

Adaptive Successor Features for Transfer Reinforcement Learning

by Oscar Kiyoshige Garces Aparicio

Thesis submitted in fulfilment of the requirements for
the degree of

Doctor of Philosophy in Computer Science

under the supervision of Dr. Hua Zuo and Dr. Junyu Xuan

University of Technology Sydney
Faculty of Engineering and Information Technology

September 2025

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Oscar Kiyoshige Garces Aparicio*, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science, Faculty of Engineering and Information Technology* at the University of Technology Sydney, Australia. This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

Production Note:

SIGNATURE: Signature removed prior to publication.

Oscar Kiyoshige Garces Aparicio

DATE: 3rd September, 2025

PLACE: Sydney, Australia

DEDICATION

*To my beloved wife, to my self, and my family which have always been there giving me
support and motivation. . . .*

ACKNOWLEDGMENTS

My PhD journey was supported by the ATN, UTS, and the LATAM Scholarship Program thanks for their support throughout these years. This scholarship helped me turn my dreams into reality. A PhD journey brings enormous changes, sacrifices, and life-changing decisions. It is a path that cannot be walked alone. For me, this journey was made possible by the unwavering support of my wife, family, friends, colleagues, the UTS staff, and my supervisors.

First and foremost, I would like to express my deepest gratitude to my principal supervisor, Dr. Hua Zuo. Thank you for your early trust in me, for giving me the opportunity to explore the fields of transfer learning and reinforcement learning, and for guiding me through every stage of this process. Thank you also for encouraging me to stay motivated during the toughest moments. To my co-supervisor, Dr. Junyu Xuan, thank you for your generosity with your time, those multiple weekly meetings, long discussions, writing sessions, and even code reviews. Thank you for recognising my limitations while always pushing me to do better. I will forever be grateful for one particular meeting, on December 7th, when after hours of discussion I finally understood what the PhD process was truly about. In that moment, I learnt that supervisors not only celebrate successes, but also carry the emotional weight alongside their students.

At the very beginning of this journey, Dr. Roberto Martínez-Maldonado helped me with the first version of my research proposal, which made it possible for me to apply for the scholarship and begin this journey. In Australia, I was fortunate to meet extraordinary people who supported me in many ways. Thank you to Professor Dragan Gašević for your guidance, mentorship, and for collaborating with me on one of the articles included in this thesis, as well as for making me feel truly welcome in the CoLAM research group. I am also grateful to Dr. Mladen Raković, whose advice in the early stages of my PhD helped me learn how to communicate my ideas more clearly and build meaningful collaborations. My sincere thanks also go to Professor Simon Buckingham Shum and Professor Kirsty Kitto for their encouragement during the first year of my PhD and for supporting Gloria throughout her doctoral journey. To my thesis reviewers, Dr. Yi Zhang, Dr. Elle Li, Dr. Xianzhi Wang, Dr. Shiping Wen, and Dr. Zhen Fang. To my external reviewers, Professor Guoxin Su and Professor Yipeng Zhou. I deeply appreciate your time, your role in reviewing, and evaluating my work.

To everyone who was part of this journey over the years in Australia, Diana Holdstock and Glenn Holdstock, thank you for welcoming us like family. To Dr. Andrea Torres, Dr. Vanessa Echavarría, Dr. Iván Silva, Itzel, Dr. Evelyn Palominos, Sayuri Takegami, Joseph

Croke, and Julian Hernandez, thank you for your presence, conversations, and friendship. I am also grateful to the UTS staff, especially Ann Yan, who has been an incredible support in managing all the administrative processes. My sincere thanks also go to my AAIL colleagues, Dr. Nelson Ma, Dr. Zihé Liu, Dr. Wei Duan, and Dr. Mengjing Wu.

But none of this would have been possible without the core of my motivation, my reward system, my family. I know you enjoy translating from English to Spanish, so I will leave this in English for you. My family has always been there, ALWAYS, and after many years of reflection, I believe that you are the ones who made the difference. Many of the good decisions I made were because you were there guiding me. You are my centre, my strength, and often my relief from stress. This PhD is for you!

To my mom, my guide, Ana Milena; my dad, my hero, my best friend, Oscar Gonzalo; and my brother, my business partner, also my best friend, and my wedding's best man, José Tsunegiro, thank you for every conversation, 'jalón de orejas', 'peleas', intervention, and every second you have spent with me to keep me motivated not only for this PhD journey but for every step in my life. You know better than anyone that this PhD was a personal dream rather than practical one and thanks for all that support, especially in the moments when I doubted myself. Anyone close to me knows that my aunties are the closest thing to being my mothers. My aunties María Amparo, a perfect example of love, discipline, and dedication, who has always motivated me to continue studying and has been there in every moment of my life with endless love; María Consuelo (I know you are watching over us), who was always present in the most important moments of my life (I will miss you at this graduation); Carmen Elisa, who always wishes me the best and encourages me to be a hard worker, every Sunday message making a difference throughout the years; and Carmen Elena, who was one of the first people to congratulate me when I received the PhD scholarship and who celebrated every academic success. To my cousins, as close to me as brothers, Takeji and Minoru, whom we grew up together building a loyal partnership and camaraderie. And to my uncle Guillermo Salazar for his motivation to build and make things happen. To my new extended family (Fernández-Nieto), Gloria and Ramón, thank you for asking every day about the progress of my PhD thesis. To all those who are not longer with us! I wanted to tell you that all of you, near or far, will always remain in my heart, I will always be thankful for everything, and I will always try to be the best person you would be proud of.

Lastly, I want to end with the person who has been by my side every step of this long, demanding, but wonderful journey: Gloria Milena, my wife, the love of my life, and my best friend. Your resilience, patience, and love have been extraordinary. In 2022, we decided to converge our pathways in Australia, I started my PhD you ended yours, and we've got married. I often say that you completed two PhDs in less than seven years, and it is true. You have been more than a life partner; you have been everything in my reward system, co-author, project manager, and motivator. Last year we reached a special milestone when we wrote and published a paper together. That moment meant the world to me and represents the beginning of a new era in our lives.

Oscar Kiyoshige Garces Aparicio.

PUBLICATIONS

RELATED TO THE THESIS

1. **Garces, Kiyoshige**, Xuan, J., & Zuo, H. (2023). Transformed successor features for transfer reinforcement learning. *AI 2023: Advances in Artificial Intelligence*. https://doi.org/10.1007/978-981-99-8391-9_24
2. **Garces, Kiyoshige**, Xuan, J., & Zuo, H. (2025). Adaptive successor features composition for transfer reinforcement learning. *IEEE Transactions on Artificial Intelligence*, 1–12. <https://doi.org/10.1109/TAI.2025.3610392>
3. **Garces, Kiyoshige**, Fernandez-Nieto, G. M., Rakovic, M., Li, X., Li, T., Zhao, L., Gašević, D., Xuan, J., & Zuo, H. (2025). Transfer Reinforcement Learning for Self-Regulated Learning Support: An Evaluation Using Successor Representations. *Artificial Intelligence in Education*. https://doi.org/10.1007/978-3-031-98465-5_55
4. **Garces, Kiyoshige**, Xuan, J., & Zuo, H. (n.d.). Successor feature adaptation for transfer reinforcement learning across dissimilar transition dynamics via gaussian process [Under review]. *RLC 2026*

TABLE OF CONTENTS

Certificate of Original Authorship	iii
Dedication	v
Acknowledgments	vii
Publications	ix
Table of Contents	xi
List of Figures	xv
List of Tables	xix
List of Abbreviations	xxi
Abstract	xxiii
1 Introduction	1
1.1 Background and Motivation	4
1.2 Research Questions and Objectives	5
1.3 Contributions	9
1.4 Thesis Organisation	10
2 Literature Review	13
2.1 Reinforcement Learning	13
2.2 Transfer Reinforcement Learning	15
2.3 Transfer Reinforcement Learning via Successor Features	17
2.3.1 Generalised Policy Improvement	18
2.4 Related Work on Successor Features	18

TABLE OF CONTENTS

2.4.1	Address Reinforcement Learning Limitations	19
2.4.2	Address Successor Features Limitations	20
2.4.3	Address Real-world Application Challenges	25
2.5	Successor Features Constraints and Limitations	26
3	Adaptive Successor Features via Transition Dynamics Transformation	27
3.1	Background and Motivation	28
3.2	Preliminaries and Problem Formulation	29
3.3	Method	30
3.3.1	Transformed Successor Features	31
3.3.2	Transfer via Transformed Successor Features	36
3.3.3	Source Transformed Successor Feature Learning	37
3.3.4	Target Transformed Successor Feature Learning	37
3.4	Experiments and Analysis	39
3.4.1	Experiment Setup	39
3.4.2	Experiment Results and Analysis	42
3.5	Summary	54
4	Adaptive Successor Features via Gaussian Process Kernel Specification	57
4.1	Background and Motivation	57
4.2	Preliminaries and Problem Formulation	59
4.3	Method	60
4.3.1	Transition Dynamics Comparison via Feature Observations	61
4.3.2	Successor Feature-based Transition Dynamics Transfer Kernel	62
4.3.3	Successor Features Approximation via Gaussian Process	68
4.3.4	Uncertainty-driven Generalised Policy Improvement	69
4.4	Experiments and Analysis	70
4.4.1	Experiment Setup	70
4.4.2	Experiment Results and Analysis	73
4.5	Summary	79
5	Adaptive Successor Features via Hierarchical Behaviour Composition	81
5.1	Background and Motivation	81
5.2	Preliminaries and Problem Formulation	84
5.3	Method	85
5.3.1	High-Level Markov Decision Process	86

