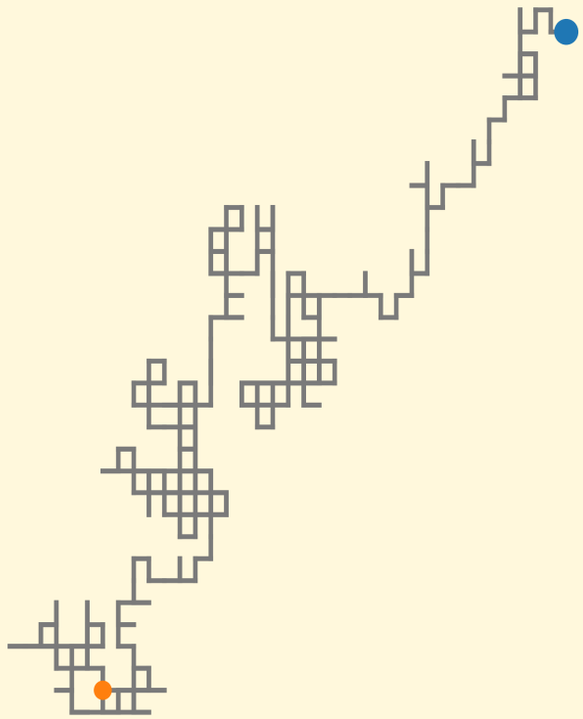
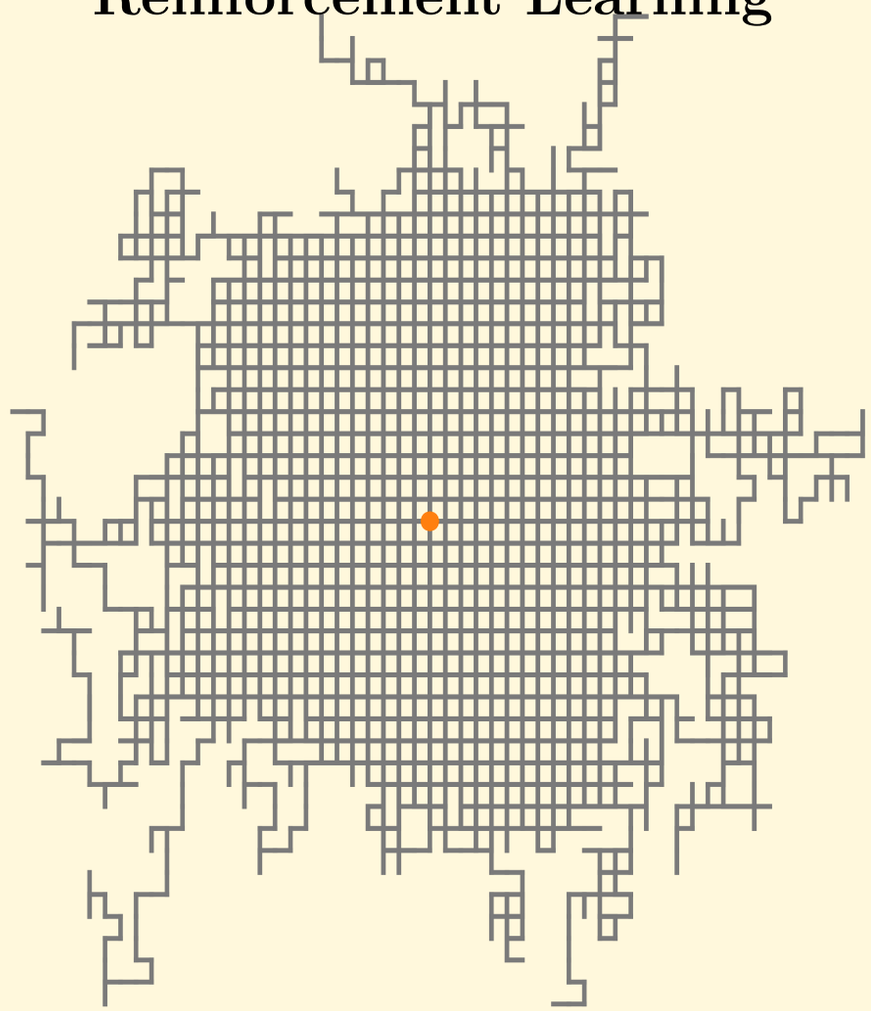
Enhancing autonomy and efficiency in goal-conditioned reinforcement learning

Yang, Z.



Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning · Zhao Yang

Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning



Yang, Zhao
杨 昭

Propositions
accompanying the thesis

Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning

1. In reset-free settings, model-based reinforcement learning is always preferable over model-free. [Chapter 3]
2. In deterministic tasks where we require a quick solution, stitching previously encountered high-reward trajectories ('episodic memory') is preferable over reinforcement learning. [Chapter 4 & 5]
3. Adaptive switching between episodic memory and reinforcement learning gives both fast and optimal solutions. [Chapter 5]
4. When the agent reaches frontier goals, additional exploration near those goals expands the agent's knowledge boundary, leading to improved performance. [Chapter 6]
5. Reinforcement learning agents should context-dependently decide when, how and for how long to explore.
6. When downstream tasks are unknown, learning a world model should be the primary objective of an reinforcement learning agent.
7. We should strive for autonomous reinforcement learning agents that require as few human interventions as possible.
8. Getting reinforcement learning to work requires reading, thinking, engineering, experimenting, and organizing, and each aspect is indispensable.
9. If you do not yet have a clear goal in life, try to learn as much as possible about the world, since it will enable you to quickly adapt to future goals.

Zhao Yang
Leiden, 26-02-2025

Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op woensdag 26 februari 2025
klokke 16:00 uur

door

Yang, Zhao 杨昭

geboren te Ningxian, China

in 1996

Promotor:
Prof.dr. A. Plaat

Co-promotores:
Dr. T. M. Moerland
Dr. M. Preuss

Promotiecommissie:

Prof.dr. J. Batenburg	
Prof.dr. M. Bonsangue	
Dr. V. François-Lavet	Vrije Universiteit Amsterdam
Dr. J. Liu	Lingnan University Hong Kong
Prof.dr. M. Spaan	Delft University of Technology

This PhD project was conducted in the Reinforcement Learning Group, Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands.

SIKS Dissertation Series No. 2025-10. The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Research presented in this thesis was supported by the China Scholarship Council (CSC). CSC No. 202007720053.

ISBN: 978-94-6506-997-5
Printed by: Ridderprint

To my grandmother.

Contents

1 Introduction **1**

1.1 Research questions 4

1.2 Outline of this thesis 6

1.3 List of publications 8

2 Preliminaries **9**

2.1 Reinforcement learning 9

2.2 Goal-conditioned reinforcement learning 11

2.2.1 Define the goal space 12

2.2.2 Select a goal 13

2.2.3 Reach the goal 14

2.2.4 Post-explore 15

3 World Models Increase Autonomy in Reinforcement Learning **17**

3.1 Introduction 19

3.2 Related Work 21

3.3 Preliminaries 23

3.3.1 Reset-free RL 23

3.3.2 Model-based RL setup 24

3.4 Method 25

3.4.1 Back-and-Forth Go-Explore 25

3.4.2 Learning to Achieve Relevant Goals in Imagination 26

3.4.3 Implementation Details 27

3.5 Experiments 27

3.5.1 Results 29

3.5.2 Analysis 30

3.5.3 Ablations . . . 32
3.6 Conclusion and Future Work . . . 33
3.7 Experimental Details . . . 34
3.7.1 Environments . . . 34
3.7.2 Baseline Implementations . . . 36
3.7.3 Hyperparameters . . . 36
3.7.4 Results Clarification . . . 38
3.7.5 Resource Usage . . . 38
3.8 More Visualizations on Replay Buffer . . . 38
3.9 Detailed Ablations . . . 38
3.10 MBRL on Sawyer Door . . . 40
3.11 More Analysis on Fetch Environments . . . 43
3.12 Analysis on R3L . . . 44

4 Continous Episodic Control **45**
4.1 Introduction . . . 47
4.2 Background . . . 49
4.3 Related Work . . . 50
4.4 Continuous Episodic Control (CEC) . . . 51
4.5 Experiments . . . 55
4.5.1 Toy Examples . . . 55
4.5.2 Continuous Navigation Tasks . . . 58
4.5.3 Robotics Control Task . . . 59
4.6 Conclusion and Future Work . . . 60

5 Two-Memory Reinforcement Learning **63**
5.1 Introduction . . . 65
5.2 Background . . . 67
5.2.1 Markov Decision Process . . . 67
5.2.2 Deep Q-Learning . . . 67
5.2.3 Episodic Control . . . 68
5.3 Related Work . . . 68
5.3.1 Episodic Control . . . 69
5.3.2 Episodic Memory for Learning . . . 69
5.3.3 Experience Replay . . . 69
5.4 Two-Memory Reinforcement Learning (*2M*) . . . 70
5.4.1 A Motivating Example . . . 71

5.4.2 Switching . . . 73
5.4.3 Learning . . . 74
5.5 Experiments . . . 74
5.5.1 Proof of Concept Under Tabular RL Setting . . . 75
5.5.2 Results on MinAtar Games . . . 76
5.5.3 Ablation Study . . . 78
5.6 Conclusion and Future Work . . . 81

6 First Go, then Post-Explore **83**
6.1 Introduction . . . 85
6.2 Related Work . . . 86
6.3 Background . . . 87
6.4 Methods . . . 89
6.4.1 IMGEP . . . 89
6.4.2 Post-exploration . . . 91
6.5 Experiments . . . 92
6.5.1 Experimental Setup . . . 92
6.5.2 Hyper-parameter Settings . . . 92
6.5.3 Results . . . 94
6.6 Conclusion and Future Work . . . 96

7 Conclusion **99**
7.1 Answers to research questions . . . 99
7.2 Limitations . . . 102
7.2.1 Limitations of model-based reset-free methods . . . 102
7.2.2 Limitations of episodic control methods . . . 103
7.2.3 Limitations of post-exploration . . . 105
7.3 Reflections on goal-conditioned reinforcement learning . . . 106
7.4 Future research . . . 107
7.5 Conclusion . . . 108

Acknowledgements **123**

Summary **125**

Samenvatting **127**

Titles in the SIKS dissertation series since 2016 **129**

Curriculum Vitae **147**

Chapter 1

Introduction

A child who learns to walk integrates sensory feedback from the environment with motor commands from the brain. We, as humans, first perceive the surrounding environment, gather visual, proprioceptive and tactile information and adjust our movements accordingly to avoid falls or setbacks to learn to walk in a stable, fast fashion. The aforementioned learning process is an example of reinforcement learning (RL): we adjust our behavior based on the feedback we collect from the external environment (Sutton & Barto, 2018). If we walk steadily and reach the desired locations sooner to get food for survival, which is positive feedback, we reinforce the behaviors that create such fast movements. In contrast, when we fall, we receive negative feedback. The impact of hitting the ground creates pain. As a result, we avoid the behaviors that lead to the fall. Although humans generally take 10 to 18 months to learn to walk by following this reinforcement learning circle, we do not require external help and can learn innately. This is the appeal of RL, that the learning occurs through trial and error, in a self-improving way. In this thesis, we will discuss a computational way to implement reinforcement learning, which is a popular subfield of artificial intelligence.

As an illustration, let's imagine there is a humanoid robot called Bob that is instructed to learn to walk. Bob is equipped with all different types of sensors for capturing the surrounding information and more importantly, it is programmed with a reward function that acts like our brain to score every action that Bob takes. The reward will be higher when the movement is stable and swift, and lower if Bob falls down. Bob dedicates itself to learning day and night, sometimes falling to the ground and requiring human assistance for repairs. After several days of effort, Bob finally masters the ability to walk.

With walking mastered, Bob is instructed to learn to cook, cycle, play soccer, and more. There are many more skills it can acquire and goals it can achieve. However, the skills Bob mastered for walking might not be useful for new tasks. For example, cooking requires Bob to stand and use its arm, rather than walk. Bob must develop new skills for newly encountered tasks. This way, if the same task is commanded in the future, Bob can execute the previously learned corresponding skill set. To learn multiple tasks, Bob begins to adjust its behaviors based on the feedback given by the reward function that changes with each new goal. Over time, Bob learns different skill sets to accomplish different goals. This process is called goal-conditioned reinforcement learning (GCRL), which is a generalization of standard RL and aims to learn multiple tasks. In the example, GCRL enables Bob to learn more than just one task, with each task linked to a specific reward function. In GCRL, the behavior learned is conditioned on the given goal and adapts when the goal changes.

To complete the GCRL loop (Colas et al., 2022), Bob should follow three primary steps and one optional step:

1. **Define the goal space**. Bob must first understand which goals are achievable within its capabilities. For instance, Bob could set goals such as learning to walk or cook. However, goals like flying would be unrealistic since Bob lacks wings.

2. **Select a goal**. Once the goal space is defined, Bob needs to choose a goal that is interesting to pursue. For example, if Bob has already mastered walking, it might find cooking more interesting and challenging as a new goal.

3. **Reach the goal**. After selecting a goal, Bob must devise a strategy to achieve it. This could involve various learning methods such as learning through trail and error, or imitating human behaviors.

4. **Post-explore** (optional). Even after achieving a goal, Bob can choose to continue exploring further and push beyond the initial goal to discover additional capabilities or improvements. For example, after mastering walking, Bob might continue to practice and eventually learn to run, finding that running is an extension of walking.

After the goal is achieved (step 3), Bob generally returns to step 2 to select a new goal and repeat the process. Or it goes back to step 1 since the newly acquired experience can also change / grow the goal space. Alternatively, Bob can opt for the post-exploration phase (step 4) to further explore new possibilities. Intuitively, when a goal is accomplished, Bob reaches the limit of its current abilities. By continuing

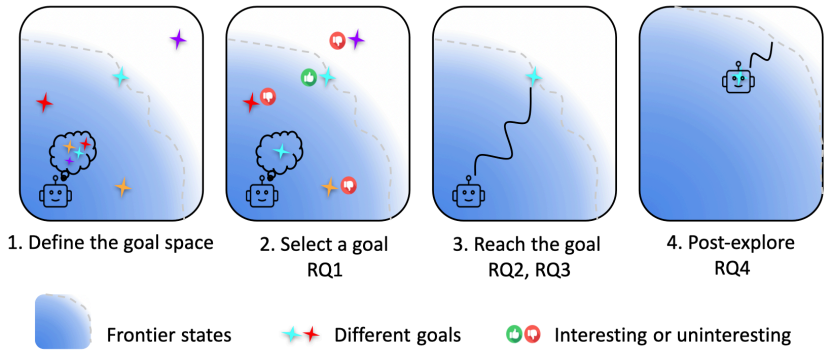


Figure 1.1: Four steps in the GCRL framework. We investigate four research questions (RQ) in total. RQ1 in Chapter 3 proposes a new goal selection mechanism to deal with the reset-free RL setting. RQ2 in Chapter 4 and RQ3 in Chapter 5 propose two non-parametric methods to learn to solve the given task faster. RQ4 in Chapter 6 investigates the role of post-exploration.

to push beyond these limits, it is likely to uncover new and exciting insights, thereby learning new skills.

GCRL (Liu et al., 2022) is an extension of conventional RL that adapts it to a multiple-goal setting. As illustrated by the example of Bob, GCRL enables the agent to learn to achieve multiple goals rather than just one. In our complex society, everyone must manage various roles and responsibilities, and ideally, we would want an AI agent to emulate this versatility, becoming a generalist agent proficient in various fields.

Recent developments demonstrate that leveraging the GCRL framework can enable an agent to achieve remarkable feats. For instance, an agent trained using GCRL is able to play StarCraft at the level of best human professional players (Vinyals et al., 2019), a game that requires rapid decision-making and precise control; perform hundreds of tasks (Mendonca et al., 2021), including object manipulation and locomotion; or solve Rubik's cube (Akkaya et al., 2019) with a robot hand. It also shows tremendous potential in other fields, such as finance (Zheng et al., 2022), healthcare (He et al., 2023) and energy management (Yi et al., 2022).

Training a generalist agent using GCRL takes four steps (illustrated in Figure 1.1), each of which are researched individually or combined to improve the GCRL loop. We now discuss each of these four steps and identify what is still missing.

As shown in Figure 1.1, the first step of the GCRL framework is to define the goal space. The goal space defines the space of interest that the agent should learn to achieve ultimately. It can be defined in several ways: the same as the state space, a

sub-space of the entire state space (Ecoffet et al., 2021; Plappert et al., 2018), or an embedding space of the original state space (Mendonca et al., 2021; Péré et al., 2018). After having a goal space, we need to specify a goal sampling strategy to sample goals from the defined goal space (see the step 2 in Figure 1.1).

A default approach for sampling goals is uniform sampling, which assumes that every goal is equally important to learn. However, previous research also shows that biasing the sampling towards frontier states (Pitis et al., 2020; Pong et al., 2019) or goals on which the agent has learning progress (Florensa et al., 2018; Portelas et al., 2020) can lead to better performance.

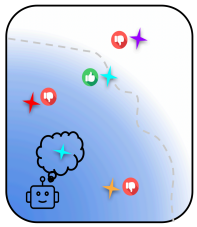
When a goal is selected (see the step 3 in Figure 1.1), the agent can be trained to reach the goal using goal-conditioned policies (Rahman et al., 2023; Schaul, Horgan, et al., 2015; Z. Zhang et al., 2023) or online planning (Hansen et al., 2022; Sancaktar et al., 2022). Afterwards, if the agent achieves the selected goal or exhausts the assigned budget for reaching it, it can opt for performing additional exploration (Ecoffet et al., 2021; E. S. Hu et al., 2023), which we call post-exploration. As illustrated in the fourth step of Figure 1.1, by engaging in post-exploration, the robot ventures into new areas and expands the frontier of its capabilities.

Although extensive research has already been conducted on each step of the GCRL framework, in this thesis, we have identified several drawbacks in current methods: 1) Current goal selection strategies are designed for conventional episodic RL settings, and they fail in other scenarios, for example, in settings where there is no reset. 2) After an interesting goal (e.g. the frontier state) is selected, an under-trained goal-conditioned policy might be lagging behind and not able to reach it, which slows down the learning process. 3) While performing additional exploration after reaching a goal (post-exploration) enhances performance, a direct comparison with the agent without post-exploration is lacking. In this thesis, we investigate the aforementioned problems of existing methods and propose various new approaches to overcome these drawbacks, thereby improving the GCRL framework.

1.1 Research questions

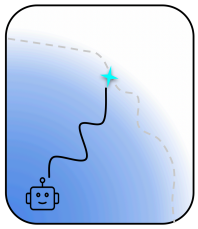
We research four questions in total (see Figure 1.1), one on step 2 (we propose a goal selection strategy to tackle the RL setting where there is no reset), two on step 3 (we propose two non-parametric methods to reach the goal faster) and one on step 4 (we investigate the role of post-exploration). The research questions we aim to answer in the thesis are as follows:

Q1 **Can RL agents learn without access to a 'reset'?[Chapter 3]**



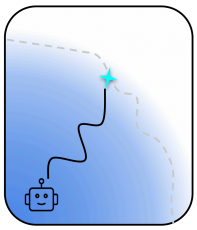
(step 2) RL agents typically need to be reset to their initial states after achieving a goal, starting the process over again. For example, once a quadruped robot completes a loading task in the factory, it will be moved to a designated location and its angles of joints will be restored to their default positions, preparing it for the next task. Additionally, if the robot gets stuck along the way, human intervention is required to free it so it can continue its task. Ideally, an RL agent should be able to operate autonomously and manage to escape from deadlock states, restore its joints, and move itself to designated locations as needed. In the GCRL framework, the behaviors of agents largely depend on commanded goals, making intelligent goal selection crucial for effective exploration and navigation. We demonstrate that by selecting goals appropriately, GCRL agents can operate without the access to the reset method. On the contrary, they can perform reset behaviors by themselves. This process occurs without human intervention, thus being more autonomous.

Q2 **Can non-parametric methods be applied to tasks with a continuous action space to achieve faster learning? [Chapter 4]**



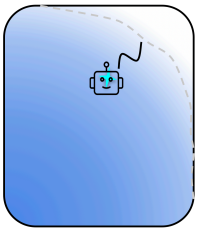
(step 3) Learning signals in RL are slowly back-propagated, especially when combined with parametric function approximations (e.g. deep neural networks). Rather than using RL to learn to achieve goals, previous research shows that non-parametric methods such as episodic control (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016) can learn faster in Atari (Bellemare et al., 2013) and Labyrinth tasks, which all have discrete action spaces. We instead study if episodic control methods can also be applied to tasks with a continuous action space and, if so, how we can implement it to achieve faster learning, as it does in tasks with a discrete action space.

Q3 **Can non-parametric and parametric methods be combined to achieve both fast learning and optimality? [Chapter 5]**



(step 3) Non-parametric methods (episodic control) overwrite the previous solution by the newly encountered better solution. Consequently, it keeps track of the best solution quickly, even if such a solution might occur with low probability. Unlike RL, in tasks with stochasticity, episodic control is unable to store the optimal solution. In essence, RL is slow but capable of finding the optimal solution, while episodic control is fast but tends to be non-optimal. We investigate whether it is possible to combine episodic control with RL methods to leverage the advantages of both. Specifically, can we achieve the speed of episodic control while also attaining the optimality of RL?

Q4 **What benefits can post-exploration introduce compared to not having it? [Chapter 6]**



(step 4) Optionally, when the selected goal is achieved, RL agents can choose to explore additionally and hopefully step into the unknown area. We call the exploration after the goal has been achieved 'post-exploration'. Go-Explore (Ecoffet et al., 2021) implements the post-exploration idea as an RL agent and shows strong performance on extremely hard exploration tasks. However, it is not clear why the agent with post-exploration outperforms the one without and whether it can be extended to more general cases. We investigate why post-exploration is helpful and conduct a more explicit comparison between the agent with it and without by turning it on and off in the same RL algorithm within a more general GCRL framework, under both tabular and deep RL settings in discrete navigation and continuous control tasks.

1.2 Outline of this thesis

The stucture of this thesis is as follows. In Chapter 2, we provide the necessary background for understanding the work presented: RL and GCRL as well as its four main algorithmic design choices. Chapter-specific preliminaries will be introduced in each chapter separately.

Then, in Chapter 3, we propose a model-based reset-free (MoReFree) agent to tackle the reset-free RL setting where there is no access to a reset mechanism. By using world models and commanding the GCRL agent on three different goal distributions, MoReFree can eliminate the need for reset and result in a more autonomous agent.

MoReFree outperforms state-of-the-art methods tailored for reset-free RL setting with regard to data efficiency and asymptotic performance.

In Chapter 4, we propose a non-parametric method called continuous episodic control (CEC) that can learn to achieve goals faster. Since the original non-parametric method episodic control (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016) only works for tasks with a discrete action space, we extend it to be able to solve tasks with a continuous action space and examine it on several robotic control tasks. Results show that CEC has a superior learning speed than conventional deep RL methods while obtaining competitive asymptotic performance.

Non-parametric methods (episodic control) overwrite the existing solutions with better ones as they are encountered. These methods learn faster but are non-optimal in stochastic tasks. In Chapter 5, we combine episodic control with RL to form one united agent, which switches between the two agents to gain benefits from both sides, i.e. the speed of episodic control and optimality and generalizability of RL.

In Chapter 6, we present the work on post-exploration, where we systematically investigate how additional exploration after the selected goal is achieved (post-exploration) enhances the performance under tabular and deep RL settings in discrete navigation and continuous control tasks. We find that the agent with post-exploration discovers more of the state space compared to the one without.

Finally, in Chapter 7 we reflect on the work presented in this thesis and the field of goal-conditioned RL as a whole, and propose potential directions for future work. At the end, we summarize the main conclusion.

1.3 List of publications

The chapters in this thesis are based on the following publications. Publications are edited only for style cohesion; all content remains unchanged from the published versions. The last one publication is related to the research, but not part of this thesis.

Chapter	Publication
3	Zhao Yang, Thomas M. Moerland, Mike Preuss, Aske Plaat, Edward S. Hu (2024). *World Models Increase Autonomy in Reinforcement Learning.* In submission
4	Zhao Yang, Thomas M. Moerland, Mike Preuss, Aske Plaat (2023). *Continuous Episodic Control.* In proceedings of the 2023 IEEE Conference on Games.
5	Zhao Yang, Thomas M. Moerland, Mike Preuss, Aske Plaat (2023). *Two-Memory Reinforcement Learning.* In proceedings of the 2023 IEEE Conference on Games.
6	Zhao Yang, Thomas M. Moerland, Mike Preuss, Aske Plaat (2023). *First Go, then Post-Explore: the Benefits of Post-Exploration in Intrinsic Motivation.* In proceedings of the 15th International Conference on Agents and Artificial Intelligence.
-	Zhao Yang, Mike Preuss, Aske Plaat (2021). *Transfer Learning and Curriculum Learning in Sokoban.* In proceedings of the 33rd Benelux Conference on Artificial Intelligence. Communications in Computer and Information Science, vol 1530 (pp. 187-200). Springer, Cham.

Chapter 2

Preliminaries

In this chapter, we explain the necessary concepts, definitions, and notations for understanding the entire thesis. Chapter-specific concepts are explained within each chapter. First, we introduce reinforcement learning, a framework used to solve decision-making problems. Then, reinforcement learning will be extended to the goal-conditioned setting, where the agent needs to learn policies conditioned on different goals. In the end, related work on four fundamental phases of the goal-conditioned reinforcement learning framework is briefly discussed.

2.1 Reinforcement learning

Reinforcement learning (RL) is a framework to solve sequential decision-making problems, which are formalized as Markovian decision processes (MDPs). In this thesis, we follow the definition and notations of MDP introduced by Sutton and Barto, 2018. A MDP is a 5-tuple, $< \mathcal{S}, \mathcal{A}, p, r, \gamma >$, where:

- $\mathcal{S}$ is a set of states, called the state space.
- $\mathcal{A}$ is a set of actions, called the action space.
- $p(s'|s,a) \doteq Pr(S_t = s'|S_{t-1} = s, A_{t-1} = a)$ is the transition dynamic, which defines the probability of transitioning from state s to s' after taking the action a at time step t. $s, s' \in \mathcal{S}$ are states and S_t is the state at time t. Similarly, $a \in \mathcal{A}$ is the action and A_t is the action at time t.

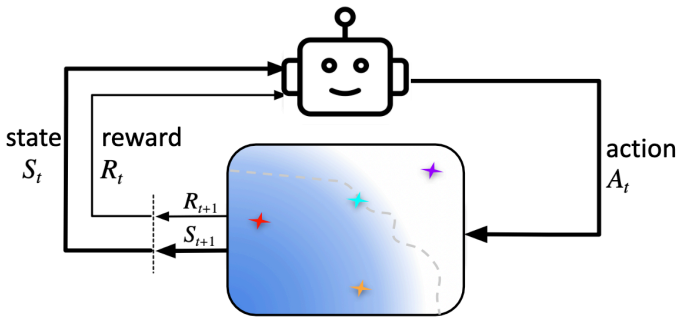


Figure 2.1: The agent-environment interaction in a MDP (Sutton & Barto, 2018). The agent observes the state S_t and takes the action A_t in the environment. The environment returns the next state S_{t+1} and reward R_{t+1}. Then the process repeats.

- $r(s, a, s')$ is the reward function, which produces the intermediate reward $R_t \doteq r(s, a, s'|S_{t-1} = s, A_{t-1} = a, S_t = s')$, i.e. after transitioning from s to s' by taking action a.
- γ is the discount factor ranging from $[0, 1]$.

The agent-environment interaction is shown in Figure 2.1. At time step t, the agent observes the state S_t. Then, according to a policy $\pi : \mathcal{S} \to \Pr(\mathcal{A})$, the agent chooses an action $A_t \sim \pi(A_t|S_t)$. Next, the action A_t is executed and influences the environment to transition from S_t to S_{t+1} based on the transition dynamic p. At the same time, the agent gets a reward R_{t+1}. The agent observes the new state S_{t+1}, then the process repeats. We are interested in the long-term performance of the agent, thus we would like to maximize the cumulative future reward that the agent can get. The cumulative future reward (also called return G) at the time step t can be written as:

$$G_t \doteq R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1}. \tag{2.1}$$

We define the value function $V_\pi(s)$ of the state s as the expected return starting from the state s, following the policy π and transition dynamic p:

$$V^\pi(s) \doteq \mathbb{E}_{\pi,p}[G_t|S_t = s]. \tag{2.2}$$

The equation can be written in a recursive format, known as the Bellman equation:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim p(\cdot|s,a)}[r(s, a, s') + \gamma \cdot V^\pi(s')]. \tag{2.3}$$

The state-action value function $Q^\pi(s,a)$ is defined as the expected return starting from the state s and taking the action a, and following the policy π afterwards, based on the transition dynamic p:

$$Q^\pi(s,a) \doteq \mathbb{E}_{\pi,p}[G_t|S_t = s, A_t = a]. \tag{2.4}$$

Similarly, the state-action value function can be rewritten in a recursive format as well:

$$Q^\pi(s,a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}[r(s,a,s') + \gamma \cdot \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q^\pi(s',a')]]. \tag{2.5}$$

The objective of RL is to find a policy π^* that maximizes the state-action value function:

$$\pi^* \doteq \underset{\pi}{\operatorname{argmax}}\, Q^\pi(s,a). \tag{2.6}$$

2.2 Goal-conditioned reinforcement learning

Goal-conditioned reinforcement learning (GCRL) extends RL to a multiple-goal setting to solve goal-conditioned MDP, which is defined as $< \mathcal{S}, \mathcal{A}, p, r, \gamma, \mathcal{G} >$, where $\mathcal{G}$ is a set of goals called the goal space. For any goal $g \in \mathcal{G}$, the reward function will also take the goal g as input, denoted as $r(s,a,s',g)$ (Schaul, Horgan, et al., 2015), which provides the reward for achieving the goal g. We can then rewrite the goal-conditioned state value function and state-action value function in a similar way as Equations (2.3) and (2.5):

$$\begin{aligned} V^\pi(s,g) &= \mathbb{E}_{a\sim\pi(\cdot|s,g),s'\sim p(\cdot|s,a)}[r(s,a,s',g) + \gamma \cdot V_g^\pi(s',g)], \\ Q^\pi(s,a,g) &= \mathbb{E}_{s'\sim p(\cdot|s,a)}[r(s,a,s',g) + \gamma \cdot \mathbb{E}_{a'\sim\pi(\cdot|s',g)}[Q^\pi(s',a',g)]], \end{aligned} \tag{2.7}$$

where π now takes an additional input g to be a goal-conditioned policy. The objective of GCRL is to find a policy π^* that maximizes the state-action value function over all goals (Liu et al., 2022):

$$\pi^* \doteq \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{g\sim p_g}\, Q^\pi(s,a,g), \tag{2.8}$$

where p_g represents the desired goal distribution.

As we discussed in Chapter 1, a typical GCRL framework comprises four phases (as detailed in Table 2.1): 1) defining the goal space; 2) selecting an interesting goal for the agent; 3) the agent learning to reach the goal; 4) the agent post-exploring. Each of

Table 2.1: Commonly used methods for each step in the GCRL framework (see Figure 1.1). Representative references are added in the table and more extensive references can be found in the text. GCP is short for goal-conditioned policy and MB is short for model-based.

1. Define the goal space	2. Select a goal	3. Reach the goal	4. Post-explore
State space (Ecoffet et al., 2021; Plappert et al., 2018)	Frontier states (Pitis et al., 2020; Pong et al., 2019)	GCP (Hafner et al., 2023; Schaul, Horgan, et al., 2015)	Random (Ecoffet et al., 2021)
Learned (Eysenbach et al., 2022; Mendonca et al., 2021)	Learning progress (Florensa et al., 2018; Portelas et al., 2020)	MB planning (Hansen et al., 2023)	Intrinsic reward (Sekar et al., 2020; Zhu et al., 2020)
Others (Fu et al., 2019; Jiang et al., 2022)	Others (E. S. Hu et al., 2023; C. Lu et al., 2024)	Others (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016; Ecoffet et al., 2021)	Others (Klissarov et al., 2023)

these steps has been studied either independently or in conjunction with others. We now provide an overview of commonly used methods for each step. A summary can be found in Table 2.1. It is important to note that while some of these works may not have been explicitly studied within the GCRL framework, they can essentially be adapted and integrated into the GCRL framework with minimal modifications.

