a score above zero. Lastly, another achievement worth mentioning is that it outperformed state-of-the-art solutions of imitation learning where an agent is trained on recorded man-made gameplay due to the fact that their approach can cheaply create such demonstrations on its own. For all these reasons, *Go-Explore* is a very promising direction for further research, especially focusing on how to combine existing methods with this novel idea.

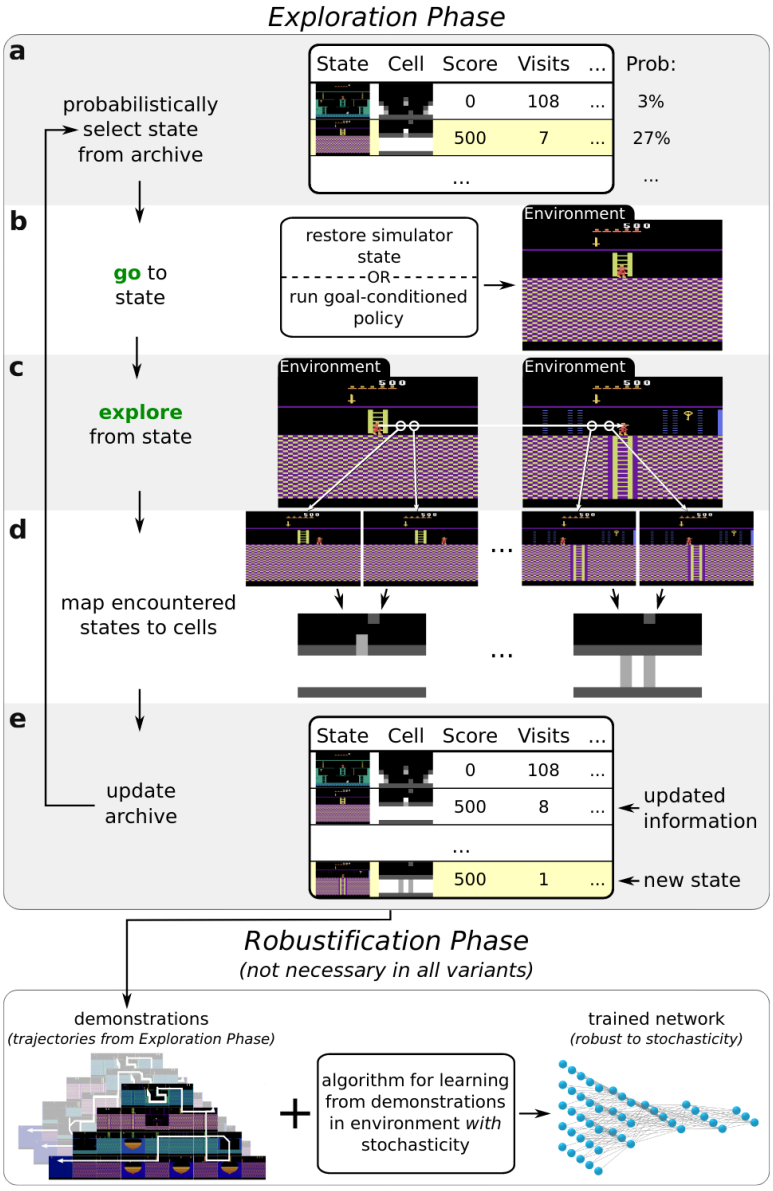


Figure 5.4: The simple, yet astonishingly good performing algorithm "Go-Explore" by Ecoffet et al. (2019) which is currently the best performing approach to *Montezuma's Revenge*, directly follow by "Random Network Distillation" from the authors Burda et al. (2018b)[7]

[7](Ecoffet et al., 2020, see p. 3)

CHAPTER 6

Implementation

In order to answer the posed questions of section 1.3, the thesis builds upon the released source codes and environments from Burda et al. (2018a) and Pathak et al. (2017) which can be found on their websites.[1,2]

The implementation relies on the same tools used by the preceding authors since it guarantees optimal reproducibility of their results. In an overview, these tools are the *Python*[3] programming language, the Machine Learning (ML) platform *TensorFlow*[4], and lastly the *OpenAI Gym*[5] which is a toolkit for developing and comparing Reinforcement Learning (RL) algorithms. Concerning the creation of plots to visually present the results, the *Matplotlib*[6] library is used.

The implementation progress was structured into the following five milestones:

1. Setting-up the environment, including the installation of the *CUDA*[7] toolkit and the *cuDNN*[8] library
2. Analyzing the implementation of the Proximal Policy Optimization (PPO) algorithm
3. Analyzing the implementation of different auxiliary tasks such as Random Features (RFs) and Inverse Dynamics Features (IDFs)
4. Tuning of the coefficients α, β and comparing them to state-of-the-art results for densely and sparsely rewarding environments

[1] https://pathak22.github.io/large-scale-curiosity
[2] https://pathak22.github.io/noreward-rl/
[3] https://www.python.org/
[4] https://www.tensorflow.org/
[5] https://gym.openai.com/
[6] https://matplotlib.org/
[7] https://developer.nvidia.com/cuda-toolkit
[8] https://developer.nvidia.com/cudnn]

5. Implementation of the synthetic generated noisy-TV problem, proposed by Burda et al. (2018a)

During this incremental process, an emphasis is laid on two out of five sparse Atari 2600 games which were categorized by Bellemare et al. (2016) and picked by Burda et al. (2018a), namely *Montezumas's Revenge* (Figure 7.1a) and *Freeway* (Figure 7.1b).

Regarding milestone three, *Random Features* and *Inverse Dynamics Features* as introduced by Burda et al. (2018a) are primarily the focus. Concerning milestone four and the comparison to state-of-the-art approaches, results from methods based on knowledge acquisition via exploration by the preceding authors are considered.

In the following few sections an overview is provided of the most significant parts of the implementation. The source code to this Bachelor's thesis is provided on the author's *GitHub* page.[9]

6.1 Proximal Policy Optimization and Auxiliary Tasks

As aforementioned, the algorithm of choice is PPO as it does not need a lot of hyperparameter tuning and thus allows for quick scalability across different environments and different auxiliary tasks. Concerning its implementation, the pseudo-code for the PPO algorithm is denoted in section 3.4 where all the objective functions are described in detail as well. In order to optimize the objective, defined in (Equation 3.11), a so-called Adam optimizer with a learning rate of 0.0001 is used.

Additionally, as the reward signal is non-stationary it is normalized since it helps the agent/the value function to learn more quickly. This is done by calculating the standard deviation continuously over the sum of the discounted rewards and further dividing the current rewards by the square root of the obtained value. The advantages are also scaled using a Z-normalization where each batch is normalized to have an approximate mean of 0 and a standard deviation of 1.

The embedding network as well as the policy network are structured identically where both are based on a Convolutional Neural Network (CNN) architecture consisting of three convolutional layers with a final reshaped feature dimension of 512 that uses the leaky Rectified Linear Unit (ReLU) as a non-linearity. The filters, kernel sizes, and strides of these layers are as follows:

1. 32 filters, a kernel size of 8×8, and a stride of 4
2. 64 filters, a kernel size of 4×4, and a stride of 2
3. 64 filters, a kernel size of 3×3, and a stride of 1

[9] `https://github.com/RafaelSterzinger/Bachelor-Thesis`

According to Burda et al. (2018a), this is a commonly used network for experiments and benchmarks of Atari 2600 games. Furthermore, the Neural Networks (NNs) responsible for learning IDFs or RFs were added on top of the embedding network. These forward dynamics networks consist of five fully connected layers with a hidden layer size of 512 as well. Regarding their non-linearity, they all use the leaky ReLU.

In order to increase the stability of the deployed algorithm multiple actors which work in parallel are considered. Throughout all of the following conducted experiments, 128 parallel actors were employed in their own environment to solely gather data by interacting with their surroundings in order to train the agent. According to algorithm 3.1, this also affects the batch size. Concerning the optimization step of the algorithm, for each iteration, three epochs were used to optimize the policy as this allows for faster progression.

A summary which not only includes the hyperparameter just mentioned but also the other ones used by the PPO algorithm is provided by Table 6.1

Hyperparameter	Value
Horizon (T)	128
Learning rate	1×10^{-4}
Num. epochs (K)	3
Minibatch size (M)	8
Discount (γ)	0.99
GAE parameter (λ)	0.95
Num. actors (N)	128
Cliprange (ϵ)	0.1
VF coeff. (c_1)	1
Entropy coeff. (c_2)	0.001

Table 6.1: PPO hyperparameters for the conducted experiments utilizing the notations introduced in section 3.4

Finally, another significant aspect of the implementation is the omission of a *done* signal which is normally given at the end of an episode. By ignoring such a signal, information about the true extrinsic reward function is not leaked which allows for truly intrinsically motivated agents (Burda et al., 2018a). Therefore, the death of an agent turns just into another state transition which solely can be avoided if passing away is considered to be boring. Although the agent is now given an option, Burda et al. (2018a) state that it usually avoids death since dying would only reset the game which puts it back into a well-known area. Unfortunately, sooner or later the agent will eventually lose and hence the agent must restart at the beginning where it might get detached from the rewarding area. This problem has been depicted in Figure 5.3.

Apart from the policy and its optimization, the implementation of the auxiliary tasks

that are responsible for the embedding are also worth mentioning. Before the embedding occurs, the raw observation which consists of pixels is pre-processed in order to reduce dimensionality. This is done by resizing the current frame to a size of 84×84 and converting it into a single colour channel, resulting in only grey-scale images with reduced dimensionality (Figure 6.1a).

Concerning the observation during the training process, a stack including the current frame as well as three previous ones is used to train both the policy and the forward dynamics network. This is due to the fact that the agent is able to recognize partial observability within its environment, if multiple subsequent frames are taken into consideration. Additionally, standard frame-skipping wrappers are used in the Atari games which enables the execution of an action four times in a row, thus speeding up the training process linearly.

In addition to the normalization of rewards and advantages, observations are normalized as well. Here, mean and standard deviation are calculated prior to the initialization of the embedding network which are based on data that is obtained by letting a random agent explore its environment for 10000 time-steps. Afterwards, the generated data is used to normalize the input frames during training which is especially useful as it ensures for features to have small variance at the beginning and for them to hold less deviation across other environments.

6.2 Hyperparameter Tuning

In addition to the hyperparameters mentioned in Table 6.1, the coefficients α and β, responsible for the weighting of intrinsic and extrinsic rewards, are used as well and must be tuned depending on the conducted experiment. During the tuning process only coefficients within a range of 0 and 1 are taken into consideration with an additionally added constraint that $\alpha + \beta$ must be equal to 1. However, the authors did not add this limitation:

> "We [the authors] would like to emphasize that this [combining extrinsic and intrinsic rewards] is not the focus of the paper, and these experiments are provided just for completeness. We just combined extrinsic (coefficient 1.0) and intrinsic reward (coefficient 0.01) directly without any tuning."
>
> (Burda et al., 2018a, see p. 9)

This poses only a minor change which is solely taken into account as it facilitates the tuning during e.g. a gridsearch[10] by more or less bounding the search space. Besides using completely intrinsically motivated agents ($\alpha = 1, \beta = 0$) as in the work by Burda et al. (2018a) and also evaluating the performance of only extrinsically rewarded agents ($\alpha = 0, \beta = 1$), a mixed form is considered.