5.3.2	High-Level Actions with Successor Features	86
5.3.3	High-Level States with Successor Features	87
5.3.4	High-Level Reward	88
5.3.5	High-Level Policy Learning and Knowledge Transfer	88
5.4	Experiments and Analysis	90
5.4.1	Experiment Setup	90
5.4.2	Experiment Results and Analysis	93
5.5	Summary	108
6	Adaptive Successor Features for Human Learning Support	111
6.1	Background and Motivation	112
6.2	Preliminaries and Problem Formulation	114
6.3	Method	118
6.3.1	Research design and context	118
6.3.2	Self-Regulated Learning Meets Transfer Reinforcement Learning and Successor Features	121
6.4	Experiments and Analysis	122
6.4.1	Experiment Setup	122
6.4.2	Experiment Results and Analysis	124
6.5	Summary	127
7	Conclusion	129
7.1	Findings	129
7.2	Limitations	131
7.3	Future Work	132
7.4	Concluding Remarks	133
	Bibliography	135

LIST OF FIGURES

FIGURE	Page
1.1 Harvesting drone across varying crops and weather conditions.	2
1.2 Harvesting drone across varying crops and weather.	3
1.3 Thesis organisation diagram.	10
2.1 Groups of related work on successor features.	19
3.1 Transformations between base distribution probability, $p^*(\cdot \cdot, \cdot)$, and source task distribution probabilities for source tasks, i.e., $p^1, p^2, p^3, \dots$	31
3.2 Grid search for g and h dimensions.	41
3.3 Robot Reacher setup. The blue dots are the coordinates for the source tasks. The orange dots are the coordinates for the target tasks. The green dot is the centre of the reacher arm.	43
3.4 Scenario 1: Average return in source and target tasks by total training steps. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.	45
3.5 Scenario 2: Average return in source and target tasks by total training steps. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.	47
3.6 Scenario 1: Average return in source and target tasks by total training steps for diverse dynamics setting. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.	48
3.7 Scenario 2: Average return in source (top) and target (bottom) tasks. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.	50
3.8 Cartpole environment. Set of source and target tasks defined by the pole lengths in meters.	51

LIST OF FIGURES

3.9	Average return in source and target tasks by total training steps. Shades show half a standard deviation over 10 runs with independent random seeds.	52
3.10	Hopper environment. Set of source and target tasks defined by the healthy ranges.	53
3.11	Average return in source and target tasks by total training steps. Shades show half a standard deviation over 5 runs with independent random seeds.	54
4.1	Seasonal drone picker. Different wind conditions occur each season, resulting in varying transition dynamics.	58
4.2	Left, each colour group in the observed feature space represents a different task with distinct transition dynamics. Right, covariance matrix using the immediate observed features metric.	63
4.3	Foraging environment. Sequence of agent's navigation across a 10×10 grid with blue and red objects. The agent (red triangle) navigates through the grid gathering objects. Each object represents different reward to the agent. The reward is given by the weight vector.	73
4.4	Foraging environment. Average reward over 10 random seeds. Figs. 4.4(a)-4.4(b) represent source tasks in foraging environment. Figs. 4.4(c)-4.4(f) represent the target tasks.	73
4.5	Foraging environment. Holdout losses in target tasks for h -GPI agent.	75
4.6	CartPole environment. Results are averaged over 10 random seeds, and a vertical line indicates the end of the adaptation phase.	77
4.7	Evaluation of uncertainty-based policy selection. Results are averaged over 10 random seeds, and a vertical line indicates the end of the adaptation phase. . .	78
5.1	Overview of proposed SF composition. (A) Source task policies. (B) High-level state space. (C) Composed policy.	83
5.2	High-level SF composition for Transfer Reinforcement Learning (TRL). Z and o denote the high-level state and action, s and a the low-level state and action, and w the current task.	85
5.3	Four-room environment. <i>Cyan</i> dots are source tasks, with $S_0 = [1.0, 0.0, 0.0]$, $S_1 = [0.0, 1.0, 0.0]$, $S_2 = [0.0, 0.0, 1.0]$. <i>Orange</i> and <i>grey</i> dots are target tasks. <i>Grey</i> dots we considered <i>GPI-challenging tasks</i> due to they represent the farthest tasks. 93	
5.4	Four-room Environment. Source tasks average rewards. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.	95

5.5	Four-room Environment. Target tasks average rewards. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.	95
5.6	Four-room Environment. Target tasks average rewards. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.	95
5.7	Results for the 16 target tasks in the simplex of $\mathbb{R}^3$ for the Four-room Environment per target tasks. Average of total reward over 10 random seeds with 1 standard deviation.	96
5.8	Four-room Environment. <i>Cyan</i> dots represent source tasks. $S_0 = [1.0, 0.0, 0.0]$, $S_1 = [0.0, 1.0, 0.0]$, $S_2 = [0.0, 0.0, 1.0]$. $T_3, T_7, T_{11}, T_{15}, T_{16}, T_{17}$, and T_{21} are <i>GPI-challenging tasks</i> . Average reward over 10 random seeds with shaded area indicating 1 standard deviation.	96
5.9	Reacher environment. <i>Cyan</i> dots are source tasks, with $S_0 = [1.0, 0.0, 0.0, 0.0]$, $S_1 = [0.0, 1.0, 0.0, 0.0]$, $S_2 = [0.0, 0.0, 1.0, 0.0]$, and $S_3 = [0.0, 0.0, 0.0, 1.0]$. <i>Orange</i> and <i>grey</i> dots are target tasks. <i>Grey</i> dot we considered <i>GPI-challenging task</i> . T_6 is the barycenter projected to one of the tetrahedron faces.	97
5.10	Reacher Environment using 2 close coordinates. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.	98
5.11	Target tasks results for the Reacher Environment using 2 close coordinates. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.	99
5.12	Reacher Environment using 2 close coordinates. <i>Cyan</i> dots are source tasks. T_6 is the barycentre projected to one of the tetrahedron faces. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.	100
5.13	Results for the remaining 6 target tasks in the simplex of $\mathbb{R}^4$ for the Reacher environment per target tasks using 2 close coordinates Fig. 5.10. Average of the 11 target tasks. Average of total reward over 5 random seeds with 1 standard deviation.	100
5.14	Reacher Environment using 4 distant coordinates. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.	101
5.16	Reacher Environment using 4 distant coordinates. <i>Cyan</i> dots are source tasks. T_6 is the barycentre projected to one of the tetrahedron faces. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.	102
5.17	Results for the 6 remaining target tasks in the simplex of $\mathbb{R}^4$ for the Reacher Environment per target tasks using 4 distant coordinates. Average of total reward over 5 random seeds with 1 standard deviation.	103

LIST OF FIGURES

5.18	Foraging environment. <i>Cyan</i> dots are source tasks, with $S_0 = [1.0, 0.0]$, $S_1 = [0.0, 1.0]$. <i>Orange</i> dots are target tasks.	104
5.19	Foraging environment. Source task average performance.	104
5.20	Foraging environment. Average of total reward over 10 random seeds with shaded area indicating 1 standard deviation.	105
5.21	Foraging environment. <i>Cyan</i> dots are source tasks, with $S_0 = [1.0, 0.0]$, $S_1 = [0.0, 1.0]$. Average of total reward over 10 random seeds with shaded area indicating 1 standard deviation.	105
5.22	Ablation and task distance analysis in the four-room environment. Average total reward over 10 random seeds with shaded area indicating 1 standard deviation.	106
5.23	Impact analysis of source and target task distances. Average total reward over random seeds with shaded area indicating 1 standard deviation.	106
5.24	Principal Component Analysis (PCA) of high-level states for task $[0.33, 0.33, 0.33]$ in four-room.	107
6.1	Learning platform that supports Self-regulated learning (SRL).	119
6.2	Parser from raw data to SRL processes.	120
6.3	A. Task (e.g., essay writing). B. Overview of Osakwe et al. (2024) approach to address model SRL using RL. C. Propose approach to scale up implementations of TRL for SRL.	120
6.4	ψ -network architecture using LSTM network.	123
6.5	A. ENA representations of SRL behaviours for source and target task. B. Network difference graph.	124
6.6	Left: Source context with balanced reward—learning steps vs. episode reward. Middle: Target context based on NLG. Right: Target context based on ES. Results are averaged over 6 random seeds, using a moving window of 50, and the shaded area representing 1 standard deviation.	126

LIST OF TABLES

TABLE	Page
3.1 Hyperparameters used for SFDQN agent.	41
3.2 Hyperparameters used for TSFDQN agent.	42
3.3 Scenario 1: Target coordinates for the Reacher environment and torque multiplier assigned to each task.	44
3.4 Scenario 2: Target coordinates and torque multipliers for the Reacher Environment.	46
3.5 Ω values after 2×10^6 training steps for Scenario 1, diverse dynamics setting. .	50
3.6 Ω values after 3×10^6 training steps for Scenario 2, diverse dynamics setting. .	51
3.7 Ω for target tasks in Cartpole environment.	53
3.8 Ω for target tasks in Hopper environment.	54
4.1 Gaussian process agents hyperparameters for Foraging and CartPole environments.	72
4.2 h -GPI agent hyperparameters for Foraging environment.	72
4.3 Successor features and ϕ model hyperparameters for Foraging and CartPole environments.	72
5.1 Hyparameter settings for low-level policies using tabular successor features. . .	91
5.2 Hyparameter settings for low-level policies using function approximation. . . .	91
5.3 Hyparameter settings for high-level policy.	93
5.4 Target tasks values for the Four-room environment.	94
6.1 SRL Processes and their definitions across different cognitive categories. . . .	115

ABBREVIATIONS

GP	Gaussian Process
GPI	Generalised policy improvement
MDP	Markov Decision Process
RL	Reinforcement Learning
SF	Successor Feature
SR	Successor Representation
SRL	Self-regulated learning
TD	Temporal Difference
TRL	Transfer Reinforcement Learning
TSF	Transformed Successor Feature

ABSTRACT

Reinforcement Learning has demonstrated remarkable results as a paradigm for machine learning, learning from trial-and-error interactions with an environment. Nevertheless, this paradigm requires an extensive number of samples to perform a specific task, and for a new, unseen task the learning process must start from scratch. Transfer Reinforcement Learning (TRL) is an emerging paradigm that aims to improve sample efficiency by reusing and adapting previously learnt knowledge in new tasks.