2.2.1 Define the goal space

To ensure that the training goals align with the evaluation goals, it is essential to define a goal space that matches or covers the area of interest for evaluation. Typically, evaluation focuses on a sub-space of the entire state space. For instance, in the Fetch-Push task [1], although the state space of the block covers the whole table, the robot is only required to push the block to goals sampled from the center area of the table during evaluation, rather than the entire table. Thus, a pre-defined sub-space of the state space is generally used for sampling goals in tasks with a continuous state space (Plappert et al., 2018; X. Yang et al., 2021; Yu et al., 2019). In tasks with a discrete state space, specific states are selected (e.g. states that can lead to more exploration potential) to form a pool representing the goal space (Ecoffet et al., 2021; Kompella

[1]The Fetch-Push task can be found in https://robotics.farama.org/envs/fetch/push/

et al., 2022).

When dealing with high-dimensional state spaces, such as images, the original state space may contain excessive distracting information. Therefore, a more compact latent space can be learned using techniques such as a self-predictive objective (Hansen et al., 2023; Schwarzer et al., 2020), contrastive loss (Eysenbach et al., 2022; Poudel et al., 2024), or reconstruction loss (Hafner et al., 2023; Mendonca et al., 2021), which can then be utilized as the goal space.

Besides using the original state space or a learned latent space thereof, language itself offers a compact and abstract representation that can be used as a goal space (Chevalier-Boisvert et al., 2024; Fu et al., 2019). Moreover, the integration of multi-modal goals, which combines both languages and images (Jiang et al., 2022), can further provide a richer goal space.

Although our thesis does not focus on how to better define a goal space, it is a crucial aspect of the GCRL framework in general.

2.2.2 Select a goal

The exploration of the GCRL agent largely depends on the selected goals, which serve as guidance to indicate the next area of interest for the agent. Once the goal space is defined, the most straightforward approach to select goals from the goal space is uniform sampling, where each goal in the goal space is sampled with equal probability. However, as learning progresses, certain goals may become more relevant than others. For instance, after the agent masters nearby goals, it may be more beneficial to focus on goals that are slightly farther away, but not too distant (e.g. the step 2 in Figure 1.1).

Various strategies have been developed to select goals dynamically. Skew-fit (Pong et al., 2019) and MEGA (Pitis et al., 2020) focus on selecting frontier states as goals by first estimating state density and then choosing states with lower densities. These methods encourage the agent to explore less-visited areas of the state space. AMIGo (Campero et al., 2021) trains two agents in an adversarial fashion: one agent is trained to select challenging goals for another agent, which then attempts to reach these selected goals. This approach ensures continuous improvement for both learning agents, encouraging the goal-selecting agent to select interesting goals progressively. Goal-GAN (Florensa et al., 2018) selects goals based on their difficulty, and goals that are neither too easy nor too difficult are selected for the agent to achieve. Similarly, selecting goals where the agent demonstrates learning progress (Portelas et al., 2020) has also shown to be effective. PEG (E. S. Hu et al., 2023) focuses on goals that offer

the most exploration potential, driving the agent toward the most informative parts of the state space.

Recently, foundation models have demonstrated a profound understanding of the real world, characterized by their ability to master a wide range of tasks and domains. Consequently, they can be directly utilized to inform the agent about which goals to pursue (C. Lu et al., 2024).

Unlike the goal selection strategy we propose in Chapter 3, which is designed especially for a more realistic and autonomous RL scenario where the expensive reset is eliminated, the methods discussed here are tailored for the conventional episodic RL setting and often fail in the more challenging reset-free setting. More specifically, these methods cannot deal with over-exploration, a challenge that is introduced by the reset-free setting.

2.2.3 Reach the goal

After the goal is selected, goal-conditioned policies / value functions (Schaul, Horgan, et al., 2015) are usually deployed to learn to reach the given goal. On the other hand, since the process of learning to reach the goal is a decision-making problem formulated as a MDP, any methods that solve an MDP can be used here. Meanwhile, RL methods might not always be the best choice due to their poor performance under certain scenarios, such as low data regime, or in the setting where a dynamic model is given or learned, methods beyond reinforcement learning are also used to fulfill the given goal.

If access to the underlying tasks / environments is available, then the agent can be directly teleported to the selected goal and then post-explores (Ecoffet et al., 2021). It saves a massive amount of learning time but having such oracle access is impossible and unrealistic in real life. When a dynamic model is given / learned, besides training goal-conditioned policies/value functions from synthetic data generated by the dynamic model (Hafner et al., 2023), model-based planning can also be utilized to find the solution to the given goal (Hansen et al., 2023; Schrittwieser et al., 2020). Other methods like imitation learning (Ding et al., 2019; Reuss et al., 2023), evolution strategies (Salimans et al., 2017) or episodic control (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016; H. Hu et al., 2021) can also be used to learn to reach the goal.

In Chapters 4 and 5, we explore the application of episodic control, a non-parametric approach known for its superior learning speed (Blundell, Uria, Pritzel, Li, Ruderman,

Leibo, Rae, et al., 2016), for reaching the goal. We first extend the previous episodic control to tasks with a continuous action space in Chapter 4. Then, to address its inherent limitation (i.e. non-optimality), we propose to combine episodic control with deep RL Chapter 5, achieving both fast learning and better final performance.

2.2.4 Post-explore

When the selected goal or the allocated budget is reached, the agent may post-explore. This differs from the exploration in typical RL paradigms, where the exploration and exploitation are alternated (e.g. ϵ-greedy) or integrated (e.g. adding policy entropy regularization). In contrast, post-exploration is pure exploration, focusing entirely on exploring the environment without considering any task-specific rewards.

Essentially, any exploration approaches can be used for post-exploration. The most straightforward approach is random exploration (Ecoffet et al., 2021; Pong et al., 2019), which, despite its simplicity, has demonstrated strong performance. More sophisticated exploration mechanisms that seek novel states based on intrinsic reward (distinct from extrinsic reward provided by the environment) are also applicable.

For instance, Random Network Distillation (RND) (Burda et al., 2018) encourages the agent to seek novel states by maximizing the prediction error between a fixed randomly initialized network and a predictor network. In this setup, the predictor network is trained to predict the random network's output, with larger prediction errors indicating more novel states. Another method, Intrinsic Curiosity Module (ICM) (Pathak et al., 2017), rewards the agent based on the prediction error between the predicted and the real next state. ICM stimulates the agent to collect states that lead to more accurate prediction, effectively targeting less frequently encountered states. Plan To Explore (P2E) (Sekar et al., 2020) is a model-based exploration method that is trained to maximize the disagreement among outputs of an ensemble of world models, encouraging the agent to visit states that can improve world models the most.

Foundation models, as we described in Section 2.2.3, show profound understandings of the world. They can be used to provide preferences over states (Klissarov et al., 2023), which are then used to guide the exploration in a manner that aligns with the heuristics and insights of the used foundation models.

Whereas the paper that first introduces the idea of post-exploration illustrates that RL agents equipped with post-exploration can solve extremely challenging exploration tasks (Ecoffet et al., 2021), it lacks a direct comparison with agents that do not employ post-exploration. The underlying reasons why post-exploration enhances performance

remains unclear. Therefore, in Chapter 6, we conduct a systematic investigation on post-exploration by turning it on and off within the same RL agent and qualitatively analyse why post-exploration facilitates stronger performance.

Four primary steps of the GCRL framework in Figure 1.1 are discussed above and the chapters presented later focus on the step 2, 3 and 4. In Chapter 3, we work on the step 2 where a goal sampling strategy is proposed to tackle the reset-free RL setting. Chapter 4 and Chapter 5 study the step 3, and propose two non-parametric methods to learn the given task faster. Post-exploration (the step 4) is investigated in Chapter 6.

Chapter 3

World Models Increase Autonomy in Reinforcement Learning

Authors: Zhao Yang, Thomas M. Moerland, Mike Preuss, Aske Plaat, Edward S. Hu
In submission

Abstract

Reinforcement learning (RL) is an appealing paradigm for training intelligent agents, enabling policy acquisition from the agent's own autonomously acquired experience. However, the training process of RL is far from automatic, requiring extensive human effort to reset the agent and environments.

To tackle the challenging reset-free setting, we first demonstrate the superiority of model-based (MB) RL methods in such setting, showing that a straightforward application of MBRL can outperform all the prior state-of-the-art methods while requiring less supervision. We then identify limitations inherent to this direct extension and propose a solution called model-based reset-free (MoReFree) agent, which further enhances the performance. MoReFree adapts two key mechanisms, exploration and policy learning, to handle reset-free tasks by prioritizing task-relevant states. It exhibits superior data-efficiency across various reset-free tasks without access to environmental reward or demonstrations while significantly outperforming privileged baselines that require supervision. Our findings suggest model-based methods hold significant promise for reducing human effort in RL. Website: https://sites.google.com/view/morefree

3.1 Introduction

Reinforcement learning presents an attractive framework for training capable agents. At first glance, RL training appears intuitive and autonomous - once a reward is defined, the agent learns from its own automatically gathered experience. However, in practice, RL training often assumes the access to environmental resets that can require significant human effort to setup, which poses a significant barrier for real world applications of RL like robotics.

Most RL systems on real robots to date have employed various strategies to implement resets, all requiring a considerable amount of effort (Levine et al., 2016; Nagabandi et al., 2020; Yahya et al., 2017; Zhu et al., 2019). In Nagabandi et al. (2020)'s work, which trains a dexterous hand to rotate balls, the practitioners had to (1) position a funnel underneath the hand to catch dropped balls, and (2) deploy a separate robot arm to pick up the dropped balls for resets, and (3) script the reset behavior. These illustrate that even for simple behaviors, proper implementation of reset mechanisms can result in significant human effort and time.

Rather than depending on human-engineered reset mechanisms, the agent can operate within a reset-free training scheme, learning to reset itself (Eysenbach et al., 2017; Haldar et al., 2023; Sharma, Gupta, et al., 2021; Sharma et al., 2022) or train a policy capable of starting from diverse starting states (Zhu et al., 2020). However, the absence of resets introduces unique exploration challenges. Without periodic resets, the agent can squander significant time in task-irrelevant regions that require careful movements to escape and may overexplore, never returning from indefinite exploration. Recent unsupervised model-based RL (MBRL) approaches (E. S. Hu et al., 2023; Mendonca et al., 2021) in the episodic setting have shown sophisticated exploration, high data-efficiency and promising results in long-horizon tasks. This prompts the question: *would MBRL agents excel in the reset-free RL setting?*

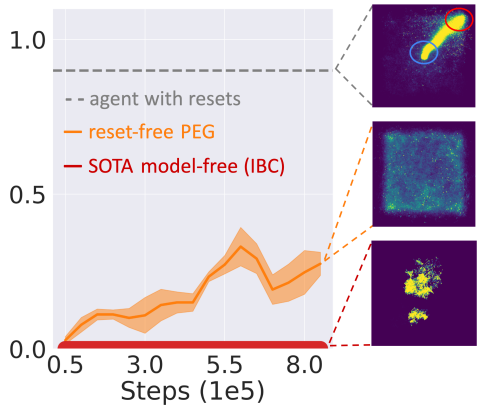


Figure 3.1: Performance and collected data of different agents on the reset-free Ant locomotion task.

As an initial attempt, we first evaluate an unsupervised MBRL agent, in a reset-

free Ant locomotion task. The ant is reset to the center of a rectangular arena, and is tasked with navigating to the upper right corner. The agent is reset only once at the start of training. The evaluation is episodic - the agent is reset at the start of each evaluation episode.

For the MBRL agent, we use PEG (E. S. Hu et al., 2023), which was developed to solve hard exploration tasks in the episodic setting. As seen in Figure 3.1, PEG, with minor modifications for the reset-free version, outperforms prior state-of-the-art, model-free agent, IBC (J. Kim et al., 2023), tailored for the reset-free setting.

In Figure 3.1, we plot state visitation heatmaps of the agents, where lighter colors correspond to more visitations. The oracle agent, with access to resets, explores the the "task-relevant" area between the initial and top right corner, which is ideal for training a policy that succeeds in episodic evaluation. IBC's heatmap (bottom) shows that it fails to explore effectively, never encountering the goal states in the top right region. In contrast, PEG exhaustively explores the entire space, as seen through its uniform heatmap. This results in an overexploration problem - PEG may devote considerable time on finding irrelevant states rather than concentrating on the task-relevant region of the task. This leads us to ask: **how can MBRL agents acquire more task-relevant data in the reset-free setting to improve its performance?**

We propose **Mo**del-based, **Re**set-**Free** (MoReFree), which improves two key mechanisms in model-based RL, exploration and policy optimization, to better handle reset-free training. Following the top row of Figure 3.2: to gather task-relevant data without resets, we define a training curriculum that alternates between temporally extended phases of task solving, resetting, and exploration. Next, as seen in the bottom row of Figure 3.2, we bias the policy training within the world model towards achieving task-relevant goals such as reaching initial states and evaluation states.

Our key contributions are as follows: **(1)** We demonstrate the viability of using model-based agents with strong exploration abilities for the reset-free setting as well as their inherent limitations. We address such limitations through the MoReFree framework which focuses exploration and policy optimization on task-relevant states. **(2)** We evaluate the adapted reset-free version of MBRL baseline and MoReFree against state-of-the-art reset-free methods in 8 challenging reset-free tasks ranging from manipulation to locomotion. Notably, both model-based approaches outperform prior state-of-the-art baselines in 7/8 tasks in final performance and data efficiency, all the while requiring less supervision (e.g. environmental reward or demonstrations). MoReFree outperforms the model-based baseline in the 3 hardest tasks. **(3)** We perform in-depth analysis of the MoReFree and baselines behaviors, and show that MoReFree

explores the state space thoroughly while retaining high visitation counts in the task-relevant regions. Our ablations show that the performance gains of MoReFree come from the proposed design choices and justify the approach.

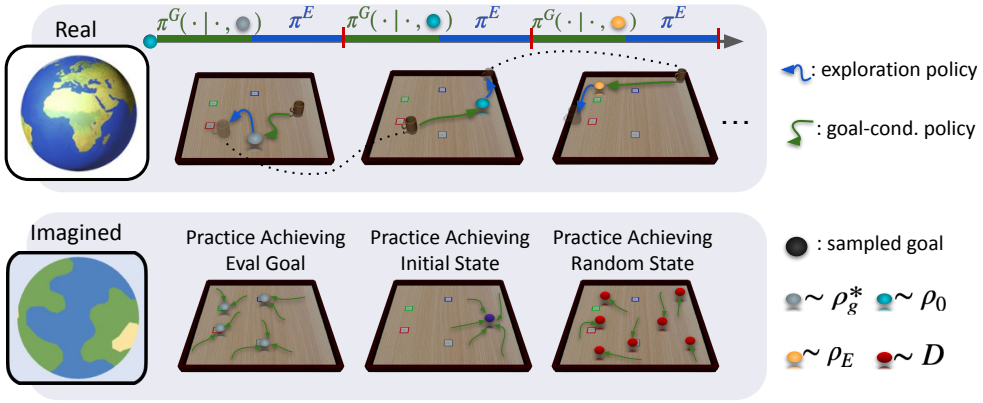


Figure 3.2: MoReFree is a model-based RL agent for solving reset-free tasks. **Top row:** MoReFree strikes a balance between exploring unseen states and practicing optimal behavior in task-relevant regions by directing the goal-conditioned policy to achieve evaluation states, initial state states (emulating a reset), and exploratory goals. **Bottom row:** MoReFree focuses the goal-conditioned policy training inside the world model on achieving evaluation states, initial states, and random replay buffer states to better prepare the policy for the aforementioned exploration scheme.

3.2 Related Work

Reset-free RL: There is a growing interest in researching reinforcement learning methods that can effectively address the complexities of reset-free training. Sharma, Xu, et al., 2021 proposes a reset-free RL benchmark (EARL) and finds that standard RL methods like SAC (Haarnoja, Zhou, Hartikainen, et al., 2018) fail catastrophically in EARL. Multiple approaches have been proposed to address reset-free training, which we now summarize. One approach is to add an additional reset policy, to bring the agent back to suitable states for learning (Eysenbach et al., 2017; J. Kim et al., 2022, 2023; Sharma, Gupta, et al., 2021; Sharma et al., 2022). LNT (Eysenbach et al., 2017) and J. Kim et al., 2022 train a reset policy to bring the agent back to initial state distribution, supervised by dense rewards and demonstrations respectively. MEDAL (Sharma et al., 2023, 2022), train a goal-conditioned reset policy and direct it to reset goal states from demonstrations. IBC (J. Kim et al., 2023) defines a curriculum for both task and reset policies without requiring demonstrations. VaPRL (Sharma,

Gupta, et al., 2021) trains a single goal-conditioned policy to reach high value states close to the initial states. Instead of guiding the agent back to familiar states, R3L (Zhu et al., 2020) and Xu et al., 2020 learn to reset the policy to diverse initial states, resulting in a policy that is more robust to variations in starting states. However, such methods are limited to tasks where exploration is unchallenging. The vast majority of reset-free approaches are model-free, with a few exceptions (K. Lu, Grover, et al., 2020; K. Lu, Mordatch, & Abbeel, 2020). Other works (Gupta et al., 2021; Smith et al., 2019) model the reset-free RL training process as a multi-task RL problem and require careful definition of the task distribution such that the tasks reset each other.

Goal-conditioned Exploration: A common theme running through the aforementioned work is the instantiation of a curriculum, often through commanding goal-conditioned policies, to keep the agent in task-relevant portions of the environment while exploring. Closely related is the subfield of goal-conditioned exploration in RL, where a goal-conditioned agent selects its own goals during training time to generate data. There is a large variety of approaches for goal selection, such as task progress (Baranes & Oudeyer, 2013; Veeriah et al., 2018), intermediate difficulty (Florensa et al., 2018), value disagreement (Y. Zhang et al., 2020), state novelty (Pitis et al., 2020; Pong et al., 2019), world model error (E. S. Hu et al., 2023; Sekar et al., 2020), and more. Many goal-conditioned exploration methods use the "Go-Explore" (Ecoffet et al., 2021) strategy, which first selects a goal and runs the goal-conditioned policy ("Go"-phase), and then switches to an exploration policy for the latter half of the episode ("Explore"-phase). PEG (E. S. Hu et al., 2023), which MoReFree uses, extends Go-Explore to the model-based setting, and utilizes the world model to plan states with higher exploration value as goals. However, such methods are not designed for the reset-free RL setting, and may suffer from *over-exploration* of task-irrelevant states.

Table 3.1: A conceptual overview of reset-free methods. Existing methods are model-free, and most of them require other forms of supervision (environmental reward or demonstrations or both). In performance, MoReFree improves over reset-free PEG, which significantly outperforms privileged baselines IBC, MEDAL and R3L.

Approach	MEDAL	IBC	VaPRL	R3L	reset-free PEG	MoReFree
Model-based	✗	✗	✗	✗	✓	✓
Demonstrations	✓	✗	✓	✗	✗	✗
Environmental reward	✓	✓	✓	✗	✗	✗

We notice that the majority of all prior work are model-free and may suffer from poor sample efficiency and exploration issues. In contrast, our model-based approaches

use world models to efficiently train policies and perform non-trivial goal-conditioned exploration with minimal supervision. See Table 3.1 for a conceptual comparison between prior work and two model-based methods (MoReFree and reset-free PEG).

3.3 Preliminaries

3.3.1 Reset-free RL

We follow the definition of reset-free RL from EARL (Sharma, Xu, et al., 2021), and extend it to the goal-conditioned RL setting. Consider the goal-conditioned Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{G}, \mathcal{A}, p, r, \rho_0, \rho_{g^*}, \gamma)$. At each time step t in the state $s_t \in \mathcal{S}$, a goal-conditioned policy $\pi(\cdot|s_t, g)$ under the goal command $g \in \mathcal{G}$ selects an action $a_t \in \mathcal{A}$ and transitions to the next state s_{t+1} with the probability $p(s_{t+1}|s_t, a_t)$, and gets a reward $r(s_t, a_t, g)$. ρ_0 is the initial state distribution, ρ_{g^*} is the evaluation goal distribution, and γ is the discount factor.

The learning algorithm $\mathbb{A}$ is defined: $\{s_i, a_i, s_{i+1}\}_{i=0}^{t-1} \mapsto (a_t, \pi_t)$, which maps the transitions collected until the time step t to the action a_t the agent should take in the non-episodic training and the best guess π_t of the optimal policy π^* on the evaluation goal distribution (ρ_{g^*}). In reset-free training the agent will only be reset to the initial state $s_0 \sim \rho_0$ a few times. The evaluation of agents is still episodic. The agent always starts from $s_0 \sim \rho_0$, and is asked to achieve $g \sim \rho_{g^*}$. The evaluation objective for a policy π is:

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0, g \sim \rho_{g^*}, a_j \sim \pi(\cdot|s_j, g), s_{j+1} \sim p(\cdot|s_j, a_j)}[\sum_{j=0}^{T} \gamma^j r(s_j, a_j, g)], \tag{3.1}$$

where T is the total time steps during the evaluation. The goal of algorithm $\mathbb{A}$ during the reset-free training is to minimize the performance difference $\mathbb{D}(\mathbb{A})$ of the current policy π_t and the optimal policy π^*:

$$\mathbb{D}(\mathbb{A}) = \sum_{t=0}^{\infty} (J(\pi^*) - J(\pi_t)). \tag{3.2}$$

In summary, the algorithm $\mathbb{A}$ should output an action a_t that the agent should take in the non-episodic data collection and a policy π_t that can maximize $J(\pi_t)$ at every time step t based on all previously collected data.

3.3.2 Model-based RL setup

Recent goal-conditioned MBRL approaches like LEXA (Mendonca et al., 2021) and PEG (E. S. Hu et al., 2023) train goal-conditioned policies purely using synthetic data generated by learned world models. Their robust exploration demonstrate significant success in solving long-horizon goal-conditioned tasks. In the reset-free setting, strong exploration is crucial, as the agent can no longer depend on episodic resets to bring it back to task-relevant areas if it gets stuck. Therefore, we select PEG as the backbone MBRL agent for its strong exploration abilities and sample efficiency.

PEG (E. S. Hu et al., 2023) is a model-based Go-Explore framework that extends LEXA (Mendonca et al., 2021), an unsupervised goal-conditioned variant of DreamerV2 (Hafner et al., 2020). The following components are parameterized by θ and learned:

$$\begin{aligned} &\text{world model: } \widehat{\mathcal{T}}_\theta(s_t|s_{t-1}, a_{t-1}) \\ &\text{goal conditioned policy: } \pi^G_\theta(a_t|s_t, g) \qquad \text{goal conditioned value: } V^G_\theta(s_t, g) \\ &\text{exploration policy: } \pi^E_\theta(a_t|s_t) \qquad \text{exploration value: } V^E_\theta(s_t) \end{aligned} \tag{3.3}$$

The world model is a recurrent state-space model (RSSM) which is trained to predict future states and is used as a learned simulator to train the policies and value functions. The goal-conditioned policy π^G_θ is trained to reach random states sampled from the replay buffer. The exploration policy π^E_θ is trained on an intrinsic motivation reward that rewards world model error, expressed through the variance of an ensemble (Sekar et al., 2020). Both policies are trained on simulated trajectory rollouts in the world model.

▶ **Self-supervised Goal-reaching Reward Function:** Rather than assuming access to the environmental reward, PEG learns its own reward function. PEG uses a dynamical distance function (Hartikainen et al., 2019) as the reward function within world models, which predicts the number of actions between a start and goal state. The distance function is trained on random state pairs from imaginary rollouts of π^G_θ. π^G_θ is then trained to minimize the dynamical distance between its states and commanded goal state in imagination. See Mendonca et al., 2021 for more details.

▶ **Phased Exploration via Go-Explore:** For data-collection, PEG employs the Go-Explore strategy. In the "Go"-phase, a goal is sampled from some goal distribution ρ. The goal-conditioned policy, conditioned on the goal is run for some time horizon H_G, resulting in trajectory τ_g.

Then, in the "Explore"-phase, starting from the last state in the "Go"-phase, the

exploration policy is run for H_E steps, resulting in τ_e. The interleaving of goal-conditioned behavior with exploratory behavior results in more directed exploration and informative data. This in turn improves accuracy of the world model, and the policies that train inside the world model. See Algorithm 1 and Algorithm 2 for pseudocode. The choice of goal distribution ρ is important for Go-Explore. In easier tasks, the evaluation goal distribution ρ_{g^*} may be sufficient. But in longer-horizon tasks, evaluation goals may be too hard to achieve. Instead, intermediate goals from an exploratory goal distribution ρ_E can help the agent explore. We choose PEG, which generates goals by planning through the world model to maximize exploration value (see E. S. Hu et al., 2023 for details).

3.4 Method

As motivated in Section 3.1 and Figure 3.1, the direct application of PEG to the reset-free setting shows promising performance but suffers from over-exploration of task-irrelevant states. To adapt model-based RL to the reset-free setting, we introduce MoReFree, a model-based approach that improves PEG to handle the lack of resets and overcome the over-exploration problem. MoReFree improves two key mechanisms of MBRL for reset-free training: exploration and policy training.

Algorithm 1 Go-Explore

Input: $g, \pi_\theta^G, \pi_\theta^E$
$\tau_g \leftarrow \{\}; \tau_e \leftarrow \{\}$
for $t = 1$ to H_G **do**:
  $a_t \sim \pi_\theta^G(\cdot\,|s_t, g)$
  $s_{t+1} \sim \mathcal{T}(\cdot\,|s_t, a_t)$
  $\tau_g \leftarrow \tau_g \cup \{s_t\}$
for $t = 1$ to H_E **do**:
  $a_t \sim \pi_\theta^E(\cdot\,|s_t)$
  $s_{t+1} \sim \mathcal{T}(\cdot\,|s_t, a_t)$
  $\tau_e \leftarrow \tau_e \cup \{s_t\}$
return τ_g, τ_e

3.4.1 Back-and-Forth Go-Explore

First, we introduce MoReFree's procedure for collecting new datapoints in the real environment. PEG (E. S. Hu et al., 2023) already has strong goal-conditioned exploration abilities, but was developed for solving episodic tasks. Without resets, PEG's Go-Explore procedure can undesirably linger in unfamiliar but task-irrelevant portions of the state space. This generates large amounts of uninformative trajectories, which in turn degrades world model learning and policy optimization.

MoReFree overcomes this by periodically directing the agent to return to the states relevant to the task (i.e. initial and evaluation goals). We call this exploration procedure "Back-and-Forth Go-Explore", where we sample pairs of initial and evaluation goals and ask the agent to cycle back and forth between the goal pairs, periodically

Algorithm 2 MBRL Backbone

Input: π_θ^G, π_θ^E, world model $\widehat{\mathcal{T}}_\theta$, goal distribution ρ (including: exploratory goal distribution ρ_E, evaluation goal distribution ρ_{g^*}, initial state distribution ρ_0)
$\mathcal{D} \leftarrow \{\}$
while within the reset-free horizon **do**:
  # reset-free PEG
  sample a goal $g \sim \rho_E$
  $\tau_g, \tau_e \leftarrow$Go-Explore$(g, \pi_\theta^G, \pi_\theta^E)$
  # MoReFree
  $\tau_g, \tau_e \leftarrow$Back-and-Forth Go-Explore$(\pi_\theta^G, \pi_\theta^E, \rho)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \tau_g \cup \tau_e$
  update $\widehat{\mathcal{T}}_\theta$ with $\mathcal{D}$
  update π_θ^E with $\widehat{\mathcal{T}}_\theta$ in imagination
  update π_θ^G with $\widehat{\mathcal{T}}_\theta$ in imagination, cond. on goals g':
  $g' \sim \mathcal{D}$
  $g' \sim \mathcal{D}$ with $\Pr = 1 - \alpha$, $g' \sim \rho_{g*}, \rho_0$ with $\Pr = \alpha$

interspersed with exploration phases (see Figure 3.2 top row).

Now, we define the "Back-and-Forth Go-Explore" strategy as seen in Algorithm 3. First, we decide whether to perform initial / evaluation state directed exploration. With probability α, we sample goals (g^*, g_0) from ρ_{g^*}, ρ_0 respectively. Then, we execute the Go-Explore routine for each goal. We name Go-Explore trajectories conditioned on initial state goals as "Back" trajectories, and Go-Explore trajectories conditioned on evaluation goals as "Forward" trajectories. With probability $1 - \alpha$, we execute exploratory Go-Explore behavior by sampling exploratory goals from PEG. The difference between reset-free PEG and MoReFree can be seen in Algorithm 2, unlike PEG, MoReFree employs the "Back-and-Forth Go-Explore".

By following this exploration strategy, the agent modulates between various Go-Explore strategies, alternating between solving the task by pursuing evaluation goals, resetting the task by pursuing initial states, and exploring unfamiliar regions via exploratory goals.

3.4.2 Learning to Achieve Relevant Goals in Imagination

Next, we describe how MoReFree trains the goal-conditioned policy in the world model. To train π_θ^G, MoReFree samples various types of goals and executes $\pi_\theta^G(\cdot \mid \cdot, g)$ inside the world model to generate "imaginary" trajectories. The trajectory data is scored

using the learned dynamical distance reward mentioned in Section 3.3.2, and the policy is updated to maximize the expected return. This procedure is called imagination (Hafner et al., 2019), and allows the policy to be trained on vast amounts of synthetic trajectories to improve sample efficiency.

First, we choose to sample evaluation goals from ρ_{g^*} since the policy will be evaluated on its evaluation goal-reaching performance. Next, recall that Back-and-Forth Go-Explore procedure also samples initial states from ρ_0 as goals for the Go-phase to emulate resetting behavior. Since we would like π_θ^G to succeed in such cases so that the task is reset, we will also sample from ρ_0. Finally, we sample random states from the replay buffer to increase π_θ^G's ability to reach arbitrary states. The sampling probability for each goal type is set to $\alpha/2, \alpha/2, 1-\alpha$ respectively. In other words, MoReFree biases the goal-conditioned policy optimization procedure to focus on achieving task-relevant goals (i.e. evaluation and initial states), as they are used during evaluation and goal-conditioned exploration to condition the goal-reaching policy (see Figure 3.2 bottom row). This leads to additional changes of line 13 in Algorithm 2.

Algorithm 3 Back-and-Forth Go-Explore

1: **Input:** $\pi_\theta^G, \pi_\theta^E, \rho_{g^*}, \rho_0, \rho_E$
2: Generate a random number r in $[0, 1]$
3: **if** $r < \alpha$ **then**
4: $\quad g^*, g_0 \sim \rho_{g^*}, \rho_0$
5: $\quad \tau_{g^*}, \tau_e^1 \leftarrow$Go-Explore$(g^*, \pi_\theta^G, \pi_\theta^E)$
6: $\quad \tau_{g_0}, \tau_e^2 \leftarrow$Go-Explore$(g_0, \pi_\theta^G, \pi_\theta^E)$
7: $\quad \tau_g \leftarrow \tau_{g^*} \cup \tau_{g_0}; \tau_e \leftarrow \tau_e^1 \cup \tau_e^2$
8: **else**
9: $\quad g \sim \rho_E$
10: $\quad \tau_g, \tau_e \leftarrow$Go-Explore$(g, \pi_\theta^G, \pi_\theta^E)$
11: **end if**
12: **return** τ_g, τ_e

3.4.3 Implementation Details

Our work builds on the top of PEG (E. S. Hu et al., 2023), and we use its default hyperparameters for world model, policies, value functions and temporal reward function. We set the length of each phase for Go-Explore (H_G, H_E) to half the evaluation episode length for each task. We set the default value of $\alpha = 0.2$ for all tasks (never tuned). See Section 3.7.3 for more details and the supplemental for MoReFree code.