[10] For further information, the literature by Goodfellow et al. (2016, see p. 420ff) is recommended

In order to find good combinations of these coefficients, a gridsearch was implemented using a shell script. The initial configuration starts with coefficients $\alpha = 1$ and $\beta = 0$ that are decremented/incremented during each iteration by a Δ of 0.1 which allows for an exhaustive combination of these values. This process is repeated multiple times for each considered game where after each completed search the range as well as the rate of change (Δ) are redefined depending on the obtained performance.

6.3 Noisy-TV Problem

For the implementation of the noisy-TV problem, introduced by Schmidhuber and described in section 1.1, an additional wrapper was implemented which wraps around the the pre-processed observation and adds a fixed region of random noise. Besides the task of combining the coefficients of the rewards, the problem of stochasticity is implemented as experiments are conducted to analyze the impact of Extrinsic Motivation (EM) on a distracted, intrinsically motivated agent. The modified agent's observation which is caused by the additional wrapper can be observed in Figure 6.1b.

(a) Original input converted and rescaled to an 84×84 grey-scale image

(b) Pre-processed input with a 20% padding of random noise

Figure 6.1: What the agent sees: a pre-processed frame during a game of Breakout where Figure 6.1a is the preprocessed frame and Figure 6.1b depicts the additional added layer of random noise

CHAPTER 7

Results

In this chapter the findings and results of the conducted experiments are presented which were carried out in order to find answers to the posed questions in section 1.3. Depending on the question various games with different properties have to be selected.

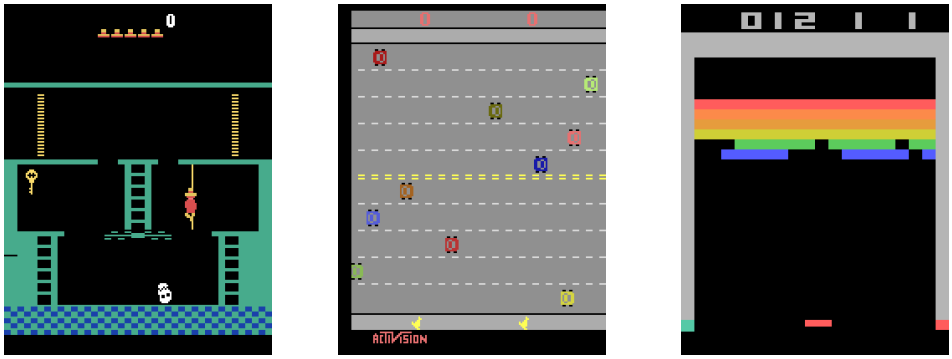
Concerning the question on how to combine extrinsic and intrinsic rewards and the possibility to observe certain patterns, the game *Montezuma Revenge* (Figure 7.1a) and *Freeway* (Figure 7.1b) were selected. The reasoning behind these two games is based on the categorization by Bellemare et al. who classified these games to be among the hardest Atari 2600 games. However, in regards to their characteristics, the two games are very dissimilar as on the one hand, *Montezuma Revenge* is one of the most sparsely rewarding games and on the other hand, *Freeway* naturally contains a lot of randomness which causes similar problems as white-noise. Hence, *Freeway* is especially challenging for intrinsically motivated agents.

For the question regarding the possibility of a performance increase in dense environments, the game *Breakout* (Figure 7.1c) was selected as it is a densely rewarding game that is commonly used, e.g. by Burda et al. (2018a) for their experiments. Concerning the final posed question which deals with the impact of stochasticity and the chance of Extrinsic Motivation (EM) to reduce those effects, *Breakout* was chosen again as its densely rewarding environment allows to see the consequences of deploying a noisy-TV more quickly and thus reduces the runtime immensely.

The gameplay of the just three mentioned games is displayed in Figure 7.1.

7.1 Evalutation

All of the experiments presented in this chapter were run on a single machine with a 4-core and 8-thread Intel Xeon E3-1230 CPU which has a maximum clock speed of 3.8 GHz. For the GPU and the RAM, solely one NVIDIA GeForce GTX 1070 card and 16



(a) Montezuma's Revenge (b) Freeway (c) Breakout

Figure 7.1: The gameplay of three selected Atari 2600 games

GB of DDR4 memory were used. Details about the employed software can be looked up in chapter 6. Experiments were executed for either $4 \times 10^6, 1 \times 10^8, 2 \times 10^8$, or 4×10^8 frames with a runtime duration of approximately 15 minutes, 6 hours, 12 hours, and 24 hours respectively, resulting in a linearly growing time complexity. Regarding the addressed frame-skipping in section 6.1, the standard *OpenAI* environments "BreakoutNoFrameskip-v4", "FreewayNoFrameskip-v4", and "MontezumaRevengeNoFrameskip-v4" were used. Therefore, one time-step in the implementation is equal to four consecutive frames where for each frame the same action is taken by the agent.

Benchmarking algorithms in Deep Reinforcement Learning (DRL) is a very challenging problem as this process includes a lot of stochasticity. In order to offer best reproducibility and fair comparability the procedure of the conducted experiments was based on methods used by Burda et al. (2018a) as well as the proposed best-practices by Francois-Lavet et al.

According to their best-practices, experiments must be carried out using multiple random seeds across many different runs. This recommendation was realized through the implementation of the gridsearch which allows for an exhaustive cross validation, thus ensuring the validity of the conducted tests. Concerning the graphical visualizations, the runs are summarized in curves using the mean of different runs where the shaded area depicts the Standard Error (SE).[1]

This is true for all displayed plots unless it is stated otherwise.

Usually, the purpose of these executed benchmarks is to test newly developed or adapted algorithms for an increase in performance. Concerning the aforementioned stochasticity of these experiments, it is not sufficient to simply compare e.g. average returns of the baseline and of the newly/adapted one as they might have a highly different standard deviation. Therefore, 2-sample paired t-tests will be utilized to show the significance of the obtained results where necessary.

[1] The SE is equal to $\sigma \div \sqrt{n}$ where σ is the standard deviation and n is the sample size

As mentioned in section 6.2, experiments regarding the optimal combination of rewards as well as the possibility of improving existing results in dense settings are started with a standard gridsearch which exhaustively tries out every possible combination of the coefficients (α, β) within 0 and 1 and a rate of change (Δ) of 0.1. In that case, only Random Features (RFs) were considered for the tuning process. After this initial step, metrics such as the *recent extrinsic episodic reward*, the *mean return*, or the *mean reward* were inspected. However, for the final evaluation mainly the recent extrinsic episodic reward was considered which stands for the sum of extrinsic rewards an agent receives per episode solely awarded by the environment's reward function. With regards to games, the recent extrinsic episodic reward is equivalent to points an agent achieves during a game.

In order to form the return, the advantages that utilize the normalized weighted rewards and the value predictions of the policy network are added together where advantages were normalized as well (see section 6.1). Furthermore, the reward is composed of the weighted extrinsic and intrinsic one where the intrinsic reward resembles the loss calculated by the forward dynamics model. Finally, the abbreviations M, SD, and SE will be utilized throughout the rest of this chapter in order to report the mean, the standard deviation as well as the standard error respectively all rounded to two decimal places. These statistical key figures are calculated using the results of the last $\sim$ 500.000 frames.

In Figure 7.2, the initial gridsearch of the game *Freeway* is depicted which acts as a general example on how the evaluation procedure was deployed. Depending on the performance of the different combinations with regards to the just mentioned metrics, the search space is further restricted.

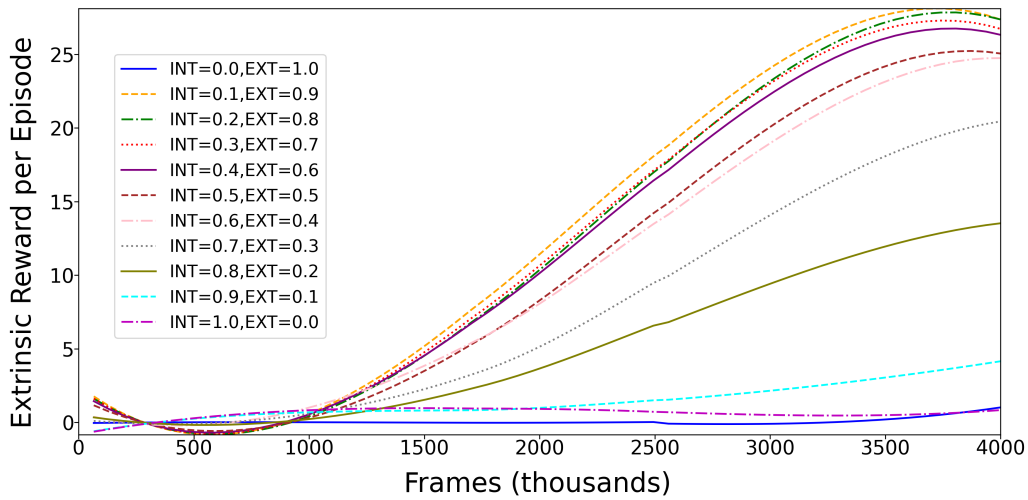


Figure 7.2: A visualization of the first iteration of a gridsearch executed on the game Freeway where the mean extrinsic reward per episode is evaluated

In regards to the performance comparison of the games *Montezuma's Revenge* and

Freeway, the result by Burda et al. (2018a, see Table 2) act as a baseline. Similarly to the evaluation procedure which is employed in this thesis, they compared the last recent mean episodic reward (±SE) after 100 million frames across three different seeds. Furthermore, they executed two different runs: one combined approach with initial weights of 1.0 for extrinsic rewards and 0.01 for intrinsic ones, and another with a purely extrinsically motivated agent. Unfortunately only the results for *Freeway* can be compared as their results on *Montezuma's Revenge* do not align with the rest of their work since they reported a significantly higher score[2] for the purely extrinsically motivated agent than for the purely intrinsically one.

The plots in the following few sections represent only the most important findings and hence additional plots concerning the *episodic return*, the *average return* and the *standard deviation* as well as the *average reward* will be attached to the Appendix.

7.2 Montezuma's Revenge

In order to find good weights which maximize an agent's episodic reward, a gridsearch is executed for 4×10^6 frames. Here, the challenge lies in analyzing the performance metrics as the agent is normally unable to obtain any extrinsic episodic rewards in *Montezuma's Revenge* given such little time. Hence, such a comparison as depicted in Figure 7.2 is not applicable and other performance metrics had to be considered. With regards to the mean return, weighting the extrinsic reward with 0.6 and the intrinsic one with 0.4 yielded the highest value ($M \approx 0.88, SD \approx 0.07$) in the last episode which is directly followed by the the weights 0.7 and 0.3 ($M \approx 0.78, SD \approx 0.05$).

Based on these received results, the search space had been restricted, thus varying the intrinsic weight solely between 0.5 and 0.2 and the extrinsic weight between 0.5 and 0.8. After the second iteration, the optimal combination was yet again selected by analyzing the mean return. This time, the optimal combination was 0.56 for the extrinsic reward and 0.44 for the intrinsic one. This procedure was repeated once again which resulted in an optimal weighting of $\alpha = 0.43$ and $\beta = 0.57$.

According to the procedure of the baseline experiment which was conducted by Burda et al. (2018a), the performance of this obtained combination was evaluated by executing a 3-fold cross validation with three different seeds utilizing both RFs and Inverse Dynamics Features (IDFs). Additionally, purely curiosity based agents as well as a solely extrinsically rewarded agent have been examined. The results of this can be observed in Figure 7.3a.