In recent years, Successor Features (SFs) have been a widely studied framework for TRL. SFs are a robust mechanism for transfer learning that excel in scenarios where rewards are a linear combination of features. They achieve transfer by decoupling transition dynamics from rewards. Moreover, together with Generalised policy improvement (GPI), they form a powerful framework able to compose and transfer solutions to new tasks. This framework has shown promising results in diverse applications such as neuroscience, robotics, autonomous driving, and control.

Despite these outstanding results and applications, SFs and GPI present two main limitations. First, they assume that transition dynamics are fixed and shared across tasks, an assumption that hinders knowledge transfer in realistic scenarios where subtle variations in physical or environmental factors can lead to significantly different dynamics. Second, the performance of SFs and GPI on target tasks is theoretically tied to their distance from the source tasks, which limits their ability to guarantee optimal policies for more distant tasks. These challenges highlight the need for new composition mechanisms.

This thesis explores mechanisms to enhance SFs and GPI, addressing their current limitations and making the framework more suitable for complex real-world applications. Furthermore, it considers a novel setting in which Reinforcement Learning (RL) is not feasible, showing how SFs can make RL applicable to support human learning.

INTRODUCTION

Gigni de nihilo nihil; in nihilum nil posse reverti

Persius, Satirae

Learning is an essential process for both biological and artificial systems (Neftci & Averbeck, 2019; Witzany, 2018). Biological systems require learning to adapt, compete, and survive in highly dynamic environments. Artificial systems, such as machines, are tools that increasingly shape and influence those same environments. Therefore, it is crucial to study how these tools can be developed to positively affect both their environments and the biological systems within them. For machines to have a positive impact, they too must become more intelligent and adaptive, which fundamentally requires learning.

Machine Learning (ML) has emerged as a major research field with growing influence on everyday life. Its origins can be traced back to foundational questions about whether machines could *think* (Turing, 1950). Over time, the question shifted from possibility to implementation, focusing on how to build artificial systems capable of imitating intelligence, recognising patterns (Rosenblatt, 1957), and ultimately learning from experience (Samuel, 1959).

A machine that learns from experience resembles how humans collect, process, and reflect upon their experiences for future use. Yet, this vision raises major challenges, how to efficiently collect experiences, define the purpose of learning, interpret the context in which experience occurs, and exploit what has already been learnt.

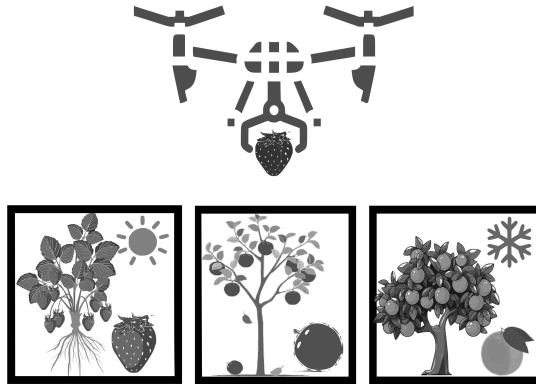
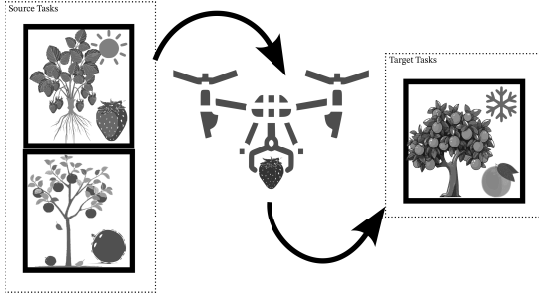


Figure 1.1: Harvesting drone across varying crops and weather conditions.

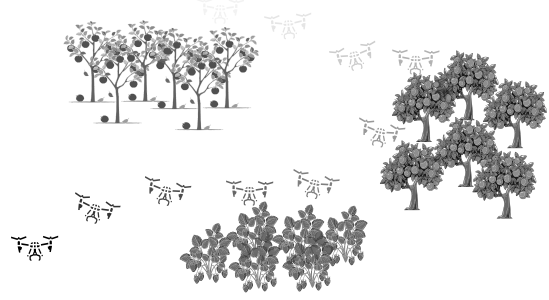
Reinforcement Learning (RL) addresses many of these challenges by providing a rigorous mathematical framework for learning through trial-and-error interactions (Sutton & Barto, 2018). In recent years, RL has gained significant momentum, strengthened by advances in supervised and unsupervised learning, as well as in deep neural networks (Hochreiter & Schmidhuber, 1997; Krizhevsky et al., 2017; R. K. Srivastava et al., 2015; Vaswani et al., 2017). These advances have enabled RL to achieve remarkable, sometimes superhuman, performance in complex games and robotic control tasks (Barreto et al., 2018; Mnih et al., 2015; Schrittwieser et al., 2020), as well as impactful real-world applications such as resource planning (Mu et al., 2024; J. Zhang, Ren, et al., 2024), robotics (Kober et al., 2013), and recommender systems (Lin et al., 2024).

Nevertheless, classical RL requires a considerably high number of interactions with the environment to learn even a single task, making it resource-intensive and often impractical in real-world scenarios. This limitation becomes even more noticeable when learning must extend across multiple tasks. For example, consider a harvesting drone, Fig. 1.1, designed to pick three different crops: oranges, strawberries, and apples. The drone receives rewards for picking each type of fruit. Each crop has unique characteristics, and classical RL would require millions of interactions to master harvesting only one type. When switching to another crop, the system typically needs to relearn from scratch, which makes the approach inefficient and unscalable. This inefficiency highlights the need for more scalable learning approaches. We use the harvesting drone example throughout this chapter as a running illustration of transfer learning challenges.

Intuitively, machines would expect to reuse the knowledge acquired in previous tasks when solving new ones. One possible solution is to assume access to multiple tasks simultaneously while learning to solve a single task using reinforcement learning (Caruana, 1997). However, this approach, known as Multi-Task Reinforcement Learning (MTRL),



(a) Harvesting drone across varying crops by reusing knowledge across tasks using transfer reinforcement learning.



(b) Harvesting drone across varying crops through successor representations perspective.

Figure 1.2: Harvesting drone across varying crops and weather.

relies on the assumption of concurrent access to all tasks. A more general idea forms a family of techniques known as Transfer Reinforcement Learning (TRL). TRL is an emerging paradigm that seeks to reduce resource demands and mitigate task specificity, and has shown outstanding results and impact in a variety of research domains and real-world applications (Wulfmeier et al., 2023), including forecasting of wind discrepancies (J. Zhang et al., 2025), robotics and control systems (Ju et al., 2022; Szoke et al., 2023; X. Wang et al., 2024; Z. Wang & Pang, 2025; Zeng et al., 2025), and human–robot collaboration (J. Wang et al., 2025).

To achieve effective transfer, two fundamental questions must be addressed: *what* knowledge can be reused across tasks and *how* should it be reused (Lazaric, 2012; Zhu et al., 2023a). TRL typically distinguishes between two sets of tasks. A set of *source tasks*, which are used to determine *what* knowledge is transferable, and a set of *target tasks*, where the previously acquired knowledge is applied, thereby answering the question of *how* to reuse it. In the harvesting drone example, different crops can be defined as source and target tasks, as illustrated in Fig. 1.2(a).

A promising answer to the question of *what* to transfer is representation transfer (Zhu et al., 2023b). In representation transfer, knowledge is encoded in learnt representations that capture expressive and interpretable task structures, which can be reused across diverse tasks in artificial intelligence problems (Barreto et al., 2017; Dayan, 1993; Le Lan et al., 2022; Ma et al., 2018, 2020; Schaul et al., 2015; Sutton et al., 2011). Following this line of research, Dayan (1993) introduced the concept of successor representation (SR), a method to predict future state visits and to define these predictions as reusable representations for different tasks. In the harvesting drone example, Fig. 1.2(b), the agent predicts future drone positions within the crops and bases its decisions on these predictions. These

representations can then be reused to transfer knowledge across multiple crops; that is, by using these representations, the drone can navigate and harvest across seasons without relying on season-specific representations.

This thesis aims to strengthen the Successor Feature (SF) framework, enhancing its ability to address the complexity and variability of real-world tasks.