3.5 Experiments

We evaluate three MBRL methods (PEG (E. S. Hu et al., 2023), the extension reset-free PEG and our proposed method MoReFree) and four competitive reset-free base-

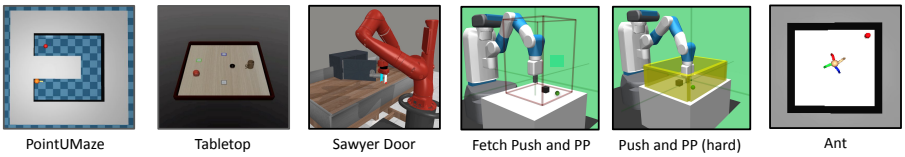


Figure 3.3: We evaluate MoReFree on eight reset-free tasks ranging from navigation to manipulation. PP is short for Pick&Place.

lines on eight reset-free tasks. We aim to address the following questions: 1) Do MBRL approaches work well in reset-free tasks in terms of sample efficiency and performance? 2) What limitations arise from running MBRL in the reset-free setting, and does our proposed solution MoReFree address them? 3) What sorts of behavior do MoReFree and baselines exhibit in such tasks, and are our design choices for MoReFree justified?

Baselines: All baselines except for R3L are implemented using official codebases, see Section 3.7.2 for details.

- **PEG** (E. S. Hu et al., 2023) is the original episodic PEG in which exploratory goals are only sampled once at the beginning of each episode (in the reset-free setting, the episode is extremely long). The goal-conditioned policy and the exploration policy are then executed for the first half and second half of the episode, respectively.

- **reset-free PEG** is a straightforward extension of PEG to the reset-free setting. Exploratory goals are sampled every $H_G + H_E$ steps. Then, the goal-conditioned policy is executed for H_G steps followed by the exploration policy being executed for H_E steps.

- **MEDAL** (Sharma et al., 2022) requires demonstrations and trains two policies, one for returning to demonstration states and another that achieves task goals.

- **IBC** (J. Kim et al., 2023) is a competitive baseline that outperforms prior reset-free work (e.g. MEDAL, VaPRL) by defining a bidirectional curriculum for the goal-conditioned forward and backwards (i.e. reset) policies trained using the environmental reward.

- **R3L** (Zhu et al., 2020) trains two policies, one for achieving task goals and another that perturbs the agent to novel states. Notably, it is the only baseline that operates without any additional assumptions (i.e. environmental rewards, demonstrations, and resets).

- **Oracle** is SAC (Haarnoja, Zhou, Hartikainen, et al., 2018) trained under the episodic setting on the environmental reward.

Note that most baselines enjoy some advantage over two MBRL methods: MEDAL, IBC and Oracle use ground truth environmental reward, while MEDAL also uses demonstrations and Oracle uses resets. See Table 3.1 for a conceptual comparison between MoReFree and prior work.

Environments: We evaluate MoReFree and baselines on eight tasks (see Figure 3.3). We select five tasks from IBC's evaluation suite of six tasks; (Fetch Reach is omitted because it is trivially solvable). Next, we increased the complexity of the two hardest tasks from IBC, Fetch Push and Fetch Pick&Place, by extending the size of the workspace, replacing artificial workspace limits (which cause unrealistic jittering behavior near the limits, see the website for videos) with real walls, and evaluating on harder goal states (i.e. Pick&Place goals only in the air rather than including ones on the ground). In addition, we contributed a difficult locomotion task, Ant, which is adapted from the PEG codebase (E. S. Hu et al., 2023). All methods are run with 5 seeds, and the mean performance and standard error are reported. During the evaluation, the performance on tasks with randomly sampled goals from ρ_{g^*} is measured by averaging over 10 episodes. See Section 3.7 for more experimental details.

3.5.1 Results

As shown in Fig 3.4, two reset-free model-based methods (MoReFree and reset-free PEG), without demonstrations or access to environmental reward, outperform other baselines with privileged access to supervision in both final performance and sample efficiency in 7/8 tasks. We observe that the two reset-free MBRL methods learn good behaviors: the pointmass agent hugs the wall of the UMaze to minimize travel time and the Fetch robot deftly pushes and picks up the block into multiple target locations. MoReFree is always competitive with or outperforms reset-free PEG, with large gains in the 3 hardest tasks: Push (hard) by 45%, Pick&Place (hard) by 13% and Ant (hard) by 36%. We observe that MoReFree learns non-trivial reset behaviors such as picking and pushing blocks back into the center of the table for the hard variants of the Fetch manipulation tasks. However, the original PEG performs poorly, suggesting that directly applying episodic MBRL methods in a reset-free setting without adaptations yields suboptimal results. See the website for videos of MoReFree and baselines.

In many tasks, the baselines fail to learn at all. We believe this is due the low sample budget, which may be too low for the baselines to fully explore the environment

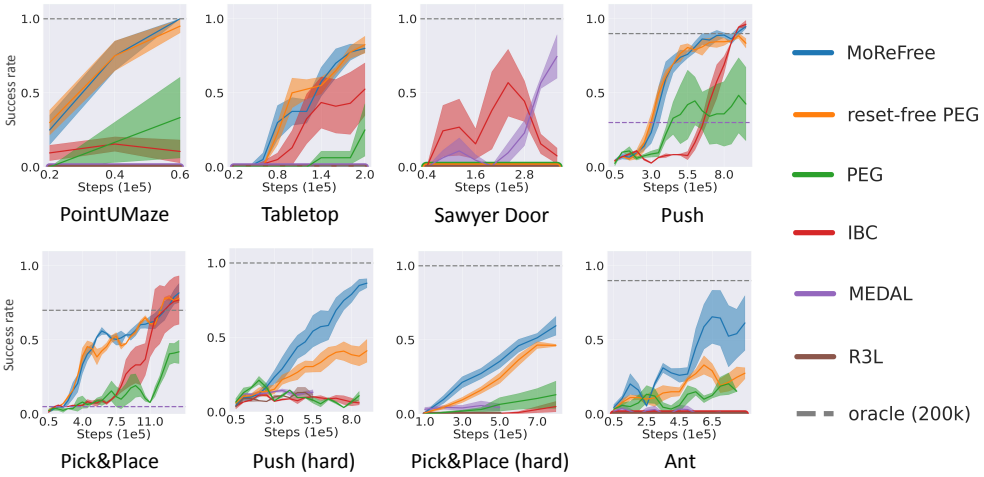


Figure 3.4: Two reset-free MBRL methods (MoReFree and reset-free PEG) significantly outperform baselines in 7/8 tasks. However, directly applying PEG works poorly. In 4 tasks, only MBRL methods are able to learn meaningful behavior, showcasing MBRL's sample efficiency in the reset-free setting. MoReFree outperforms reset-free PEG in the 3 most difficult tasks.

and learn the proper resetting behaviors necessary to train the actual task policy. In Section 3.11, we increased the training budget by 3× for the IBC baseline and it still fails, underscoring the difficulty of the tasks and the sample-efficiency gains of MoReFree and MBRL. On the other hand, we noticed that one environment, Sawyer Door, seemed particularly hard for MBRL agents to solve. We hypothesize that the dynamics of the task are hard to model, resulting in performance degradation for model-based approaches (see Section 3.10 for more analysis).

3.5.2 Analysis

To explain the performance differences between MoReFree and baselines, we closely analyze the exploration behaviors.

MoReFree focuses on task-relevant states. In Figure 3.5 we visualize the state visitation heatmaps of methods in various environments, and also compute the percentage of "task-relevant" states (initial and goal regions, highlighted with white borders). We highlight two trends. First, the heatmaps show that MoReFree and reset-free PEG explore thoroughly while baselines have more myopic exploration patterns, as seen in the Ant heatmaps at the top.

Next, performance differences between reset-free PEG and MoReFree are intu-

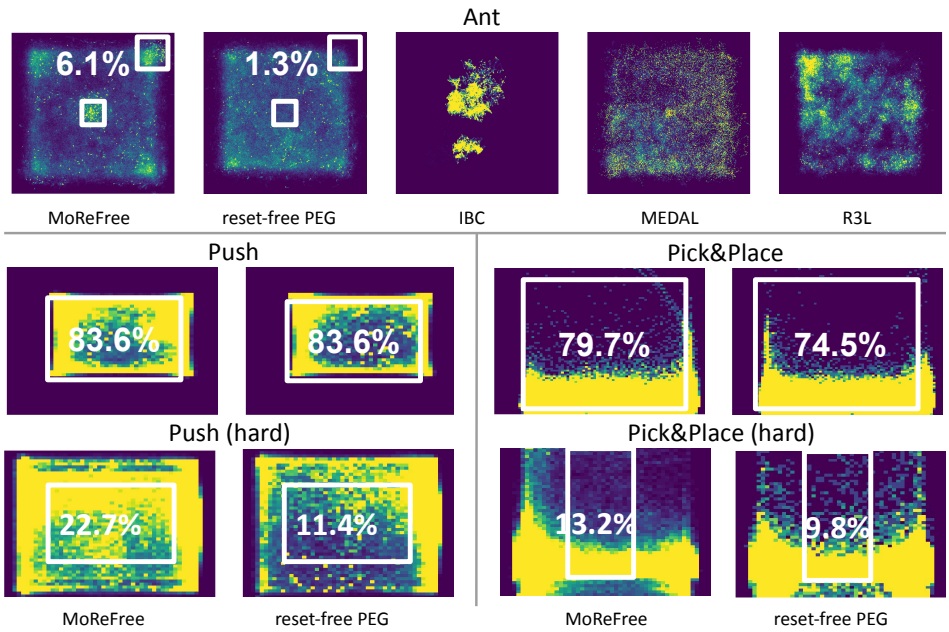


Figure 3.5: State visitation heatmaps of different agents. White areas are task-relevant states (including initial and goal state distributions) and we overlay the percentages of task-relevant states. reset-free MBRL methods explore more and in harder environments, MoReFree experiences more task-relevant states.

itively explained by the amount of task-relevant data collected by each agent. In easier environments like Push or Pick&Place where both reset-free PEG and MoReFree encounter similar amounts of task-relevant states, the performance is roughly similar between reset-free PEG and MoReFree. But in harder environments (Ant, Push (hard), Pick&Place (hard)) with larger state spaces and more complicated resetting dynamics, MoReFree collects $1.3 - 5\times$ more task-relevant data and has large performance gains over reset-free PEG. By experiencing more task-relevant states and training policies on them in imagination, MoReFree policies are more suited towards succeeding at the episodic evaluation criteria. See Section 3.8 for additional visualizations.

MoReFree effectively resets. Next, we investigate the qualitative behavior of MoReFree's Back-and-Forth Go-Explore. To see if "Back" trajectories help free the agent from the sink states, we analyze the replay buffer of MoReFree for the environments, and plot the starting locations of the agent / object up to 100 timesteps before a successful "Back" trajectory is executed in Figure 3.6. The color intensity of the dots correspond to state density over the last 100 steps (i.e. dark red means the

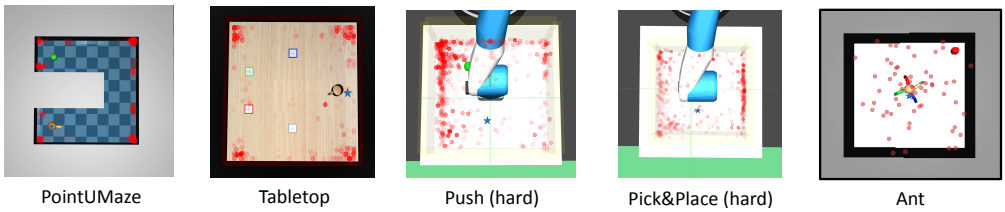


Figure 3.6: We visualize the start position (red dots) of successful "Back" trajectories of MoReFree's Back-and-Forth Go-Explore, where π_θ^G is directed to reset the environment. The color intensity of the dots correspond to state density over the last 100 steps.

agent / object has rested there for a while). We observe that the starting locations (red dots) of the agent / object are in corners or next to walls in all environments. This suggests that these areas act as sink states, where the agent / object would remain for long and waste time. We observe that MoReFree learns reset behaviors like picking the block out of corners and walls in Fetch Push and Fetch Pick&Place. See detailed videos of the reset behavior on the website.

3.5.3 Ablations

To justify our design choices, we ablate the two mechanisms of MoReFree, the back-and-forth exploration and task-relevant goal-conditioned policy training, and plot the results in Figure 3.7. First, removing all mechanisms (**MF w/o Explore & Imag.**) reduces to reset-free PEG, and we can see a large gap in performance. Next, **MF with Only Task Goals** sets $\alpha = 1$, which causes an extreme bias towards task-relevant states in the exploration and policy training. This also degrades performance, due to the need for strong exploration in the reset-free setting. Examinations of more values for α can be found in Section 3.7.3.

Finally, we isolate individual components of MoReFree. First, we disable Back-and-Forth Go-Explore by disallowing the sampling of initial or evaluation goals during Go-Explore. Only exploratory goals are used in Go-Explore for this ablation (named **MF w/o BF-GE**). Next, in **MF w/o Imag.** we turn off the initial / evaluation goal sampling in imagination, so only random replay buffer goals are used to train π_θ^G. We see that both variants perform poorly. This is somewhat intuitive, as the two components rely on each other. Having both forms a synergistic cycle where 1) the goal-conditioned policy' s optimization is more focused towards reaching initial / goal states, and 2) the exploration is biased towards reaching initial / goal states by using the goal-conditioned policy we just optimized in step 1. If we remove one without the

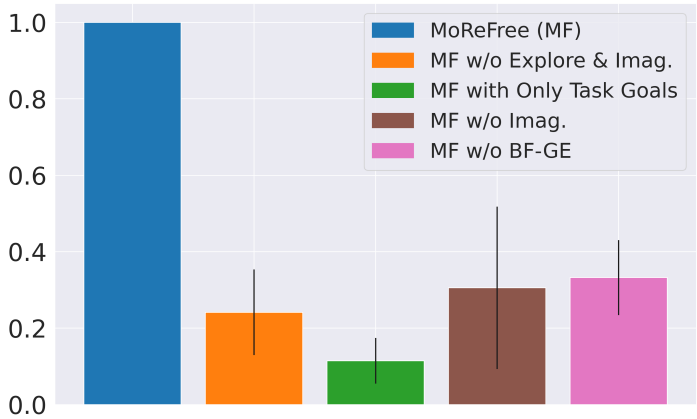


Figure 3.7: Ablations on 5 variants of MoReFree over 3 hard environments, Push (hard), Pick&Place (hard) and Ant, with normalized final performance.

other, then the cycle breaks down. In **MF w/o Imag.**, Back-and-Forth Go-Explore will suffer since π_θ^G trained on random goals cannot reliably reach initial / evaluation goals. In **MF w/o BF-GE**, the exploration strategy will not seek initial / evaluation states, resulting in an inaccurate world model and degraded policy optimization. In summary, the ablations show that MoReFree's design is sound and is the major factor behind its success in the reset-free setting. See Section 3.9 for details.

3.6 Conclusion and Future Work

As a step towards reset-free training, we adapt model-based methods to the reset-free setting and demonstrate their superior performance. Specifically, we show that with minor modifications, unsupervised MBRL method substantially outperforms the state-of-the-art model-free baselines tailored for the reset-free setting while being more autonomous (requires less supervision like environmental reward or demonstrations). We then identify a limitation of unsupervised MBRL in the reset-free setting (over-exploration on task-irrelevant states), and propose MoReFree to address such limitations by focusing model-based exploration and goal-conditioned policy training on task-relevant states. We conduct a through experimental study of MoReFree and baselines over 8 tasks, and show considerable performance gains over the MBRL baseline and prior state-of-the-art reset-free methods. Despite its overall success, MoReFree is not without limitations. Being a model-based approach, it inherits all associated disadvantages. For example, we believe Sawyer Door is a task where learning the

dynamics is harder than learning the policy (see Section 3.10), disadvantaging MBRL approaches. Next, MoReFree uses a fixed percentage of task-relevant goals for exploration and imagination, whereas future work could consider an adaptive curriculum. Finally, scaling MoReFree to high-dimensional observations would be a natural extension. We hope MoReFree inspires future efforts in increasing autonomy in RL.

3.7 Experimental Details

3.7.1 Environments

PointUMaze: The state space is 7D and the action space is 2D. The initial state is $(0, 0)$, which located in the bottom-left corner, and noise sampled from $\mathcal{U}(-0.1, 0.1)$ is added when reset. The goal during the evaluation is always located in at the top-left corner of the U-shape maze. The maximum steps during the evaluation is 100. Hard reset will happen after every $2e5$ steps. In the whole training process we performed, it only reset once at the beginning of the training. Taken from the IBC (J. Kim et al., 2023) paper.

Tabletop: The state space is 6D, and the action space is 3D. During the evaluation, four goal locations are sampled in turn, the initial state of the agent is always fixed and located in the center of the table. The maximum steps during the evaluation is 200. Hard reset will happens after every $2e5$ steps. In the whole training process we performed, it only reset once at the beginning of the training. Taken from the EARL (Sharma, Xu, et al., 2021) benchmark and also used in the IBC paper.

Sawyer Door: The state space is 7D and the action space is 4D. The position of door is initialized to open state (60 degree with noise sampled from $(0, 18)$ degree) and the goal is always to close the door (0 degree). The arm is initialized to a fixed location. Maximum number of steps is 300 for the evaluation. Hard reset will happen after every $2e5$ steps. In the whole training process we performed, it resets twice. Taken from the EARL (Sharma, Xu, et al., 2021) benchmark and also used in the IBC paper.

Fetch Push and Pick&Place: The state space is 25D and action space is 4D. These are taken from the IBC paper. Authors converted the original Fetch environments to a reversible setting by defining a constraint on the block position. The initial and goal distributions are identical to the original Fetch Push and Pick&Place. More details can be found in the IBC paper.

Push (hard): Different from the original Fetch Push task, in our case walls are

added to prevent the block from dropping out of the table. The workspace of the robot arm is also limited. The block is always initialized to a fixed location, and goal distribution during the evaluation is $\mathcal{U}(-0.15, 15)$. Fetch Push used in the IBC paper, the block is limited by joint constraint, which shows unrealistic jittering behaviors near the limits (we observe such phenomenon by running model-based go-explore, the exploration policy prefers to always interact with the block and keep pushing it towards the limit boundary, see videos on our project website [1]). Meanwhile, the gripper is blocked, which makes the task easier. In our case, we release the gripper and it can now open and close again which add two more dimension of the state space. We found it is important to release the gripper in our version of Push task, when the block is in corners, it will need to operate the gripper to drag the block escape from corners. The maximum steps the agent can take in 50 during the evaluation. Hard reset will happen after every $1e5$ steps. In the whole training process we performed, it resets 5 times in total.

Pick&Place (hard): We add walls in the same way as we did for Push (hard). We make it more difficult by only evaluating the agent on goals that are in the air. Then it has to learn to perform picking behavior properly, whereas goals on the ground can just be solved by pushing. The goal will be uniformly sampled from a $5 \times 5 \times 10$ cm cubic area above the table. It has the same observation space, action space, initial state and maximum steps with Fetch Push described above. Hard reset will happens after every $1e5$ steps. In the whole training process we performed, it resets 5 times in total. See the visual difference between our Pick&Place and IBC's in Figure 3.3. Since the workspace of the robot is limited within the walls as well in Push (hard) and Pick&Place (hard), when the block gets stuck in corners, the robot needs to precisely move to the corner and bring the block back. In contrast, the robot in IBC's version can move to everywhere, being able to create various circumstance to solve such difficult position.

Ant: We adapt the AntMaze task from environments[2] codebase of PEG and change the shape of the maze to square, also change the evaluation goal distribution to be a uniform distribution $\mathcal{U}(2, 3)$ for both x and y location, which lies on the top-left corner of the square. The ant is always initialized to the center point (0, 0) of the square to start from, with uniform noise ($\mathcal{U}(-0.1, 0.1)$) added. The state space is 29D and the action space is 8D. The maximum steps for evaluation is 500. Hard reset will happen after every $2e5$ steps. In the whole training process we performed, it reset 4

[1] https://sites.google.com/view/morefree
[2] https://github.com/edwhu/mrl

times in total.

3.7.2 Baseline Implementations

PEG: We use the official implementation of PEG[3] and only optimize the exploratory goal distribution once at the beginning of each reset-free training episode, i.e. H_G and H_E are set to half of the reset-free episode length.

reset-free PEG: We extend the official implementation of PEG[4] to reset-free setting by 1) set H_G and H_E to half of the evaluation episode length; 2) optimizing the goal distribution every $H_G + H_E$ steps; 3) keeping all other hyperparameters the same as MoReFree.

IBC: We use the official implementation from authors[5] and keep hyperparameters unchanged.

MEDAL: We follow the official implementation of MEDAL[6] and use the deafult setting for experiments. Since MEDAL requires demonstrations, for tasks from EARL benchmark, demonstrations are provided. For other environments, we generate demonstrations by executing the final trained MoReFree to collect data. 30 episodes are generated for each task.

R3L: We implement R3L agent by modifying the FBRL agent from MEDAL codebase. The backward policy is replaced by an exploration policy trained using the random network distillation (RND) objective (Burda et al., 2018). The RND implementation we follow is from DI-engine[7].

Oracle: This is a episodic SAC agent, we use the implementation from MEDAL codebase and keep all the hyper-parameters unchanged.

MoReFree: Our agent is built on the model-based go-explore method PEG (E. S. Hu et al., 2023), we extend their codebase by adding back-and-forth goal sampling procedure and training on evaluation initial and goal states in imagination goal-conditioned policy training. See our codebase in the supplemental.

3.7.3 Hyperparameters

Train ratio (i.e. Update to Data ratio) is an important hyper-parameter in MBRL. It controls how frequently the agent is trained. Every n steps, a batch of data is sampled

[3] https://github.com/penn-pal-lab/peg
[4] https://github.com/penn-pal-lab/peg
[5] https://github.com/snu-larr/ibc_official
[6] https://github.com/architsharma97/medal
[7] https://opendilab.github.io/DI-engine/12_policies/rnd.html

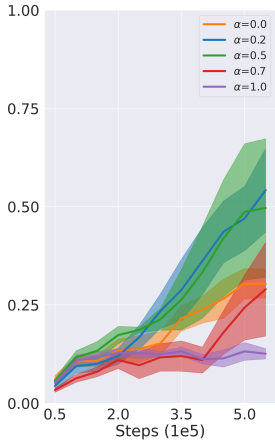


Figure 3.8: Performance of MoReFree with different values of α in Push (hard).

from the replay buffer, the world model is trained on the batch, and then policies and value functions are trained in imagination. In all our experiments, we only vary n on different tasks. See the table below for different values on different tasks we used through experiments. MoReFree also introduces a new parameter α, which we keep $\alpha = 0.2$ for all tasks and did not tune it at all. All other hyperparameters we keep the same as the original code base.

Table 3.2: Different train ratio we used for different tasks. We keep all other hyperparameters the same as default ones.

PointUMaze	2	Sawyer Door	5	Tabletop	1	Fetch Push	2
Fetch Pick&Place	2	Push (hard)	2	Pick&Place (hard)	2	Ant	2

Different values for α. We examine different values of α in MoReFree on Fetch Push task, which affects how much MoReFree focuses on task-relevant goals in exploration and imagination. In Figure 3.8, we see that introducing a moderate amount of task-relevant goals (α=0.2, α=0.5) results in sensible performance, while too many task-relevant goals (α=0.7, α=1.0) degrades performance. We use the same value of alpha, 0.2, across all tasks, which showcases MoReFree ' s consistency.

3.7.4 Results Clarification

In Push and Pick&Place results, we retrieved the final performance of MEDAL directly from the IBC paper (dashed purple lines) and did not have time to run R3L in these two environments. R3L is shown to be a lot worse than MEDAL in the MEDAL paper and performs obviously bad in other tasks shown in Figure 3.4. In Push (hard) and Pick&Place (hard), we ran R3L and MEDAL with less budget since other methods clearly outperform and their learning curves do not show any evidence for going up.

3.7.5 Resource Usage

We submit jobs on a cluster with Nvidia 2080, 3090 and A100 GPUs. Our model-based experiments take 1-2 days to finish, and the model-free baselines take half day to one day to run.

3.8 More Visualizations on Replay Buffer

We visualize the replay buffer of different agents on more tasks. See Figure 3.9 for XY location of the mug in Tabletop, Figure 3.11 for XY location data of the agent in PointUMaze, Figure 3.10 for XZ location of the block in Pick&Place (hard) and Figure 3.12 for XY location data of the block in Push (hard) and Pick&Place (hard). Overall, we see MoReFree explores the whole state space better. Meanwhile, due to back-and-forth procedure, MoReFree collects more data near initial / goal states, which are important for the evaluation. However, IBC, MEDAL, R3L and Oracle all fail to explore well; their heatmaps are mostly populated with low visitation cells.

3.9 Detailed Ablations

We report learning curves for each variant agent we ablate in Section 3.5.3 on every task in Figure 3.13. Since MoReFree does not learn at all in Saywer Door task, we exclude the ablation for it. In each task, MoReFree is better or on par with all other ablations. Through learning curves, we see different components contribute differently on different tasks.

We further analyze the ablation on PointUMaze as an example by visualizing the replay buffer of different variants, see Figure 3.14. In the performance on PointUMaze from Figure 3.13, sampling exploratory goals for data collection is important (MF w/o Explore & Imag. outperforms other ablations). But we see in 3.14, MF w/o Explore

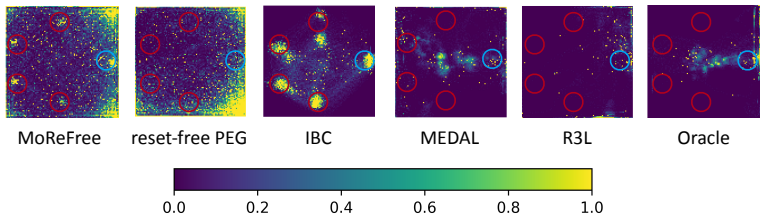


Figure 3.9: XY state visitation heatmap of the mug in Tabletop of various approaches. MoReFree' s heatmap shows high state diversity while retaining high visitation counts near the task-relevant states (red circles are goal states, the blue circle is the initial state). reset-free PEG also shows diverse exploration, but it over-explores the bottom-right corner which is entirely task-irrelevant. IBC's bi-directional curriculum leads the exploration shuttles between the initial state and goal states, but fails to explore well. All other methods fail to explore, visited states mostly cluster in few spots.

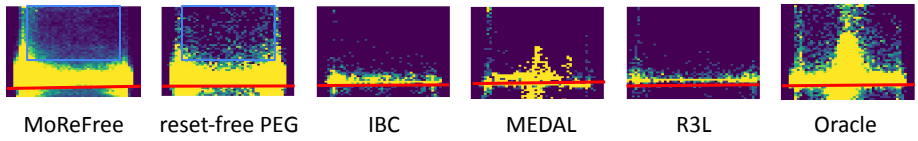


Figure 3.10: XZ state visitation heatmap of the block in Pick&Place (hard). States above the red line are in the air, which are crucial for solving the picking task. Two MBRL methods collect more data diversely in the air, while other reset-free methods barely pick up the block.

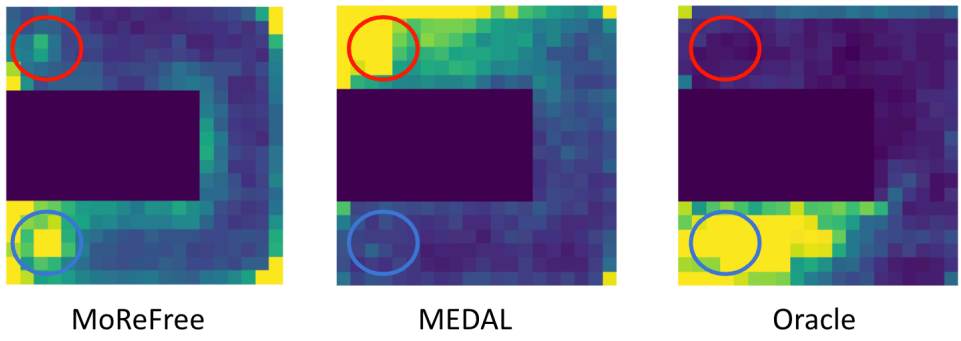


Figure 3.11: State visitation heatmap on point maze. MoReFree has special focuses on both initial state (blue circles) corner and goal state (red circles), while explore much uniformly. MEDAL collects lots of data near the goal state and little data on the initial state. Both MEDAL and Oracle explore less extensively.

& Imag. does not have focus on the initial / goal state which we care about for the evaluation, which makes it slightly worse than MoReFree. MF with Only Task Goals has a strong preference on initial / goal state, we think it is because in the later phase of

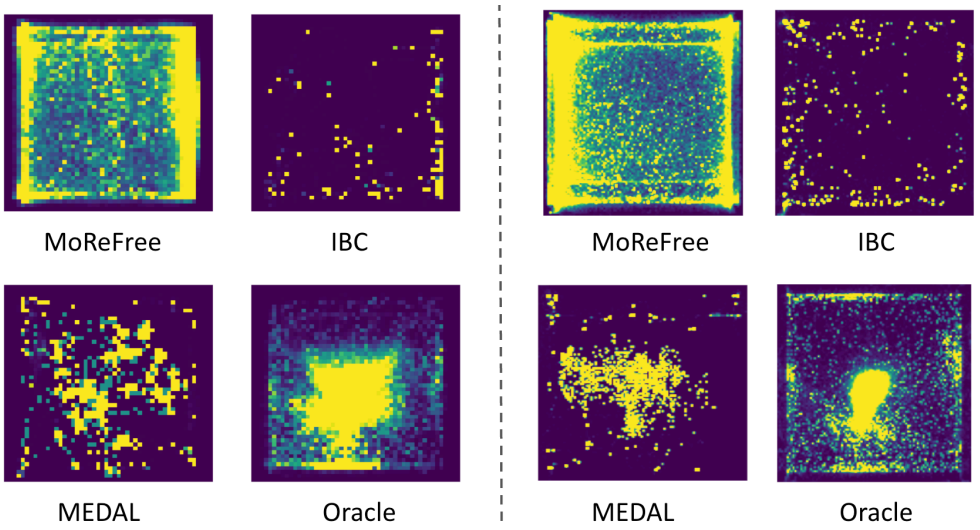


Figure 3.12: Block state visitation heatmap on Fetch Push (left) and Fetch Pick&Place (right) of different agents. MoReFree better explores the whole state space, while IBC and MEDAL do not have too much interactions with the block, thus lighted areas are scattered everywhere.

the training when the agent is able to solve the task, it goes back-and-forth consistently to collect data. But in the early phase of the training, it might lack exploration which causes the degraded performance compare with MoReFree. MF w/o Explore and MF w/o Imag. only either go to initial / goal state for data collection and do not practice on it during the imagination training, or practice without really going, which both does not form the positive cycle, and end up with poor performance.

3.10 MBRL on Sawyer Door

We investigate why two MBRL methods fail on Sawyer Door tasks. Note that MoReFree is able to solve intermediate goals such as closing the door in some angles, but is unable to solve the original IBC evaluation goal (see website for more videos).

We simplify Sawyer Door task by limiting the movement range of the robot to a box and also having a block holds the door to prevent it from opening it too much, see Figure 3.15. Although MBRL methods are trained on the simplified environment, we see learning curves on Sawyer Door are completely flat in Figure 3.4, compared with other baselines trained on the original task. We wonder why MBRL methods can show the same performance and gain benefits as it does in other environments.

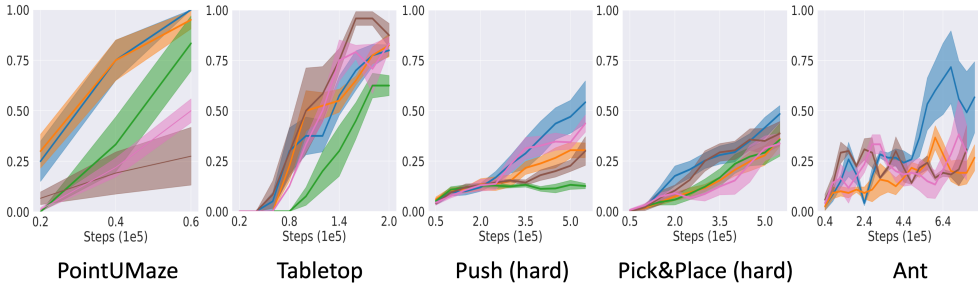


Figure 3.13: Learning curves of ablation study on 5 tasks. We see different components contribute differently in different tasks. For instance, in Tabletop, **MF w/o Imag.** even performs better than MoReFree, maybe because the whole state space can be explored quickly, then randomly sampling states from the replay buffer as goals for training already has good coverage on evaluation initial / goal states.