Surprisingly the performance of the purely extrinsically rewarded agent was by far the best ($M \approx 266.67, SD \approx 188.56, SE \approx 34.43$) followed by the mixed motivated RF-curious agent ($M \approx 133.33, SD \approx 188.56, SE \approx 34.43$). Concerning the IDF-curious agents as

[2] The difference between the purely intrinsically (~ 250) and extrinsically (~ 1800) motivated agent is about ~ 1550. After asking the authors Burda et al. (2018a) for clarification, no answer has been received. Therefore, it is assumed that these differences are based on different extrinsic reward signals or an error by a power of ten.

wells as the purely intrinsically motivated RF-curious agent, they almost continuously achieved a recent episodic reward of 0 ($M \approx 0.0, SD \approx 0.0, SE \approx 0.0$).

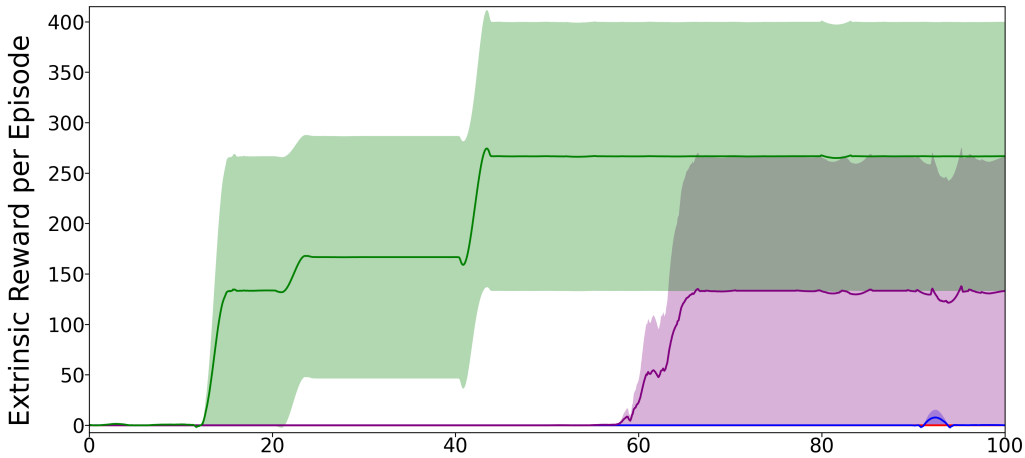
However, Burda et al. (2018a) showed that purely intrinsically motivated agents can perform much better, if given enough time. A the training duration of 100 million frames seems to be insufficient as e.g. the pure RF-curious agent did not achieve a single extrinsic reward in the final episode whereas when trained for 400 million frames it is able to achieve an episodic reward of ~ 250 (Burda et al., 2018a, compare Figure 2). Usually, the authors evaluated the performance of those agents for at least 400 million frames which makes their proposed baseline of training mixed motivated agents for only 100 million frames questionable.

Hence, the run time was increased to 400 million frames which allowed for a fairer overall comparison between different combinations of coefficients and dynamics. The results of this experiment can be observed in Figure 7.3b. Here, it is important to mention that the mixed motivated agents as well as the purely extrinsically one were solely evaluated with 1-fold cross validation due the massive amount of needed computational power. Observing the results, one can clearly notice that the extrinsically motivated agent as well as the mixed RF-curious agent performed equally well ($M \approx 400.0, SD \approx 0.0, SE \approx 0.0$) with the extrinsically motivated agent converging around 140 million frames earlier. Although the purely motivated RF-curious agent was able to drastically increase its performance ($M \approx 249.33, SD \approx 180.95, SE \approx 33.04$) when compared to the previous evaluation, it still performed worse than the other two. However, it was the only one which was ever able to go beyond an extrinsic episodic reward of 400 as it achieved a maximum reward of 425. Lastly, the IDF-curious agents were almost always unable to obtain any extrinsic rewards.

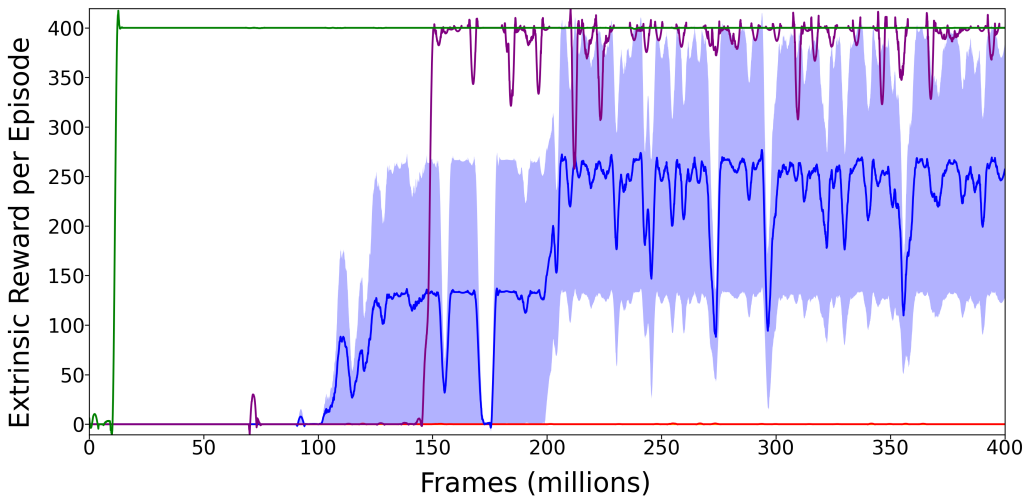
Although it seems that the extrinsically motivated agent as well as the optimally combined RF-curious agent are identical performance-wise, there is a big difference between them. In fact, while they both achieved they same extrinsic episodic reward, the RF-curious agent was able to discover far more rooms than the extrinsic one. In total the mixed motivated RF-curious agent visited five different rooms in *Montezuma's Revenge* while the purely extrinsically one had only discovered two. This phenomenon is depicted in Figure 7.4 where the number of rooms visited by the purely intrinsically motivated agents and the mixed IDF-curious one are plotted as well.

7.3 Freeway

In regards to the evaluation of the game *Freeway*, the same gridsearch method was applied again with a runtime of 4×10^6 frames. However, due to the fact that this game credits rewards far more densely than *Montezuma's Revenge*, it is now possible to compare the performance of the different coefficients by utilizing the recent extrinsic episodic reward as a metric. The initial run for *Freeway* where coefficients are varied between 0 and 1 is depicted in the example Figure 7.2. When evaluating the performance of the first iteration, it is clearly noticeable that the coefficient tuples of $\alpha = 0.1$,$\beta = 0.9$



(a)



(b)

Random Features	Extrinsic Only	Inverse Dynamic Features
INT=1.0, EXT=0.0	INT=0.0, EXT=1.0	INT=1.0, EXT=0.0
INT=0.43, EXT=0.57		INT=0.43, EXT=0.57

Figure 7.3: The performance of an optimally weighted agent in the game of Montezuma's Revenge after 100 million frames and 400 million frames with coefficients $\alpha = 0.43$ and $\beta = 0.57$

$(M \approx 28.90, SD \approx 0.14)$ and $\alpha = 0.2$,$\beta = 0.8$ $(M \approx 28.37, SD \approx 0.26)$ achieved the highest extrinsic episodic reward.

Hence, the search space was restricted to exhaustively evaluate intrinsic weights between 0.3 and 0.0 and extrinsic weights between 0.7 and 1.0 respectively. Concerning the second

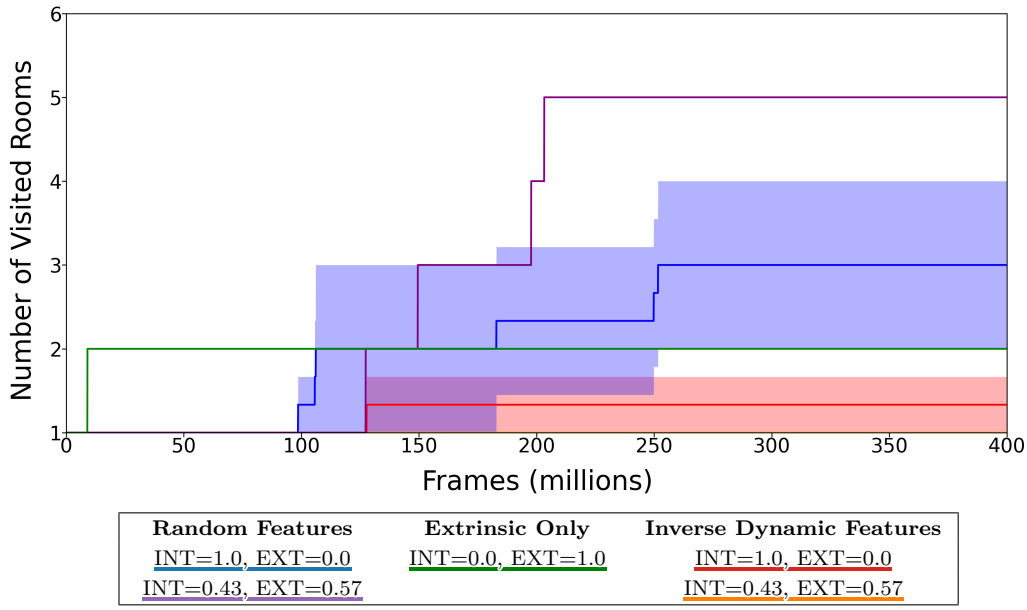


Figure 7.4: A visualization of the number of rooms visited by an agent over the course of 400 million frames

iteration, the coefficients were yet again selected by observing the mean extrinsic episodic reward where the most promising results were obtained with a combination of $\alpha = 0.15$ and $\beta = 0.85$. Since the optimal combination was once again achieved with a weighting of intrinsic rewards between 0.1 and 0.2, the final iteration was executed varying the coefficients solely within these seemingly optimal bounds.

After the final iteration and the obtained optimal coefficients of $\alpha = 0.12$ and $\beta = 0.88$, an exhaustive evaluation of purely motivated and mixed motivated agents was conducted. The results of this evaluation can be observed in Figure 7.5. Here, the purely intrinsically motivated RF-curious agent ($M \approx 0.06, SD \approx 0.06, SE \approx 0.01$) as well as the IDF-curious agent ($M \approx 0.18, SD \approx 0.12, SE \approx 0.02$) performed the worst. Concerning the three other agents, they performed almost equally well and they all achieved a maximum extrinsic episodic reward of 34. Since Burda et al. (2018a) used a purely extrinsically motivated agent as a baseline, the performances of the mixed RF-curious agent ($M \approx 33.68, SD \approx 0.2, SE \approx 0.04$) and the IDF-curious agent ($M \approx 33.46, SD \approx 0.16, SE \approx 0.03$) are compared to the extrinsic one ($M \approx 33.54, SD \approx 0.43, SE \approx 0.08$).

Starting with the comparison between the RF-curious agent and the extrinsic one, the results ($t(58) = -1.6, p = 0.11$) of the t-test for means, with a significance level of 0.05, suggest that the null hypothesis cannot be rejected and thus the means do not vary significantly. Regarding the means of the IDF-curious agent and the extrinsic one, the t-test with the same significance level yet again recommends to not reject the null hypothesis ($t(58) = 0.94, p = 0.35$).

Even though combining extrinsic and intrinsic motivation does not yield a better performance than only using extrinsic one, it seems that using mixed curious agents, at least for the game *Freeway*, allow for a more stable performance and a somewhat faster convergence as the SE is barely visible as opposed to the extrinsic one.