1.1 Background and Motivation

Successor-style representations have been demonstrated to be useful in biological and artificial intelligence scenarios. Recent studies have revealed neural evidence that the human brain predicts future states through successor representations (Momennejad et al., 2017; Russek et al., 2021). Successor representations have also served to model spatial navigation in the hippocampus (De Cothi & Barry, 2020), bridging computational and human-brain models (Madarasz & Behrens, 2019).

Building on this foundation, Barreto et al. (2017) generalised successor representations to state–feature representations, proposing the SF framework. This generalisation enabled the use of function approximators and continuous spaces. Recently, Barreto et al. (2018) introduced generalised policy improvement (GPI), a mechanism for reusing and composing successor features across tasks. Collectively, SF and Generalised policy improvement (GPI) provide a principled foundation for representation transfer, forming a central foundation for scalable TRL methods.

Formally, successor-style representations assume that reward functions can be decoupled as a linear combination of features, $r(s, a) = \phi(s, a)^\top \mathbf{w}$, where ϕ is a feature vector and $\mathbf{w}$ represents task-specific weights (Barreto et al., 2017; Dayan, 1993). By varying $\mathbf{w}$, different reward structures can be yielded while reusing the same learnt feature representation. This decomposition enables transfer across tasks that differ in reward functions. However, successor-style representations assume that tasks share the same transition dynamics; that is, the probability distribution over next states, given the current state and action, remains unchanged across environments. This assumption limits transfer to settings where only reward structures vary (Abdolshah et al., 2021; Lehnert et al., 2022).

Similarly, GPI can be viewed as a mechanism for composing learnt from source tasks (Carvalho et al., 2023; Tasse et al., 2020). It leverages the knowledge from previously learnt successor representations to build improved policies for new tasks. While GPI enjoys strong theoretical guarantees and tight performance bounds, its effectiveness depends on the quality and diversity of the source policies (Alegre et al., 2022; Machado et al., 2023).

Therefore, its performance is bounded by the behaviours from the source task set (Nemecsek & Parr, 2021). A detailed analysis of these theoretical bounds is provided in Chapter 2.

SF and GPI have been studied jointly as representation transfer and composition methods, with compelling results in traffic control (Szoke, Aradi, & Tettamanti, 2022), self-driving (Szoke, Aradi, Becsi, & Gaspar, 2022; J. Zhang et al., 2017), medical robotics (Hu et al., 2025), and in addressing fundamental RL problems such as unsupervised pre-training (Kim et al., 2024). Despite these results, the original SF approach is built on assumptions and limitations, such as tasks sharing the same underlying environment transition dynamics, and policy composition based on GPI having a tight upper bound (Abdolshah et al., 2021; Nemecsek & Parr, 2021).

This assumption is uncommon in real-world problems. Concrete examples include a self-driving automobile that must navigate in real scenarios across different types of roads, each requiring specialised driving skills (Dulac-Arnold et al., 2021); an agent balancing a cart-pole system where the pole dimensions vary under certain conditions (Abdolshah et al., 2021); and an agent controlling robotic arms with different physical specifications, such as length or weight. In addition, our harvesting drone (Fig. 1.2) must navigate across multiple crop layouts and changing weather conditions. In all these cases, transition dynamics differ, rendering transfer based solely on shared dynamics impractical.

Moreover, while GPI facilitates both rapid policy improvement and behaviour composition, its reliance on combining previously learnt successor features introduces structural limitations. The tight performance bounds of GPI suggest that improvements are constrained by the span of available source policies, motivating exploration of alternative approaches that relax these limitations.

1.2 Research Questions and Objectives

In this thesis, we address these limitations of the SF and GPI framework in order to make SF-based agents more suitable for real-world problems. Accordingly, we propose a set of algorithms, theoretical proofs, and mechanisms aimed at answering the central question of **How can *successor features* be enhanced to better support real-world problems?** From this fundamental question, we derive the following research questions and objectives.

-  **RQ-1** How can *successor features* be generalised to support transfer across environments with diverse transition dynamics?

Successor features have proven effective as a method of representation transfer under constrained conditions, such as when transition dynamics remain fixed. However, this assumption is unrealistic in many real-world scenarios, for instance, when harvesting crops under varying weather or physical conditions. By exploring generalisation as a guiding principle for knowledge transfer (Schaul et al., 2015; Sutton et al., 2011; Zhu et al., 2023b), *successor features* can be extended and reused in contexts where transition dynamics differ, thereby broadening their applicability to real-world tasks.

- RQ-2** How can *successor features* learnt in source tasks be adapted to target tasks with different transition dynamics and reward structures while modelling epistemic uncertainty?

Building *successor features* on the basis of generalisation introduces further constraints, particularly regarding the relationships between transition dynamics. In practice, such generalisation often requires prior knowledge of how tasks and their dynamics are related, which is rarely available in real-world scenarios. This research question focuses on adaptation as an alternative to generalisation, exploring methods for reusing knowledge from source tasks to solve target tasks where both transition dynamics and reward structures differ, while also accounting for epistemic uncertainty (Dimitrakakis & Ortner, 2022; Eysenbach & Levine, 2022).

- RQ-3** How can *successor features* be adaptively composed to achieve transfer reinforcement learning?

Successor features, along with *Generalised Policy Improvement (GPI)*, form a powerful framework for representation transfer and composition. Nevertheless, GPI has been shown to have a tight performance bound (Nemecsek & Parr, 2021). This limitation motivates the exploration of alternative or complementary methods for composing *successor features*. Therefore, this research question focuses on developing and evaluating such alternative composition strategies beyond GPI.

- RQ-4** To what extent does transfer reinforcement learning using *successor features* enhance reinforcement learning in real-world scenarios?

Reinforcement learning has been successfully applied across a wide range of research fields and real-world applications. In recent years, it has also been explored as a tool to support human learning. Despite the demonstrated success of *successor features* in TRL for domains such as robotics, self-driving cars (Szoke, Aradi, & Tet-

tamanti, 2022), traffic control (Szoke et al., 2023), and neuroscience (Abdolshah et al., 2021; De Cothi & Barry, 2020; Momennejad et al., 2017; Russek et al., 2021), they have not yet been directly applied to support human learning. This research question addresses this gap by exploring how SF-based TRL can be leveraged in real-world applications where classical RL methods struggle to scale effectively.

To address these research questions, we outline the following research objectives (ROs):

- RO-1 Propose a novel definition of *successor feature* grounded in latent transition dynamics, providing a conceptual basis for transfer reinforcement learning in environments with differing transition dynamics.

This research objective addresses RQ-1. In exploring the generalisation of *successor features* across transition dynamics, we propose a method that operates at the level of the dynamics themselves. Specifically, we assume the existence of latent transition dynamics that can be leveraged to transform *successor features*, enabling transfer across tasks with differing dynamics.

- RO-2 Design a method to transfer the learnt *successor features* to target tasks, leveraging shared latent transition dynamics across tasks with differing transition dynamics.

This research objective also addresses RQ-1. While the proposed latent shared transition dynamics provides a pathway to generalisation across contexts with differing dynamics, RO-1, it also introduces the challenge of learning *successor features* within this latent space. To address this, we propose a transformed feature function that encodes the shared latent transition dynamics, while simultaneously finding optimal policies for source tasks. To encode the latent transition dynamics, we made use of transformations of probability distributions, where the transition dynamics of source tasks are mapped onto the shared latent transition dynamics. In doing so, this objective contributes to answering the primary question of TRL, *what* knowledge should be transferred. Once generalised *successor features* that encode the shared latent transition dynamics have been learnt from source tasks, the next challenge concerns the TRL question of *how* to transfer this knowledge. To tackle this, we propose a method based on ensemble models to enable transfer across tasks with differing transition dynamics. Moreover, the proposed ensemble is formulated of a convex combination of *successor feature* representation for target tasks, and the latent transition dynamics is expressed as a convex combination of transformed source transition dynamics.

- RO-3 Develop a method to adapt *successor features* from environments with varying transition dynamics and reward structures to target tasks.

This research objective addresses RQ-2. Our approach made use of the inherent transfer mechanism of Gaussian Processes and the flexibility of kernel functions, which can be tailored to adapt the *successor features* of source tasks for solving new, unseen target tasks. In doing so, this objective enables knowledge adaptation across varying transition dynamics within the context of TRL.

- RO-4 Develop a method to capture epistemic uncertainty from successor features to enhance policy selection within the Generalised Policy Improvement (GPI) framework.

This research objective also addresses RQ-2. In addition to the flexibility offered by Gaussian Processes to specify kernel functions, they also capture epistemic uncertainty. Building on this property, we propose a method that leverages this advantage to examine the extent to which the proposed approach enhances Generalised Policy Improvement (GPI). Together with RO-3, this objective supports knowledge adaptation across varying transition dynamics within the context of TRL, while explicitly capturing epistemic uncertainty

- RO-5 Design a method to adaptively learn and compose *successor features*.

This research objective addresses RQ-3. Generalised Policy Improvement (GPI) has been widely studied as a primary mechanism for composing *successor features* in TRL. GPI can be interpreted as a hierarchical strategy in which the policy to follow is selected at each timestep. However, this method relies on the assumption of a fixed relationship between the vector weights of source and target tasks, and this assumption has been shown to establish a tight performance bound. Our aim is to introduce an alternative mechanism for composing *successor features* that can overcome these performance limitations.

- RO-6 Identify a real-world scenario where classical reinforcement learning falls short, and demonstrate how transfer reinforcement learning using *successor features* can make reinforcement learning a practical solution.