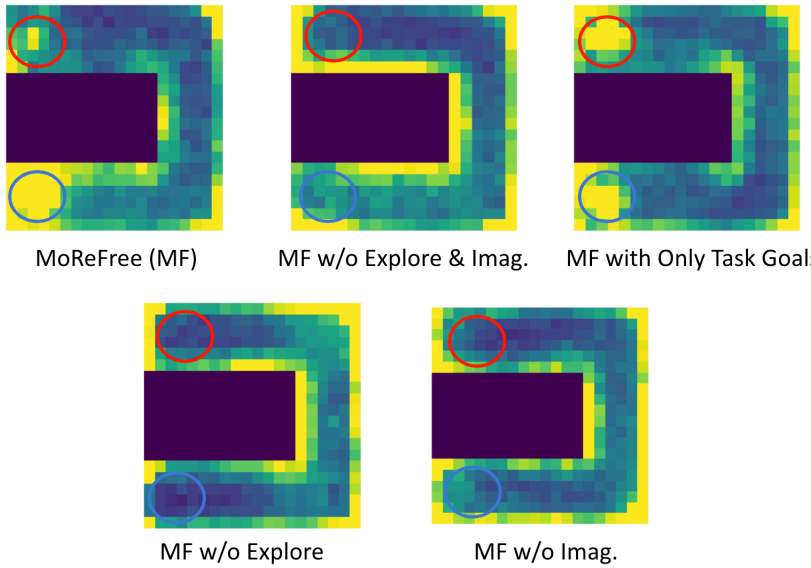


Figure 3.14: State visitation heatmap on PointUMaze task of all ablations. Red circles are evaluation goal states and blues are initial states. We see MoReFree collect good amount of data near initial / goal states while stronger exploration. MF w/o Explore and MF w/o Imag. could not gather task-relative data, which further causes poor performance.

MoReFree and reset-free PEG use DreamerV2 as backbone agents and extend it to reset-free settings. We hypothesize that Dreamer itself, even under the episodic setting with task reward function, would not work well. If that's the case, then MBRL methods in the reset-free setting with self-supervised reward function would almost certainly not work either. For example, if the backbone agent cannot model the

dynamics precisely, then both policy learning and dynamical distance reward learning, will be degraded.

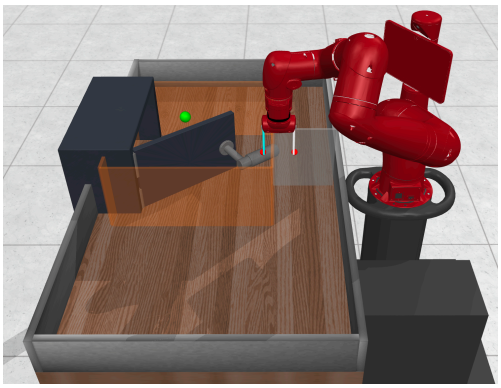


Figure 3.15: Simplified version of Sawyer Door. Orange walls show the limited workspace for the robot arm, and a grey wall is added to limit the movement of the door. The door can only move to maximum 60 degrees.

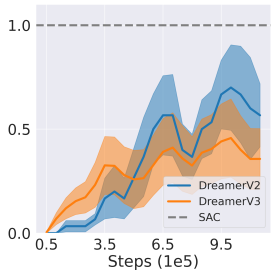


Figure 3.16: Performance of DreamerV2 and V3 on episodic Sawyer Door task. SAC can solve the task in 200k steps, while after 1 million steps MBRL is still not able to steadily solve the task.

We then run the underlying MBRL backbones under the episodic setting. Figure 3.16 shows DreamerV2 [8], and Dreamerv3 [9] struggle to solve the task, while model-free method SAC can steadily solve the task after 200k steps. This might be a potential reason that MBRL methods do not work on the more difficult reset-free setting. We hypothesize that the combination of the sparse environmental reward and dynamics of the door result in a hard prediction problem for world modelling approaches. We leave further investigation for the future work.

[8] https://github.com/danijar/dreamerv2
[9] https://github.com/danijar/dreamerv3

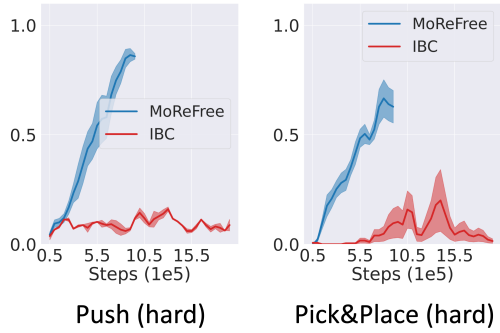


Figure 3.17: Longer training of IBC in our Fetch tasks, where the state space is larger and artificial constraints are replaced with surrounded walls. IBC still can not learn meaningful behaviors.

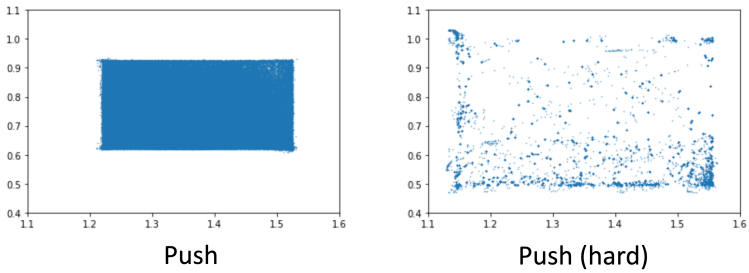


Figure 3.18: XY location of the block collected by IBC on Push (hard) and its original version (Push). IBC covers the whole state space very well in Push while fails in Push (hard), where the block stays for long time in corners or areas next to walls.

3.11 More Analysis on Fetch Environments

Although IBC gains good final performance in Push and Pick&Place, it starts learning late compared with MBRL methods and fails entirely in our harder versions. We suspect IBC might need more computational budget to start learning in harder tasks. Thus we train IBC with two millions environment steps and results in Figure 3.17 show that it still fails to solve the harder version of Push.

Figure 3.18 shows 600k data of the obejct (XY view) collected by IBC on our Push (hard) and IBC's Push. We see the block stays in corners or next to walls a lot in Push (hard), while goes everywhere and covers the whole space in IBC's Push, indicating object interaction is more difficult in Push (hard) due to the larger state space, surrounded walls and limited work space. In IBC's Push, the block can bounce back when it hits the limit of joint constraints. However, in Push (hard), the

block needs to be explicitly brought back from the corner or walls, requiring more sophisticated behaviors. Meanwhile, larger size of the limited area (our version is 3× larger than IBC's.) also increases the difficulty of the task.

3.12 Analysis on R3L

R3L trains two policies, one for reaching the goal and another that brings the agent to novel states. The goal-reaching policy is trained using a learned classifier to classify the goal state and other states. Original R3L takes images as inputs, thus the trained classifier can successfully classify goal images from random state images. In our work, we use low-dimensional state input. Outputs of the trained classifier on the whole state space of PointUMaze is shown in Figure 3.19. We see that the classifier learns to output higher values for states close to the goal state (red dot) and lower values for states further away. Nonetheless, due to the smoothness of the output scope, states near the initial state (blue circle) that are numerically closer but spatially further to the goal state also have higher values. R3L agent trained using such reward function will always tend to follow states with higher values to the corner instead of going forward. See the website for more videos. These trained reward functions are misleading for learning reasonable policies which result in poor performance we see in Figure 3.4.

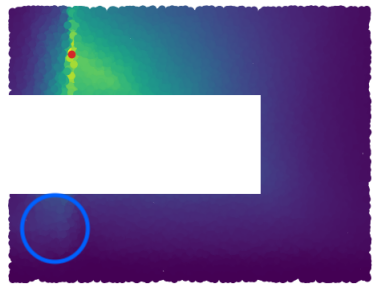


Figure 3.19: Outputs of the learned classifier on the whole state space. Due to the smoothness of the output scope, states near the initial state (blue circle) also have higher values.

Chapter 4

Continous Episodic Control

Authors: Zhao Yang, Thomas Moerland, Mike Preuss, Aske Plaat
Published in IEEE Conference on Games (CoG), 21-24 August 2023, Boston, USA.

Abstract

Non-parametric episodic memory can be used to quickly latch onto high-rewarded experience in reinforcement learning tasks. In contrast to parametric deep reinforcement learning approaches in which reward signals need to be back-propagated slowly, these methods only need to discover the solution once, and may then repeatedly solve the task. However, episodic control solutions are stored in discrete tables, and this approach has so far only been applied to discrete action space problems. Therefore, this paper introduces **C**ontinuous **E**pisodic **C**ontrol (CEC), a novel non-parametric episodic memory algorithm for sequential decision making in problems with a continuous action space. Results on several sparse-reward continuous control environments show that our proposed method learns faster than state-of-the-art model-free RL and memory-augmented RL algorithms, while maintaining good long-run performance as well. In short, CEC can be a fast approach for learning in continuous control tasks. [1]

[1] Code can be found at https://github.com/yangzhao-666/cec.

4.1 Introduction

Deep reinforcement learning (RL) methods have recently demonstrated superhuman performance on a wide range of tasks, including Gran Turisma (Wurman et al., 2022), StarCraft (Vinyals et al., 2019), Go (Silver et al., 2017), etc. However, in these methods, the weights of neural networks are slowly updated over time to match the target predictions based on the encountered reward signal. Therefore, even if the agent finds the solution to the problem, the learning still can be fairly slow, and the agent may not be able to solve the problem in subsequent episodes. This is one of the reasons state-of-the-art RL methods generally require many interactions with the environment and substantial computational resources (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, et al., 2015; Silver et al., 2016). Non-parametric episodic memory (EM) is introduced to directly store high-rewarded experiences, enabling the agent to quickly latch onto these experiences in reinforcement learning problems. Compared to parametric deep RL methods, these methods only need to solve the task once, because the solution is stored in memory and can be quickly retrieved. This is especially helpful for tasks where the agent only gets a final reward once it succeeds, i.e. sparse reward problems (Ecoffet et al., 2021). Although parametric RL methods do perform well in the long run (due to its generalizability, optimality and ability of dealing with stochasticity), sometimes we prefer a quick and less computationally expensive solution to the problem (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016). The ability to quickly latch onto recent successful experiences is also clearly evident in nature, where scrub jays for example store food and directly remember the exact location using episodic memory (Clayton & Dickinson, 1998).

Episodic memory is a term that originates from neuroscience (Tulving, 1972), where it refers to memory that we can quickly recollect. In the context of RL, this concept has generally been implemented as a non-parametric (or semi-parametric) table that can be read from and written into rapidly. Information stored in the memory can then either be used directly for control (for action selection) or to generate training targets for parametric deep RL methods. When used for control (usually called episodic control), episodic memory stores state-action pairs and their corresponding episodic returns in memory and extracts the policy by taking actions with maximum returns. It is therefore conceptually similar to tabular RL methods (C. J. C. H. Watkins, 1989) but with a different update rule, where stored values are directly replaced with larger encountered ones instead of being gradually updated towards targets.

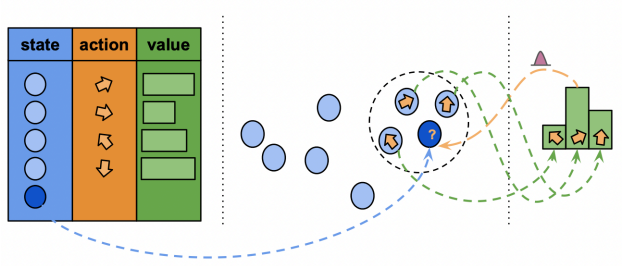


Figure 4.1: Conceptual illustration of the Continuous Episodic Control (CEC) algorithm. Left: We store a non-parametric table/buffer of state-action-value estimates obtained from previous episodes. Middle & Right: Action selection for a novel query state (dark blue circle) is performed by collecting its nearest neighbours in the buffer (dashed circle), and then sampling one of the actions of these neighbours proportional to their value (right). After possibly injecting exploration noise the selected action for the query state is returned.

Since current episodic control methods maintain a value estimate for each state-action pair (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016), they cannot naturally deal with continuous action spaces. One solution to this problem is to combine episodic memory with deep RL methods, where the episodic memory part generates training targets for a value network that is used to act (value-based approaches (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, et al., 2015)) or to guide act (actor-critic approaches (Lillicrap et al., 2015)) in the environment. The drawback of this approach is that we not only have to maintain extra memory, but again rely on the relatively slow deep RL agent for action selection and thereby performance is affected. The natural question that arises is: **can we directly use episodic memory for action selection in continuous control tasks, without relying on parametric RL methods?**

This work therefore introduces **C**ontinuous **E**pisodic **C**ontrol (CEC), an algorithm that uses episodic memory directly for action selection in tasks with a continuous action space. The main idea is based on the principle of generalisation: similar states generally require similar actions to obtain high returns. However, given a new query state, we 1) may not have that exact state in our buffer, and 2) we definitely only have value estimates for a few points in the continuous space of possible actions. We solve these issues by 1) specifying the nearest neighbour search in state space and 2) selecting actions proportionally to their value estimate, injected with Gaussian noise per action. The overall process of CEC is illustrated in Figure 4.1.

Experiments on a range of continuous control tasks show that our method outperforms state-of-the-art model-free and memory-augmented RL methods in various

continuous control problems. We also visualize how CEC manages to capture state-action values quickly and reliably in the beginning of the training, which results in steeper learning curves. In short, CEC seems to be a promising approach to learn quickly in continuous control tasks.

4.2 Background

In reinforcement learning (Sutton & Barto, 2018), an agent interacts with the environment. This process is formed as a Markov Decision Process (MDP) defined as the tuple $M = \langle S, A, P, R, \gamma \rangle$, where S is a set of states, A is a set of actions the agent can take, P specifies the transition dynamic of the environment, R is the reward function, and γ is the discount factor. At timestep t, the agent observes a state $s_t \in S$, and selects an action $a_t \in A$. The environment returns a next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and an associated reward $r_t = R(s_t, a_t, s_{t+1})$ after the action is executed. The agent selects actions according to a policy $\pi(\cdot|s_t)$ that maps a state to a distribution over actions. The goal is to learn a policy that can maximize the expected return: $\mathbb{E}_{s_{t+1}\sim P(\cdot|s_t,a_t),a_t\sim\pi(\cdot|s_t),r_t\sim R(\cdot|s_t,a_t)}[\sum_{t=0}^{T}\gamma^t \cdot r_t]$, where $\gamma \in [0, 1]$.

Define the state-action value $Q(s_t, a_t)$ as the expected return:

$$Q(s_t, a_t) = \mathbb{E}_{P,\pi,R}[\sum_{k=t}^{T}\gamma^k \cdot r_k],$$

when the agent starts from state s_t and action a_t at time-step t. Q-learning learns the optimal state-action value function by updating the current estimated value towards a target value (TD-target) following:

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_{a' \in A} \hat{Q}(s_{t+1}, a') - \hat{Q}(s_t, a_t)]\,,$$

where α is the learning rate, $\hat{Q}(s_t, a_t)$ is estimated state-action value.

In deep Q-learning (Mnih et al., 2013), state-action value function $Q(s_t, a_t)$ is approximated by a neural network denoted by $Q_\phi(s_t, a_t)$. Then the function is updated by minimizing the mean squared error $L(\phi, D)$ between the TD-target and the estimated value computed using samples from D:

$$L(\phi, D) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim D}\Big(r_t + \gamma \cdot \max_{a' \in A} \hat{Q}_\phi(s_{t+1}, a') - \hat{Q}_\phi(s_t, a_t)\Big)^2$$

After the optimal state-action value function is learned, the optimal policy can be exacted by greedily taking actions that have highest state-action values:

$$a_t = \operatorname*{argmax}_{a_t} Q(s_t, a_t). \tag{4.1}$$

4.3 Related Work

In this section, we will discuss RL methods that use or are augmented with episodic memory. There are mainly two directions, one that directly uses episodic memory for control ('episodic control'), and another which uses data in the episodic memory to guide the learning update of parametric RL agents ('memory-augmented RL'). We will discuss both.

Episodic control Model-free episodic control (MFEC) (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016) uses a non-parametric table to store episodic return $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ for each state-action pair and only overwrites it when a higher return of the state-action pair is encountered:

$$\hat{Q}_{em}(s_t, a_t) \leftarrow \max[\hat{Q}_{em}(s_t, a_t), G_t]$$

The optimal policy is extracted by greedily taking actions with the greatest values stored in the EM table, similar with Equation (4.1). It outperforms state-of-the-art model-free RL methods on some Atari games in the short run. Neural episodic control (Pritzel et al., 2017b) proposed to use a semi-tabular approach called differentiable neural dictionary (DND) which can store trainable representations which are updated during the training process. NEC selects action in the same way as MFEC. The slowly changing representations and quickly updating values lead to better performance than MFEC. To the best of our knowledge, no previous method has studied the use of episodic memory for action selection in continuous action spaces.

Memory-augmented RL Instead of using non-parametric models directly for action selection, information stored can also be used to enhance learning of a deep RL agent. During updates of value-based deep RL methods (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, et al., 2015) or critic parts of actor-critic RL methods (Lillicrap et al., 2015), the neural network is updated by minimizing the error between a TD target and the currently estimated Q value. Therefore, values from the episodic memory can be used to form a new update target alongside the original TD target:

$$L = \alpha \cdot (\hat{Q} - \hat{Q}_{original})^2 + \beta \cdot (\hat{Q} - \hat{Q}_{em})^2, \tag{4.2}$$

where L is the loss/error to minimize, α and β are hyper-parameters to balance the contribution of the original TD target $\hat{Q}_{original}$ and the episodic target $\hat{Q}_{em}$, and $\hat{Q}$ is

the current Q value estimation. Episodic Memory Deep Q-networks (EMDQN) (Lin et al., 2018), Episodic Memory Actor-Critic (EMAC) (Kuznetsov & Filchenkov, 2021), Model-based Episodic Control (MBEC) (Le et al., 2021) and Generalizable Episodic Memory (GEM) (H. Hu et al., 2021) all use this approach, but in different ways. EMDQN combines values from the EM with 1-step TD target in deep Q learning using fixed α and β. EMAC combines values from the EM with 1-step TD target in DDPG for solving tasks with continuous action space. MBEC brings episodic memory into the model-based RL settings, and combines values from the EM with 1-step TD target using a learned weight instead of fixed α and β. GEM uses EM for implicit planning and performs the update with an n-step TD target (Sutton & Barto, 2018), first taking n steps based on the EM then bootstrapping from a Q-network. Aforementioned methods are all trying to distill information stored in EMs into parametric models. Although by doing so the performance of parametric models is enhanced, meanwhile it also loses the underlying benefit of EM which is fast reading and writing.

Table 4.1: Overview of related memory-based RL approaches. None of these previous works uses episodic memory for action selection in problems with a continuous action space.

Method	EM usage	Action space
MFEC	action selection	discrete
NEC	action selection	discrete
EMAC	target update	continuous
EMDQN	target update	discrete
GEM	target update	continuous
MBEC	target update	discrete
Neural Map	feature storage	discrete
CEC (ours)	**action selection**	**continuous**

Finally, note that there are various other methods, such as Neural Map (Parisotto & Salakhutdinov, 2018), that learn to interact with an external (tabular) memory module to deal with partial observability, but these approaches do not store any value or policy estimates in the table, like episodic memory. A summary of related work is provided in Table 4.1.

4.4 Continuous Episodic Control (CEC)

The principle of CEC is that actions the agent takes in similar states should also be similar. It is implemented by maintaining a table that contains a state-action-value

tuple for each row. Each entry stores the cumulative discounted reward the agent received in an episode after taking the specific action in the specific state. To interact with the memory, CEC mainly has two functions: an **update** function that adds new encountered data into the memory or updates old data after an episode is terminated, and an **action selection** function that is used during the episode to select actions.

Update We update the episodic memory after each collected episode. Intuitively, we want our episodic memory to have good coverage of the visited state space, for which we use the following mechanism. We first collect state-action-value pairs (s, a, v) from episodes, where each value

$$v = \hat{Q}_{em}(s, a) = \sum_{i=0}^{T} (\gamma)^i \cdot r_{t+i} | s_t = s, a_t = a$$

is the cumulative discounted reward obtained after taking action a in state s (for readability we often omit the dependence of v on s and a when it is clear from the context). For each collected (s, a, v), we then find its closest state neighbour s^c in the current memory. The new state-action-value tuple is directly added into the CEC memory if the state is far away from its closest neighbor. Thereby, if the CEC memory does not contain any information about the area near the incoming state, we directly add it into the memory.

When its closest neighbor is within a certain distance threshold d, and its corresponding value is smaller than the value of the incoming state, we replace the previous state-action-value tuple with the new one. The intuition is that when the closest neighbor is close enough, we assume that the CEC memory has information about the area near the new state. However, if the stored value is worse than the one of the new state-action-value tuple, we do overwrite it since we now have information about a better action in the specific state region. The update rule can be expressed as:

$$\text{EM}(s^c, a^c, v^c) \leftarrow \begin{cases} (s, a, v) & \text{if } f(s, s^c) < d \\ & \text{and } v > v^c, \\ (s^c, a^c, v^c) & \text{otherwise} \end{cases} \tag{4.3}$$

Here, superscript c indicates the closest neighbour in the buffer, superscript l indicates the least recently updated entry in the buffer, EM() refers to the slot of the specific entry in the memory buffer, $f()$ specifies a distance measure and d specifies a distance threshold. Once the buffer has filled up, when a new coming state is out of the distance

threshold of its closest state, the least updated tuple will be thrown away and the new coming tuple will be added, otherwise, it will follow the aforementioned update rule:

$$\mathrm{EM}(s^l, a^l, v^l) \leftarrow \begin{cases} (s, a, v) & \text{if } f(s, s^c) > d, \\ (s^l, a^l, v^l) & \text{otherwise} \end{cases} \tag{4.4}$$

Here, superscript l indicates the least recently updated entry in the buffer.

Action selection When a state s is encountered, a corresponding action a is selected based on information stored in the CEC memory. We first find the k nearest-neighbors of state s, denoted by S_k, then filter out neighbors which are too far away. Especially when the memory is sparse, the closest neighbors could still be very far away and we do not consider these faraway neighbors as 'similar' states. Next we let values stored for state-action pairs of the filtered neighbors S_k decide which action should be selected. Higher values indicate better actions, but for exploration purposes we still want to give other actions a chance to be chosen. Therefore, we sample an action from the set S_k proportional to their value estimates, by taking a softmax:

$$p(a_s|S_k) = \frac{e^{v_s/\tau}}{\sum_{s' \in S_k} e^{v_{s'}/\tau}}, s \in S_k \tag{4.5}$$

where subscripts s and s' indicate the state the specific action and value belong to in the buffer, and τ is a temperature factor to scale exploration. We now have a sample from the discrete set of a_i points in the buffer, but our true action space is of course continuous. We therefore transform our sample to a continuous distribution by adding Gaussian noise to the sampled action with probability ϵ. During the evaluation, all exploration is turned off and actions are selected by greedily taking the action of the closest neighbour.

Feature embedding While dealing with a high-dimensional state space, storing the original state and finding its neighbors is expensive. We therefore utilize random projections to project the original state space into a smaller space. We first sample a random matrix $A \in \mathbb{R}^{D_{ori} \times D_{new}}$ for a standard Gaussian distribution, where D_{ori} and D_{new} are the dimensions of the original and projected state space, respectively. We then (matrix) multiply the original state by A to get an embedded state, i.e. $s_{new} = A \cdot s_{ori}$, where s_{new} and s_{ori} are the embedded and original state, respectively.

These transformations will still preserve the relative distance in the original state space. We only use the random projection for **FetchReach** and **Safexp-CarGoal** environments, while for other environments the original state is directly stored and queried.

The overall algorithm can be found in Algorithm 1. $\texttt{knn}(k, s, n, d)$ is a function that first finds the k nearest neighbors of the state s, then filters out states whose distances are larger than n times of the distance threshold d.

Algorithm 1 Continuous Episodic Control (CEC)

Initialize: CEC Memory M, environment `Env`, episode memory M_{eps}, distance threshold d, filter factor n, k for k-nearest-neighbors, discount factor γ, noise standard deviation σ, exploration factor ϵ.
while training budget left **do**
  $s \leftarrow$ `Env.reset()` {Reset the environment.}
  $done$ = false
  while not $done$ **do**
    // SELECT ACTION
    $S_k = \texttt{knn}(k, s, n, d)$ {Find k nearest-neighbors of the state s within threshold $n \times d$.}
    $a \sim p(a_s|S_k)$ {Select actions based on Equation (4.5)}.
    if $\Delta \sim U(0, 1) < \epsilon$ **then**
      $\psi \sim \mathcal{N}(0, \sigma^2)$
      $a \leftarrow a + \psi$
    end if
    $s', r, done \leftarrow \texttt{Env.step}(a)$
    Append $[s, a, r]$ to M_{eps}
    $s \leftarrow s'$
  end while
  // UPDATE
  $v = 0$
  while M_{eps} **not** empty **do**
    $s, a, r \leftarrow M_{eps}.\texttt{pop()}$
    $v \leftarrow r + \gamma \cdot v$
    Update M using (s, a, v). {Update CEC memory based on (Equations (4.3) and (4.4))}.
  end while
end while

4.5 Experiments

In this section, we will first give some insights of CEC by using simple toy examples. Then we scale up to more complex continuous control tasks and compare with state-of-the-art RL and memory-augmented RL methods. Dimensions of environments we used for this work are shown in Table 4.2. Although **Safexp-PointGoal** and **Safexp-CarGoal** have the same action space (two actuators, one for turning and one for moving), in **Safexp-CarGoal** both turning and moving require coordinating both of the actuators, which is more complex. Every experiment is averaged over 5 independent runs and standard errors are plotted as well.

Table 4.2: Dimensions of state spaces and action spaces of different environments.

Env	State Space	Action Space
GrowingTree	1	1
MountainCarContinuous	2	1
PointUMaze	4	2
Point4Rooms	4	2
Safexp-PointGoal	24	2
Safexp-CarGoal	36	2
FetchReach	13	4

4.5.1 Toy Examples

In order to first get an overview and more insights into the method, we create a toy example called GrowingTree. In this environment, the agent needs to 'grow' the 'tree' to the given target height. More details can be found in Tab. **??**. The continuous action space ranges from -0.1 to 0.1, which means the agent can chose to 'cut' (< 0) the 'tree' or 'grow' (> 0) the 'tree'. The observation for the agent is the height of the current 'tree' and the height changes every step by simply adding the action to the previous observation. The optimal policy for the agent to get the final reward is to take 0.1 as the action every step whatever the height of the tree is, which causes the 'tree' to grow linearly.

We train the CEC agent on the toy example, with results shown in Figure 4.2. Every row of the figure shows an evaluation of the CEC agent after another $10k$ training steps, from top to bottom. The n^{th} row therefore represents the evaluation results after $n \times 10k$ training steps. The left part of the figure shows the heights of the tree during the evaluation episode. If the optimal policy is learned by the agent,

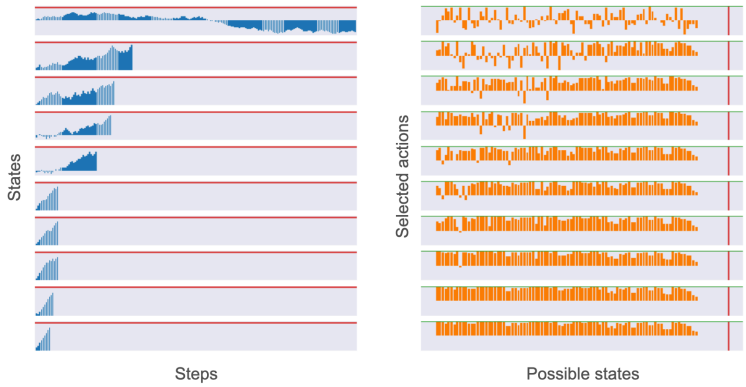


Figure 4.2: Evaluation on the toy example. We train the CEC agent on the toy example for 100 k steps and evaluate it every 10k steps. Training are from top to bottom. **Left**: the performance during the evaluation. It gradually takes shorter episode to reach the goal. x-axis is taken steps during the evaluation episode and y-axis is the state (the height of the current 'tree'). Red horizontal lines are targets we set. **Right**: actions selected by the trained CEC agent. x-axis represents different states while y-axis represents actions. Red horizontal lines are final target states and green vertical lines are optimal actions (0.1).

Table 4.3: Details about GrowingTree environment. The agent only gets a final reward when it successfully reaches the given goal (with a distance tolerance).

state space	[-2, 2]
action space	[-0.1, 0.1]
maximum steps	200
goal	1.0
final reward	1.0
distance tolerance	0.1
start height	0

the height of the tree will increase quickly with the number of steps (x-axis) increases. The right figure shows the actions the CEC agent takes for every possible state (we only evaluate states that lie within [0,1]). The optimal action in every state is (near) 0.1.

In the top rows, the agent still chooses to 'cut' the 'tree' sometimes, but the overall trend of the 'tree' is still growing. Although the solution is not the optimal, the agent is able to solve the problem successfully. In the middle rows, the agent can solve the task fairly quickly. However, if we look at the middle rows of the right figure, the chosen action for every possible state is still far from the optimal and a few of them are still smaller than 0. Gradually, selected actions of all possible states are getting

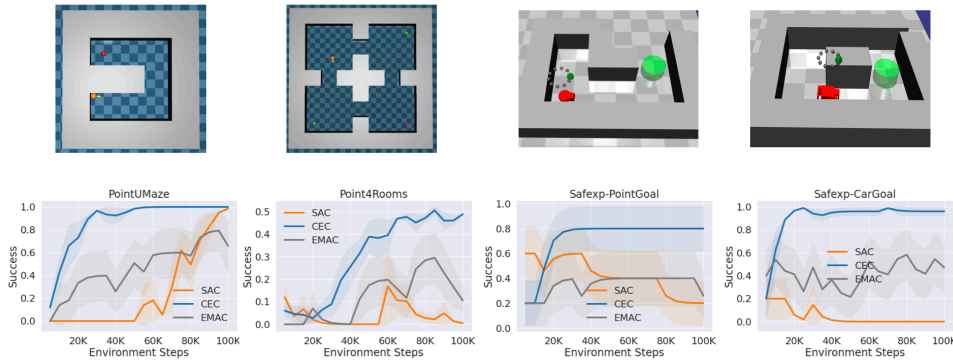


Figure 4.3: Four continuous navigation environments we used in this work and their corresponding results. From left to right, they are 'PointUMaze', 'Point4Rooms', 'Safexp-PointGoal' and 'Safexp-CarGoal', respectively. We see the proposed method CEC (blue lines) outperform SAC (orange lines) and EMAC (grey lines) in all these four tasks. Results are averaged over five independent runs and the shaded area represents standard errors.

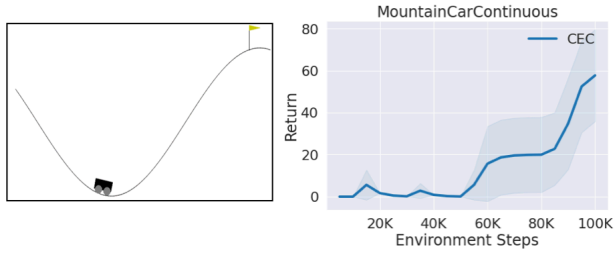


Figure 4.4: MountainCarContinuous environment and learning curve of the CEC agent on it. The agent gradually learns to solve the problem.

closer and closer to the optimal values (bottom rows).

Through this simple example, we see that our principle "similar states should have similar actions" does work and can indeed quickly solve the problem with a non-optimal solution, while it may also discover a near-optimal solution given more time.

We next test our method in a simple classical RL task, **MountainCarContinuous**. We use the sparse reward version of the task, where the agent only gets a final reward (+100) once it brings the car successfully to the flag on the top of the mountain. Since the initial position of the car is randomly generated near the bottom of the mountain, the agent also needs to deal with proper generalization. The learning curve of CEC on this task is shown in Figure 4.4. The CEC agent gradually learns to solve the task, which gives a first indication that our method can indeed be used for basic RL

problems that require generalization.