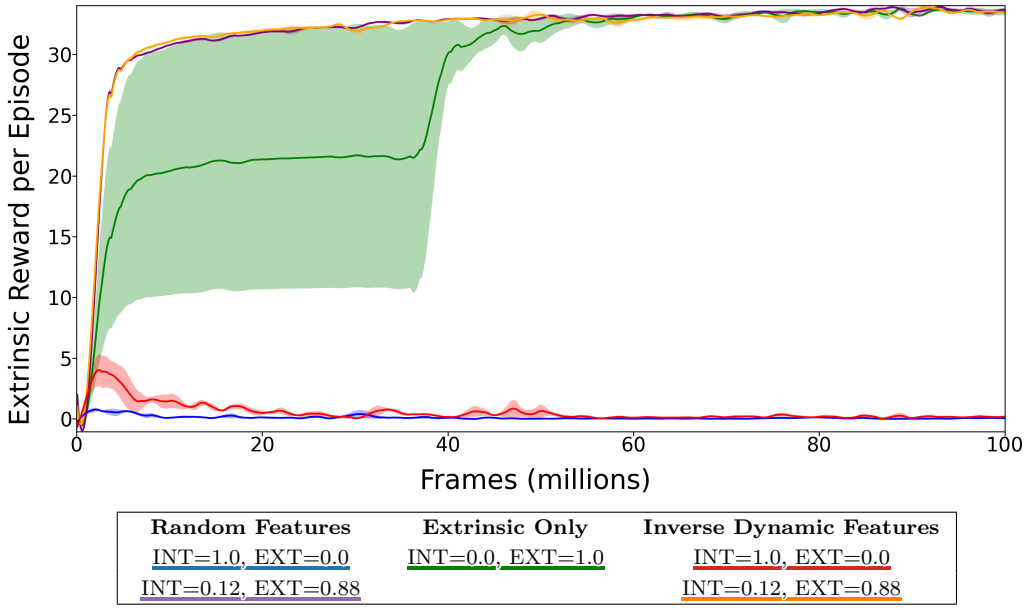


Figure 7.5: The performance of an optimally weighted agent in the game of Freeway after 100 million frames with coefficients $\alpha = 0.12$ and $\beta = 0.88$

7.4 Breakout

Concerning the performance evaluation of mixed motivated agents in densely rewarding environments and their possibility to improve state-of-the-art approaches, a gridsearch was executed in order to ensure a fair comparison by solely validating the performance of optimal coefficients. However this time, the gridsearch was only performed once while still varying the coefficients between 0 and 1 for a total of 4×10^6 frames. As it is expected within environments with dense rewards, the weighting tended to be more on the extrinsic side with an optimal combination of $\alpha = 0.1$ and $\beta = 0.9$ ($M \approx 15.3, SD \approx 2.12$).

With the obtained optimal weights for the mixed agent, the final evaluation was started considering only RF-curious agents and pure extrinsic ones with an overall training duration of 200 million frames. Furthermore, the results of this experiment are depicted in Figure 7.6. The reason behind the omission of IDF-curious agents is based on the findings of Burda et al. (2018a) who state that RFs usually deliver better results (see section 5.1). Evaluating the performance after 200 million frames shows that the pure intrinsic agent had the worst overall performance and SE ($M \approx 194.95, SD \approx 51.53, SE \approx 9.43$). In

regards to the mixed agent ($M \approx 393.53, SD \approx 32.53, SE \approx 5.93$) as well as the pure extrinsic one ($M \approx 385.27, SD \approx 23.78, SE \approx 4.34$), they almost had an identical performance with the mixed agent seemingly performing slightly better. However, after analyzing the mean with a two-sided t-test ($t(58) = -1.10, p = 0.27$) with a significance level of 0.05 the null hypothesis cannot be rejected and hence the speculation that the mixed agent performs better turned out to be false. Interestingly, there were big discrepancies between the three evaluated agents in regards to the best overall extrinsic episodic reward. While the purely intrinsically motivated agent only achieved a maximum reward of 382, the mixed motivated agent obtained a score of 743 and the extrinsically motivated agent a score of 462.

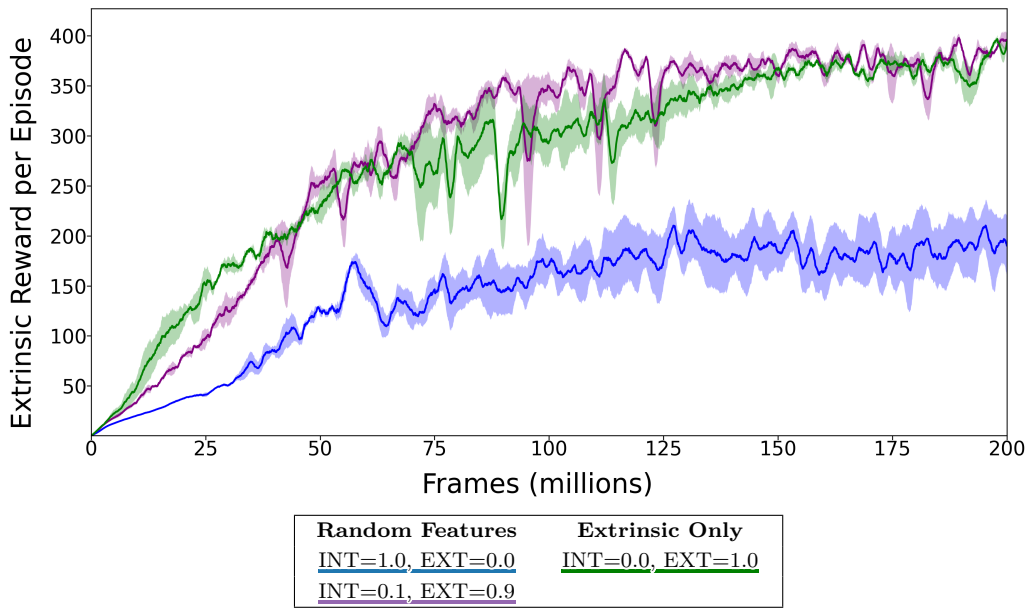


Figure 7.6: The performance of a mixed RF-curious agent in the game of Breakout after 200 million frames with coefficients $\alpha = 0.1$ and $\beta = 0.9$

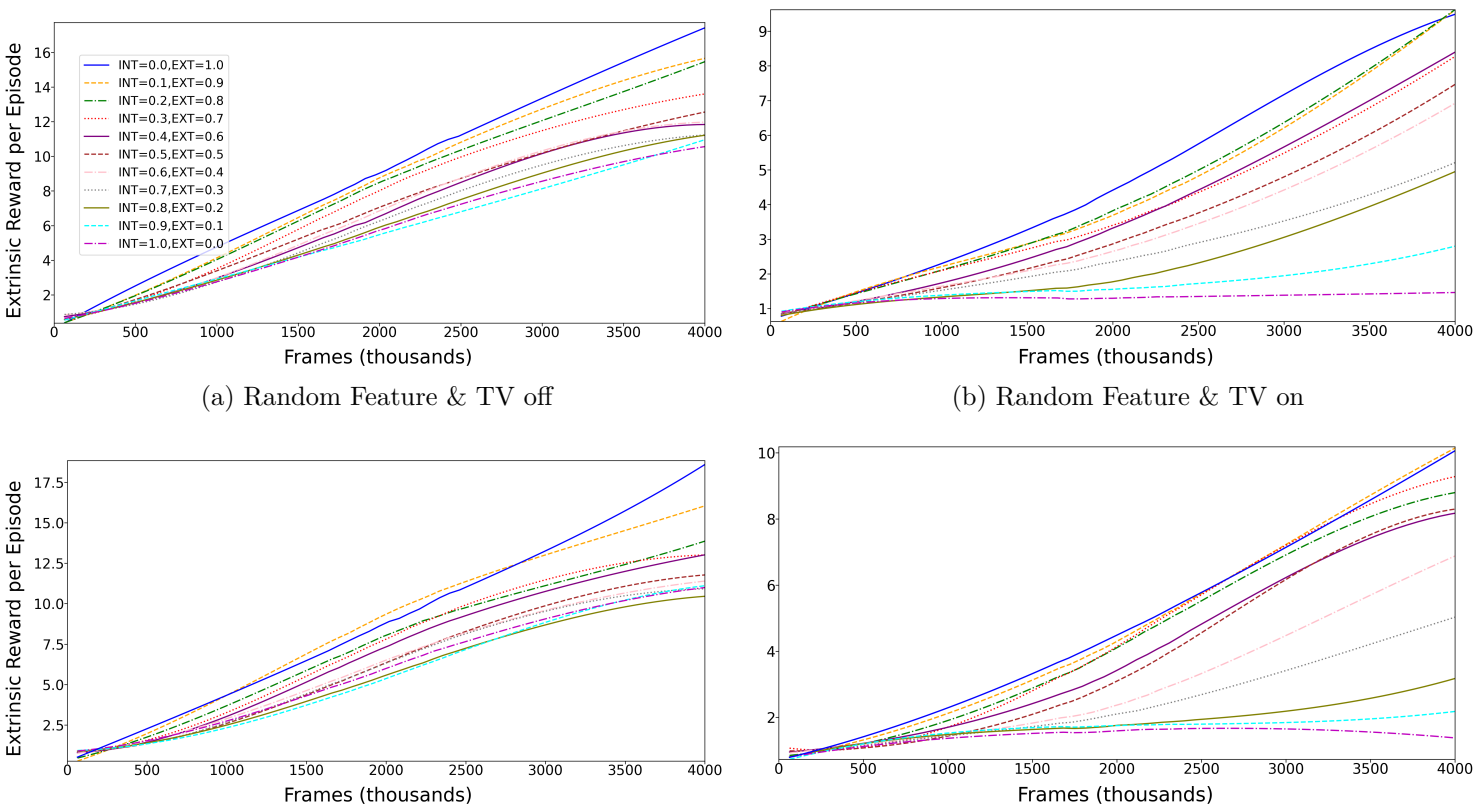
In order to pursue the final posed question regarding the impact of white-noise using different forward dynamics models, the agents were evaluated twice: once with a TV turned on and once with a TV turned off. The results of this experiment are depicted in Figure 7.7 where the curves represent the average extrinsic reward per episode. The SE is omitted to improve readability.

On a first glance, it is immediately apparent that agents which were exposed to white-noise performed worse than non-exposed ones and additionally, the range between the best and the worst performing agent is much broader as well. Beginning with non-exposed agents, the purely extrinsically motivated agent performed the best ($M \approx 16.09, SD \approx 1.68, SE \approx 0.43$) whereas the pure RF-curious one performed the worst ($M \approx 10.35, SD \approx 0.81, SE \approx 0.21$). Likewise, exposed agents were almost ranked identically where the

extrinsic agent again achieved the best final result ($M \approx 9.2, SD \approx 0.73, SE \approx 0.19$) and the intrinsic one the worst ($M \approx 1.46, SD \approx 0.25, SE \approx 0.06$).

Regarding IDF-curious agents, a similar behaviour can be observed. However this time, concerning the non-exposed agents, a weighting of $\alpha = 0.8$ and $\beta = 0.2$ performed the worst ($M \approx 10.4, SD \approx 1.07, SE \approx 0.28$). For the sake of completeness, the top-ranked exposed agent was also replaced by a weighted agent of $\alpha = 0.1$ and $\beta = 0.9$ ($M \approx 9.69, SD \approx 1.07, SE \approx 0.28$). Concerning the discrepancies between exposed and non-exposed agents, it very surprising that the purely extrinsically motivated agent experienced such a big difference in performance since it should not be affected by the white-noise.