This research objective addresses RQ-4. In exploring research fields that could benefit from TRL, we identified the area of human learning, specifically learning analytics, as a promising domain. Reinforcement learning has already been applied in

this field to support how humans can benefit from intelligent systems. In particular, self-regulated learning (SRL) has been widely studied as a crucial skill for human learning processes, and reinforcement learning is being investigated as a means to scaffold and support it. However, we argue that classical reinforcement learning is not practical for SRL due to its scalability and adaptability limitations. This objective therefore examines the feasibility of using TRL, and in particular *successor features*, as a way to support SRL and, more broadly, human learning.

1.3 Contributions

This thesis enhances *successor features* to better support real-world problem solving. To achieve this, we propose methods that relax common assumptions and limitations of *successor features*, and we introduce a novel application domain that bridges machine learning with human learning. The aim is to design machine learning techniques that have a positive impact in biological learning environments. Accordingly, the contributions of this thesis are as follows:

- A method to predict *successor features* within a shared latent transition dynamics, we named *Transformed Successor Feature*.
- A method to learn *Transformed Successor Feature* and derive optimal policies for source tasks by using probability transformation techniques.
- An ensemble model-based method to transfer *Transformed Successor Feature* from source tasks onto target tasks, thereby enabling *successor features* to be transferred across tasks with differing transition dynamics.
- A novel *successor features*-based Dynamics Transfer Kernel for adapting and transferring *successor features* across tasks with dissimilar dynamics, supported by theoretical evidence of its validity.
- An uncertainty-based policy selection method made us of Gaussian Process inherit uncertainty computation.
- A high-level Markov decision process with high-level state and action spaces using *successor features* to support decision-making and leverage prior knowledge for unseen tasks.

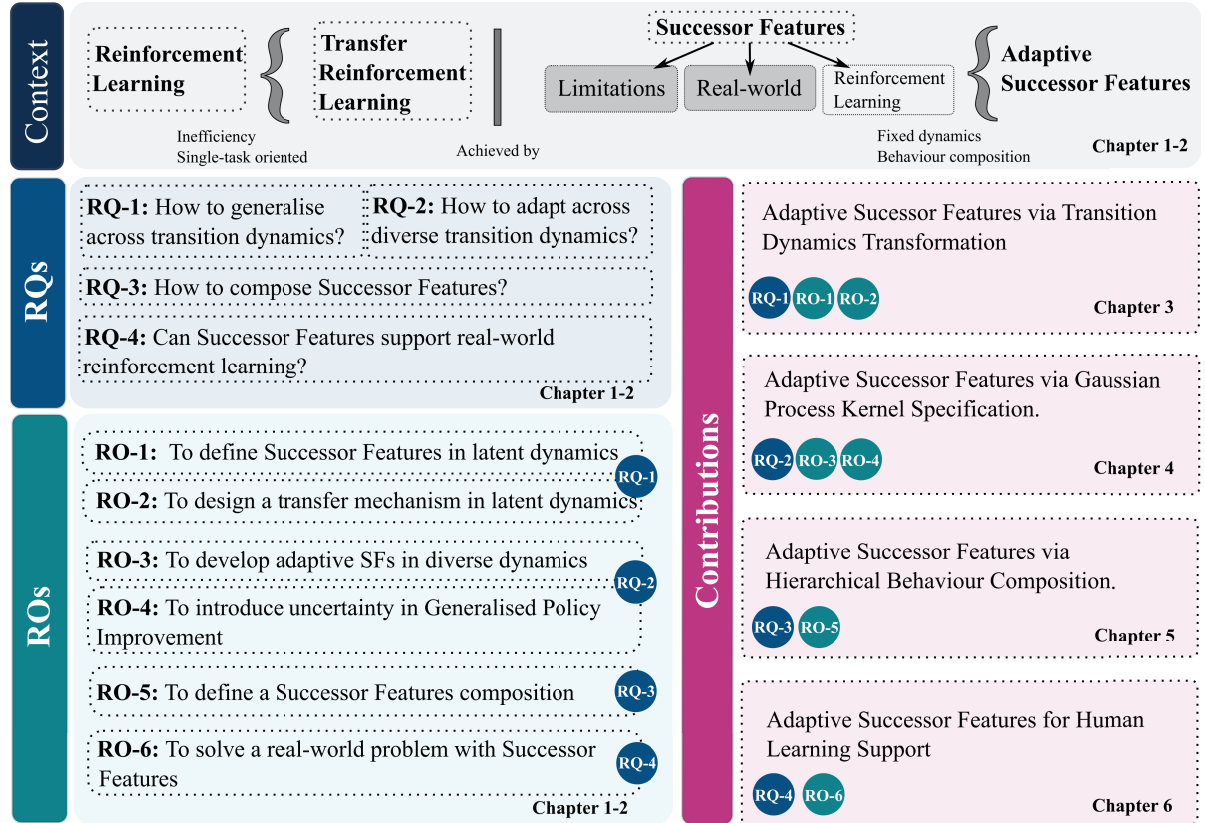


Figure 1.3: Thesis organisation diagram.

- A novel algorithm for learning and composing *successor features* from source tasks, considering them as composable behaviours.
- Problem formalisation of scaffolding self-regulated learning with reinforcement learning as a Transfer Reinforcement Learning setting and demonstrated the feasibility of transferring learnt self-regulated learning strategies to new study sessions with different rewards via *successor features*, thus enabling scalable transfer of optimal policies.

1.4 Thesis Organisation

An overview of the thesis organisation is presented in Fig. 1.3. The diagram outlines the context of this thesis, the main objective, and the derived research questions and objectives. It also illustrates the relationships between the research objectives and their corresponding research questions, and shows how the contributions are linked to both. Overall, the diagram serves as a guide to navigate the structure of the thesis.

Chapters 1 and 2 provide the context, motivation, and literature review. *Chapter 3* addresses **RQ-1** by accomplishing research objectives **RO-1**, **RO-2**, and **RO-3**. *Chapter 4* introduces methods to adapt successor features across different transition dynamics, thereby achieving **RO-4** and addressing **RQ-2**. Similarly, *Chapter 5* presents an alternative mechanism to compose successor features, **RO-5**, in order to answer **RQ-3**. *Chapter 6* introduces a novel application context, addressing research objective **RO-6** and **RQ-4**. Finally, *Chapter 7* summarises the thesis, discusses the findings, and outlines future research directions.

LITERATURE REVIEW

*The scientific method required a shift from
believing to doubting.*

*Yuval Noah Harari, Sapiens: A Brief History of
Humankind*

Knowledge transfer in reinforcement learning through Successor Feature (SF) representations has been extensively studied, spanning both theoretical foundations and practical applications to real-world problems. This chapter provides the context and theoretical background needed to understand SF and its role in Transfer Reinforcement Learning (TRL). We begin by introducing the fundamental concepts of Reinforcement Learning (RL). We then outline the main principles of TRL, describing how it is typically formulated, and explain how SFs has been proposed as a mechanism to achieve transfer. Following this, we review the literature on SF, identify the main gaps in existing approaches, and organise the discussion according to these gaps.

2.1 Reinforcement Learning

RL is a paradigm for machine learning algorithms to acquire knowledge through trial and error. The acquired knowledge is used to solve sequential decision making problems. The decision maker is an *agent*, and everything it interacts with is called the *environment*. In RL problems, the agent interacts with the environment and, as a consequence, obtains an *observation* and a *reward*. The observation is the new environment state after the interac-

tion, and the reward is a stimulus for the chosen decision. Therefore, RL can be viewed as an optimisation framework concerned with choosing the most long-term rewarding decisions (Sutton & Barto, 2018).

The complex agent interactions are commonly modelled as an Markov Decision Process (MDP). An MDP is defined by the tuple $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p, r, \gamma)$ (Puterman, 1990; Sutton & Barto, 2018), where $\mathcal{M}$ is the MDP; $\mathcal{S}$ and $\mathcal{A}$ are the states and action spaces; $p(\cdot|s, a)$ is the transition dynamics, *i.e.*, the transition probability distribution that defines the distribution of the next state given a state, s , taking action, a ; r is a reward function that can be defined as $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$; and $\gamma \in [0, 1)$ corresponds to the discount factor on future rewards.

The sequence of decisions followed by the agent is defined by the policy $\pi = \mathcal{S} \rightarrow \mathcal{A}$; that is, the policy gives the next action $a \in \mathcal{A}$ to choose given the current agent state $s \in \mathcal{S}$. As mentioned above, the aim of RL is to discover the best sequential decisions, hence the optimal policy. This optimal policy $\pi^* : \mathcal{S} \mapsto \mathcal{A}$ returns the best action given a state, expecting to maximize the future rewards $\sum_{t=0}^{\infty} \gamma^t r_t$.

To find the π^* , the literature divides the mechanisms into two sets: *model-free* and *model-based* approaches. *Model-based* approaches aim to model the environment to predict its behaviour; hence, they have a notion of the future rewards given by the environment. These methods are well-known for their planning capacity, therefore optimise the number of interactions; but they are also expensive to model complex environments. On the other hand, *Model-free* approaches are inexpensive, as they require a greater quantity of experiences gained through trial and error (Sutton & Barto, 2018). Sutton and Barto (2018) sustains that they are not mutually-exclusive and can be used to support the learning process.