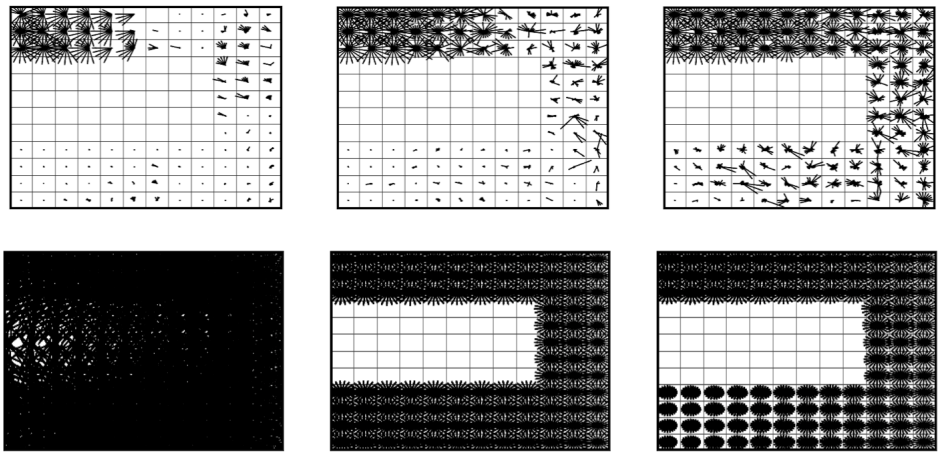


Figure 4.5: State values produced by CEC (top) and SAC (bottom) during the training on PointUMaze. Figures from left to right are after 5k, 50k, and 100k steps of training, respectively. Since the state space is continuous, we discretize the location information into 12×12 grids and the orientation information into 20 directions (represent as different directed arrows). Longer arrows indicate larger state-action values, and SAC outputs very large values at the beginning which is not rational while CEC has larger values for the area near the goal.

4.5.2 Continuous Navigation Tasks

We next move to more complicated test environments, in which we will also compare CEC to two baselines: the model-free deep RL method SAC (Haarnoja, Zhou, Abbeel, & Levine, 2018) and a memory-augmented deep RL method EMAC (Kuznetsov & Filchenkov, 2021) which uses EM for learning targets. We test these in four continuous Mujoco navigation tasks, shown in Figure 4.3. They are PointUMaze, Point4Rooms, Safexp-PointGoal, and Safexp-CarGoal, respectively. These tasks are all sparse-reward problems, in which the agent only gets a final reward when the task is successfully completed. Locations of both goals and agents are initialized with small randomness every episode. Hyper-parameters we are using for CEC on these tasks can be found in Table 4.4.

Results are shown in Figure 4.3. CEC agents can quickly solve the task and outperform SAC and EMAC agents in all environments. In these four navigation tasks, there are explicit bottlenecks and CEC can quickly remember the discovered solution by only discovering it once. In contrast, the deep RL methods need to discover

Table 4.4: Hyper-parameters of CEC we are using for continuous navigation tasks. Notations can be found in Algorithm 1.

Env	k	τ	σ	n	d
PointUMaze	5	0.1	0.3	1	0.1
Point4Rooms	5	1	0.3	3	0.1
PointGoal	5	0.1	0.1	1	0.1
CarGoal	1	10	0.1	3	0.1

the solution multiple times in order to back-propagate reward signals for learning, and this rediscovery is time-consuming due to the bottlenecks in the tasks.

We also visualize state-action values of CEC and SAC during the training for **PointUMaze**, which is shown in Figure 4.5. The agent always starts from the bottom-left corner and the goal is always located at the top-left corner. Both of these locations are initialized with small randomness. Since the state space of the environment is 3 dimensional (x location of the agent, y location of the agent, orientation of the agent) and continuous, we discretize x, y locations as a grid world (12×12) and orientations as a group of different directed arrows (20 different directions). The length of the arrow represents the state-action value of the state that is jointly indicated by the arrow and the position of the cell and the action for that specific state (only one action is maintained for each state). The longer the arrow is, the larger the state-action value is. From Figure 4.5, we see that in the beginning the SAC agent strongly overestimates all state-actions values, due to the random initialization(arrows went across the whole state space, meaning all state-action values are extremely large). On the contrary, the CEC agent obtains reasonable estimates more quickly, also identifying larger value estimates near the goal region. While training progresses, both agents gradually improve their estimates to better propagate high-value estimates towards the start region (i.e., identifying a good policy). Both agents eventually solve the task, but CEC was much faster, as can be seen in Figure 4.3 as well.

4.5.3 Robotics Control Task

We finally investigate our method on a more complicated robotics control task: **FetchReach**. The goal of the task is to control the robotics arm to reach the red point, which is reset to a randomly sampled position every episode. The environment and its associated learning curve are shown in Figure 4.6. We use $k = 5, \tau = 1, \sigma = 0.1, n = 3, d = 0.5$ for CEC. We see that CEC still outperforms the state-of-the-art baseline methods, although the difference is less pronounced compared to the previous navigation tasks.

We attribute this to the lack of bottlenecks in this task. CEC is probably most useful when a task has some narrow passages, which are challenging from an exploration point of view, and in which CEC can quickly latch onto a few successful trials. Nevertheless, even in a robotics task with no bottleneck characteristics, CEC performs on par with current state-of-the-art model-free baselines.

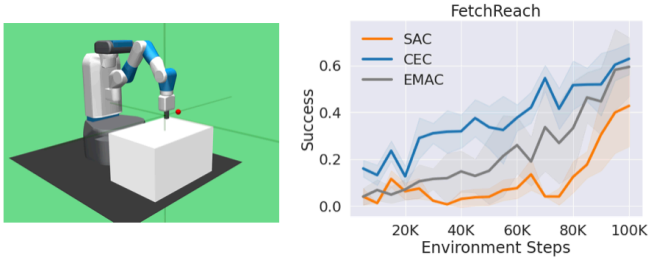


Figure 4.6: FetchReach environment and performance of different agents. CEC still outperforms other RL agents but the benefits are not as large as the ones in previous navigation tasks.

4.6 Conclusion and Future Work

In this work, we propose a new episodic control method called continuous episodic control (CEC) which can handle tasks with a continuous action space. It is the first work that uses episodic memory for continuous action selection. By following the principle that "similar states should have similar actions", CEC agents outperform the state-of-the-art RL method SAC and memory-augmented RL method EMAC in various sparse-reward continuous control environments. Although RL methods might outperform episodic control in the long run, sometimes we do prefer a quick and cheap solution to the problem. Several experiments show CEC has strong performance on continuous navigation tasks, especially when there are bottlenecks in the environment.

Although non-parametric methods (like episodic control) require increasing memory with new data, this can be eliminated by employing a throw-away mechanism that only stores important data. In fact, experimental results show that episodic control is able to scale up to Atari games (Silver et al., 2016), such as Ms.Pac-Man, Space Invaders, and Frostbite, where pixel images are used as observations. However, episodic control also has its limitations. For example, using Monte-Carlo returns as its state-action values has high variance, and also it will not be able to learn the optimal solution in stochastic environments by definition. But we believe sometimes we do

prefer a quick (might be sub-optimal) solution.

In the future, it would be very interesting to investigate better feature embedding methods (such as using latent features of a pre-trained VAE (Kingma & Welling, 2013)) and exploration strategies that work well with CEC. In addition, we could also look at methods to improve the nearest neighbour search, for example, based on clustering. Finally, we believe there is potential to combine episodic control and parametric reinforcement learning methods to distill strengths from both sides and then end up with a more powerful agent.

Chapter 5

Two-Memory Reinforcement Learning

Authors: Zhao Yang, Thomas Moerland, Mike Preuss, Aske Plaat

Published in: IEEE Conference on Games (CoG), 21-24 August 2023, Boston, USA.

Abstract

While deep reinforcement learning has shown important empirical success, it tends to learn relatively slow due to slow propagation of rewards information and slow update of parametric neural networks. Non-parametric episodic memory, on the other hand, provides a faster learning alternative that does not require representation learning and uses maximum episodic return as state-action values for action selection. Episodic memory and reinforcement learning both have their own strengths and weaknesses. Notably, humans can leverage multiple memory systems concurrently during learning and benefit from all of them. In this work, we propose a method called Two-Memory reinforcement learning agent (*2M*) that combines episodic memory and reinforcement learning to distill both of their strengths. The *2M* agent exploits the speed of the episodic memory part and the optimality and the generalization capacity of the reinforcement learning part to complement each other. Our experiments demonstrate that the *2M* agent is more data efficient and outperforms both pure episodic memory and pure reinforcement learning, as well as a state-of-the-art memory-augmented RL agent. Moreover, the proposed approach provides a general framework that can be used to combine any episodic memory agent with other off-policy reinforcement learning algorithms. [1]

[1] Code can be found at https://github.com/yangzhao-666/2m.

5.1 Introduction

Deep reinforcement learning (DRL) achieves impressive results in a wide range of domains. It reaches super-human performance in games such as Atari (Mnih et al., 2013), Go (Silver et al., 2017) and Gran Turismo (Wurman et al., 2022). Recently, it also has shown promise in scientific applications such as controlling nuclear plasma fusion (Degrave et al., 2022) and discovering new matrix multiplication algorithms (Fawzi et al., 2022). However, DRL is well-known for being data inefficient, since back-propagation of reward signals and learning updates (including representation learning) can be slow.

In contrast to such parametric learning approaches, non-parametric episodic memory approaches maintain a memory buffer to store high-rewarded trajectories for either action selection (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016; Pritzel et al., 2017a) or for enhancing other reinforcement learning methods (H. Hu et al., 2021; Kuznetsov & Filchenkov, 2021; Lin et al., 2018). These papers have been demonstrated to outperform conventional reinforcement learning methods in certain tasks, such as Atari (Bellemare et al., 2013), and Labyrinth (Mnih et al., 2016). In episodic memory, reward signals are back-propagated considerably faster than in one-step temporal difference (TD) learning that is commonly used in reinforcement learning. In addition, one-step TD learning is further slowed down by representation learning and the use of function approximation. A potential problem of episodic memory is that fast back-propagation is problematic in stochastic tasks, and the lack of learnable feature representations can make generalization difficult in episodic memory. The question therefore becomes: can we combine the best of both approaches in a single algorithm?

Evidence from neuroscience shows that multiple memory systems are activated when humans are learning, and these also interact with each other (Poldrack & Packard, 2003; Schott et al., 2005). Previous research (H. Hu et al., 2021; Kuznetsov & Filchenkov, 2021; Lin et al., 2018) has shown that integrating episodic memory and reinforcement learning can improve overall performance. In these works, episodic memory is mainly used to provide learning signals for DRL methods, but they again face the same challenges that are inherent to DRL. To fully capitalize on the advantages of episodic memory and reinforcement learning, we propose a novel approach called Two-Memory reinforcement learning (*2M*), in which both approaches complement eachother.

The workflow of the *2M* agent is shown in Figure 5.1. The *2M* agent maintains two memories, namely 'episodic control' (episodic memory for control) (*2M-EC*) and 'rein-

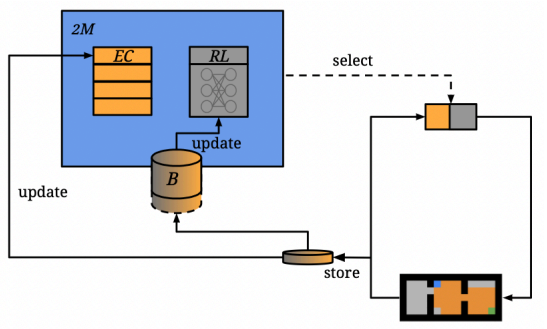


Figure 5.1: The workflow of the *2M* agent. Before the episode starts, the *2M* agent selects one type of memory for action selection in the next episode, which could either be episodic control (EC) or parametric reinforcement learning (RL). The data subsequently collected is used to update the EC solution (directly), and also enters an experience replay buffer (B) for future updating of the parametric RL solution.

forcement learning' (*2M-RL*). In the beginning, the *2M* agent decides which memory to employ for action selection in the upcoming episode. Then the collected episodic data is pushed into the experience replay buffer where data for training *2M-RL* is sampled from occasionally. Meanwhile, episodic trajectories are used to update *2M-EC*.

The intuition is that after *2M-EC* discovers high-reward trajectories, the *2M* agent is able to retrieve these trajectories quickly although they might be suboptimal. However, at this stage, *2M-RL* still has not learned anything substantial due to the one-step reward backup and the slow updates required with function approximation (in DRL). Thus, the *2M* agent should prefer to use *2M-EC* initially. As the amount of data collected increases and training continues, *2M-RL* becomes increasingly good at dealing with stochasticity and developing better feature representations; the *2M* agent should then gradually switch to *2M-RL*. This is the conceptual approach we study in this work.

In short, our work makes two main contributions:

- We propose a novel framework called *2M* that combines episodic control and reinforcement learning methods, exploiting the strengths from both sides. The framework can be used with any type of EC method and any type of off-policy RL method.

- We conduct experiments showing that *2M* outperforms state-of-the-art baselines, and we also include ablation studies that examine when the *2M* works well, and where it may still be improved.

5.2 Background

We will first introduce the formal problem definition and the two main approaches combined in this paper: parametric reinforcement learning (in the form of deep Q-learning) and non-parametric episodic memory.

5.2.1 Markov Decision Process

We follow standard definitions (Sutton & Barto, 2018) and define decision making problems as a Markov Decision Process (MDP) represented by $\langle S, A, R, P, \gamma \rangle$. S denotes the state space, A denotes the action space, R denotes the reward function and P denotes the dynamic transition. γ is the discount factor, which is usually close to 1. The agent interacts with the environment by taking action $a_t \in A$ according to some policy π given the current state $s_t \in S$ at time step t, then the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and reward $r_t = R(s_t, a_t, s_{t+1})$ is returned by the environment. A policy π maps a state to a distribution over actions. The agent will then take the next action a_{t+1} based on the new state s_{t+1}. This process repeats until the terminal time step T. The goal of the agent is to learn a policy π that maximizes the expected cumulative reward: $\mathbb{E}_{s_{t+1} \sim P(\cdot|s_t,a_t), a_t \sim \pi(\cdot|s_t), r_t \sim R(\cdot|s_t,a_t)}[\sum_{t=0}^{T} \gamma^t \cdot r_t]$.

5.2.2 Deep Q-Learning

Define the state-action value function $Q(s_t, a_t)$ as the expected cumulative reward the agent will receive by starting from state s_t and taking action a_t. Q-learning is an algorithm to learn the optimal Q value function, it updates the state-action value function by iteratively following the Bellman equation. The update rule of Q-learning is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t))$$

where α is the learning rate and γ is the discount factor. The solution of Q-learning is generally stored in a table. When the state space is large, storing all possible states becomes infeasible. We may then use a neural network to approximate the state-action value function. The update rule of deep Q network (Silver et al., 2017) is:

$$y(s_t) \leftarrow r_{t-1} + \gamma \max_{a' \in A} Q(s_t, a')$$

$$\mathbb{E}_{s_t, a_t, r_t, s_{t+1} \in D}(y(s_{t+1}) - Q(s_t, a_t))^2 \tag{5.1}$$

where D is the sampled data for training. After we have the state-action value function, the policy is exacted by always taking the action with the highest state-action value greedily in every state.

5.2.3 Episodic Control

Episodic control refers to a class of methods that directly use non-parametric episodic memory for action selection (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016; Pritzel et al., 2017a). It is generally implemented as a table (Q^{ec}), and rows represent different actions while columns represent different states. Each entry is associated with a state-action (s_t, a_t) pair and denotes the highest encountered episodic return ($G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$) after taking action a_t in the state s_t. The update only occurs after one episode has terminated, which is similar to tabular Monte-Carlo back-up but with a more aggressive update rule. Instead of incrementally updating state-action values, episodic control replaces the stored episodic return with the best value observed so far. The update rule of episodic control is:

$$Q^{ec}(s_t, a_t) \leftarrow \begin{cases} G_t & \text{if } (s_t, a_t) \notin Q_{ec}, \\ \max\{G_t, Q^{ec}(s_t, a_t)\} & \text{otherwise} \end{cases} \tag{5.2}$$

During action selection, if the queried state already has Q values for all actions, we take the action that is with the highest Q value. Otherwise, missed Q values are estimated by averaging over the queried state's K-nearest neighbors' Q values. When the memory is full, the least updated entry will be dropped from the table.

5.3 Related Work

We will first discuss previous work on episodic control, and how it has been used as a training target for deep reinforcement learning. This last approach differs from our work, where EC is used for action selection (see Figure 5.1). Afterwards, we also briefly discuss related work on experience replay, since it plays a central role in our approach as well (see Figure 5.1).

5.3.1 Episodic Control

Model Free Episodic Control (MFEC) (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016) is the first episodic control method and it is implemented as a table with untrainable features. Thus, it enjoys limited generalization over either a randomly projected or pre-trained feature space. Neural Episodic Control (NEC) (Pritzel et al., 2017a) solves this limitation by maintaining a differentiable table to learn features while using the same update and action selection rule (shown in Equation (5.2)) as MFEC but achieves better performance. Since Monte-Carlo returns must be stored for each state-action pair, episodic control methods can not deal with continuous action spaces naturally. Continuous Episodic Control (CEC) (Z. Yang et al., 2022) extends episodic memory to select actions directly in continuous action space. The *2M* agent integrates MFEC as a core component, and partially uses it for action selection.

5.3.2 Episodic Memory for Learning

While using episodic memory for control is fast, reinforcement learning methods might still be preferred in the long run due to their strength. Many approaches use the returns stored in episodic memory to provide richer learning targets for reinforcement learning methods. Episodic Memory Deep Q Network (EMDQN) (Lin et al., 2018) uses returns in episodic memory to enhance learning targets in deep Q-Learning. Episodic Memory Actor-Critic (EMAC) (Kuznetsov & Filchenkov, 2021) and Generalizable Episodic Memory (GEM) (H. Hu et al., 2021) uses episodic returns to enhance learning targets in actor-critic methods to solve tasks with continuous action space. Episodic Backward Update (EBU) (Lee et al., 2019) utilizes structural information of states that are in the same episode and executes a one-step backup for each state along the trajectory. The aforementioned methods take advantage from richer learning signals of episodic memory, but underlying neural network training still progresses slowly. Thereby, the benefit of the EC solution will not affect action selection quickly. In contrast, the *2M* agent does use episodic memory for action selection, which may give it a fast head-start. As a second difference, in *2M* the collected data is also used to train the *2M-RL* agent (in contrast to previous methods).

5.3.3 Experience Replay

Experience replay was originally proposed to improve data efficiency and break correlations of training data for off-policy reinforcement learning methods. Uniform sampling

is the most naive and commonly used way to sample data from the replay buffer for training, where transitions are sampled at the same frequency as they were experienced regardless of their significance (Schaul, Quan, et al., 2015). To address this limitation, prioritized Experience Replay (PER) (Schaul, Quan, et al., 2015) prioritizes transitions that have larger TD errors during the training, and samples these transitions more often because larger TD errors indicate there is more information to learn. Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) is proposed for the multi-goal/goal-conditioned RL setting; it treats states that the agent actually achieves as desired goals and learns from failures. Since not all failures are equally important, Curriculum-guided Hindsight Experience Replay (CHER) (Fang et al., 2019) adaptively replays failed experiences according to their similarities to true goals. An event table (Kompella et al., 2022) is defined as a buffer to store transitions related to important events. The authors theoretically proved that sampling more data from the event table will lead to better performance. Although the *2M* agent doesn't employ any special sampling strategies to sample from the replay buffer, the data stored in the buffer is actually from two different sources (*2M-EC* and *2M-RL*). We can thus vary sampling preference by pushing different amounts of data from different sources.

5.4 Two-Memory Reinforcement Learning (*2M*)

We will now formally introduce the *2M* framework. It consists of two 'memories', where one represents a fast learning memory (episodic control agent) and another one represents a slow learning memory (reinforcement learning agent, in our work we use 1-step (deep) Q-learning). Intuitively, we should prefer to use the fast (but sub-optimal) learning memory initially, then gradually switch to the slow (but with better asymptotic performance) learning memory. In this section, we will first motivate this intuition by using a very simple example (Section 5.4.1). Then we explain the designs of the proposed approach. Since we combine two different methods and want to switch between them, we also need to discuss a scheduling mechanism to decide when to switch (Section 5.4.2) and need to decide how to utilize collected data (Section 5.4.3). The overall algorithm is detailed in Algorithm 2. At the beginning of each episode, *2M-EC* or *2M-EC* is selected and will be used for action selection in the whole episode. Collected data is stored into the replay buffer used for training *2M-RL* which happens every N steps. *2M-EC* is trained after every episode is finished. Eventually, the one performs better during the training will be used for evaluation.

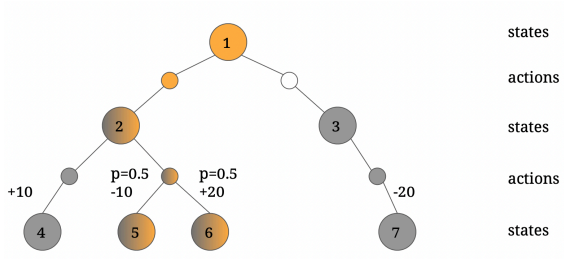


Figure 5.2: A motivating example with seven states in the state space and two actions in the action space. After the agent discovers all possible trajectories (from s^1 to all possible terminal states), EC (orange) finds the sub-optimal solution while one-step off-policy RL (grey) only back-propagates reward signals to direct predecessor states and thus does not change the estimates at the root state. However, after many iterations, off-policy RL will converge to the true optimal values (preferring action left followed by action left), while EC will commit to a suboptimal solution (action left followed by action right). Colour shading indicates the preferred solution path for each method after a single visit to every possible path.

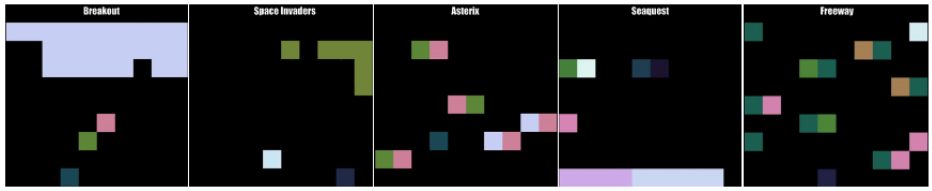


Figure 5.3: The five MinAtar games used in the experiments (from left to right): Breakout, Space Invaders, Asterix, Seaquest, Freeway.

5.4.1 A Motivating Example

We first consider a very simple example domain shown in Figure 5.2, circles with numbers represent different states while smaller circles represent actions. There are seven states in the state space: $S = \{s^1, s^2, s^3, s^4, s^5, s^6, s^7\}$, two actions (left and right) in the action space: $A = \{a^1, a^2\}$. Most of dynamic transitions are deterministic, except in state s^2, after taking action a^2, there is 50% probability the next state ends up with s^5 and 50% probability ends up with s^6: $P(s^2, a^2, s^5) = P(s^2, a^2, s^6) = 0.5$. Leaf nodes (s^4, s^5, s^6, s^7) are terminal states where the agent will receive a reward: $R(s^2, a^1, s^4) = +10$, $R(s^2, a^2, s^5) = -10$, $R(s^2, a^2, s^6) = +20$, and $R(s^3, a^2, s^7) = -20$. Color shading indicates the decisions that different agents will make after a single visit to every possible trajectory (orange for episodic control and grey for 1-step Q-learning). We can see that after the agent discovers all possible trajectories, the episodic control agent is already able to follow a sub-optimal trajectory starting from the initial state.

Algorithm 2 Two-Memory Reinforcement Learning

Input: Environment *env*, non-parametric episodic control agent *ec*, parameterized reinforcement learning agent *rl*, Two-memory agent *2M*, replay buffer B, probability p^{ec}, exploration factor ϵ

$G^{ec} \leftarrow [\,]$ {`initialize return list of ec`}
$G^{rl} \leftarrow [\,]$ {`initialize return list of rl`}
2M-RL $\leftarrow$ *rl*
2M-EC $\leftarrow$ *ec*
while training budget left **do**
  $s \leftarrow$ reset the *env*
  done = false
  $\tau \leftarrow [\,]$ {`initialize episodic trajectory`}
  if $\Delta \sim U(0,1) < p^{ec}$ **then**
    2M $\leftarrow$ *2M-EC* {`2M switches to EC`}
  else
    2M $\leftarrow$ *2M-RL* {`2M switches to RL`}
  end if
  $G \leftarrow 0$
  while not *done* **do**
    execute action a selected by *2M* based on s with ϵ-greedy exploration
    observe reward r and next state s' in *env*
    store (s, a, r, s') to B and τ
    $s \leftarrow s'$
    $G \leftarrow G + r$
    update *2M-RL* using D according to (Equation (5.1)) if needed, $D \sim B$
  end while
  update *2M-EC* using τ according to (Equation (5.2))
  if *2M* is *2M-RL* **then**
    add G to G^{rl}
  else
    add G to G^{ec}
  end if
  update p^{ec} {`decay or increase or consistent`}
end while

However, the 1-step Q-learning agent only back-propagates the reward signal from terminal states to their predecessors, which means the 1-step Q-learning agent still is not able to make decisions in the initial state. The optimal policy for this MDP is to take a^1 in s^1 and take a^1 in s^2 as well. By definition, episodic control will converge to a sub-optimal policy that always takes a^2 in s^2 (shown in Equation (5.3)). With more updates, 1-step Q-learning will learn the optimal state-action value (shown in Equation (5.4)) for each state-action pair, which will result in the optimal policy.

$$Q^{ec}(s^2, a^1) = 10, Q^{ec}(s^2, a^2) = 20 \tag{5.3}$$

$$Q^{*}(s^2, a^1) = 10, Q^{*}(s^2, a^2) = 5 \tag{5.4}$$

Thus, we conclude that episodic control is fast but sub-optimal, and reinforcement learning (1-step Q-learning in our case) is slow but optimal. There should exist an intersection point between these two different methods where reinforcement learning surpasses episodic memory. One might ask why we do not compare with (full-episode) Monte-Carlo backup. The 1-step back-up can utilize off-policy learning to learn from data collected by other policies, which is more data efficient. We therefore aim to combine the fast learning of EC with the data efficiency of 1-step off-policy learning.

5.4.2 Switching

The previous example highlighted that EC may learn a solution faster, while RL may eventually converge to a better solution. Therefore, we ideally want a switching mechanism, that transitions from EC action selection to RL action selection. We need to decide both **when** and **how** to switch between the two different 'memories'.

Regarding the 'when', we propose to decide which memory to use for every episode. This way, we ensure that action selection within the episode stays consistent. To determine which memory we will use in a particular episode, we need to define a probability p^{ec} that specifies the probability that we will use EC in the next episode. Obviously, we then select *2M-RL* with probability $1 - p^{ec}$. Since we favor the use of *2M-EC* at the beginning and *2M-RL* near the end, we want to gradually decay p^{ec} from a high value to a lower value according to the equation:

$$p^{ec} \leftarrow p^{e} + (p^{s} - p^{e}) \cdot e^{-i/\tau} \tag{5.5}$$

where p^s and p^e are starting value and end value of p^{ec}, i is the number of steps the agent takes so far and τ is the temperature that controls the speed of decay. Smaller τ decays p^{ec} faster while lager τ decays p^{ec} slower. In the ablation experiments, we also experiment with different scheduling mechanisms.

During evaluation, we exploit knowledge stored in both memories in a greedy manner. However, we still need to decide which memory to use during evaluation. The scores *2M-RL* and *2M-EC* obtained during training are used as metrics to evaluate their respective performances, and the memory with the higher recent score is selected for action selection during evaluation. More specifically, we keep track off cumulative

rewards of both memories during training, and the score (s^{rl} and s^{ec} for *2M-RL* and *2M-EC*, respectively) is defined as the average cumulative reward over the last n episodes per memory method. We fix $n = 50$ in this work. Thus, the *2M* agent will choose among two memories using Equation (5.6).

$$2M \leftarrow \begin{cases} 2M\text{-}RL & \text{if } s^{rl} \geq s^{ec}, \\ 2M\text{-}EC & \text{otherwise} \end{cases} \tag{5.6}$$

5.4.3 Learning

To foster mutual improvement of the two memories, the collected data is shared between *2M-EC* and *2M-RL*. Regardless of which memory the data is collected by, it is used to update *2M-EC* according to Equation (5.2), ensuring that *2M-EC* is always up to date. On the other hand, since we use off-policy reinforcement learning methods, data collected by *2M-EC* can also be used to update *2M-RL*. This is implemented by maintaining a replay buffer, and all data collected during the training will be pushed into it. *2M-RL* is then trained (Equation (5.1)) every x timesteps (we use $x = 10$) by sampling minibatches from the buffer. It should be noted that the value of p^{ec} indirectly determines the amount of data in the replay buffer originating from *2M-EC*, and thereby the proportion of such data used for training *2M-RL*.

5.5 Experiments

We first present experimental results on a simple WindyGrid environment to demonstrate the efficiency of the proposed *2M* agent. Subsequently, we perform extensive experiments on five MinAtar (Young & Tian, 2019) games, namely Breakout, SpaceInvaders, Asterix, Seaquest, Freeway, as illustrated in Figure 5.3. These games present diverse challenges, such as exploration (Seaquest and Freeway), classical control tasks (Breakout and SpaceInvaders), and so on. MinAtar is a simplified version of Atari Learning Environment (Bellemare et al., 2013), which simplifies representations of games while still capturing general mechanics. This allows the agent to focus on behavioural challenges rather than representation learning. Finally, an ablation study is conducted to investigate the crucial choices of the proposed method.

5.5.1 Proof of Concept Under Tabular RL Setting

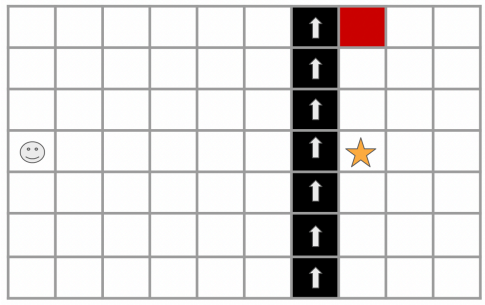


Figure 5.4: An illustration of the WindyGrid environment. The agent (smile face) needs to navigate from the starting point to the given goal (star). There is the stochastic wind that will (black column with up arrows) blow the agent up and the agent gets a penalty when it reaches the trap (red cell).

The *2M* agent integrates tabular 1-step Q-learning with episodic control to solve a WindyGrid task (shown in Figure 5.4). The WindyGrid instance we use here is 7×10 large, with a stochastic wind in the 7^{th} row and a trap where the agent will get a large penalty. The agent needs to navigate from the initial state $(3, 0)$ to the terminal state $(3, 7)$. The results presented in Figure 5.5 demonstrate the performance of various

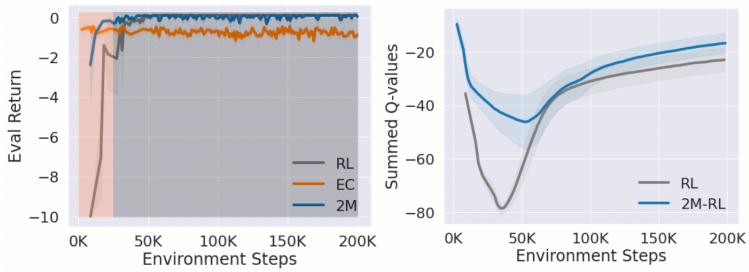


Figure 5.5: Results on WindyGrid under tabular settings. **Left**: Evaluation returns of different agents, EC learns very fast but converges to local optima while RL learns slowly but converges to global optima. 2M learns faster (compare to RL) at the beginning and has better asymptotic performance (compare to EC). Colors in the background indicate memories used during evaluation, orange and grey represent the use of *2M-EC* and *2M-RL*. **Right**: Learned Q-values summed across the entire state-action space, *2M-RL* learns Q-values faster than pure RL which is trained using monotonic RL data.

agents. The left figure shows returns the agent obtains during the evaluation, while the right figure shows learned Q-values summed across the whole state-action space. The orange line represents the *EC* agent, which achieves quick learning and decent

performance from the start but only has sub-optimal asymptotic performance. In contrast, the grey line corresponds to the *RL* agent, which initially performs poorly but gradually improves and eventually converges to the optimal solution. The blue line corresponds to the *2M* agent, which learns faster at the beginning compared to *RL* and achieves better asymptotic performance compared to *EC*. The background colors indicate the use of different memories by the *2M* agent during the evaluation, with the agent using *2M-EC* (orange) for evaluation at the beginning and then switching to *2M-RL* (grey) after approximately $25k$ steps.