Finally, multiple two-sided t-test were carried out to see whether or not the differences in means between exposed and corresponding non-exposed RF-curious agents are significant. With a significance level of 0.05, all the conducted tests suggested to reject the null hypothesis which seems obvious due to the large deviation.



(a) Random Feature & TV off

(b) Random Feature & TV on

(c) Inverse Dynamics Feature & TV off

(d) Inverse Dynamics Feature & TV on

Figure 7.7: A performance comparison of agents with different weightings of motivation regarding the noisy-TV problem

CHAPTER 8

Discussion, Further Work and Conclusion

Before the discussion is started the core questions pursued by this thesis will be recalled:

(a) Does the combination of extrinsic and intrinsic rewards obtain better results?

(b) Are there any patterns concerning optimally combined rewards?

(c) Is it possible to further improve the performance of agents in densely rewarding environments by utilizing Extrinsic Motivation and Intrinsic Motivation simultaneously?

(d) Can local stochasticity, also known as the white-noise problem, be tackled by employing Extrinsic Motivation?

Concerning question (a), an evaluation process was conducted which exhaustively tried out various possible weightings for intrinsic and extrinsic rewards on the game *Montezuma's Revenge* as well as *Freeway*. The results obtained through these experiments both suggest that it is not possible to improve existing results in the two selected games. However, it seems that the combination of intrinsic rewards yield other benefits which are not prominent on first glance.

For instance, in the game of *Montezuma's Revenge* the agent which utilized both kinds of rewards was able to explore five different rooms in total, far more than the others. This phenomenon was also discovered by Burda et al. who state that they even achieved a run where their agent was able to visit as many as ten rooms. One would assume that this advancement would increase the agent's performance but this is not the case. What is more, it looks as if the agent is still more interested in pursuing its internal

drive instead of collecting extrinsic rewards even tough it would be able to do so. A further reason could be that the agent gets detached from the intrinsically rewarding area (see Figure 5.3) which, according to Ecoffet et al. (2019), is a fundamental problem of Intrinsic Motivation (IM).

Another example would be the much quicker and less varying convergence of the obtained episodic extrinsic rewards which was witnessed during the evaluation of *Freeway*. Here, it seems like the mixed rewarded agent is able to find a good compromise between being completely distracted by the naturally occurring stochasticity of the game and dodging the initial hurdles with which an extrinsic rewarded agent has to deal with. Therefore, the results suggest that in games where stochasticity is part of the gameplay, a different weighting of the reward signals seems to be the preferred choice. As a side note, *Freeway* is a time based game where each round ends after approximately two minutes. Since both the mixed agents as well as the extrinsic one achieved a top score of 34, a further improvement could not be expected anyway. However, games that include stochasticity combined with different weightings of rewards seem to be a promising direction for further research.

Regarding question (b), two patterns were detected concerning the optimal weighting. On the one hand, through out all the various conducted experiments, the extrinsic reward signal tended to be weighted more heavily and on the other hand, it seems like the heaviness is correlated with the sparseness of the extrinsic rewards credited by the environment. This makes a lot of sense as agents were primarily evaluated considering extrinsic based performance metrics and hence, this must somehow aline with the observed patterns.

In order to answer question (c), the densely rewarding game *Breakout* was selected. After a profound evaluation of the obtained results, apparent improvements caused by the combined reward signal were disproved with the employment of a two-sided t-test. Hence, it is concluded that a performance increase via a combination of intrinsic and extrinsic rewards is not possible in settings were rewards are credited densely. Although different weightings of rewards did not manage to beat existing extrinsically rewarding approaches, they perform significantly better than purely intrinsically rewarding methods.

For question (d), an identical approach to the one proposed by Pathak et al. (2017) and Burda et al. (2018a) was reproduced as the observation which the agent receives was modified by adding a padding of white-noise around each frame. Evaluating the obtained results of the four conducted experiments clearly showed that intrinsically motivated agents get distracted by the added noisy-TV and hence their performance is drastically decreased which reinforce the findings by the just mentioned authors. Furthermore, agents with different weightings were employed during these experiments in order to analyze their ability to tackle the caused distraction. In regards to different forward dynamics models, both Random Features (RFs) as well as Inverse Dynamics Features (IDFs) seem to be affected similarly as in both cases they took a drastic hit performance-wise. Additionally, the range between the best and the worst performing agent increased by half.

Surprisingly, the purely extrinsically motivated agent was effected by the white-noise as well although in theory it should not cause any difference. However, there is a simple explanation for the resulting discrepancies as one significant side effect was disregarded. By adding a padding of white-noise around each input frame, the dimensionality of the agent's observation increases and thus, the environment gets more complex. One solution to this would be to e.g. add a padding of fixed white-noise to the non-exposed agent's observation as this ensures equal conditions while at the same time it should not cause any distraction. Although this important side effect was neglected, it is still possible to draw a conclusion. When evaluating the differences between the exposed and non-exposed agents, it seems that the impact of white-noise is heavily reduced by an extrinsic weighting of 0.5 and above. Hence, the data suggests that as soon as the extrinsic reward signal outweighs the intrinsic one the impact of the noisy-TV is not significant anymore which makes sense as the irritation is caused by IM in the first place. Therefore, one can conclude that Extrinsic Motivation (EM) is an effective option to tackle local stochasticity.

Regarding further research, one direction has already been pointed out for question (a). Additionally to that, the experiments for question (d) should be carried out again but this time taking into account the occurring side effects e.g. by means of the just proposed solution. This would ensure the validity of the drawn conclusion and might yield additional insights.

During research of this thesis, one highly limiting factor was the lack of computational power. Although all of the findings are somehow tangible, a large-scale study utilizing better hardware similar to the one by Burda et al. (2018a) would be recommended to guarantee more valid results. In addition to that, greater focus should be laid on IDFs since according to Burda et al. (2018a) they allow for better generalization across different environments. Furthermore, the results showed that the promising performance of the selected prediction-error based approach carried out by a purely intrinsically motivated agent can certainly be undermined by means of a combined reward signal. Therefore, extrinsic rewards are still needed in order for an agent to achieve superhuman performance.

A final very interesting and promising further research direction would be to combine the algorithm introduced in "Go-Explore: A New Approach for Hard-Exploration Problems" (Ecoffet et al., 2019) with IM since so far the exploration process has solely been based on a random agent.

In conclusion, while all of the conducted experiments concerning the different weighting of intrinsic and extrinsic reward signals did not improve existing, mainly extrinsically motivated approaches, they certainly perform better than purely intrinsically motivated ones. Additionally, further beneficial properties had been observed such as a faster and a more stable convergence which, if employed in the right context, can establish an advantage.

List of Figures

2.1 The Reinforcement Learning framework formalized as a Markov Decision Process . . . 7

3.1 From a biological neuron to a mathematical model: an artificial Neural Network in its simplest form . . . 18
3.2 Various activation functions and their relationships between the domain and the codomain . . . 20
3.3 The XOR problem . . . 22
3.4 A non-exhaustive taxonomy of Deep Reinforcement Learning algorithms . 24

4.1 Adapting the classic Reinforcement Learning framework to incorporate Intrinsic Motivation . . . 36

5.1 The Intrinsic Curiosity Module introduced in "Curiosity-Driven Exploration by Self-Supervised Prediction" . . . 41
5.2 A visualisation of the algorithm introduced in "Self-Supervised Exploration via Disagreement" . . . 44
5.3 An illustration of intrinsically motivated exploration methods where an agent is unable to rediscover promising areas . . . 45
5.4 The simple, yet astonishingly good performing algorithm "Go-Explore" . . 47

6.1 What the agent sees: a pre-processed frame during a game of Breakout . 53

7.1 The gameplay of three selected Atari 2600 games . . . 56
7.2 A visualization of the first iteration of an executed gridsearch for the game Freeway . . . 57
7.3 The performance of an optimally weighted agent in the game of Montezuma's Revenge after 100 million frames and 400 million frames . . . 60
7.4 A visualization of the number of rooms visited by an agent over the course of 400 million frames . . . 61
7.5 The performance of an optimally weighted agent in the game of Freeway after 100 million frames . . . 62
7.6 The performance of a mixed RF-curious agent in the game of Breakout after 200 million frames . . . 63

7.7 A performance comparison of agents with different weightings of motivation regarding the noisy-TV problem 64

List of Tables

4.1 A taxonomy on the different types of learning 31
4.2 Classification of Intrinsic Motivations in Reinforcement Learning 35

6.1 Proximal Policy Optimization hyperparameters for the conducted experiments 51

List of Algorithms

2.1 SARSA for estimating $Q \approx q_*$ 12
2.2 Q-learning for estimating $\pi \approx \pi_*$ 13

3.1 Proximal Policy Optimization, Actor-Critic Style 27

Acronyms

CNN Convolutional Neural Network. 40, 50

DL Deep Learning. 17, 22–24

DP Dynamic Programming. 5, 10, 11

DRL Deep Reinforcement Learning. 17, 23, 24, 26, 31, 56, 69

EM Extrinsic Motivation. 3, 29, 30, 43, 53, 55, 65, 67

GAE Generalized Advantage Estimation. 51

ICM Intrinsic Curiosity Module. 41, 69

IDF Inverse Dynamics Feature. 41, 42, 49, 51, 58, 59, 61, 62, 64, 66, 67

IM Intrinsic Motivation. 1–3, 29–37, 39, 40, 42, 45, 65–67, 69

M Mean. 57

MC Monte-Carlo. 5, 10, 11

MDP Markov Decision Process. 5–10, 35, 69

ML Machine Learning. 5, 6, 31, 49

MLP Multi-Layer Perceptron. 19–22

NN Neural Network. 17–22, 26, 30, 40, 43, 44, 51, 69

PG Policy Gradient. 5, 14, 24

PPO Proximal Policy Optimization. 17, 24–27, 49–51, 71, 73

ReLU Rectified Linear Unit. 20, 22, 50, 51

RF Random Feature. 40–43, 49, 51, 57–59, 61–64, 66, 69

RL Reinforcement Learning. 1–3, 5–11, 17, 23, 24, 29–36, 49, 69

SD Standard Deviation. 57

SE Standard Error. 56–58, 62, 63

TD Temporal-Difference. 5, 10–12

TRPO Trust Region Policy Optimization. 25, 26

VAE Variational Auto-Encoder. 40, 42

VF Value Function. 51

Bibliography

C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, Zug, Sept. 2018. ISBN 978-3-319-94462-3. doi: 10.1007/978-3-319-94463-0.

A. Aubret, L. Matignon, and S. Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv:1908.06976*, Nov. 2019.

M. Awad and R. Khanna. *Efficient Learning Machines*. Apress, Berkeley, California, Apr. 2015. ISBN 978-1-4302-5989-3. doi: 10.1007/978-1-4302-5990-9.

M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying Count-Based Exploration and Intrinsic Motivation. *arXiv:1606.01868*, Nov. 2016.

R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, Nov. 1954. ISSN 0002-9904. doi: 10.1090/S0002-9904-1954-09848-8.

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *arXiv:1206.5538*, Apr. 2014.

W. Böhmer, J. T. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer. Autonomous Learning of State Representations for Control: An Emerging Field Aims to Autonomously Learn State Representations for Reinforcement Learning Agents from Their Real-World Sensor Observations. *KI - Künstliche Intelligenz*, 29(4):353–362, Nov. 2015. ISSN 1610-1987. doi: 10.1007/s13218-015-0356-1.

Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-Scale Study of Curiosity-Driven Learning. *arXiv:1808.04355*, Aug. 2018a.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by Random Network Distillation. *arXiv:1810.12894*, Oct. 2018b.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-Explore: A New Approach for Hard-Exploration Problems. *arXiv:1901.10995*, May 2019.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return then explore. *arXiv:2004.12919*, May 2020.

V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, Mar. 2018. ISSN 1935-8237. doi: 10.1561/2200000071.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, Nov. 2016. ISBN 978-0-262-03561-3. doi: 10.5555/3086952.

K. Gregor, D. J. Rezende, and D. Wierstra. Variational Intrinsic Control. *arXiv:1611.07507*, Nov. 2016.

S. H. Huang, M. Zambelli, J. Kay, M. F. Martins, Y. Tassa, P. M. Pilarski, and R. Hadsell. Learning Gentle Object Manipulation with Curiosity-Driven Deep Reinforcement Learning. *arXiv:1903.08542*, Mar. 2019.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996. ISSN 1076-9757. doi: 10.1613/jair.301.

H. Kim, J. Kim, Y. Jeong, S. Levine, and H. O. Song. EMI: Exploration with Mutual Information. *arxiv:1810.01176*, June 2019a.

Y. Kim, W. Nam, H. Kim, J.-H. Kim, and G. Kim. Curiosity-Bottleneck: Exploration By Distilling Task-Specific Novelty. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3379–3388, Long Beach, Claifornia, May 2019b. doi: missing.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971*, July 2019.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec. 1943. ISSN 1522-9602. doi: 10.1007/BF02478259.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arxiv:1312.5602*, Dec. 2013.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2778–2787, Sidney, New South Wales, Aug. 2017. doi: 10.5555/3305890.

D. Pathak, D. Gandhi, and A. Gupta. Self-Supervised Exploration via Disagreement. *arxiv:1906.04161*, June 2019.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, Nov. 1958. ISSN 1939-1471. doi: 10.1037/h0042519.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986. ISSN 0028-0836. doi: 10.1038/323533a0.

R. M. Ryan and E. L. Deci. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25(1):54–67, Jan. 2000. ISSN 0361-476X. doi: 10.1006/ceps.1999.1020.

J. Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation. *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, Sept. 2010. ISSN 1943-0612. doi: 10.1109/TAMD.2010.2056368.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1889–1897, June 2015. doi: 10.5555/3045118.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*, Aug. 2017.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arxiv:1712.01815*, Dec. 2017.

S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, June 2010. ISSN 1943-0604, 1943-0612. doi: 10.1109/TAMD.2010.2051031.

B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models. *arxiv:1507.00814*, Nov. 2015.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition, May 2018. ISBN 978-0-262-03924-6.

C. Szepesvári. Algorithms for Reinforcement Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, Jan. 2010. doi: 10.2200/S00268ED1V01Y201005AIM009.

E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct. 2012. doi: 10.1109/IROS.2012.6386109.

A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. FeUdal Networks for Hierarchical Reinforcement Learning. arxiv:1703.01161, Mar. 2017.

C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698.

R. W. White. Motivation reconsidered: The concept of competence. *Psychological Review*, 66(5):297–333, Sept. 1959. ISSN 1939-1471. doi: 10.1037/h0040934.

Non-Scientific Bibliography

D. Silver. Lecture 3: Planning by Dynamic Programming. Accessed: 08/07/2020, a. URL https://www.youtube.com/watch?v=Nd1-UUMVfz4&t=2985s.

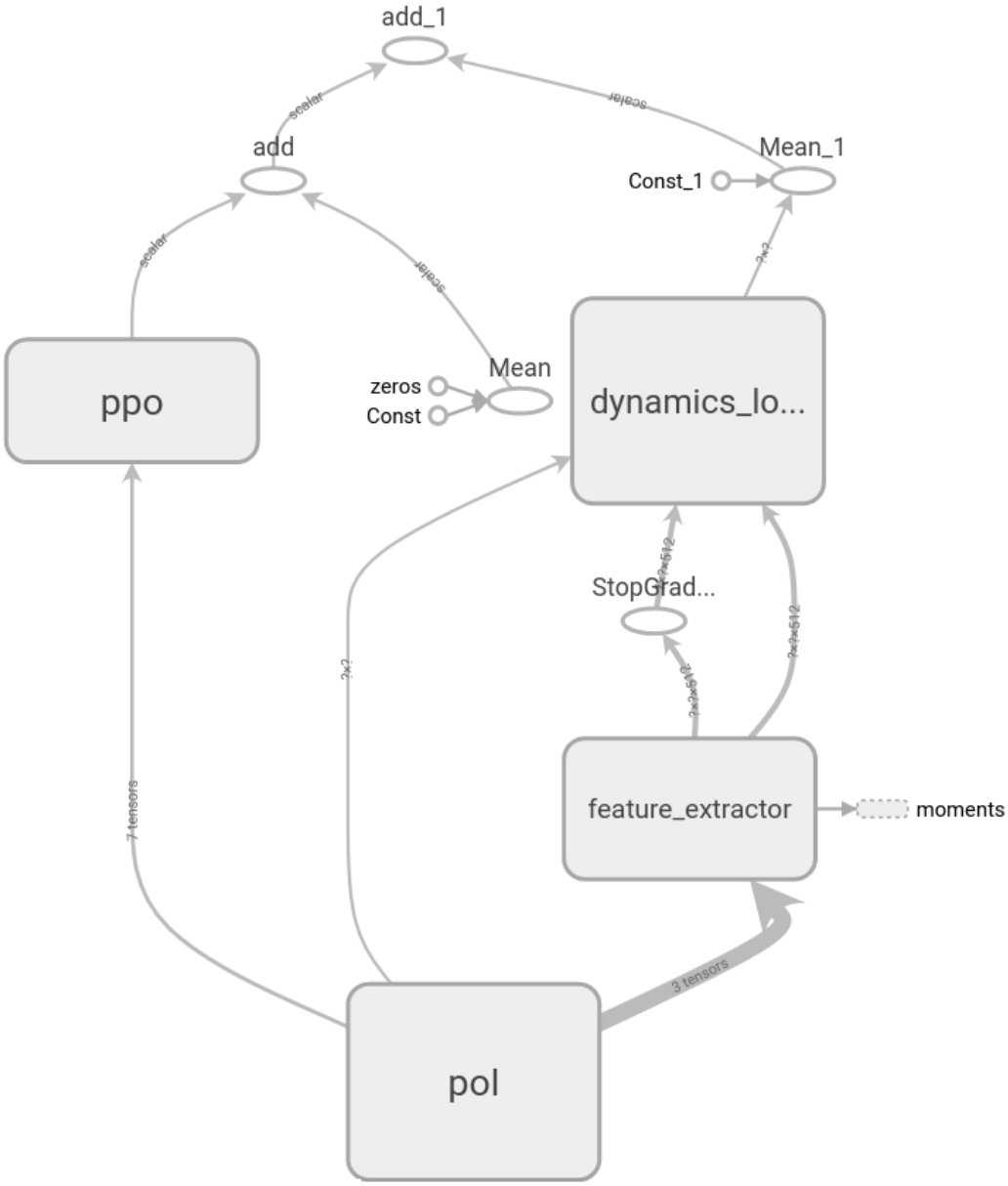
D. Silver. Lecture 2: Markov Decision Processes. Accessed: 07/07/2020, b. URL https://www.youtube.com/watch?v=lfHX2hHRMVQ&t=2627s.

D. Silver. Lecture 1: Introduction to Reinforcement Learning. Accessed: 07/07/2020, c. URL https://www.youtube.com/watch?v=2pWv7GOvuf0.