This thesis mainly focuses on *model-free* algorithms, specifically on the set of algorithms derived from Q-learning, due to inexpensive solutions in large scale environments; besides, the recent advances in deep RL demonstrate outstanding performance in complex environments designed to challenge humans (Mnih et al., 2015; Schrittwieser et al., 2020), addressing real-world problems such as stock markets (Lee et al., 2019), autonomous-aerial vehicles, and the Internet of Things (Seid et al., 2021).

Q-based learning algorithms rely on the computation of state-action values. This describes the expected future reward of an agent in a certain state, $s \in \mathcal{S}$ takes the action

$a \in \mathcal{A}$ and follows the policy π . This value is computed as,

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_{\substack{a_{t+1} \sim \pi \\ s_{t+1} \sim p}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right] \\ &= \mathbb{E}_{\substack{a_{t+1} \sim \pi \\ s_{t+1} \sim p}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a \right]. \end{aligned} \quad (2.1)$$

Many dynamic programming-based RL algorithms alternate between two processes (Puterman, 1994): *policy evaluation* to compute Q^π and *policy improvement*, which obtains policies that take greedy actions upon the Q function $\pi'(s) \in \arg \max_a Q^\pi(s, a)$. Constant alternation of both processes will converge in π^* (Puterman, 1994; Sutton & Barto, 2018).

Despite the inexpensive computation of Q values and the straightforward derivation of optimal solutions suitable for large and complex environments. These algorithms require extensive samples from the environment to achieve an optimal solution (Lazaric, 2012; Zhu et al., 2023b). Moreover, the agents solve that task; any slight change can cause the agent to perform poorly. That being said, performance in new or unseen environments can be disastrous. For example, a harvesting drone, Fig. 1.2, may require millions of interactions within a specific crop field to learn how to navigate it effectively. However, if the drone is deployed in a different field with a new layout or varying weather conditions, it would again require the same number of interactions to adapt to the new environment. This makes this body of approaches inconvenient for real-world problems.

To alleviate this, researchers have investigated approaches to reduce task-specificity and improve sample inefficiency by reusing past-learnt knowledge without completely restarting the learning process. TRL is one of these optimisation techniques (Zhu et al., 2023b).

2.2 Transfer Reinforcement Learning

TRL is a compelling set of techniques to answer *what* knowledge is learnt and *how* to reuse it in new tasks (Lazaric, 2012). TRL commonly defines two sets of tasks. The first set is used by the agents to extract knowledge, named the set of source tasks $\mathbf{M}^s \equiv \{M_i | i \in \{1, \dots, n\}\}$. The second set corresponds to the set of target tasks, in which knowledge is transferred to and is defined by $\mathbf{M}^t \equiv \{M_j | j \in \{1, \dots, m\}\}$ for the target tasks m . Again, the main goal of TRL is to reuse the knowledge learnt in $\mathbf{M}^s$ in $\mathbf{M}^t$ (Lazaric, 2012; Zhu et al., 2023b).

Numerous approaches have been proposed to reuse the previously learned knowledge from M^s into M^τ (Lazaric, 2012; Zhu et al., 2023b). To categorise existing work, Zhu et al. (2023b) classifies it by the type of knowledge that is transferred by answering the question of *what knowledge is transferred*. One of these approaches is *disentangled representation transfer*. *Disentangled representation transfer* aims to build feature representations to be used for the Q value computations, under the assumption that the state and action spaces are shared and can be disentangled.

A few works have proposed to perform disentangled representation transfer. We understand that the following works can be classified into two main groups: *value function generalisation* and *successor representation*. *Value function generalisation* aims to disentangle the Q value function in order to build a reusable value function in M^τ . A generalisation of the value function allows the capacity to reuse knowledge. In Sutton et al. (2011), the authors state that value functions can be used as semantics to represent knowledge and can be reused among tasks. In addition, Sutton et al. (2011) proposed the *Horde* architecture as a mechanism to learn a set of value functions called general value functions (GVF). In further research, Schaul et al. (2015) used the notion of *Horde* architecture to build universal value function approximators (UVFA). UVFAs are generalised over state spaces and goals through the decomposition of learning state and goal embeddings. Furthermore, the authors defined the goals as a subset of the final states. However, in such generalisation approaches, learning a good GVF and UVFA is challenging (Ma et al., 2018); moreover, it is assumed that the *Horde* architecture has continuous access to the current environment in order to generate a good quality function.

On the other hand, Successor Representations (SRs) were formally described by Dayan (1993). He formulated Q value could be obtained based on the expected frequency of future state visits and a reward vector. All of these are only for tabular scenarios. In this context, a few works have been proposed. Madarasz and Behrens (2019) propose a faster convergent solution for SR, which adapts rapidly to new environmental changes, taking advantage of the fact that optimal policies should be close to similar reward functions. Therefore, Madarasz and Behrens (2019) proposes that the use of Dirichlet process to create a cluster of tasks and an optimal policy works for all the cluster. Machado et al. (2023) formalises the relevance of SR in different levels of temporal abstraction in RL; however, their work does not focus on TRL. The aforementioned works were limited to tabular scenarios or finite state spaces. To abstract finite states into state feature representations, Kulkarni, Saeedi, et al. (2016) brings a neural network architecture to learn low-dimensional state representations. To extend SR into continuous spaces, Barreto et al. (2017) proposes SFs,

which is the focus of this thesis.

2.3 Transfer Reinforcement Learning via Successor Features

Barreto et al. (2017, 2018) extended the concept to SF to represent expected continuous state space features. In SF approaches, the reward function is decomposed into a simple product of a feature function and a weight as

$$r_i(s_t, a_t, s_{t+1}) = \phi(s_t, a_t, s_{t+1})^\top \mathbf{w}_i, \quad (2.2)$$

where $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ denotes the feature function and $\mathbf{w}_i$ is a task-specific vector for the i -th task.

The main idea is that the feature function could be shared by all tasks, which facilitates the knowledge sharing among them. This new representation of the reward function allows us to rewrite the state-value function or Q value function as

$$\begin{aligned} Q_i^\pi(s, a) &= \mathbb{E}_{\substack{a_{t+1} \sim \pi \\ s_{t+1} \sim p}} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a \right]^\top \mathbf{w}_i \\ &= \psi^\pi(s, a)^\top \mathbf{w}_i, \end{aligned} \quad (2.3)$$

where ψ is named *successor feature* and encapsulates the dynamic of the environment. This factorisation of the rewards and Q function grants the agent the possibility of reusing the SFs in the target tasks. Furthermore, this enables fast computation of Q values in tasks with the same transition dynamics $Q_j^\pi(s, a) = \psi^\pi(s, a)^\top \mathbf{w}_j$ and only requires fitting the preferences over the features for the task-specific $\mathbf{w}_j$.

To fit the vector weights and the respective SF, it is crucial to learn from source tasks. It should be noted that SF has the characteristic of following the Bellman equation; therefore, the SF for a task can be learnt by minimising the loss function,

$$\mathcal{L}_\psi = \| \phi(s, a, s') + \gamma \psi^\pi(s', a') - \psi^\pi(s, a) \|, \quad (2.4)$$

and for fitting the vector weights,

$$\mathcal{L}_\mathbf{w} = \| r(s, a, s') - \phi(s, a, s')^\top \mathbf{w} \|. \quad (2.5)$$

Additionally, Barreto et al. (2018) extended the Q learning *policy improvement* to a generalised Policy Improvement (GPI). This makes it possible to obtain the best policy

from a set of policies induced by the set of SFs. In other words, the Generalised policy improvement (GPI) algorithm merges the learnt knowledge from n source tasks and a set of source SFs.

2.3.1 Generalised Policy Improvement

GPI is a generalisation of the processes of *policy improvement* and *policy evaluation*. Instead of evaluating one policy at a time, GPI considers multiple policies. It assumes access to a set of source tasks, M^s , and their corresponding SF, $\Psi = \{\psi^{\pi_{w_i}}\}_{i=0}^n$. GPI enables the computation of an improved policy as,

$$\pi^{GPI}(s; \mathbf{w}) \in \arg \max_{a \in \mathcal{A}} \max_{\psi \in \Psi} \psi(s, a)^\top \mathbf{w}. \quad (2.6)$$

Importantly, Barreto et al. (2017) found a theorem demonstrating that $\pi^{GPI}(s; \mathbf{w})$ performs at least as well as any other policy in the set. The theorem states that the performance gap between the optimal policy for a source task and the GPI policy for a target task is bounded by the difference in task weights and the feature function, as shown in the following theorem,

Theorem 2.1. *Barreto et al. (2017) Given a task defined by an MDP $\mathcal{M}(S, A, p, r, \gamma)$. Let $M^s = \{M(S, A, p, r_i, \gamma) \mid r_i(s, a, s') = \phi(s, a, s')^\top \mathbf{w}_i\}_{i=0}^{n-1}$ be the set of n source tasks, and $M^t = \{M(S, A, p, r_j, \gamma) \mid r_j(s, a, s') = \phi(s, a, s')^\top \mathbf{w}_j\}_{j=0}^{m-1}$ be the set of m target tasks. Let $\Psi = \{\psi^{\pi_{w_i}}\}_{i=0}^{n-1}$ be the successor features (SFs) for the optimal policies for the n source tasks, $Q_{\mathbf{w}_i}^{\pi_{w_i}}(s, a) = \psi^{\pi_{w_i}}(s, a)^\top \mathbf{w}_i$ be the state-action value function of an optimal policy of $\mathbf{w}_i \in M^s$ when executed in target task $\mathbf{w}_j$, and let,*

$$\pi^{GPI}(s; \mathbf{w}_j) \in \arg \max_a \max_{\psi \in \Psi} \psi(s, a)^\top \mathbf{w}_j.$$

Then,

$$Q_{\mathbf{w}_j}^{\pi_{w_j}}(s, a) - Q_{\mathbf{w}_j}^{\pi^{GPI}}(s, a) \leq \frac{2}{1 - \gamma} (\max_{s, a} \|\phi(s, a)\| \min_{\mathbf{w}_i \in M^s} \|\mathbf{w}_j - \mathbf{w}_i\|). \quad (2.7)$$

2.4 Related Work on Successor Features

Successor features literature is extensive and represents an active, ongoing area of research. Recent studies and contributions can be categorised into three main groups, as illustrated in Fig. 2.1.