Since the memories *2M-RL* and *2M-EC* share data, and *2M-RL* is trained on data collected by both of them, we investigate whether this approach has a positive impact compared to training solely on data collected by pure *RL*. As Varun et al. (Kompella et al., 2022) demonstrated theoretically and experimentally, learning from data correlated to the optimal policy will results in lower complexity (which is an intuitive result). If we assume the EC data is at least somewhat correlated to the optimal policy, we expect that use of EC data in RL updates will lead to faster learning. We experimentally test this idea in Figure 5.5. The right panel of Figure 5.5 shows that using mixed data to train the *RL* agent (*2M-RL*) can indeed result in faster learning of state-action values compared to using data solely collected by *RL*. This suggest EC data may actually improve RL sample efficiency.

5.5.2 Results on MinAtar Games

Next, we perform experiments on five MinAtar games, which also vary in the amount of stochasticity. We optimize hyper-parameters over the values shown in Table 5.1, where the settings used for the final results are highlighted in bold.

Table 5.1: Tuned hyper-parameters, where the ones we use for the final experiments are highlighted in bold. p^s is the starting value from Equation (5.5)

ϵ (for exploration)	$\mathbf{0.1}, 0.9$
k (for k-NN in EC)	$1, \mathbf{3}, 10$
$p^s \rightarrow p^e$ (for switching)	$\mathbf{0.9 \rightarrow 0.1}, \mathbf{0.1 \rightarrow 0.1}, \mathbf{0.1 \rightarrow 0.9}$
learning rate	$0.001, \mathbf{0.0001}$

In the results presented in Figure 5.6, the top rows depict the various memories the *2M* agent selected during the evaluation, whereas the bottom rows display the returns agents obtain. The performance of *EC* (orange lines) and *DQN* (grey lines) is consistent with the findings observed in the toy example, where *EC* learns quickly at the beginning but converges to sub-optimal solutions while *DQN* learns slowly but has

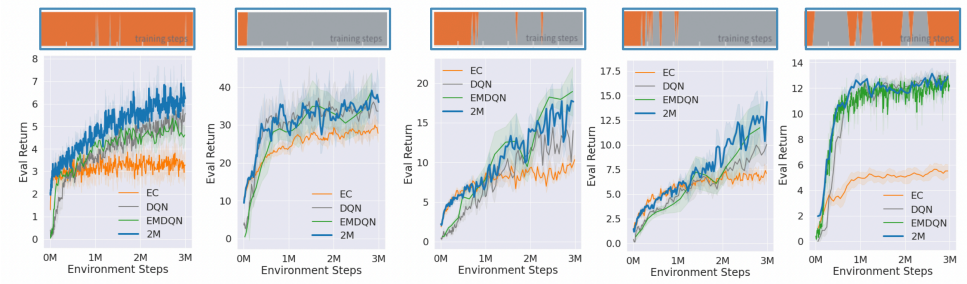


Figure 5.6: Results on MinAtar games: Breakout, SpaceInvaders, Asterix, Seaquest, Freeway. The top row shows the relevant memory that the *2M* agent chooses for evaluation: orange represents *2M-EC* and grey represents *2M-RL*. The bottom row shows the returns obtained during training (running independent evaluation epsisodes). The *2M* agent either outperforms or is on par with all baseline comparisons (EC, DQN and EMDQN). EC generally learns fast, but then reaches a plateau. *2M* learns equally fast initially, but then adopts the better long-term performance of RL.

better asymptotic performance. *2M* agents (blue lines) perform the best (comparably to *EC*) at the beginning on all five games, and in most games, they also exhibit better asymptotic performance (or at least equally good performance) compared to other baselines (except maybe for Asterix). In Asterix, the *2M* agent underperforms *EMDQN* (green lines), but is still better than *DQN* and *EC*.

During the evaluation, most *2M* agents demonstrate a preference for using *2M-EC* initially and then switching to *2M-RL* over time. In Freeway, the agent continually switches between *2M-EC* and *2M-RL*, suggesting a mutually beneficial relationship between these two memories. However, in Breakout, the agent consistently favors to use *2M-EC*. This may be attributed to the fact that stochastic dynamic transitions have a less pronounced impact on performance in this game, with the most critical time step being the one at which the paddle must bounce the ball. We hypothesize *2M-RL* can help *2M-EC* escape local optima and then *2M-EC* can rapidly learn improved solutions and continue to progress. To test this hypothesis, we need to check whether a *2M* agent that does not share collected data between both memories indeed 1) has worse performance, and 2) prefers RL in the long run, instead of getting to a more optimal EC solution.

Figure 5.7 shows the performance of an *2M* agent with data sharing enabled and disabled. With data sharing (2Mw/DaS), the agent mostly prefers to use EC during evaluation (top-left figure), as expected from Figure 5.6. When we deactivate data sharing (2Mw/oDaS, two memories are only trained using data collected by the corresponding memory separately), the *2M* agent only prefers *2M-EC* at the beginning

and then sticks to *2M-RL* (bottom-left graph of the figure). The performance graph on the right of the figure confirms these results. Without data sharing, 2M does not reach the same performance (blue line stays above the orange line). The circles show the performance of *2M-EC* at the end of training for both methods. Without data sharing, *2M-EC* (the orange circle in Figure 5.7) converges to a sub-optimal solution. With data sharing enabled, *2M-EC* (the blue circle in Figure 5.7) has a way higher performance. This observation provides evidence to support the aforementioned notion that *2M-RL* and *2M-EC* complement eachother.

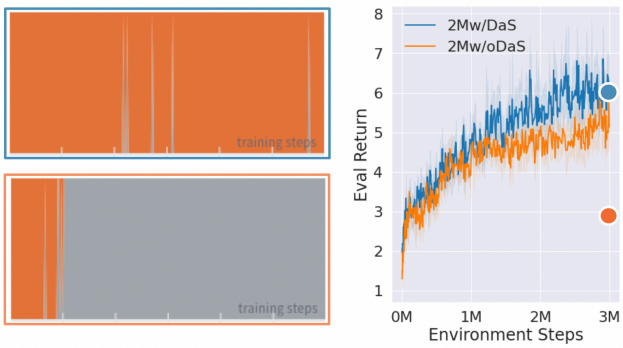


Figure 5.7: Left: Switching schedule of *2M* agent with data sharing (top) and without data sharing (bottom) during evaluation on Breakout. **Right**: Returns two different *2M* agents are able to get in evaluation. The final performance of *2M-EC* is represented by coloured circles. We see that with data sharing, EC reaches a return of 6, while without data sharing, EC only manages to reach a return of 3.

5.5.3 Ablation Study

In this section, we conduct two groups of ablation experiments to study the design choices in our work. First, we would like to investigate the impacts of data sharing. Deactivating *data sharing* (2Mw/oDS), resulting in *2M-RL* being solely trained on data collected by *2M-RL* and *2M-EC* being solely trained on data collected by *2M-EC*. This transforms our proposed method becomes a 'scheduler' that schedules the training between two distinct models and uses the better one for evaluation. Second, we aim to study different ways of scheduling p^{ec}. Specifically, we examine three different scheduling approaches: decayed scheduling (2Mw/DS), constant scheduling (2Mw/CS) and increased scheduling (2Mw/IS).

Intuitively, data sharing can be helpful since each memory will get different data from another memory, hopefully, they could also learn from each other. It should

result in a better performance compared to only training the agent on its own collected data. In fact, such sharing has different impacts on different games, shown in Figure 5.8. In Seaquest, data sharing improves *2M* agent's performance, and harms the performance of the agent in Asterix. To understand the reasons for these opposite impacts, we separately track the performance of *2M-EC* and *2M-RL* during the training, and final performance are represented by circles and triangles in Figure 5.8 and larger size represents more use of *2M-EC* (larger p^s) during the training. In Asterix, data sharing pulls down the performance of *2M-RL* (blue triangles are always below orange ones), and training *2M-RL* on more data collected by *2M-EC* leads to even worse performance (the large blue triangle is way below the large orange one), indicating that data collected by *2M-EC* is actually harmful for training *2M-RL* in this game. Conversely, in Seaquest, data sharing improves the performance of *2M-EC* (blue circles are always above orange ones), which again indicates that *2M-RL* can help *2M-EC* escape from local optima. However, overusing data collected by *2M-EC* to train *2M-RL* also leads to worse performance (the large blue triangle is way below the large orange one). All in all, *2M-RL* should not use too much data collected by *2M-EC*. Although we show that such training is helpful to learn the optimal state-action values when collected data is correlated to the optimal policy, this assumption is not always satisfied. Meanwhile, *2M-RL* can help *2M-EC* escape from local optima but not always. We presume it helps when stochastic transitions have fewer negative impacts, then *2M-EC* is able to catch improved solutions provided by *2M-RL*. For example, in Asterix, there are many enemies and the agent dies immediately when it touches enemies, meaning the agent will more likely die if a single wrong (sub-optimal) action is made. Therefore, once *2M-EC* discovers a sub-optimal trajectory with a very high return (luckily manage to survive for a long time, like ending up with s_6 in the motivating example in Figure 5.2) with a tiny probability, it will stick to it. Then it is difficult for *2M-EC* to escape from local optima even though improved solutions are provided. We leave more systematic investigations on this phenomenon for future work.

Next we examine how different scheduling mechanisms affect performance. The *2M* agent with decayed scheduling (*2Mw/DS*) will initially give a higher preference to use *2M-EC* for data collection during the training, then gradually shifts towards *2M-RL*. On the contrary, increased scheduling (*2Mw/IS*) will start with a strong preference for using *2M-RL*, then gradually switch to *2M-EC*. Constant scheduling maintains a constant preference for *2M-EC* and *2M-RL* throughout the training process. Given that *2M* agents with *2Mw/DS* perform the best or one of the best on all games,

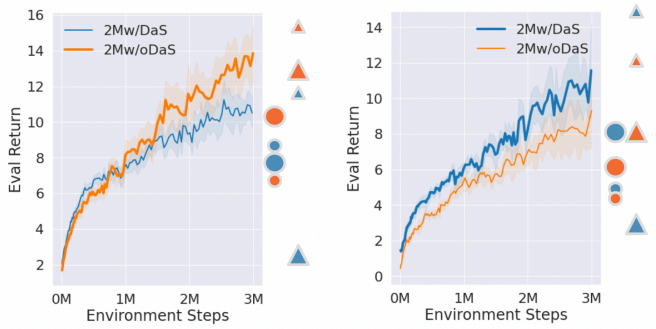


Figure 5.8: Performance of 2M agents with and without data sharing on Asterix (left) and Seaquest (right). Results are averaged over 2 different settings, one is with larger p^{ec} and one is with smaller p^{ec}. Data sharing has different impacts on these two games. Circles represent the final performance of *2M-EC* while triangles represent the performance of *2M-RL*. The larger size means the larger value of p^{ec} during the training.

we only present the performance on Seaquest as a representative in Figure 5.9. The results demonstrate that the agent with *2Mw/DS* outperforms the other two scheduling mechanisms explicitly.

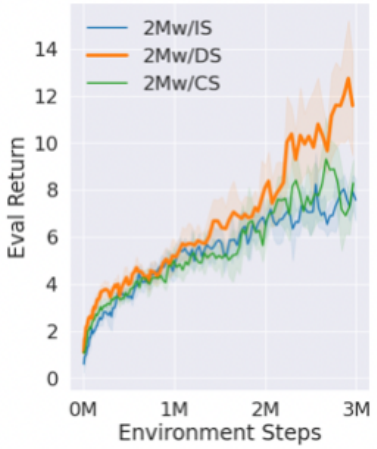


Figure 5.9: Performance of *2M* agents with different scheduling mechanisms on Seaquest. The one with the decayed scheduling mechanism works the best while other two mechanisms have similar performance.

To sum up, our experimental results demonstrate that the *2M* agent surpasses its constituent components, i.e. a pure reinforcement learning approach (Q-learning in tabular settings and DQN (Mnih et al., 2013) in deep RL settings) and a pure episodic

memory approach (MFEC (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016) in both tabular and deep RL settings). Furthermore, our proposed method exhibits better performance than a state-of-the-art memory-augmented reinforcement learning method EMDQN (Lin et al., 2018). Since the proposed framework allows for the integration of various pure EC and RL methods, we only compare the performance of the *2M* agent with those methods that are integrated into it in this work. Therefore, we do not compare our results with other pure episodic memory and reinforcement learning approaches. Lastly, we conduct ablation studies to investigate the impact of two essential design choices: the utilization of data sharing for the mutual improvement of the memories, and the scheduling of p^{ec} for switching between the two memories.

5.6 Conclusion and Future Work

In this work, we proposed a novel approach, termed **Two-Memory (*2M*) reinforcement learning** agent, which integrates two distinct learning methods, namely non-parametric episodic memory and (parametric) reinforcement learning. This approach capitalizes on the advantages of both methods by leveraging the episodic memory's rapid starting and the reinforcement learning's superior generalization and optimality to create an agent that achieves higher data efficiency than the individual methods. Experimental results show that *2M* matches or outperforms both DQN and EMDQN on a range of MinAtar games. In addition, we show that these two distinct memory modules may complement eachother, leading to even better final performance.

For future work, it would be interesting to automatically determine when data sharing is useful, for example based on the discrepancy between both memories. Another clear direction to improve the *2M* agent could be an adaptive scheduling mechanism to switch between *2M-EC* and *2M-RL*, instead of the hand-designed decay schedules used in this work. Moreover, combining stronger episodic memory methods (such as NEC) with off-policy reinforcement learning methods could lead to further improvements in performance. In our work, we uniformly replay data from the replay buffer to train the *2M-RL* component, but a more sophisticated replay strategy, such as prioritized experience replay (PER), may further enhance performance. Overall, our proposed approach provides a general framework for combining two fundamentally different approaches to sequential decision-making, combining their respective strengths.

Chapter 6

First Go, then Post-Explore: the Benefits of Post-Exploration in Intrinsic Motivation

Authors: Zhao Yang, Thomas Moerland, Mike Preuss, Aske Plaat

Published in 15th International Conference on Agents and Artificial Intelligence (ICAART), 22-24 February 2023, Lisbon, Portugal.

Abstract

Go-Explore achieved breakthrough performance on challenging reinforcement learning (RL) tasks with sparse rewards. The key insight of Go-Explore was that successful exploration requires an agent to first return to an interesting state ('Go'), and only then explore into unknown terrain ('Explore'). We refer to such exploration after a goal is reached as 'post-exploration'. In this paper, we present a clear ablation study of post-exploration in a general intrinsically motivated goal exploration process (IMGEP) framework, that the Go-Explore paper did not show. We study the isolated potential of post-exploration, by turning it on and off within the same algorithm under both tabular and deep RL settings on both discrete navigation and continuous control tasks. Experiments on a range of MiniGrid and Mujoco environments show that post-exploration indeed helps IMGEP agents reach more diverse states and boosts their performance. In short, our work suggests that RL researchers should consider using post-exploration in IMGEP when possible since it is effective, method-agnostic, and easy to implement.

6.1 Introduction

Go-Explore (Ecoffet et al., 2021) achieved breakthrough performance on challenging reinforcement learning (RL) tasks with sparse rewards, most notably achieving state-of-the-art, 'super-human' performance on Montezuma's Revenge, a long-standing challenge in the field. The key insight behind Go-Explore is that proper exploration should be structured in two phases: an agent should first attempt to get back to a previously visited, interesting state ('Go'), and only then explore into new, unknown terrain ('Explore'). Thereby, the agent gradually expands its knowledge base, an approach that is visualized in Figure 6.1. We propose to call such exploration after the agent reached a goal *post-exploration* (to contrast it with standard exploration).

There are actually two variants of Go-Explore in the original paper: one in which we directly reset the agent to an interesting goal (*restore-based* Go-Explore), and one in which the agent has to act to get back to the proposed goal (*policy-based* Go-Explore). In this work, we focus on the latter approach, which is technically part of the literature on intrinsic motivation, in particular intrinsically motivated goal exploration processes (IMGEP (Colas et al., n.d.)). Note that post-exploration does not require any changes to the IMGEP framework itself, and can therefore be easily integrated into other existing work in this direction.

While Go-Explore gave a strong indication of the potential of post-exploration, it did not structurally investigate the benefit of the approach. Go-Explore was compared to other baseline algorithms, but post-exploration itself was never switched on and off in the same algorithm. Thereby, the isolated performance gain of post-exploration remains unclear.

Therefore, the present paper performs an ablation study of post-exploration in both tabular and deep RL settings on both discrete and continuous tasks. Experiments in a range of MiniGrid and Mujoco tasks show that post-exploration provides a strong isolated benefit over standard IMGEP algorithms. As a smaller contribution, we also cast Go-Explore into the IMGEP framework, and show how it can be combined with hindsight experience replay (HER) (Andrychowicz et al., 2017) to make more efficient use of the observed data. In short, our work presents a systematic study of post-exploration and shows its effectiveness on both discrete navigation tasks and continuous control tasks.

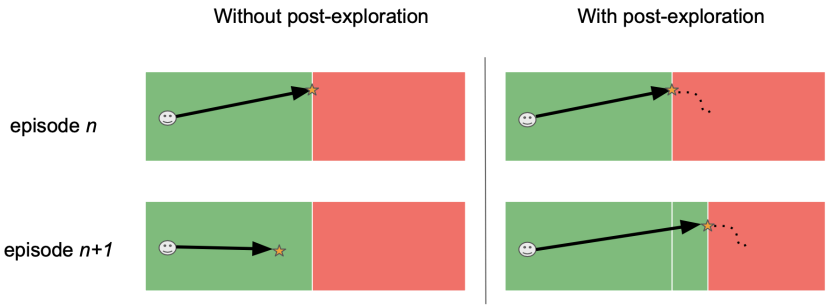


Figure 6.1: Conceptual illustration of post-exploration. Each box displays the entire state space, where green and red denote the (currently) explored and unexplored regions, respectively. **Left**: Goal-based exploration without post-exploration. The top graph shows the agent reaching a current goal, after which the episode is terminated (or the next goal in the green area is chosen). Therefore, in the next episode (bottom) the agent will again explore within the known (green) region), often leaving the unknown (red) area untouched. **Right**: Goal-based exploration with post-exploration. The top figure shows the agent reaching a particular goal, from which the agent now post-explores for several steps (dashed lines). Thereby, the known area (green) is pushed outwards. On the next episode, the agent may now also select a goal in the expanded region, gradually expanding its knowledge of the reachable state space.

6.2 Related Work

Go-Explore is a variant of intrinsically motivated goal exploration processes (IMGEP) (surveyed by Colas et al. (n.d.)) which generally consists of three phases: 1) defining/learning a goal space, 2) sampling an interesting goal from the goal space, and 3) getting to the sampled goal based on knowledge from previous episodes. Regarding the first step, Go-Explore (Ecoffet et al., 2021) pre-defines goals as down-scaled states, and the goal space is adaptively extended by adding new encountered states. Other IMGEP approaches, like Goal-GAN (Florensa et al., 2018) uses a generative adversarial network to learn a goal space of appropriate difficulty. As an alternative, AMIGo (Campero et al., 2021) trains a teacher to propose goals for a student to reach. In this work, we largely follow the ideas of Go-Explore, and pre-define the goal space as the visited state space, since our research questions are directed at the effects of post-exploration.

After a goal space is determined, we need a sampling strategy to set the next goals. Generally, desired next goals should be novel/diverse enough or lie on the border of the agent's knowledge boundary. Example strategies to select next goals include novelty (Ecoffet et al., 2021; Pitis et al., 2020), learning progress (Colas et al., 2019;

Table 6.1: Overview of moment of exploration in IMGEP papers. Most IMGEP approaches, like Goal-GAN (Florensa et al., 2018), CURIOUS (Colas et al., 2019), DIAYN (Pong et al., 2019), MEGA (Pitis et al., 2020) and AMIGo (Campero et al., 2021), only explore during goal reaching. Restore-based Go-Explore, which directly resets to a goal, only explore after the goal is reached. Our work, similar to policy-based Go-explore, explores both during goal-reaching and after goal reaching.

Approach	Exploration	Post-exploration
IMGEP	✓	✗
Restore-based Go-Explore	✗	✓
Policy-based Go-Explore + this paper	✓	✓

Portelas et al., 2020), diversity (Pong et al., 2019; Warde-Farley et al., 2019), and uniform sampling (Eysenbach et al., 2018). In this work, we use novelty (count-based) for sampling, since it has less tuneable hyper-parameters.

In the third IMGEP phase, we need to get back to the proposed goal. One variant of Go-Explore simply resets the agent to the desired goal, but this requires an environment that can be set to an arbitrary state. Instead, we follow the more generic 'policy-based Go-Explore' approach, which uses a goal-conditioned policy network to get back to a goal. The concept of goal-conditioning, which was introduced by (Schaul, Horgan, et al., 2015), is also a common approach in IMGEP approaches. A well-known addition to goal-based RL approaches is hindsight experience replay (Andrychowicz et al., 2017), which allows the agent to make more efficient use of the data through counterfactual goal relabelling. Although Go-Explore did not use hindsight in their work, we do include it as an extension in our approach.

After we manage to reach a goal, most IMGEP literature samples a new goal and either reset the episode or continue from the reached state. The main contribution of Go-Explore was the introduction of post-exploration, in which case we temporarily postpone the selection of a new goal (Table 6.1). While post-exploration is a new phenomenon in reinforcement learning literature, it is a common feature of most planning algorithms, where we for example select a particular node on the frontier, go there first, and then unfold the search tree into unknown terrain.

6.3 Background

We adopt a Markov Decision Processes (MDPs) formulation defined as the tuple $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ (Sutton & Barto, 2018). Here, $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions the agent can take, P specifies the transition function of the environment, and R

is the reward function. At timestep t, the agent observes state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, after which the environment returns a next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and associated reward $r_t = R(s_t, a_t, s_{t+1})$. We act in the MDP according to a policy $\pi(a|s)$, which maps a state to a distribution over actions. When we unroll the policy and environment for T steps, define the cumulative reward (return) of the resulting trace as $\sum_{k=0}^{T} \gamma^k \cdot r_{t+k+1}$, where $\gamma \in [0, 1]$ denotes a discount factor. The goal in RL is to learn a policy that can maximize the expected return.

Define the state-action value $Q^\pi(s, a)$ as the *expected* cumulative reward under some policy π when we start from state s and action a, i.e.,

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi,P}\Big[\sum_{k=0}^{T} \gamma^k \cdot r_{t+k} | s_t = s, a_t = a\Big] \tag{6.1}$$

Our goal is to find the *optimal* state-action value function $Q^*(s, a) = max_\pi Q^\pi(s, a)$, from which we may at each state derive the optimal policy $\pi^*(s) = argmax_a Q^*(s, a)$. A well-known RL approach to solve this problem is *Q-learning* (C. J. Watkins & Dayan, 1992). Tabular Q-learning maintains a table of Q-value estimates $\hat{Q}(s, a)$, collects transition tuples (s_t, a_t, r_t, s_{t+1}), and subsequently updates the tabular estimates according to

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)] \tag{6.2}$$

where $\alpha \in [0, 1]$ denotes a learning rate. Under a policy that is greedy in the limit with infinite exploration (GLIE) this algorithm converges on the optimal value function (C. J. Watkins & Dayan, 1992).

In deep RL, state-action value function $Q(s, a)$ is approximated by a parameterized neural network, denoted by $Q_\phi(s, a)$. While dealing with continuous action space, taking *max* operation over state-action values like in Equation (6.2) is intractable, we could learn a parameterized deterministic policy μ_θ for the agent instead. A deterministic policy maps a state to a deterministic action rather than a distribution over all actions. Then we can approximate $\max_a Q_\phi(s, a)$ with $Q_\phi(s, \mu_\theta(s))$.

$$L(\phi, D) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim D}\Big(r_t + \gamma \cdot \hat{Q}(s_{t+1}, \mu(s_{t+1})) - \hat{Q}(s_t, a_t)\Big)^2 \tag{6.3}$$

Deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015) is an actor-critic method that parameterizes the deterministic policy ('actor') and the state-action value function ('critic') as neural networks. The critic is updated by minimizing the mean squared TD error $L(\phi, D)$ in Equation (6.3), where D is a set of collected transitions. The actor is updated by solving the Equation (6.4).

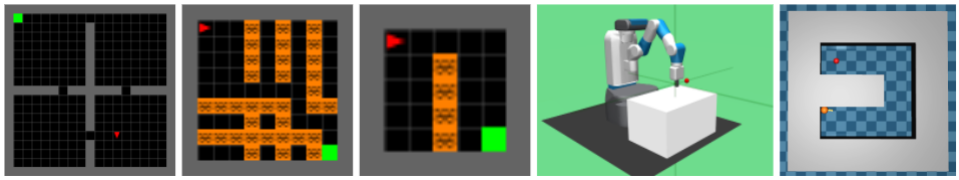


Figure 6.2: FourRooms, LavaCrossing, LavaGap, FetchReach, and PointUMaze. Room layouts of two lava environments are procedurally generated.

$$\max_{\theta} \mathbb{E}_{s_t \sim D} \Big[Q_\phi(s_t, \mu_\theta(s_t)) \Big] \tag{6.4}$$

6.4 Methods

We will first describe how we cast Go-Explore into the general IMGEP framework (Section 6.4.1), and subsequently introduce how we do post-exploration (Section 6.4.2).

6.4.1 IMGEP

Our work is based on the IMGEP framework shown in Algorithm 3. Since this paper attempts to study the fundamental benefit of post-exploration, we try to simplify the problem setting as much as possible, to avoid interference with other issues (such as goal space representation learning, goal sampling strategy design, reward function design, etc). We, therefore, define the goal space as the set of states we have observed so far. For continuous tasks, we discretize the whole state space into bins, so the goal space for continuous control tasks is the set of bins we have observed so far. Firstly, a bin will be sampled, then we further uniformly sample a goal from that bin. The goal space G is initialized by executing a random policy for one episode, while new states/bins in future episodes augment the set.

For goal sampling, we use a count-based novelty sampling strategy which aims to sample goals that are less visited more often and more visited less often from the goal space. We therefore track the number of times $n(g)$ we visited a particular goal g, and specify the probability of sampling it $p(g)$ as

$$p(g)_{g \in G} = \frac{(\frac{1}{n(g)})^\tau}{\sum_{g \in G} (\frac{1}{n(g)})^\tau}. \tag{6.5}$$

Algorithm 3 IMGEP with post-exploration and hindsight (blue part is post-exploration)

Initialize: Goal-conditioned RL agent RL, environment `Env`, episode memory M, goal space G, goal sampling probability $p(g)$, goal-conditioned reward function $R_g(s, a, s')$.
while training budget left **do**
  $g \sim p(g)$ {Sample a goal g from the goal space G according to the goal sampling probability $p(g)$.}
  $s =$ `Env`.reset() {Reset the environment.}
  Execute a roll-out using RL and collect transitions T along the way.
  Update G and $p(g)$ using T. {Extend G by adding new encountered states from T.}
  Store collected transitions T to M.
  if g is reached **then**
    Execute a fixed number of steps using the random policy and collect transitions T_r along the way.
    Update G and $p(g)$ using transitions T_r.
    Store collected transitions T_r to M.
  end if
  Sample a batch B from M and hindsight relabel according to the reward function R_g.
  Update RL agent RL using batch B.
end while

Here, τ is a temperature parameter that allows us to scale goal sampling. When τ is 0, goals will be sampled uniformly from the goal space, while larger τ will emphasize less visited goals more.

Note that we could also use other methods, both for defining the goal space and for sampling new goals (for example based on curiosity or learning progress).

To get (back) to a selected goal, we train a goal-conditioned policy, for tabular settings we use Q-learning, and for deep RL settings we use DDPG. Agents are trained on the goal-conditioned reward function R_g, which is a one-hot indicator that fires when the agent manages to reach the specified goal:

$$R_g(s, a, s') = 1_{s'=g}. \tag{6.6}$$

Note that different goal-conditioned rewards functions, and different update methods are of course possible as well.

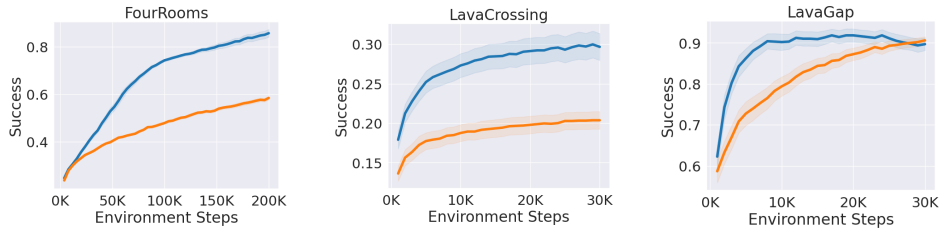


Figure 6.3: IMGEP agents with (blue) and without (orange) post-exploration. **Left**: Performance on FourRooms, with results averaging over three different values of ϵ $(0, 0.1, 0.3)$. **Middle**: Performance on LavaCrossing, averaged over 10 different environment seeds. **Right**: Performance in LavaGap, averaged over 10 different environment seeds. Overall, agents with post-exploration outperform agents without post-exploration.

6.4.2 Post-exploration

The general concept of post-exploration was already introduced in Figure 6.1. The main benefit of this approach is that it increases our chance to step into new, unknown terrain. Based on this intuition, we will now discuss when and how to post-explore, as well as how we utilize data collected during the post-exploration.

As is directly visible from Figure 6.1, post-exploration is aiming to expand the agent's knowledge boundary further when necessary. Since the less visited goals are sampled more often, when a selected goal is reached by the agent, that means the agent's knowledge boundary (the ability to reach goals) is already near the selected goal, then we decide to expand the boundary by doing post-exploration. In this work, we simply let the agent post-explore every time the selected goal is reached. More constraints can be applied here, such as only doing post-exploration when the reached goal is lying on the border of the unknown area or it is the bottleneck of a graph of the visited states, etc. We leave the design of smarter strategies for future work.

We assume that post-exploration could lead the agent to step into new naive areas, thus it is important to design an effective way to do post-exploration. In Go-Explore (Ecoffet et al., 2021), they mentioned random post-exploration is a simple and effective way. So in this work, we also use the random policy during the post-exploration. The agent will take a fixed number of random actions for post-exploration.

The agent in principle only observes a non-zero reward when it successfully reaches the goal, which may not happen at every attempt. We may improve sample efficiency (make more efficient use of each observed trajectory) through *hindsight experience relabelling* (HER) (Andrychowicz et al., 2017). With hindsight, we imagine states in our observed trajectory were the goals we actually attempted to reach, which allows us

to extract additional information from them ('if my goal had been to get to this state, then the previous action sequence would have been a successful attempt'). For tabular settings, we choose to always relabel 50% of the entire trajectory (goal reaching plus post-exploration), i.e., half of the states in each trajectory are imagined as if they were the goal of that episode. We choose to always relabel the full post-exploration part, because it likely contains the most interesting information, and randomly sample the remaining relabelling budget from the part of the episode before the goal was reached. For deep RL settings, we use the 'future' relabel strategy from the original HER work (Andrychowicz et al., 2017).

6.5 Experiments

6.5.1 Experimental Setup

We test our work on three MiniGrid environments (Chevalier-Boisvert et al., 2018) (tabular settings), and two Mujoco environments (Kanagawa, 2021; Plappert et al., 2018) (with using the function approximation), visualized in Figure 6.2. Results on two lava environments are averaged over 10 seeds, i.e., 10 independently drawn instances of the task. For evaluation, not like in Go-Explore where the agent needs to reach a specific final goal (the state with the highest score), we instead test the ability of the agent to reach every possible state in the whole state space. This checks to what extent the goal-conditioned agent is able to reach a given goal, when we execute the greedy policy (turn exploration off). All our results report the total number of environment steps on the x-axis. Therefore, since an episode with post-exploration takes longer, it will also contribute more steps to the x-axis (i.e., we report performance against the total number of unique environment calls). Curves display mean performance over time, including the standard error over 5 repetitions for each experiment (10 repetitions for DRL experiments).