D. Silver. Lecture 7: Policy Gradient. Accessed: 25/07/2020, d. URL https://www.youtube.com/watch?v=KHZVXao4qXs&t=610s.

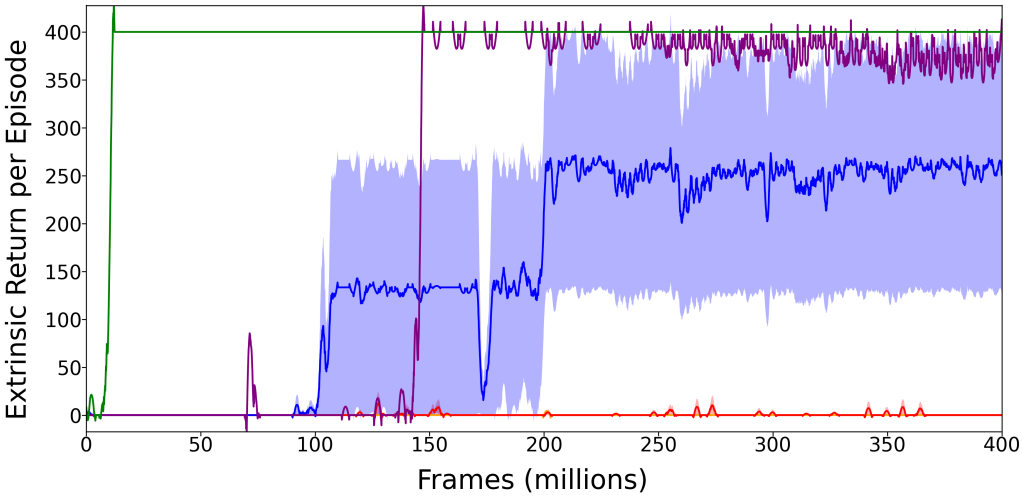
Appendix

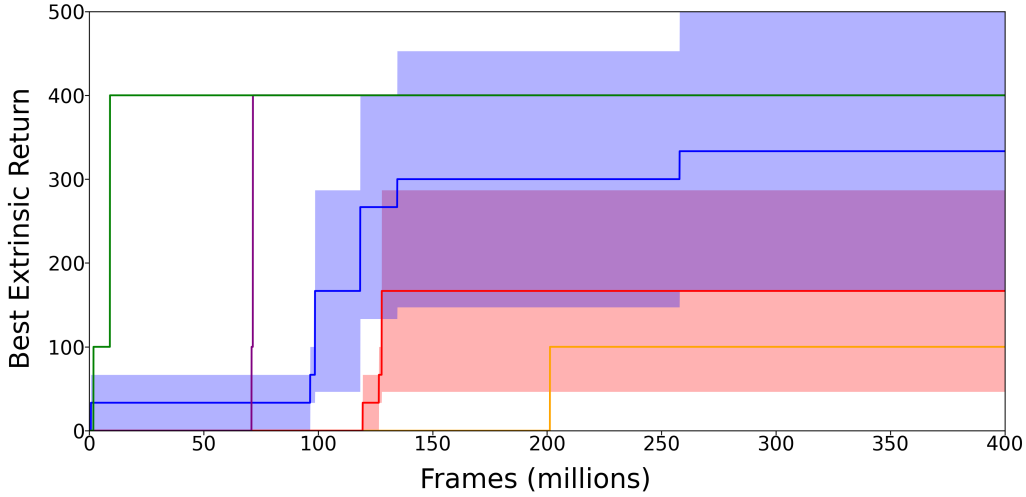
Main Graph of the TensorFlow Session

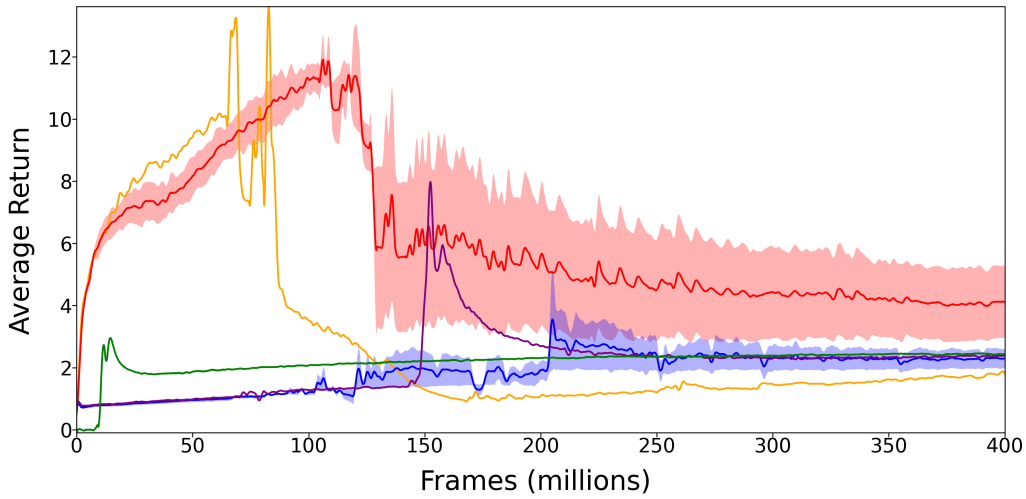


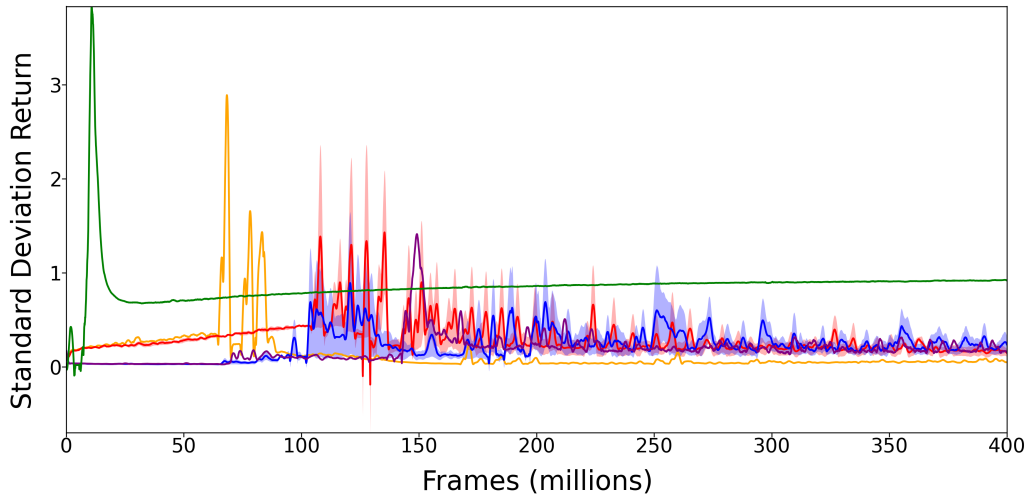
Montezuma's Revenge

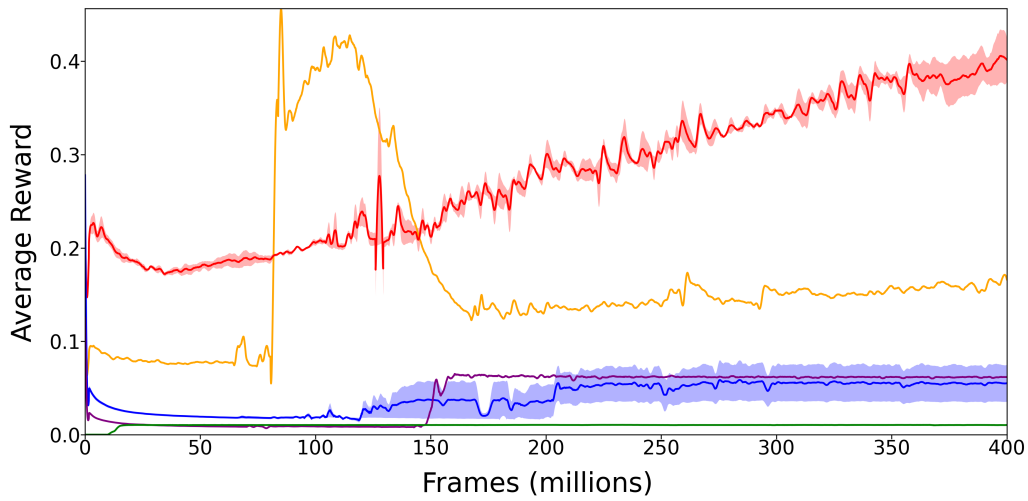
Random Features	Extrinsic Only	Inverse Dynamic Features
INT=1.0, EXT=0.0	INT=0.0, EXT=1.0	INT=1.0, EXT=0.0
INT=0.43, EXT=0.57		INT=0.43, EXT=0.57




Best Extrinsic Return
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
Frames (millions)

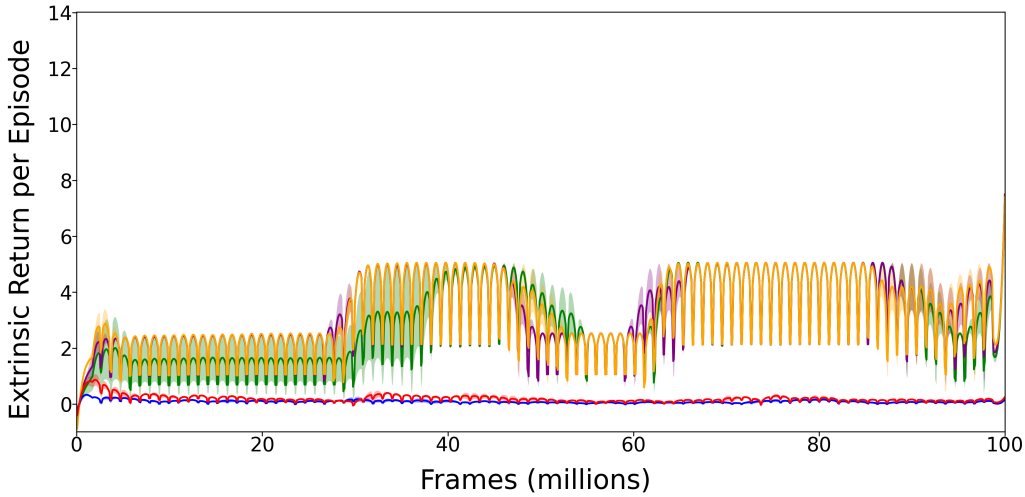

Average Return
12
10
8
6
4
2
0
0
50
100
150
200
250
300
350
400
Frames (millions)

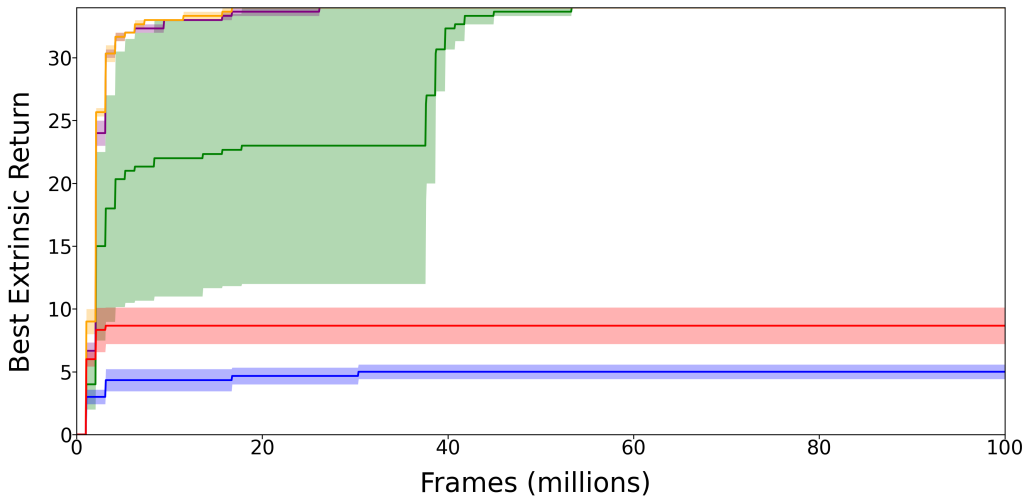

Standard Deviation Return
3
2
1
0
0
50
100
150
200
250
300
350
400
Frames (millions)

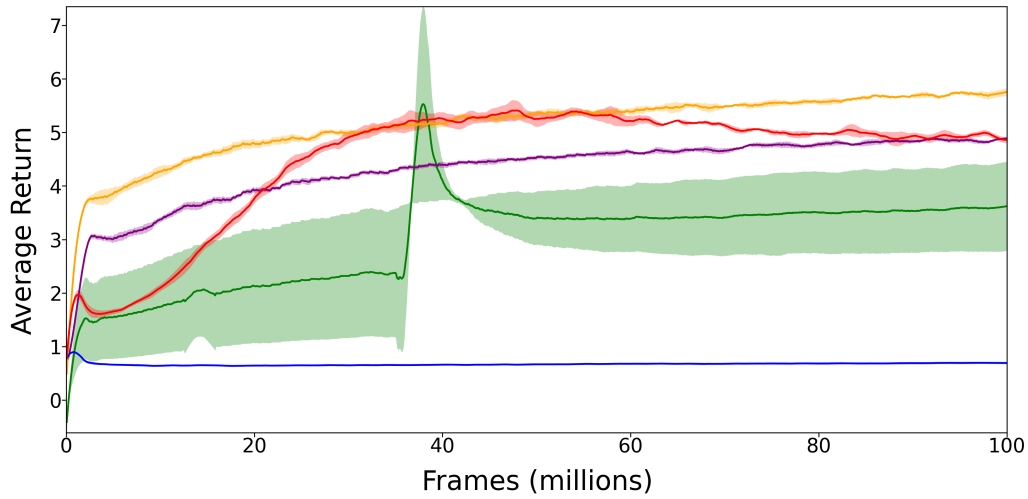

Average Reward
0.4
0.3
0.2
0.1
0.0
0
50
100
150
200
250
300
350
400
Frames (millions)