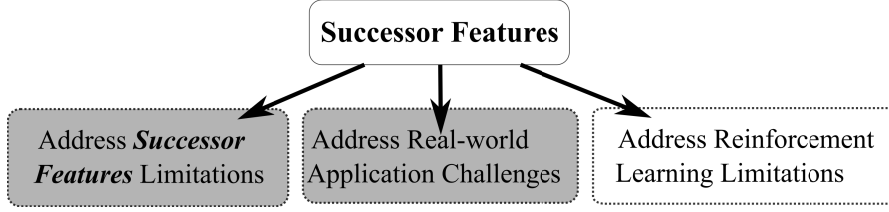


Figure 2.1: Groups of related work on successor features.

2.4.1 Address Reinforcement Learning Limitations

This first body of work focuses on implementing SFs and seizing their properties to address existing limitations of RL. In particular, Tao et al. (2021) proposed *REPAINT* as a solution to improve learning times using transfer learning in similar and dissimilar environments. The goal of *REPAINT* is to transfer knowledge in two stages: the transfer of knowledge about the representation of the policy and the transfer of knowledge about the off-policy instance. To achieve this, the approach used *Policy Distillation* for the instance transfer.

Lehnert and Littman (2020) studied SF as a mechanism that intrinsically bridged model-free and model-based RL. In the same vein, Janner et al. (2020) proposed a hybrid approach that bridges model-free and model-based methods, using SF to enable long-horizon prediction. This approach leverages conditioned policies and infinite-horizon value functions while avoiding reward dependence on model-based methods, though it remains limited to tabular settings.

Hansen et al. (2020) propose *Variational Intrinsic SFs*, combining two techniques: *Mutual Behaviour Information* and *SFs* to improve the computation of features and task inference for unseen tasks. Their implementation is complex due to their attempt to solve two problems: unsupervised pre-training and *Mutual Behaviour Information*. H. Liu and Abbeel (2021a) proposed an improvement to the *Variational Intrinsic SFs* method. This was achieved by incorporating an improved exploration guided by entropy maximisation H. Liu and Abbeel (2021b) and mutual behaviour information. Later, Kim et al. (2024) utilised SF to decouple the problem of exploration and exploitation for unsupervised pre-training. Recent work in unsupervised pre-training utilised SF to yield policies capable of achieving zero-shot transfer (Park et al., 2024; Touati et al., 2023).

Scarpellini et al. (2024) combined foundation models, SFs, and offline data to propose a method $\pi 2vec$ which main focus is to represent black-box policies. Black-box policies are problematic since there is no direct access to the policy, only certain offline data. SFs were used to represent the black-box policy expected behaviours. C. Wang et al. (2024) used SF to address the problem of transferring knowledge from offline-to-online in RL.

Similarly, Filos et al. (2021) used offline data to propose an inverse temporal difference to leverage policies from offline-to-online in RL. Other work combined UVFA and SF, to enhance representation transfer and yield generalisation across tasks (Borsa, Barreto, Quan, Mankowitz, et al., 2019).

Other areas of RL have also leveraged the properties of SF. For instance, Alegre et al. (2022) successfully modelled SFs as a multi-objective RL problem. Then, they exploited this relationship to propose multi-objective RL algorithms for the context of SF. Similarly, Machado et al. (2023) used SFs to address the skill discovery problem in hierarchical RL, showing the potential of SF to model behaviour across temporal abstractions. Gimelfarb et al. (2021) utilised SF to derive robust policies and address risk-constrained settings.

Moreover, Madarasz and Behrens (2019) proposed a faster convergent solution for *Successor Representations* that adapts more rapidly to environmental changes, leveraging the assumption that optimal policies are similar when reward functions are similar. Their approach clusters tasks using a Dirichlet process, under the premise that tasks within a cluster share similar optimal policies. However, this method relies on the assumption that environments are sufficiently similar in both transition dynamics and reward structure for clustering to be effective. When environments exhibit substantial variation in transition dynamics, the similarity assumptions underlying clustering may no longer hold. Consequently, transfer performance becomes sensitive to the degree of environmental similarity.

Despite the demonstrated properties of SFs in addressing existing limitations of RL and yielding state-of-the-art algorithms, in this thesis, we mainly focus on the other two groups of related work: How works have addressed *Successor Features Limitations* and *Real-world Application challenges* using SFs.

2.4.2 Address Successor Features Limitations

SFs and GPI have demonstrated strong performance, particularly under the constraint of shared, fixed transition dynamics. Additionally, GPI is constrained by tight theoretical performance bounds (Nemecek & Parr, 2021). The framework can be formalised as follows,

$$\psi^{\pi^{\text{GPI}}}(s, a) = \underbrace{\mathbb{E}_{\substack{a_{t+1} \in \pi^{\text{GPI}} \\ \text{GPI policy} \\ s_{t+1} \sim p \\ \text{Fixed Transition Dynamics}}}}_{\substack{a_{t+1} \in \pi^{\text{GPI}} \\ \text{GPI policy} \\ s_{t+1} \sim p \\ \text{Fixed Transition Dynamics}}} \left[\sum_{t=0}^{\infty} \gamma^t \underbrace{\phi(s_t, a_t, s_{t+1})}_{\text{Fixed Feature Function}} \mid s_0 = s, a_0 = a \right], \quad (2.8)$$

where π^{GPI} is given by,

$$\pi^{\text{GPI}}(s; \mathbf{w}) \in \underbrace{\arg \max_{a \in \mathcal{A}} \underbrace{\max_{\pi_i \in \Pi} \psi^{\pi_i}(s, a)}_{\text{Set of Optimal Policies}}^\top \mathbf{w}}_{\text{Policy Composition}}. \quad (2.9)$$

As shown in (2.8), SFs the underlying transition dynamics p are fixed and shared among tasks, and GPI policy, (2.9), is a two-level decision making process where a set of policies is composed to derive a new policy. We use (2.8) and (2.9) to organize this literature review by understanding how literature in SFs has addressed these limitations.

Fixed Transition Dynamics A first body of work on SFs has aimed to relax the assumption of *fixed transition dynamics*. This assumption is restrictive in RL settings, where transition dynamics are typically unknown and often subject to variation. To our knowledge, two works have aimed at decoupling dynamics in the successor feature context.

J. Zhang et al. (2017) proposed a linear relationship between the source and target feature functions. However, this formulation assumes that differences between environments can be modelled through linear transformations, which restricts applicability to settings with approximately linear dynamics shifts.

More recently, Abdolshah et al. (2021) proposed representing target SFs as noisy transformations of source SFs using Gaussian Processes. While this approach offers greater flexibility, Gaussian Processes introduce cubic computational complexity, limiting scalability in high-dimensional environments (Goumiri et al., 2020). Furthermore, these methods rely on kernel-based similarity assumptions that may not generalise when transition dynamics differ substantially.

Beyond the SF setting, Gaussian Process models and transferable kernels have been studied extensively in multi-task and transfer learning. For instance, Álvarez et al. (2010) and Bonilla et al. (2007) proposed Intrinsic Coregionalisation Model to capture task relationships but assumes joint access to all task data. Similarly, transferable kernel approaches, where Tighineanu et al. (2022) and Wei et al. (2023) model structured source-target relationships, require explicit similarity structures between tasks.

Collectively, these approaches either assume linear dynamics transformations, rely on strong similarity priors, or suffer from limited scalability. As a result, a principled and scalable method for generalising successor features across environments with substantially differing transition dynamics remains an open challenge.

This gap directly motivates **RQ-1**, which investigates how successor features can be generalised beyond shared or linearly related dynamics, and **RQ-2**, which explores scalable mechanisms for modelling task dissimilarities under dynamic variations.

Beyond direct transfer, SFs have also been studied for their role in bridging model-free and model-based RL (Lehnert et al., 2022). Similarly, Alegre et al. (2023a) integrated model-based planning with SFs by approximating environment dynamics for GPI. These efforts further emphasise the importance of explicitly handling transition dynamics in SF-based methods, reinforcing the central focus of this thesis.

Policy Composition A second body of work is based on problems similar to *composition mechanisms to reuse policies* in target tasks. In this line, Barreto et al. (2020) work centres on managing SFs to apply policy evaluation and policy improvement from a list of tasks and policies independently. Later, Brantley et al. (2021) introduces a solution to remove the static list of tasks and policies. This work is a model-based algorithm that manages multiple policies simultaneously on the assumption that they have access to a perfect environment model. In this line, Tasfi et al. (2023) proposes *DynSF*; this solution also manages and reasons with multiple policies at once, allowing the agent to adapt dynamically to different tasks; however, their work focuses only on settings where feature functions can be expressed solely by the next states. The *DynSF* is capable of predicting SFs.