6.5.2 Hyper-parameter Settings

Tabular Q-learning

All hyper-parameters we used for tabular Q-learning in this work are shown in Table 6.2. Learning rate α and discount factor γ are for Q-learning update (Equation (6.2)). τ is the temperature for goal sampling. (Equation (6.5)). ϵ is the exploration factor in ϵ-greedy policy. We average results over 3 different values of ϵ. p_{pe}

is the percentage of the whole trajectory that the agent will post-explore. $p_{pe} = 0.5$ means the agent will post-explore $0.5 * l$ steps after the given goal is reached, l is the length of the trajectory the agent takes to reach the given goal.

Parameters	Values
learning rate	0.1
discount factor	0.99
τ	0
ϵ	0, 0.1, 0.3
p_{pe}	0.5

Table 6.2: All hyper-parameters we used for tabular Q-learning experiments.

DDPG

All hyper-parameters we used for DDPG agent are shown in Table 6.3. The learning rate α is for both the policy and the critic update. τ is the temperature for goal sampling. (Equation (6.5)). Random ϵ is the probability of taking a random action and noise ϵ is the probability of adding noise to selected actions. n_{pe} is the number of steps that the agent will post-explore. n_{bins} is the number of bins we discretize the whole goal space on each dimension. Batch size is the size of the batch sampled for training every time. Replay k is the portion of data we will relabel. If $k = 4$, that means we will relabel 4/5 of the whole sampled training data. The replay strategy we used here is 'future'.[1]

Parameters	FetchReach	PointUMaze
learning rate	0.001	0.001
discount factor	0.98	0.98
τ	0	0.01
random ϵ	0.01	0.1
noise ϵ	0.01	0.1
n_{pe}	30	50
n_{bins}	20	100
batch size	16	2
replay k	4	4
replay strategy	future	future

Table 6.3: All hyper-parameters we used for DDPG agents.

[1]For the remaining parameters of DDPG agents, we use the default settings from https://github.com/TianhongDai/hindsight-experience-replay.

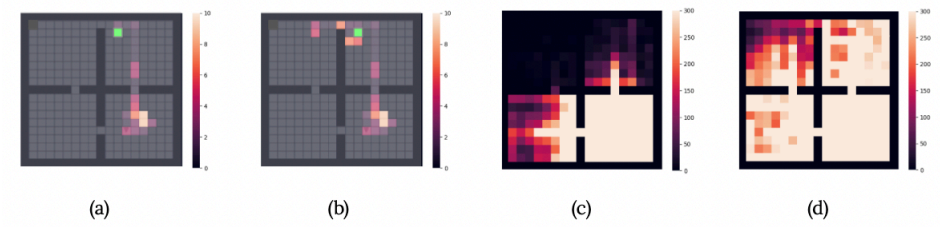


Figure 6.4: Comparison of characteristic traces (a,b) and coverage (c,d) for agents without post-exploration (a,c) and with post-exploration (b,d). Colour bars indicate the number of visitations, green square indicates the selected goal in a particular episode. (a): Standard exploration towards a goal without post-exploration. (b): With post-exploration, the agent manages to reach the next room. (c): Coverage after 200k training steps without post-exploration. (d): Coverage after 200k training steps with post-exploration. The boundary of the coverage with post-exploration clearly lies further ahead.

6.5.3 Results

Figure 6.3 compares an IMGEP agent with post-exploration to an IMGEP agent without exploration in the three MiniGrid environments. On all three environments, the agent with post-exploration outperforms the baseline agent significantly (on its average ability to reach an arbitrary goal in the state space).

A graphical illustration of the effect of post-exploration is provided in Figure 6.4. Part a shows a characteristic trace for an agent attempting to reach the green square *without* post-exploration, while b shows a trace for an agent *with* post-exploration. In a, we see that the agent is able to reach the goal square, but exploration subsequently stops, and the agent did not learn anything new. In part b, the agent with post-exploration subsequently manages to enter the next new room. This graph, therefore, gives a practical illustration of our intuition from Figure 6.1.

To further illustrate this effect, in Figure 6.4, c and d show a visitation heat map for the agent after 200k training steps, without post-exploration (c) and with post-exploration (d). The agent with post-exploration (c) has primarily explored goals around the start region (in the bottom-right chamber). The two rooms next to the starting room have also been visited, but the agent barely managed to get into the top-left room. In contrast, the agent with post-exploration (d) has extensively visited the first three rooms, while the coverage boundary has also been pushed into the final room already. This effect translates to the increased goal-reaching performance of post-exploration visible in Figure 6.3. We think such a principle holds in other environments as well.

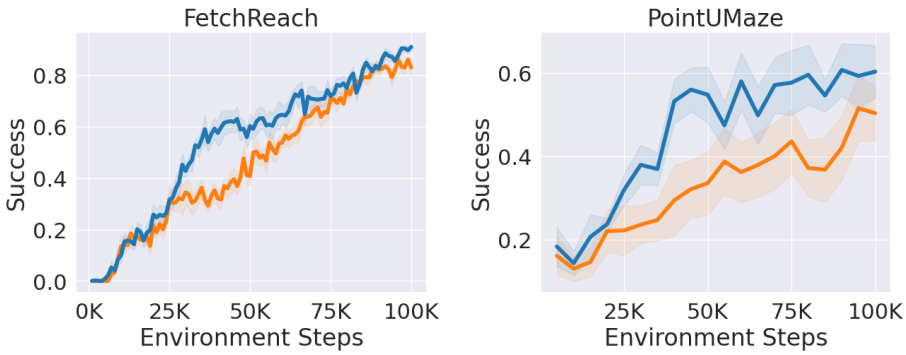


Figure 6.5: IMGEP agents with (blue) and without (orange) post-exploration. **Left**: Performance on FetchReach, with results averaging over 10 different random seeds. **Right**: Performance on PointUMaze, averaged over 10 different random seeds. Overall, agents with post-exploration outperform agents without post-exploration.

Comparisons between agents with post-exploration and ones without on Mujoco environments are shown in Figure 6.5 and we use the DDPG agent with HER as the baseline (orange lines in Figure 6.5) to compare with. With post-exploration, agents (blue lines in Figure 6.5) outperform baseline agents in both FetchReach and PointUMaze environments. We observed that at the beginning and the end, two agents (with post-exploration and without) tend to have similar performances. We interpret the phenomenon as follows: the agent is not able to reach any selected goals in the beginning so there are no post-exploration and no benefits; then gradually the agent starts being able to reach selected goals and post-exploration comes in thus further improving the performance; in the end, when the whole state space is been entirely discovered post-exploration will not add benefits anymore. We think this phenomenon happens in environments where the baseline agent (without post-exploration) can also fully discover the whole state space over time, however, it would less happen in environments where there are many bottlenecks so that the baseline agent can hardly fully discover the entire state space.

By adding post-exploration, the agent is more likely to step into new unknown areas and thus can further improve the state coverage. Ideally, we want the agent's visitation of the state space to be as uniform as possible. More specifically, the higher entropy indicates that the visitation of the whole state space is more uniform. The

entropy is computed using the equation below:

$$-\sum_{i=1}^{N} p(x_i)\log(p(x_i)) \tag{6.7}$$

where the whole state space is discretized into N bins and $p(x_i) = n(x_i)/\sum_{i=0}^{N} n(x_i)$, $n(x_i)$ is the number of visitations of the i^{th} bin. Figure 6.6 shows the entropy on agents' visitation of the state space during the training. With post-exploration (blue lines), the entropy always lies above the one without (orange lines). Therefore, the principle (Figure 6.4) that we saw in the discrete navigation tasks is still true in continuous control tasks.

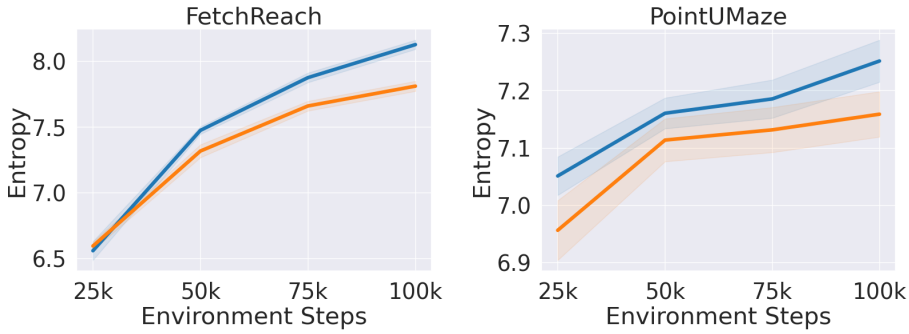


Figure 6.6: The entropy of the visitation on the state space for the agent with (blue) and without (orange) post-exploration. **Left**: Entropy on FetchReach, with results averaging over 10 different random seeds. **Right**: Entropy on PointUMaze, averaged over 10 different random seeds. Adding post-exploration can increase the entropy of the agent's visitation on the state space.

6.6 Conclusion and Future Work

An intrinsically motivated agent not only needs to set interesting goals and be able to reach them but should also decide whether to continue exploration from the reached goal ('post-exploration'). In this work, we systematically investigated the benefit of post-exploration in the general IMGEP framework under different RL settings and tasks. Experiments in several MiniGrid and Mujoco environments show that post-exploration is not only beneficial in navigation tasks under tabular settings but also can be scaled up to more complex control tasks with neural networks involved. Ac-

cording to our results, agents with post-exploration gradually push the boundaries of their known region outwards, which allows them to reach a greater diversity of goals. Moreover, we realize that 'post-exploration' is a very general idea and is easy to be plugged into any IMGEP method. Researchers should put more attention to this idea and consider using it when possible.

The current paper studied post-exploration in a simplified IMGEP framework, to better understand its basic properties. In the future, it would be interesting to plug post-exploration into other existing IMGEP methods directly to show its benefits. Moreover, our current implementation uses random post-exploration, which turned out to already work reasonably well. So an interesting direction for future work is to post-explore in a smarter way, like using macro actions or options. For example, in tasks where we need to control a more complex agent such as an ant or a humanoid robot, then random actions will never help the agent stand up properly and it is even harder to lead the agent step into new areas. Another promising future direction is to investigate the 'adaptive' post-exploration. Intuitively, post-exploration will be most likely useful when it starts from a state that is new or important enough and the agent should post-explore more if the reached area is more naive, etc. In short, the agent should adaptively decide when and for how long to post-explore based on its own knowledge boundary or properties of reached goals. Altogether, post-exploration seems a promising direction for future RL exploration research.

Chapter 7

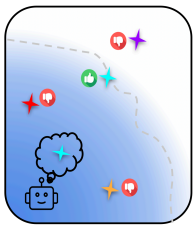
Conclusion

Goal-conditioned reinforcement learning is pivotal in training a generalist agent. Throughout this thesis, several new methods are proposed to enhance the goal-conditioned reinforcement learning framework. Whereas these methods show superior performance compared to the baselines, they are not without limitations. In this chapter, we first answer all research questions that are asked at the beginning, and then discuss the limitations of the methods we proposed. Then, we zoom out and provide a more general reflection on the current goal-conditioned reinforcement learning framework and how we can potentially improve it in the future. At the end, we summarize the main conclusion.

7.1 Answers to research questions

In Section 1.1, four research questions were formulated. Here, each research question is answered based on the results as described in Chapters 3-7, and the main conclusions for each question are given.

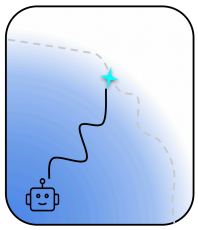
Q1 **Can RL agents learn without access to a 'reset'? [Chapter 3]**



(step 2) Having access to a 'reset' seems natural, but can be expensive in real-world tasks. For instance, training a robotic hand to spin a pen requires human intervention to reset when the pen falls from the hand (i.e. put the pen back in the robotic hand). Since such training schemes that require humans in the loop are infeasible to scale up, ideally, we want agents to operate with minimal human effort, i.e. without a reset (reset-free).

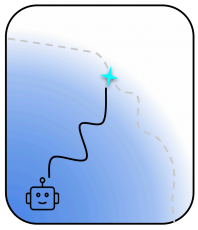
However, eliminating reset imposes challenges on exploration. We show model-based reinforcement learning (MBRL) methods can be applied 'out-of-the-box' to the reset-free setting and outperform state-of-the-art model-free methods due to their sophisticated exploration and high data efficiency, while requiring less human effort, such as environmental reward function or demonstrations. We then identify that applying MBRL methods directly causes over-exploration, i.e. the agent will squander a significant amount of time on 'task-irrelevant' states. To overcome over-exploration of MBRL methods, we propose a model-based reset-free agent (MoReFree), which biases exploration and policy training towards 'task-relevant' states to get better performance. More specifically, during the data collection in the real environment, MoReFree is commanded on three different state distributions, i.e. a goal state distribution and an initial state distribution for collecting more 'task-relevant' data, and an exploratory state distribution to encourage better exploration and collect novel data. During imagination training, the policy is explicitly trained to reach the initial and goal state distributions. In short, MBRL methods show promising results under the reset-free setting due to their sophisticated exploration and high data efficiency, and biasing towards task-relevant states during both data collection and policy training creates a synergistic cycle for MBRL methods, resulting in even stronger performance.

Q2 **Can non-parametric methods be applied to tasks with a continuous action space to achieve faster learning? [Chapter 4]**



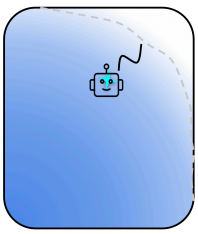
(step 3) Non-parametric methods, in our case, episodic control methods, store solutions in tabular forms and maintain values (episodic returns) for each possible action separately. Consequently, they can in principle only handle tasks with a discrete action space. We approach the proposed research question by maintaining both the action and the value for a state, replacing the existing action and its corresponding value with the newly encountered action which has a higher value. Since a continuous state will never be encountered twice, during action selection, we use the k-nearest neighbors algorithm to make an approximate estimation for the new coming state. We first find its k-nearest neighbor states, and then actions attached to neighbors with higher values are more likely to be selected. Our experimental results show that the proposed method achieves faster learning speed and better performance compared with deep RL baselines in various continuous control tasks.

Q3 **Can non-parametric and parametric methods be combined to achieve both fast learning and optimality? [Chapter 5]**



(step 3) Since episodic control methods replace existing values using newly encountered better ones, only the best solution the agent ever discovered is stored, even if such a solution only occurs with low probabilities. Consequently, although they learn faster, solutions stored by episodic control are non-optimal in stochastic tasks. RL, on the other hand, learns slowly but can handle stochasticity. We combine these two approaches together to form a single agent called the Two-Memory (2M) agent. For each episode during training, either episodic control or RL is selected to collect data, which is then used to train both of them. During evaluation, the one with better historical performance is used for action selection. Quantitative results show that 2M outperforms both sole RL and episodic control agents in five simplified Atari games, illustrating the success and effectiveness of such a combination. Interestingly, qualitative results show that initially, the 2M agent prefers to select the episodic control agent for data collection since it learns reasonably good behaviors quickly. But gradually, the RL agent catches up on the performance while maintaining better solutions, and thus the 2M agent switches to RL for data collection. Thereby, by switching between two methods and employing the better one for evaluation, 2M achieves both fast learning and optimality.

Q4 **What benefits can post-exploration introduce compared to not having it? [Chapter 6]**



(step 4) Post-exploration is defined as the exploration that occurs after the selected goal is reached. The intuition behind post-exploration is that the agent should explore when it is in states where exploration is beneficial. For instance, if the agent reaches frontier states, performing additional exploration will likely bring the agent into new regions, thus collecting more novel data. Our experimental results show that post-exploration indeed can lead the agent into new, unseen areas, where it can acquire more novel and diverse data. Therefore, the goal space extends to new areas, and the goal-conditioned policy is trained on new goals, resulting in a better performance compared with the agent without post-exploration (which resets directly after reaching the goal), using the same amount of environmental interactions. In conclusion, post-exploration can allocate the environmental interaction budget to more interesting areas, collecting more valuable data and leading to better performance.

7.2 Limitations

Although methods proposed in this thesis show promising results and outperform baselines, they are not without limitations. In each chapter, we already briefly discussed them, and here, we provide a more detailed and thorough discussion.

7.2.1 Limitations of model-based reset-free methods

In Chapter 3, MoReFree shows superior performance compared to baselines in various simulated robotic tasks. However, as a model-based unsupervised agent, it has all the disadvantages associated with model-based unsupervised approaches. Furthermore, evaluating reset-free methods using tasks designed for episodic settings limits their ability to fully showcase their potential. We discuss the limitations of MoReFree from these two perspectives.

Unsupervised MBRL backbone MoReFree is built on an unsupervised MBRL backbone, PEG (E. S. Hu et al., 2023), thus it inherits all drawbacks of both unsu-

pervised RL and MBRL methods. Unsupervised RL methods learn a self-supervised reward function, which is subsequently used to train the behavior policy. Although this eliminates the need for human-defined reward functions, in tasks requiring complex and precise behaviors, such self-supervised reward functions might be inadequate for providing effective learning signals. In these cases, incorporating a sparse reward function, which is generally easy to obtain, into the learned reward function can be beneficial. Similarly, MoReFree is constrained by its model-based backbone. MBRL methods first fit a model on collected data, and then use the model to either train the policy or perform online planning. Consequently, the upper bound of the performance of MBRL methods depends on the accuracy of the learned model. Namely, if the model is inaccurate, the trained policy or the online planning procedure will also be imprecise. Thereby, enhancing the underlying MBRL backbone, such as by employing more accurate world model architectures (Deng et al., 2023; Gu & Dao, 2023) or developing better self-supervised reward functions, will further improve the performance of MoReFree.

Benchmark Results shown in Chapter 3 illustrate the strength of MoReFree on various simulated robotics tasks, which are all adapted from existing tasks designed for the conventional episodic setting. Although these tasks are already quite challenging for RL agents, they are constrained in ways that rely on resets, which are not considered in the reset-free setting. For instance, in the Fetch-Push and Pick&Place tasks, a fixed-location robot cannot reach an object that has fallen to the ground if the object is not reset back to the table. However, if the robot were equipped with motility, it could then navigate around the table, pick up the block from the ground, and successfully complete the task in a reset-free setting. This increased flexibility would bring the simulated tasks closer to real-world scenarios, where robots need to operate in environments with less predefined constraints. To achieve this, we should specifically design tasks for the reset-free setting, instead of directly adapting them from episodic benchmarks.

7.2.2 Limitations of episodic control methods

In Chapter 4 and Chapter 5, we show that non-parametric methods, i.e., episodic control, can latch onto discovered solutions quickly, resulting in better performance than deep RL methods in some scenarios. We now discuss the main limitations of episodic control from three perspectives: generalizability, optimality, and representation. In

the end, we also provide a reflection on why we think it is still worth investigating episodic control methods in the future.

Generalizability Episodic control methods store discovered solutions in a tabular format. Therefore, in continuous tasks where it is impossible to encounter exactly the same state twice, we need mechanisms to retrieve solutions of states that are similar in some way to the newly encountered state. The most commonly used mechanism is the K-Nearest Neighbors algorithm (KNN), which outputs the top-k states that are most similar to the queried state according to a given distance metric (Euclidean distance, cosine similarity, etc). Unfortunately, in high-dimensional spaces, KNN does not work well due to the curse of dimensionality, which might output unreasonable neighbors and limit the generalizability of episodic control methods.

Stochasticity Episodic control methods learn quickly due to their aggressive update rule: instead of taking the expectation over all possible trajectories, they overwrite the existing solution with a better encountered one. Consequently, the best solution ever encountered is stored, even though it might have an extremely low probability of occurring due to stochastic transition dynamics. This limitation is inherent and cannot be eliminated unless combined with other methods. For example, in Chapter 5, we combine episodic control methods with deep RL methods to form one single agent, which gains the ability of dealing stochasticity from the RL side, resulting in better performance.

Representation Unlike parametric methods (e.g. deep RL) where representations are learnable and trained to be informative for learning behavior policies, episodic control methods do not utilize any learnable parameters. Episodic control methods either store original states as representations or employ dimensionality reduction methods like random projection to preprocess the original states. However, these non-parametric dimensionality reduction methods do not scale up well and might filter out information that is important for learning good behaviors. Consequently, Pritzel et al. (2017a) integrates trainable features into episodic control methods, leading to improved performance and highlighting the advantages of using better representations. Furthermore, employing pre-trained features in episodic control methods could be beneficial, as the significant success of pre-trained representations in fields such as computer vision or natural language processing.

Reflection Episodic control methods draw inspiration from episodic memory in the human brain, which refers to the ability to recall specific events, experiences, and situations from one's past (Tulving, 1983). The primary reason we believe these methods are still worth investigating is that humans frequently rely on episodic memory in daily life. For example, when driving home after work, we constantly use episodic memory to recall the route and landmarks along the way to ensure a quick return. More importantly, even when unexpected events occur, such as encountering traffic we have never seen before or meeting unfamiliar people, we can still navigate home correctly and successfully. Apparently, humans demonstrate the ability to handle stochasticity when using episodic memory, a capability that current episodic control methods lack, as seen in Algorithm 1 and Chapter 5. This discrepancy raises questions: Is it because our brain has such good attention mechanisms that it selectively filters out all irrelevant information, such as the new traffic? Or is it because the generalization ability of our brain is so strong that it can adapt to these unseen situations seamlessly? Or is there something even more sophisticated at play? These questions should motivate us to continually explore the potential of better implementation of episodic memory in artificial systems.

7.2.3 Limitations of post-exploration

Post-exploration is exploration after the goal is achieved. In Chapter 6, we made an attempt to study post-exploration and key design choices are made as straightforward as possible:

- **When the agent should post-explore**: post-exploration is only performed when the goal is successfully achieved, otherwise the agent is reset back to the initial state, and the next goal is selected for the agent to start over again.
- **For how long the agent should post-explore**: post-exploration is always performed with a fixed number of steps, which is treated as a hyper-parameter.
- **How the agent should post-explore**: during post-exploration, the agent is always taking random actions.

Although these simple choices allow us to isolate irrelevant factors and better study the properties of post-exploration, they also limit the scope of our analysis. As tasks become more complex, the effectiveness and applicability of the aforementioned simple design choices may need to be reconsidered.

To improve the current post-exploration scheme, these processes should be dynamically adapted. For instance, dynamically determining the number of steps for post-exploration. Intuitively, the areas that take longer to reach deserve more exploration. Or, as in PEG (E. S. Hu et al., 2023), replacing random exploration with more sophisticated exploration strategies mentioned in Chapter 2.

7.3 Reflections on goal-conditioned reinforcement learning

The current goal-conditioned reinforcement learning (GCRL) framework consists of four fundamental steps: defining the goal space, selecting goals for the agent, training the agent to achieve these goals, and post-exploration. However, upon reflection, several considerations arise:

- *Autonomy*: Each step in the GCRL framework relies heavily on human intervention, which hinders autonomy and scalability. The design of crucial components such as the reward function, reset mechanisms, and strategies for goal selection often require human expertise. However, human involvement in these processes introduces bottlenecks, making it challenging to develop agents that can scale.

- *Goal space*: Exploration in the GCRL framework is largely influenced by the commanded goals selected from the predefined goal space. When humans perform goal-conditioned learning, we utilize a broad spectrum of goal types (Berkman, 2018), ranging from abstract to concrete, and from objective to subjective. Humans can also pursue goals that may not yet exist, driven by personal aspirations or creative visions. Importantly, different types of goals motivate individuals in distinct ways. In contrast, the goal space typically defined in the current GCRL framework is too simplistic and often aligns closely with the state space or a latent representation thereof, which may restrict the exploration of the agent, potentially limiting the performance.

- *State space*: A large portion of the current GCRL research focuses on tasks that utilize proprioceptive state spaces. These internal state representations are relatively easy to obtain in simulated environments but challenging to acquire in real-world settings. This limitation restricts the applicability of GCRL methods in real-world applications where only visual perception is available. Consequently, incorporating visual or even multi-modal inputs is essential to enable

GCRL agents to operate effectively in diverse and more realistic environments.

Excessive reliance on human expertise, unimaginative goal space definition, and unrealistic state space representation limit the potential for training GCRL agents at scale in the real world, highlighting the need for improvements in several key areas.

7.4 Future research

As future research of each work has already been discussed in the corresponding chapters separately, here we provide high-level future research directions on the GCRL field.

Recent developments in foundation models (Mu et al., 2024; Touvron et al., 2023) that are trained on internet-scale data have demonstrated a strong ability in generalization and human-level understanding across a wide range of tasks. All these foundation models came out during the course of my PhD study, and they are already being integrated into the RL loop, yielding promising results. For instance, they are demonstrated to be able to design better reward functions (Ma et al., 2023) for complex robotic tasks than human experts or provide bonuses for better exploration (Klissarov et al., 2023). Essentially, the findings of previous studies indicate that it is possible to replace human-designed components with those designed by foundation models, enhancing the overall performance and enabling the RL framework to scale up effectively.

The GCRL framework would greatly benefit from the integration of foundation models, especially given the increased necessity for human design in the training loop. Foundation models can play a crucial role in several aspects of GCRL:

- *Goal selection*: With their extensive understanding across various domains, foundation models are likely to comprehend the tasks at hand, thus they can be used to replace human experts in providing meaningful goals under different contexts for the RL agents to pursue (C. Lu et al., 2024). For instance, multi-modal foundation models like vision-language models (Radford et al., 2021) can be used to propose goals in text space while the robot has a vision input.

- *Reward signals*: Foundation models can also be employed to provide step-wise reward signals for learning. For example, given a goal, they can distinguish good states that are likely to lead to the goal from bad ones that are far off. This distinction can then be used as a reward to train the agent, guiding it to reach the given goal more effectively.

However, most powerful foundation models like ChatGPT (OpenAI, 2024), Claude (Anthropic, 2024) or Gemini (Google, 2024) are close-sourced. This means users can only query these models and cannot access their intermediate output, such as embedding, which limits their utility. In contrast, researchers worldwide are developing open-source domain-specific pre-trained models in areas like robotics (M. Kim et al., 2024; Padalkar et al., 2023), self-driving cars (J. Yang et al., 2024) and MineCraft (Fan et al., 2022). If these initiatives can replicate the success seen with pre-training in natural language processes and computer vision, RL could benefit immensely from such pre-trained models:

- *Pre-trained representations*: As seen in natural language processes and computer vision, pre-trained representations capture the compact and informative features of the original input space. In RL, where learning representations only from reward signals can be time-consuming, the use of such pre-trained representations could bypass the need for encoding steps and facilitate the policy learning. Meanwhile, effective representations often ensure that similar states are also close in the latent space. This property also enables leveraging similarities between representations as reward signals.
- *Dynamic models*: The pre-trained predictive models can also be used as world models (Escontrela et al., 2024). These models are trained to predict the subsequent states given the current state-action pairs and enable zero-shot model-based RL or planning, where agents can simulate future states and plan actions without interactions with the environment.

These applications illustrate just a part of the potential of using foundation models and domain-specific pre-trained models to enhance GCRL agents. As research progresses, we anticipate that even more groundbreaking possibilities will emerge, further improving the efficiency and performance of GCRL agents.

7.5 Conclusion

In the thesis, we proposed four research questions and focused on three parts of the goal-conditioned reinforcement learning framework. First, in Chapter 3, we studied a more autonomous goal-conditioned reinforcement learning setting where there is no access to reset, which poses challenges on exploration. By using world models and selecting various goals to command the agent for data collection, our proposed agent can autonomously operate in reset-free tasks and outperform state-of-the-art baselines.

Once a goal is selected, a goal-conditioned policy is typically trained to reach the goal. Previous research on a non-parametric method, called model-free episodic control (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016), shows that episodic control can quickly latch onto previously discovered solutions and learn faster than deep RL methods which are known to be slow. Instead of training a goal-conditioned policy, episodic control can also be employed to train the agent to reach the selected goal. However, model-free episodic control was designed for tasks with a discrete action space. In Chapter 4, we extend the episodic control to continuous episodic control that can now tackle tasks with a continuous action space, and demonstrate its strong performance on various continuous robotic control tasks. Moreover, limitations of episodic control methods are identified in Chapter 5, namely, they will be non-optimal in stochastic tasks. Then we proposed to combine episodic control with deep reinforcement learning methods to gain from both approaches. Experimental results show that by combining these two methods, the unified agent achieves both the fast learning attributed to episodic control and the optimality attributed to reinforcement learning.

In Chapter 6, we demonstrated that post-exploration is an important mechanism to improve efficiency of GCRL agents. By performing post-exploration, agents successfully step into new, unseen areas and acquire more diverse data, resulting in better performance compared to agents without post-exploration.

The methods proposed in the thesis improved components of the goal-conditioned reinforcement learning framework, including goal selection, policy learning and exploration, consequently enhancing the performance and increasing the autonomy of the entire goal-conditioned reinforcement learning framework. In future work, we hope that the goal-conditioned reinforcement learning framework will serve as a recipe for training a generalist agent, and ultimately, an embodied artificial general intelligent agent.

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., & Zaremba, W. (2017). Hindsight experience replay. *Advances in neural information processing systems*, *30*.

Anthropic. (2024, September). Claude.

Baranes, A., & Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, *61*(1), 49–73.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Berkman, E. T. (2018). The neuroscience of goals and behavior change. *Consulting Psychology Journal: Practice and Research*, *70*(1), 28.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., & Hassabis, D. (2016). Model-free episodic control. *arXiv preprint arXiv:1606.04460*.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J. W., Wierstra, D., & Hassabis, D. (2016). Model-free episodic control. *CoRR*, *abs/1606.04460*.

Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Campero, A., Raileanu, R., Kuttler, H., Tenenbaum, J. B., Rocktäschel, T., & Grefenstette, E. (2021). Learning with amigo: Adversarially motivated intrinsic goals. *International Conference on Learning Representations*.

Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., & Terry, J. (2024). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, *36*.

Chevalier-Boisvert, M., Willems, L., & Pal, S. (2018). Minimalistic gridworld environment for openai gym.

Clayton, N. S., & Dickinson, A. (1998). Episodic-like memory during cache recovery by scrub jays. *Nature*, *395*(6699), 272–274.

Colas, C., Karch, T., Sigaud, O., & Oudeyer, P. (n.d.). Intrinsically motivated goal-conditioned reinforcement learning: A short survey. arxiv 2020. *arXiv preprint arXiv:2012.09830*.

Colas, C., Fournier, P., Chetouani, M., Sigaud, O., & Oudeyer, P.-Y. (2019). Curious: Intrinsically motivated modular multi-goal reinforcement learning. *International conference on machine learning*, 1331–1340.

Colas, C., Karch, T., Sigaud, O., & Oudeyer, P.-Y. (2022). Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: A short survey. *Journal of Artificial Intelligence Research*, *74*, 1159–1199.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, *602*(7897), 414–419.

Deng, F., Park, J., & Ahn, S. (2023). Facing off world model backbones: Rnns, transformers, and s4. *arXiv preprint arXiv:2307.02064*.

Ding, Y., Florensa, C., Abbeel, P., & Phielipp, M. (2019). Goal-conditioned imitation learning. *Advances in neural information processing systems*, *32*.

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

Escontrela, A., Adeniji, A., Yan, W., Jain, A., Peng, X. B., Goldberg, K., Lee, Y., Hafner, D., & Abbeel, P. (2024). Video prediction models as rewards for reinforcement learning. *Advances in Neural Information Processing Systems*, *36*.

Eysenbach, B., Gu, S., Ibarz, J., & Levine, S. (2017). Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*.

Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2018). Diversity is all you need: Learning diverse skills without a reward function.

Eysenbach, B., Zhang, T., Levine, S., & Salakhutdinov, R. R. (2022). Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, *35*, 35603–35620.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., & Anandkumar, A. (2022). Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, *35*, 18343–18362.