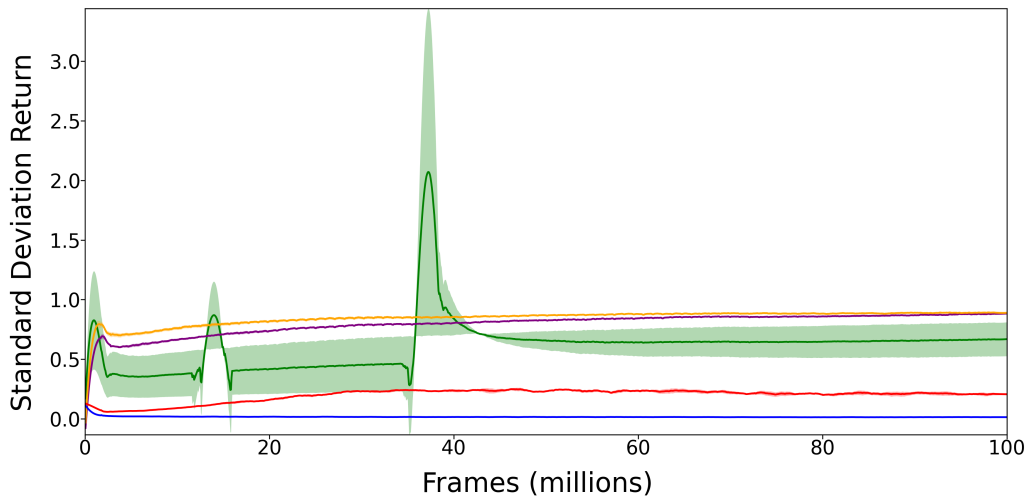
Freeway

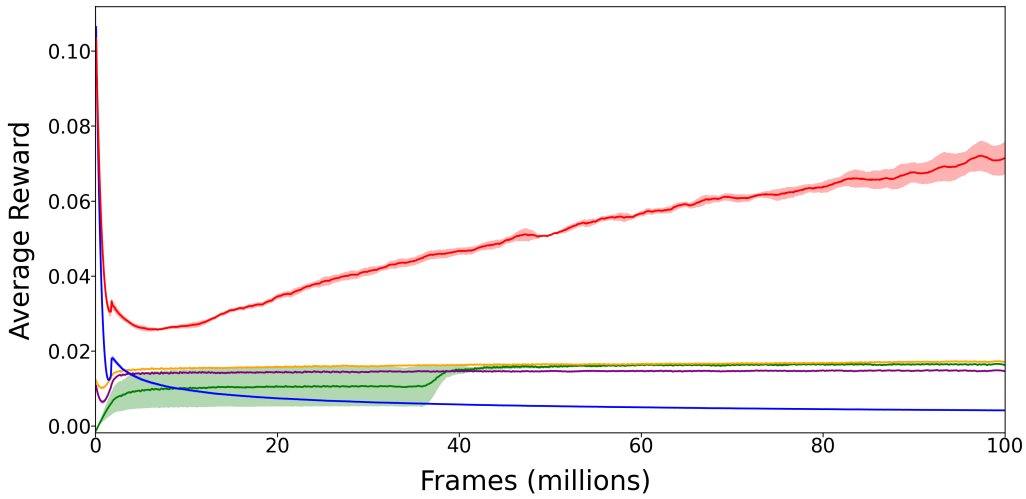
Random Features	Extrinsic Only	Inverse Dynamic Features
INT=1.0, EXT=0.0	INT=0.0, EXT=1.0	INT=1.0, EXT=0.0
INT=0.12, EXT=0.88		INT=0.12, EXT=0.88





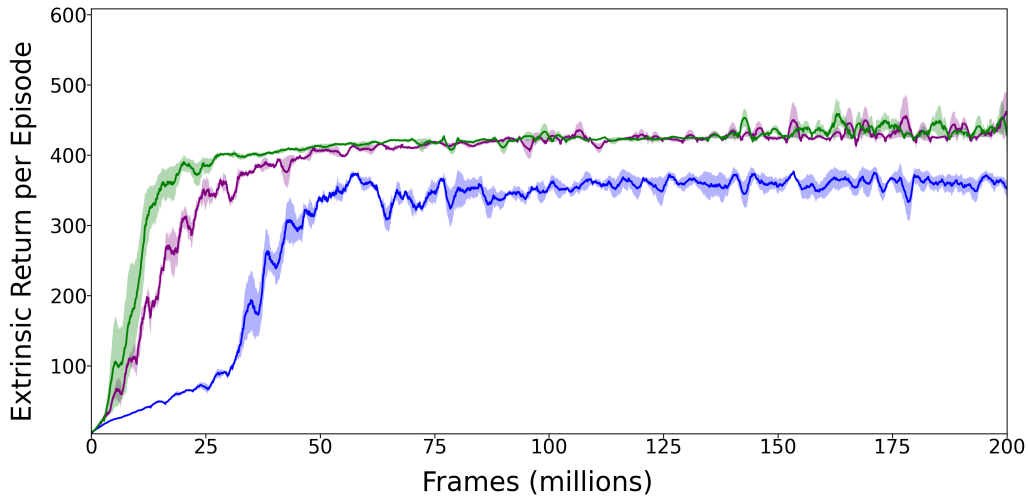

Average Return
Frames (millions)
0
1
2
3
4
5
6
7
0
20
40
60
80
100

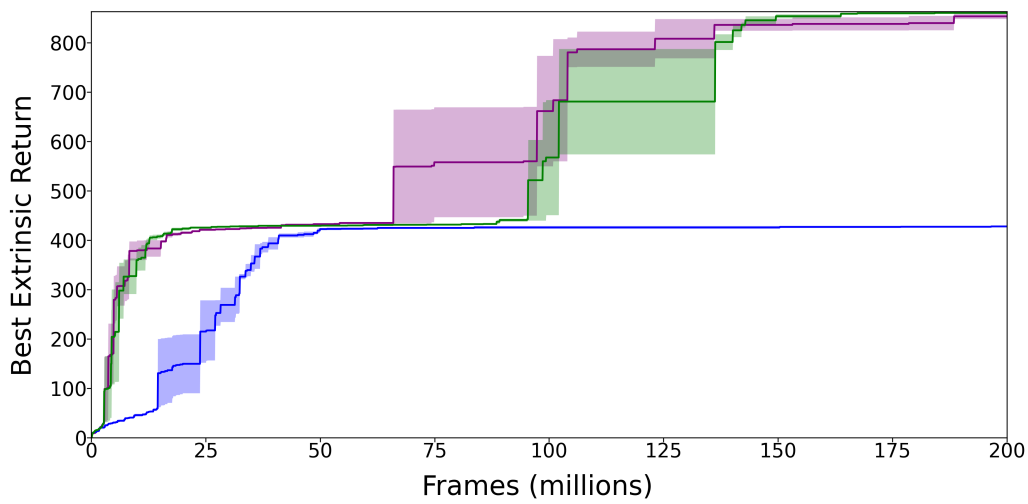

Standard Deviation Return
Frames (millions)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
20
40
60
80
100

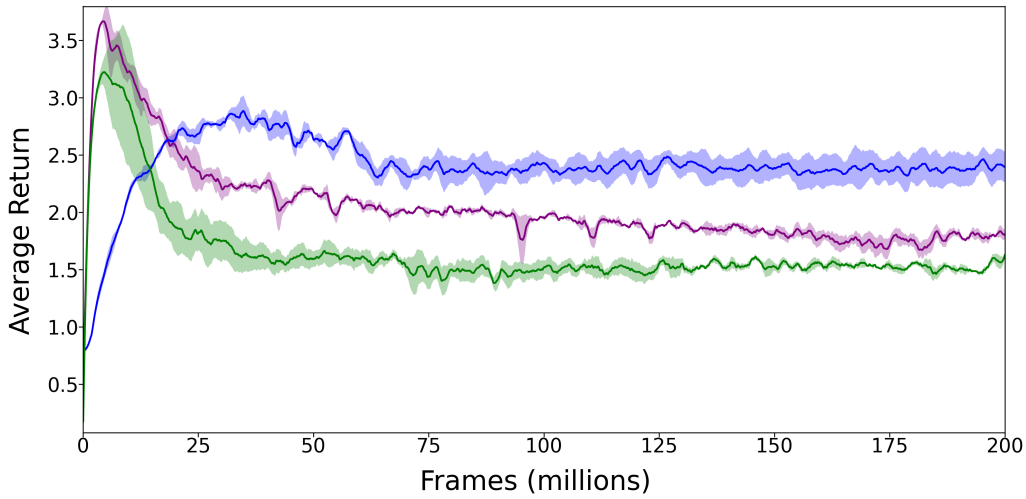

Average Reward
0.10
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100
Frames (millions)

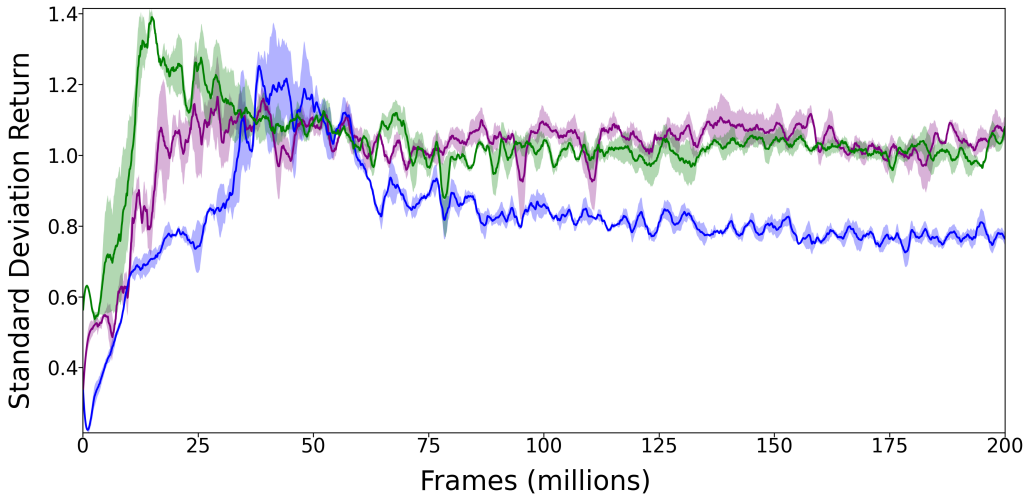
Breakout

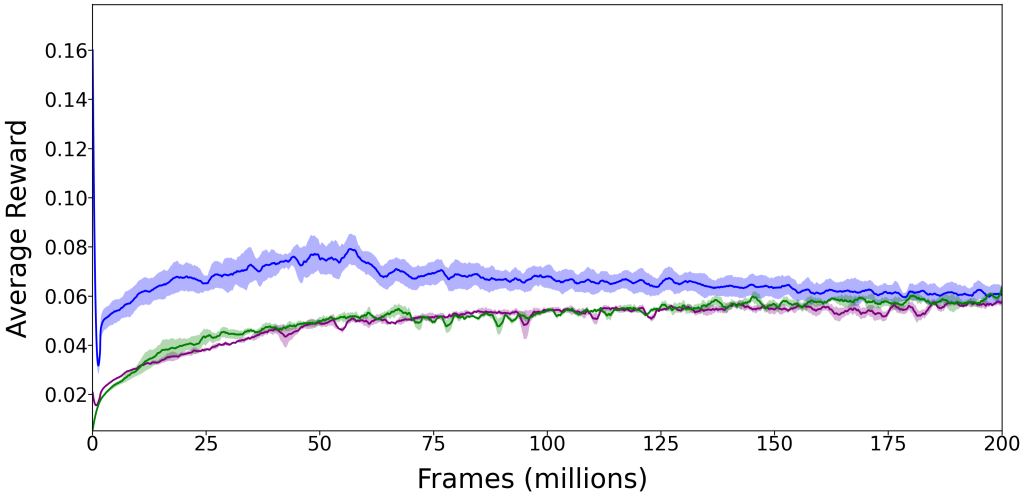
Random Features	Extrinsic Only
INT=1.0, EXT=0.0	INT=0.0, EXT=1.0
INT=0.1, EXT=0.9	






Average Return
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
25
50
75
100
125
150
175
200
Frames (millions)


Standard Deviation Return
1.4
1.2
1.0
0.8
0.6
0.4
0
25
50
75
100
125
150
175
200
Frames (millions)


Average Reward
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0
25
50
75
100
125
150
175
200
Frames (millions)