Fang, M., Zhou, T., Du, Y., Han, L., & Zhang, Z. (2019). Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, *32*.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R Ruiz, F. J., Schrittwieser, J., Swirszcz, G., et al. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, *610*(7930), 47–53.

Florensa, C., Held, D., Geng, X., & Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. *International conference on machine learning*, 1515–1528.

Fu, J., Korattikara, A., Levine, S., & Guadarrama, S. (2019). From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*.

Google. (2024, September). Gemini.

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Gupta, A., Yu, J., Zhao, T. Z., Kumar, V., Rovinsky, A., Xu, K., Devlin, T., & Levine, S. (2021). Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 6664–6671.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. G. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning, ICML 2018, stockholmsmässan, stockholm, sweden, july 10-15, 2018* (pp. 1856–1865, Vol. 80). PMLR.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2019). Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Hafner, D., Lillicrap, T., Norouzi, M., & Ba, J. (2020). Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*.

Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Haldar, S., Pari, J., Rai, A., & Pinto, L. (2023). Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*.

Hansen, N., Su, H., & Wang, X. (2023). Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*.

Hansen, N., Wang, X., & Su, H. (2022). Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*.

Hartikainen, K., Geng, X., Haarnoja, T., & Levine, S. (2019). Dynamical distance learning for semi-supervised and unsupervised skill discovery. *arXiv preprint arXiv:1907.08225*.

He, Q., Feng, Z., Fang, H., Wang, X., Zhao, L., Yao, Y., & Yu, K. (2023). A blockchain-based scheme for secure data offloading in healthcare with deep reinforcement learning. *IEEE/ACM Transactions on Networking*, *32*(1), 65–80.

Hu, E. S., Chang, R., Rybkin, O., & Jayaraman, D. (2023). Planning goals for exploration. *The Eleventh International Conference on Learning Representations*.

Hu, H., Ye, J., Zhu, G., Ren, Z., & Zhang, C. (2021). Generalizable episodic memory for deep reinforcement learning. *arXiv preprint arXiv:2103.06469*.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., & Fan, L. (2022). Vima: General robot manipulation with multimodal prompts. *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Kanagawa, Y. (2021). Mujoco-maze.

Kim, J., Cho, D., & Kim, H. J. (2023). Demonstration-free autonomous reinforcement learning via implicit and bidirectional curriculum. *International Conference on Machine Learning*.

Kim, J., hyeon Park, J., Cho, D., & Kim, H. J. (2022). Automating reinforcement learning with example-based resets. *IEEE Robotics and Automation Letters*, *7*(3), 6606–6613.

Kim, M., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., & Finn, C. (2024). Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., & Henaff, M. (2023). Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*.

Kompella, V., Walsh, T. J., Barrett, S., Wurman, P., & Stone, P. (2022). Event tables for efficient experience replay. *arXiv preprint arXiv:2211.00576*.

Kuznetsov, I., & Filchenkov, A. (2021). Solving continuous control with episodic memory. *arXiv preprint arXiv:2106.08832*.

Le, H., Karimpanal, T. G., Abdolshah, M., Tran, T., & Venkatesh, S. (2021). Model-based episodic memory induces dynamic hybrid controls. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems 34: Annual conference on neural information processing systems 2021, neurips 2021, december 6-14, 2021, virtual* (pp. 30313–30325).

Lee, S. Y., Sungik, C., & Chung, S.-Y. (2019). Sample-efficient deep reinforcement learning via episodic backward update. *Advances in Neural Information Processing Systems*, *32*.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, *17*(1), 1334–1373.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, Z., Zhao, T., Yang, G., & Zhang, L. (2018). Episodic memory deep q-networks. *arXiv preprint arXiv:1805.07603*.

Liu, M., Zhu, M., & Zhang, W. (2022). Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*.

Lu, C., Hu, S., & Clune, J. (2024). Intelligent go-explore: Standing on the shoulders of giant foundation models. *arXiv preprint arXiv:2405.15143*.

Lu, K., Grover, A., Abbeel, P., & Mordatch, I. (2020). Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*.

Lu, K., Mordatch, I., & Abbeel, P. (2020). Adaptive online planning for continual lifelong learning.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., & Anandkumar, A. (2023). Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., & Pathak, D. (2021). Discovering and achieving goals via world models.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nat.*, *518*(7540), 529–533.

Mu, Y., Zhang, Q., Hu, M., Wang, W., Ding, M., Jin, J., Wang, B., Dai, J., Qiao, Y., & Luo, P. (2024). Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, *36*.

Nagabandi, A., Konolige, K., Levine, S., & Kumar, V. (2020). Deep dynamics models for learning dexterous manipulation. *Conference on Robot Learning*, 1101–1112.

OpenAI. (2024, September). Chatgpt.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al. (2023). Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*.

Parisotto, E., & Salakhutdinov, R. (2018). Neural map: Structured memory for deep reinforcement learning. *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *ICML*.

Péré, A., Forestier, S., Sigaud, O., & Oudeyer, P.-Y. (2018). Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*.

Pitis, S., Chan, H., Zhao, S., Stadie, B., & Ba, J. (2020). Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. *International Conference on Machine Learning*, 7750–7761.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., & Zaremba, W. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research.

Poldrack, R. A., & Packard, M. G. (2003). Competition among multiple memory systems: Converging evidence from animal and human brain studies. *Neuropsychologia*, *41*(3), 245–251.

Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., & Levine, S. (2019). Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*.

Portelas, R., Colas, C., Hofmann, K., & Oudeyer, P.-Y. (2020). Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. *Conference on Robot Learning*, 835–853.

Poudel, R. P., Pandya, H., Liwicki, S., & Cipolla, R. (2024). Recore: Regularized contrastive representation learning of world model. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22904–22913.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., & Blundell, C. (2017a). Neural episodic control. *International Conference on Machine Learning*, 2827–2836.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., & Blundell, C. (2017b, June). Neural episodic control. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 2827–2836, Vol. 70). PMLR.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *International conference on machine learning*, 8748–8763.

Rahman, A. A., Agarwal, P., Michalski, V., Noumeir, R., & Kahou, S. (2023). Empowering clinicians with medt: A framework for sepsis treatment. *NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning*.

Reuss, M., Li, M., Jia, X., & Lioutikov, R. (2023). Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*.

Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Sancaktar, C., Blaes, S., & Martius, G. (2022). Curious exploration via structured world models yields zero-shot object manipulation. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. *International conference on machine learning*, 1312–1320.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schott, B. H., Henson, R. N., Richardson-Klavehn, A., Becker, C., Thoma, V., Heinze, H.-J., & Düzel, E. (2005). Redefining implicit and explicit memory: The functional neuroanatomy of priming, remembering, and control of retrieval. *Proceedings of the National Academy of Sciences*, *102*(4), 1257–1262.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604–609.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., & Bachman, P. (2020). Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., & Pathak, D. (2020). Planning to explore via self-supervised world models. *ICML*.

Sharma, A., Ahmad, R., & Finn, C. (2022). A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*.

Sharma, A., Ahmed, A. M., Ahmad, R., & Finn, C. (2023). Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. *arXiv preprint arXiv:2303.01488*.

Sharma, A., Gupta, A., Levine, S., Hausman, K., & Finn, C. (2021). Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, *34*, 18474–18486.

Sharma, A., Xu, K., Sardana, N., Gupta, A., Hausman, K., Levine, S., & Finn, C. (2021). Autonomous reinforcement learning: Formalism and benchmarking. *arXiv preprint arXiv:2112.09605*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nat.*, *529*(7587), 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, *550*(7676), 354–359.

Smith, L., Dhawan, N., Zhang, M., Abbeel, P., & Levine, S. (2019). Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tulving, E. (1983). Elements of episodic memory.

Tulving, E. (1972). Episodic and semantic memory. *Organization of memory*.

Veeriah, V., Oh, J., & Singh, S. (2018). Many-goals reinforcement learning. *ArXiv*, *abs/1806.09605*.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T., Ionescu, C., Hansen, S., & Mnih, V. (2019). Unsupervised control through non-parametric discriminative rewards. *International Conference on Learning Representations*.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3), 279–292.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., et al. (2022). Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, *602*(7896), 223–228.

Xu, K., Verma, S., Finn, C., & Levine, S. (2020). Continual learning of control primitives: Skill discovery via reset-games. *Advances in Neural Information Processing Systems*, *33*, 4999–5010.

Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., & Levine, S. (2017). Collective robot reinforcement learning with distributed asynchronous guided policy search. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 79–86.

Yang, J., Gao, S., Qiu, Y., Chen, L., Li, T., Dai, B., Chitta, K., Wu, P., Zeng, J., Luo, P., et al. (2024). Generalized predictive model for autonomous driving. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14662–14672.

Yang, X., Ji, Z., Wu, J., & Lai, Y.-K. (2021). An open-source multi-goal reinforcement learning environment for robotic manipulation with pybullet. *Annual Conference Towards Autonomous Robotic Systems*, 14–24.

Yang, Z., Moerland, T. M., Preuss, M., & Plaat, A. (2022). Continuous episodic control. *arXiv preprint arXiv:2211.15183*.

Yi, Z., Luo, Y., Westover, T., Katikaneni, S., Ponkiya, B., Sah, S., Mahmud, S., Raker, D., Javaid, A., Heben, M. J., et al. (2022). Deep reinforcement learning based optimization for a tightly coupled nuclear renewable integrated energy system. *Applied Energy*, *328*, 120113.

Young, K., & Tian, T. (2019). Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., & Levine, S. (2019). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *Conference on Robot Learning (CoRL)*.

Zhang, Y., Abbeel, P., & Pinto, L. (2020). Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, *33*, 7648–7659.

Zhang, Z., Li, Y., Bastani, O., Gupta, A., Jayaraman, D., Ma, Y. J., & Weihs, L. (2023). Universal visual decomposer: Long-horizon manipulation made easy. *arXiv preprint arXiv:2310.08581*.

Zheng, S., Trott, A., Srinivasa, S., Parkes, D. C., & Socher, R. (2022). The ai economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science advances*, *8*(18), eabk2607.

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., & Kumar, V. (2019). Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost.

2019 International Conference on Robotics and Automation (ICRA), 3651–3657.

Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., & Levine, S. (2020). The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*.

Acknowledgements

I would like to express my deepest gratitude to my supervisors and promoters, Thomas, Mike, and Aske. Thomas, I am really grateful you joined my supervision team after the first year of my PhD study—it has been an incredible journey. Your consistent support has greatly shaped my research, and I am truly thankful for the expertise you have shared with me. Thank you for challenging me to think critically and I appreciate everything you have done. Mike, thank you so much for always being there when I needed you. Your enthusiasm for research has inspired me, and your constant belief in my abilities has helped me navigate challenges with greater confidence. I am deeply grateful for your kindness and the personal support you have provided throughout this journey. Aske, I sincerely appreciate your encouragement and guidance during my study, as well as your swift responses and inspiring discussions. Your willingness to explore new ideas and guide me toward meaningful solutions has always been motivating. Working with you has been an absolute privilege.

I would also like to thank my collaborator Edward Hu from the University of Pennsylvania. Thank you for the wonderful collaborations and insightful discussions. Additionally, I extend my thanks to all my colleagues, whose collaboration and friendship made this journey enjoyable. Special thanks go to all members of RLG—especially Alan, Alvaro, Andrius, Annie, Andreas, Bram, Felix, Koen, Matthias, Michiel, Mike, and Tom—for their assistance with various aspects of my research and for always being willing to share their expertise and time. I would also like to thank all colleagues and friends at LIACS, especially Dr. Xueqin Chen, Dr. Wenjing Chu, Dr. Hui Feng, Dr. Xuhan Liu, Dr. Nan Pu, Dr. Hui Wang, Dr. Furong Ye, Qi Huang, Sengim Karayalcin, Shuaiqun Pan, Serban Vadineanu, Yumeng Wang, Xiaohan Wei, Weikang Wen, and Yuxuan Zhao. Thank you for the coffee breaks and the discussions we shared.

I am grateful to my surfing buddy Lixuan Zhang and my bouldering buddies, especially Dr. Yiming Dong. Thanks as well to Dr. Hao Wang, Zhaoxing Wang, and

Dr. Bing Xu for their guidance on both life and research. I am also deeply grateful to all friends from ACSSNL, especially Dr. Wei Chen, Dr. Yu Dong, Dr. Yupeng He, Dr. Wenyi Wang, Dr. Xiaoyang Zhong, Wei Fang, Nasi Liu, Tiantian Tang, Wenyu Wan, Jiayi Yang, Ying Yin, and Yuxuan Zhu. Thank you all—your continued friendship and support mean the world to me.

To my friends in China, thank you for your unwavering support. Yuan Liang and Chen Wang, your encouragement was especially meaningful during the early stages of my PhD, amidst the challenges of COVID-19. Zhao Yan, thank you for always being there. Kaiqi Fu, your research and life ambitions have been inspiring. Yuxuan Jiang, your kindness has been a constant source of comfort. To my long-time friend, Minkang Tan, and along the way, I was fortunate to meet F3—thank you Jiayang, Zhong, and Lincen for all the coffee and beers we shared.

Finally, I would like to express my deepest appreciation to my family. My heartfelt thanks go to my grandma, parents, my brother and my sister-in-law, and all my cousins, nephews, and nieces. My sincere gratitude also goes to my girlfriend for accompanying me to the end of this journey. Without your support and love, I would never have made it this far.

I am also grateful to the China Scholarship Council for their financial support, which made this endeavor possible.

This thesis would not have been possible without the support, guidance, and encouragement of all these remarkable individuals. Thank you all!

Summary

Reinforcement learning is a framework that enables agents to learn in a manner similar to humans, i.e. through trial and error. Ideally, we would like to train a generalist agent capable of performing multiple tasks and achieving various goals. Goal-conditioned reinforcement learning is a step toward training such an agent. The goal-conditioned reinforcement learning framework comprises four steps: 1) defining the goal space; 2) selecting an interesting goal for the agent; 3) the agent learning to reach the goal; 4) the agent post-exploring.

In the thesis, we proposed four research questions and focused on three parts of the goal-conditioned reinforcement learning framework. First, in Chapter 3, we studied a more autonomous goal-conditioned reinforcement learning setting where there is no access to reset, which poses challenges on exploration. By using world models and selecting various goals to command the agent for data collection, our proposed agent can autonomously operate in reset-free tasks and outperform state-of-the-art baselines.

Once a goal is selected, a goal-conditioned policy is typically trained to reach the goal. Previous research on a non-parametric method, called model-free episodic control (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016), shows that episodic control can quickly latch onto previously discovered solutions and learn faster than deep RL methods which are known to be slow. Instead of training a goal-conditioned policy, episodic control can also be employed to train the agent to reach the selected goal. However, model-free episodic control was designed for tasks with a discrete action space. In Chapter 4, we extend the episodic control to continuous episodic control that can now tackle tasks with a continuous action space, and demonstrate its strong performance on various continuous robotic control tasks. Moreover, limitations of episodic control methods are identified in Chapter 5, namely, they will be non-optimal in stochastic tasks. Then we proposed to combine episodic control with deep reinforcement learning methods to gain from both approaches. Experimental results

show that by combining these two methods, the unified agent achieves both the fast learning attributed to episodic control and the optimality attributed to reinforcement learning.

In Chapter 6, we demonstrated that additional exploration after reaching frontier goals—referred to as post-exploration—enhances the efficiency of GCRL agents. Without post-exploration, the agent resets after reaching the frontier goals, preventing it from expanding its knowledge boundary. By engaging in post-exploration, agents effectively step into new, unseen areas, acquiring more diverse data. This leads to improved performance compared to agents that do not perform post-exploration.

The methods proposed in the thesis improved components of the goal-conditioned reinforcement learning framework, including goal selection, policy learning and exploration, consequently enhancing the performance and increasing the autonomy of the entire goal-conditioned reinforcement learning framework. In future work, we hope that the goal-conditioned reinforcement learning framework will serve as a recipe for training a generalist agent, and ultimately, an embodied artificial general intelligent agent.

Samenvatting

Reinforcement learning is een methode om te leren door vallen en opstaan, en dat werkt voor computers net als bij mensen. Idealiter willen we een generiek algoritme hebben dat een agent zodanig traint dat deze verschillende taken kan uitvoeren en verschillende doelen kan bereiken. Goal-directed reinforcement learning is een manier die een stap in deze richting zet. Het Goal-directed reinforcement learning raamwerk omvat vier stappen: 1) het definiëren van de doelruimte; 2) het selecteren van een interessant doel voor de agent; 3) een algoritme om de agent te leren om dat doel te bereiken; 4) een agent die post-exploration uitvoert. In het proefschrift poneren we vier onderzoeksvragen. We richten ons op drie delen van het goal-directed reinforcement learning raamwerk. Ten eerste hebben we in Chapter 3 een meer autonoom goal-directed reinforcement learning setting bestudeerd die werkt zonder reset; dit levert complicaties op bij exploratie. Door zogenaamde World-Models te gebruiken en verschillende doelen te selecteren in de fase van gegevensverzameling, kan onze voorgestelde agent autonoom werken in reset-vrije taken en beter presteren dan state-of-the-art baselines. Zodra een doel is geselecteerd, wordt normaalgesproken een goal-directed policy getraind om het doel te bereiken. Eerder onderzoek naar een niet-parametrische methode (genaamd model-free episodic control (Blundell, Uria, Pritzel, Li, Ruderman, Leibo, Rae, et al., 2016)), toont aan dat episodic control snel kan inhaken op eerder ontdekte oplossingen en sneller kan leren dan deep RL-methoden (waarvan bekend is dat ze traag zijn). In plaats van het trainen van een goal-directed policy, kan episodic control ook worden gebruikt om de agent te trainen om het geselecteerde doel te bereiken. Model-free episodic control is echter ontworpen voor taken met een discrete actieruimte. In Chapter 4 breiden we episodic control uit naar continue episodische methoden die nu taken met een continue actieruimte kunnen aanpakken. We laten zien dat dit goed presteert op een aantal continue problemen uit de robotica. Ook noemen we een aantal beperkingen van episodic control, in Chapter 5, namelijk dat ze niet optimaal zullen

zijn in stochastische problemen. Vervolgens stelden we voor om episodic control te combineren met deep reinforcement learning-methoden om van het beste van twee werelden te bereiken. Onze experimenten laten zien dat door deze twee methoden te combineren, de gecombineerde agent zowel het snelle leren bereikt van episodic control, als de optimaliteit van reinforcement learning. In Chapter 6 laten we zien dat extra exploratie na het bereiken van de doelen op de search-boundary (post-exploratie genoemd) de eciëntie van GCRL-agenten verbetert. Zonder post-exploratie reset de agent zich na het bereiken van deze doelen, waardoor de agent zijn kennisgrens niet kan uitbreiden. Met post-exploratie stappen agenten eectief in nieuwe, onbekende, gebieden en verwerven ze meer diverse gegevens. Dit leidt tot betere prestaties in vergelijking met agenten die geen post-exploratie uitvoeren. De methoden die in het proefschrift worden voorgesteld, verbeterden onderdelen van het goal-directed reinforcement learning-framework, waaronder doelselectie, policy-learning en exploratie, wat heeft geleid tot betere prestaties en de autonomie van het gehele goal-directed reinforcement learning-framework verhoogde. In de toekomst hopen we dat goal-directed reinforcement learning gebruikt zal worden voor het trainen van een generieke agent en uiteindelijk een embodied AI agent (robot).

Titles in the SIKS dissertation series since 2016

2016 01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines

02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow

03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support

04 Laurens Rietveld (VUA), Publishing and Consuming Linked Data

05 Evgeny Sherkhonov (UvA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers

06 Michel Wilson (TUD), Robust scheduling in an uncertain environment

07 Jeroen de Man (VUA), Measuring and modeling negative emotions for virtual training

08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data

09 Archana Nottamkandath (VUA), Trusting Crowdsourced Information on Cultural Artefacts

10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms

11 Anne Schuth (UvA), Search Engines that Learn from Their Users

12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems

13 Nana Baah Gyan (VUA), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach

14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization

15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UvA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VUA), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VUA), Refining Statistical Data on the Web
19 Julia Efremova (TU/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UvA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VUA), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UvA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VUA), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (TiU), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UvA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TU/e), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UvA), Monitoring Social Media: Summarization, Classification and Recommendation

36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies

37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry

38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design

39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect

40 Christian Detweiler (TUD), Accounting for Values in Design

41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance

42 Spyros Martzoukos (UvA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora

43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice

44 Thibault Sellam (UvA), Automatic Assistants for Database Exploration

45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control

46 Jorge Gallego Perez (UT), Robots to Make you Happy

47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks

48 Tanja Buttler (TUD), Collecting Lessons Learned

49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis

50 Yan Wang (TiU), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

---

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime

02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation

03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines

04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store

05 Mahdieh Shadi (UvA), Collaboration Behavior

06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search

07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly

08 Rob Konijn (VUA), Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TU/e), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (TiU), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UvA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UvA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VUA), Logics for causal inference under uncertainty
23 David Graus (UvA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VUA), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VUA), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (TiU), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"

30 Wilma Latuny (TiU), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VUA), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TU/e), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TU/e), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction

04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UvA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UvA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TU/e), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (TiU), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TU/e), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OU), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TU/e), Hierarchical Process Mining for Scalable Software Analysis

28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (TiU), Emotion Recognition from Mandarin Speech
30 Wouter Beek (VUA), The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TU/e), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
06 Chris Dijkshoorn (VUA), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UvA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VUA), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TU/e), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents

20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TU/e), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VUA),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VUA), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VUA), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OU), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VUA), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TU/e), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TU/e), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OU), Gamification with digital badges in learning programming
36 Kevin Ackermans (OU), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OU), Learning visually grounded and multilingual representations

2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour

02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models

03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding

04 Maarten van Gompel (RUN), Context as Linguistic Bridges

05 Yulong Pei (TU/e), On local and global structure mining

06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support

07 Wim van der Vegt (OU), Towards a software architecture for reusable game components

08 Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search

09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research

10 Alifah Syamsiyah (TU/e), In-database Preprocessing for Process Mining

11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data Augmentation-Methods for Long-Tail Entity Recognition Models

12 Ward van Breda (VUA), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment

13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming

14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases

15 Konstantinos Georgiadis (OU), Smart CAT: Machine Learning for Configurable Assessments in Serious Games

16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling

17 Daniele Di Mitri (OU), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences

18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems

19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems

20 Albert Hankel (VUA), Embedding Green ICT Maturity in Organisations

21 Karine da Silva Miras de Araujo (VUA), Where is the robot?: Life as it could be

22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar

23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging

24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots

25 Xin Du (TU/e), The Uncertainty in Exceptional Model Mining

26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization

27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context

28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality

29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference

30 Bob Zadok Blok (UL), Creatief, Creatiever, Creatiefst

31 Gongjin Lan (VUA), Learning better – From Baby to Better

32 Jason Rhuggenaath (TU/e), Revenue management in online markets: pricing and online advertising

33 Rick Gilsing (TU/e), Supporting service-dominant business model evaluation in the context of business model innovation

34 Anna Bon (UM), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development

35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

---

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space

02 Rijk Mercuur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models

03 Seyyed Hadi Hashemi (UvA), Modeling Users Interacting with Smart Devices

04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning

05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems

06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot

07 Armel Lefebvre (UU), Research data management for open science
08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning
14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues
17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks
18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
19 Roberto Verdecchia (VUA), Architectural Technical Debt: Identification and Management
20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems
21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes
22 Sihang Qiu (TUD), Conversational Crowdsourcing
23 Hugo Manuel Proença (UL), Robust rules for prediction and description
24 Kaijie Zhu (TU/e), On Efficient Temporal Subgraph Query Processing
25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications
26 Benno Kruit (CWI/VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables
27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You

	28	Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs
2022	01	Judith van Stegeren (UT), Flavor text generation for role-playing video games
	02	Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey
	03	Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare
	04	Ünal Aksu (UU), A Cross-Organizational Process Mining Framework
	05	Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization
	06	Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding
	07	Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics
	08	Maikel L. van Eck (TU/e), Process Mining for Smart Product Design
	09	Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
	10	Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
	11	Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
	12	Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
	13	Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
	14	Michiel Overeem (UU), Evolution of Low-Code Platforms
	15	Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
	16	Pieter Gijsbers (TU/e), Systems for AutoML Research
	17	Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
	18	Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation

19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation

20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media

21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments

22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations

23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents

24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values

25 Anna L.D. Latour (UL), Optimal decision-making under constraints and uncertainty

26 Anne Dirkson (UL), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences

27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences

28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems

29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality

30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays

31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management

32 Cezara Pastrav (UU), Social simulation for socio-ecological systems

33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management

34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change

35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction

---

2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions

02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts

03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations

04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval

05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications

06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment

07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning

08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning

09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques

10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing

11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications

12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries

13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation

14 Selma auevi (TUD), Energy resilience through self-organization

15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models

16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters

17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision – From Oversight to Insight

18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation

19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals

20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning

21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
22 Alireza Shojaifar (UU), Volitional Cybersecurity
23 Theo Theunissen (UU), Documentation in Continuous Software Development
24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning
25 Jurian Baas (UU), Entity Resolution on Historical Knowledge Graphs
26 Loek Tonnaer (TU/e), Linearly Symmetry-Based Disentangled Representations and their Out-of-Distribution Behaviour
27 Ghada Sokar (TU/e), Learning Continually Under Changing Data Distributions
28 Floris den Hengst (VUA), Learning to Behave: Reinforcement Learning in Human Contexts
29 Tim Draws (TUD), Understanding Viewpoint Biases in Web Search Results

2024 01 Daphne Miedema (TU/e), On Learning SQL: Disentangling concepts in data systems education
02 Emile van Krieken (VUA), Optimisation in Neurosymbolic Learning Systems
03 Feri Wijayanto (RUN), Automated Model Selection for Rasch and Mediation Analysis
04 Mike Huisman (UL), Understanding Deep Meta-Learning
05 Yiyong Gou (UM), Aerial Robotic Operations: Multi-environment Cooperative Inspection & Construction Crack Autonomous Repair
06 Azqa Nadeem (TUD), Understanding Adversary Behavior via XAI: Leveraging Sequence Clustering to Extract Threat Intelligence
07 Parisa Shayan (TiU), Modeling User Behavior in Learning Management Systems
08 Xin Zhou (UvA), From Empowering to Motivating: Enhancing Policy Enforcement through Process Design and Incentive Implementation
09 Giso Dal (UT), Probabilistic Inference Using Partitioned Bayesian Networks
10 Cristina-Iulia Bucur (VUA), Linkflows: Towards Genuine Semantic Publishing in Science
11 withdrawn
12 Peide Zhu (TUD), Towards Robust Automatic Question Generation For Learning

13 Enrico Liscio (TUD), Context-Specific Value Inference via Hybrid Intelligence
14 Larissa Capobianco Shimomura (TU/e), On Graph Generating Dependencies and their Applications in Data Profiling
15 Ting Liu (VUA), A Gut Feeling: Biomedical Knowledge Graphs for Interrelating the Gut Microbiome and Mental Health
16 Arthur Barbosa Câmara (TUD), Designing Search-as-Learning Systems
17 Razieh Alidoosti (VUA), Ethics-aware Software Architecture Design
18 Laurens Stoop (UU), Data Driven Understanding of Energy-Meteorological Variability and its Impact on Energy System Operations
19 Azadeh Mozafari Mehr (TU/e), Multi-perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior
20 Ritsart Anne Plantenga (UL), Omgang met Regels
21 Federica Vinella (UU), Crowdsourcing User-Centered Teams
22 Zeynep Ozturk Yurt (TU/e), Beyond Routine: Extending BPM for Knowledge-Intensive Processes with Controllable Dynamic Contexts
23 Jie Luo (VUA), Lamarck's Revenge: Inheritance of Learned Traits Improves Robot Evolution
24 Nirmal Roy (TUD), Exploring the effects of interactive interfaces on user search behaviour
25 Alisa Rieger (TUD), Striving for Responsible Opinion Formation in Web Search on Debated Topics
26 Tim Gubner (CWI), Adaptively Generating Heterogeneous Execution Strategies using the VOILA Framework
27 Lincen Yang (UL), Information-theoretic Partition-based Models for Interpretable Machine Learning
28 Leon Helwerda (UL), Grip on Software: Understanding development progress of Scrum sprints and backlogs
29 David Wilson Romero Guzman (VUA), The Good, the Efficient and the Inductive Biases: Exploring Efficiency in Deep Learning Through the Use of Inductive Biases
30 Vijanti Ramautar (UU), Model-Driven Sustainability Accounting
31 Ziyu Li (TUD), On the Utility of Metadata to Optimize Machine Learning Workflows
32 Vinicius Stein Dani (UU), The Alpha and Omega of Process Mining

33 Siddharth Mehrotra (TUD), Designing for Appropriate Trust in Human-AI interaction

34 Robert Deckers (VUA), From Smallest Software Particle to System Specification - MuDForM: Multi-Domain Formalization Method

35 Sicui Zhang (TU/e), Methods of Detecting Clinical Deviations with Process Mining: a fuzzy set approach

36 Thomas Mulder (TU/e), Optimization of Recursive Queries on Graphs

37 James Graham Nevin (UvA), The Ramifications of Data Handling for Computational Models

38 Christos Koutras (TUD), Tabular Schema Matching for Modern Settings

39 Paola Lara Machado (TU/e), The Nexus between Business Models and Operating Models: From Conceptual Understanding to Actionable Guidance

40 Montijn van de Ven (TU/e), Guiding the Definition of Key Performance Indicators for Business Models

41 Georgios Siachamis (TUD), Adaptivity for Streaming Dataflow Engines

42 Emmeke Veltmeijer (VUA), Small Groups, Big Insights: Understanding the Crowd through Expressive Subgroup Analysis

43 Cedric Waterschoot (KNAW Meertens Instituut), The Constructive Conundrum: Computational Approaches to Facilitate Constructive Commenting on Online News Platforms

44 Marcel Schmitz (OU), Towards learning analytics-supported learning design

45 Sara Salimzadeh (TUD), Living in the Age of AI: Understanding Contextual Factors that Shape Human-AI Decision-Making

46 Georgios Stathis (Leiden University), Preventing Disputes: Preventive Logic, Law & Technology

47 Daniel Daza (VUA), Exploiting Subgraphs and Attributes for Representation Learning on Knowledge Graphs

48 Ioannis Petros Samiotis (TUD), Crowd-Assisted Annotation of Classical Music Compositions

---

2025 01 Max van Haastrecht (UL), Transdisciplinary Perspectives on Validity: Bridging the Gap Between Design and Implementation for Technology-Enhanced Learning Systems

02 Jurgen van den Hoogen (JADS), Time Series Analysis Using Convolutional Neural Networks

03 Andra-Denis Ionescu (TUD), Feature Discovery for Data-Centric AI

04 Rianne Schouten (TU/e), Exceptional Model Mining for Hierarchical Data

05 Nele Albers (TUD), Psychology-Informed Reinforcement Learning for Situated Virtual Coaching in Smoking Cessation

06 Daniël Vos (TUD), Decision Tree Learning: Algorithms for Robust Prediction and Policy Optimization

07 Ricky Maulana Fajri (TU/e), Towards Safer Active Learning: Dealing with Unwanted Biases, Graph-Structured Data, Adversary, and Data Imbalance

08 Stefan Bloemheuvel (TiU), Spatio-Temporal Analysis Through Graphs: Predictive Modeling and Graph Construction

09 Fadime Kaya (VUA), Decentralized Governance Design - A Model-Based Approach

10 Zhao Yang (UL), Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning

Curriculum Vitae

Yang, Zhao was born in Ningxian, China, in 1996. He completed his middle and high school education in Ningxian. In 2014, he began studying computer science at Beijing Language and Culture University. After earning his bachelor's degree in 2018, Zhao moved to Leiden, where he enrolled in the master's program in computer science and data science at Leiden University. He obtained his master's degree in 2020 and subsequently joined the Reinforcement Learning Group (RLG) at Leiden University as a PhD student the same year.

Zhao's research interests include deep reinforcement learning, and its application in games and robotic tasks. During his PhD studies, he took courses in academic writing, communication in science and scientific conduct, among others.