Recently, Y. T. Liu and Ahmad (2023) uses value composition to compute a new Q value, which is therefore derived from a composed policy. These interesting approaches aid the successor feature in managing a set of policies simultaneously. Nevertheless, these approaches are based on strict assumptions, such as access to the environment model, specific settings, or application to low-dimensional action and state spaces. More importantly, these works similar to GPI, higher differences in the rewards vector can lead to suboptimal policies in target tasks.

Relevant works have explored the composition of fixed behaviours. Barreto et al. (2019) proposed the *Option Keyboard (OK)*, which defines 2 sets of weights: one for low-level policies and another for the high-level policy, where the latter governs the extended execution of options. However, their approach relies on GPI in the cumulant space for composition. Building upon the OK, Carvalho et al. (2023) extended it to compose dynamic behaviours while simultaneously discovering features and tasks.

Machado et al. (2023) and Ramesh et al. (2019) proposed frameworks for option discovery within successor features. Other works have explored composing SFs over long horizons, often leveraging a mix of model-based solutions and SF-GPI (Alegre et al., 2023b;

Janner et al., 2020). All of these works rely on GPI for the policy composition process; it is well-known that GPI has a tight upper-bound with respect to the vector weights distance (Nemecek & Parr, 2021).

Beyond SF, behaviour composition has been a key area of focus for solving new, unseen tasks, primarily through value-function and policy composition. Value-function composition aims to derive new behaviours by combining existing value functions. For instance, several works have explored this approach to produce novel behaviours, including those by Adamczyk et al. (2023), Tasse et al. (2020, 2022), and Van Niekerk et al. (2019).

Within the SF and GPI literature, an example of this type of composition was specifically studied by Y. T. Liu and Ahmad (2024). Policy composition, on the other hand, directly combines existing policies to form new behaviours. Approaches such as those presented in Haarnoja, Pong, et al. (2018), Peng et al. (2019), and L. Wang et al. (2024) have achieved this, with Hunt et al. (2019) demonstrating its application within the SF and GPI framework. Kuric et al. (2024) proposed a method to compose SFs by planning behaviours using automata. Other research, including (Barreto et al., 2017; Barreto et al., 2019; Gupta et al., 2017; Han & Tschitschek, 2022; Shah et al., 2022), has explored hierarchical and representational RL as alternative composition strategies, which motivates **RQ-3** of this thesis.

In SF and GPI, few works have addressed epistemic uncertainty modelling. Malekzadeh et al. (2022) enhanced exploration and exploitation by modelling learning-time SF uncertainty. Madarasz and Behrens (2019) combined successor representations with non-parametric clustering in task space; while their algorithm handled uncertainty across clusters, it did not model epistemic uncertainty within them. C. Wang et al. (2024) addressed epistemic uncertainty via an ensemble. These methods primarily improve SF learning under specific conditions, without tackling the shared dynamics assumption or modelling uncertainty for the GPI process. This identified gap motivates **RQ-2** of this thesis.

Set of Optimal Source Tasks Another body of work focuses on constructing a finite set of basis successor features to guarantee optimal transfer across target tasks. Alver and Precup (2022) developed a mechanism to build a library of basis policies that, when composed with GPI, represented optimal policies. This approach assumed that, for features encapsulating unique behaviours, learning policies focused on those specific features would be sufficient for optimal transfer when rewards were convex combinations of them.

Nemecek and Parr (2021) also defined conditions for building a set of basis behaviours under convex combinations of shared features. In contrast, their method dynamically ex-

panded the set whenever a more relevant policy was discovered that outperformed the existing ones. Zahavy et al. (2021, 2022) explored automatic ways of constructing the set of basis behaviours under convex MDPs.

Finally, Alegre et al. (2022) bridged multi-objective RL by using a high-level algorithm to build the optimal set of source tasks and basis behaviours. Once constructed, this set guaranteed an optimal behaviour for any target task. However, all these approaches rely on GPI and its properties, which demand significant resources to build and maintain the libraries.

Instead of dynamically adding behaviours, the question of whether there is an alternative to GPI for composing fixed behaviours and testing approaches under more challenging conditions, particularly in tasks that are significantly more distant from the source tasks, remains unanswered in the literature and forms the core of  in this thesis.

Fixed Feature Function A third group of approaches aimed at enhancing SFs framework states the difficulty of learning the feature function. Works in this vein can be categorised into two subgroups.

First, there is a body of work that manually designs the feature function, such as (Barreto et al., 2017; Borsa, Barreto, Quan, Mankowitz, et al., 2019; Szoke, Aradi, Becsi, & Gaspar, 2022; Szoke, Aradi, & Tettamanti, 2022). In these approaches, the feature function includes expert knowledge tailored to specific environments.

The second subgroup includes approaches that learn from the data and apply supervised strategies. For example, Carvalho et al. (2023) proposed a new modular architecture for SFs which aims to improve the prediction of latent space representations. Their method relaxes the need for a common state representation, allowing the agent to learn independent state representations and enabling the combination of task solutions. This solution assumes that the feature functions belong to a partition of the latent space. Similarly, Mozifian et al. (2022) presented an algorithm to decompose the learning state representation process from reward characteristics using actor-critic methods.

On the other hand, relevant mechanisms for learning without explicit reward have also been proposed. Hansen et al. (2020) introduced *variational intrinsic SFs*, combining two techniques: *mutual behaviour information* and SFs to improve the computation of the feature function and task inference for unseen tasks. However, this method limits the exploration of new states and actions. To address this constraint, H. Liu and Abbeel (2021a) proposed *active pre-training with SFs* to combine the successor feature framework and active pre-training in unsupervised settings. These approaches rely on intrinsic rewards to

discover features in an unsupervised manner. Nevertheless, they are restricted to discrete state and action spaces, and although they claim state representation independence, performance remains sensitive to the chosen representation H. Liu and Abbeel (2021a).

From a theoretical perspective, learning the feature function can be problematic. As discussed in Machado et al. (2023), $\phi(s, a, s) = 0$ is a fixed point, which may hinder meaningful representation learning. Recall that $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$, where d is the feature dimension. As noted by Barreto et al. (2017) and Dayan (1993), there is no benefit in choosing $d > |\mathcal{S}|^2|\mathcal{A}|$, implying that $d \leq |\mathcal{S}|^2|\mathcal{A}|$.

More recent work on shared feature function learning has sought to overcome the limitations of purely state-based approaches by developing more expressive representations. For instance, R. Chua et al. (2024) proposed a Temporal Difference (TD) mechanism and employed a stop-gradient operator to avoid collapsed states during learning in complex pixel-based scenarios. Filos et al. (2021) introduced an inverse temporal difference from offline data and learnt the shared feature function from demonstration. Ollivier (2025) showed that different families of features that describe rewards also share a common structure, which can be expressed in terms of Laplacian eigenfunctions; however, this structure holds only in deterministic environments, and no practical method currently exists to learn such eigenfunctions. Similarly, Park et al. (2024) proposed learning feature functions through Hilbert representations, abstracting states while preserving global relationships. Their method re-parameterises relationships by treating one state as a goal, capturing temporal dependencies between states and the goal.

In this thesis, the design and learning of the feature function itself are considered outside the scope of the proposed contributions.

2.4.3 Address Real-world Application Challenges

Finally, SF and GPI together have become a widely used framework, with applications extending beyond theoretical studies into real-world domains. For example, Szoke, Aradi, Becsi, and Gaspar (2022) and Szoke, Aradi, and Tettamanti (2022) studied SFs in the context of autonomous driving, whereas Szoke et al. (2023) demonstrated their use in traffic light control, while Xu et al. (2020) explored applications in natural language processing. In addition, J. Zhang et al. (2017) implemented SFs for navigation problems. Similarly, J. Zhang, Ren, et al. (2024) studied planning and control in multiple unmanned surface vehicles (USVs), demonstrating the capacity to coordinate multiple USVs in ocean and water conditions.

Beyond control and robotic problems, SF has been used to study human learning. Momennejad et al. (2017) studied SF to model how humans learn, whereas De Cothi and Barry (2020) utilised SF to model how the hippocampus in the human brain plans over spatio-temporal spaces.

Despite their broad adoption across diverse research fields and practical domains, existing work with SF and GPI has not explored their potential to support human learning, nor demonstrated how TRL can make RL feasible in real-world educational contexts, which motivates .

2.5 Successor Features Constraints and Limitations

SFs and GPI form a powerful framework that has achieved broad adoption and delivered impressive results across a wide range of fields. Nevertheless, important limitations remain. Among these is the assumption of fixed transition dynamics, which poses a major barrier to real-world applicability. Addressing this challenge requires methods that neither assume prior knowledge of the relationship between transition dynamics nor rely on their direct approximation.

In addition, while GPI has become the *de facto* mechanism for composing SFs, its reliance on the sole composition strategy highlights the need to explore alternative mechanisms that could offer greater flexibility and scalability.

To the best of our knowledge, these constraints remain unresolved, representing critical open problems. Tackling them is essential for advancing the literature on SFs and GPI and for making these frameworks more suitable for complex, real-world tasks.

ADAPTIVE SUCCESSOR FEATURES VIA TRANSITION DYNAMICS TRANSFORMATION

Facilidad, sin embargo, por que lo que el hombre teme por encima de todo no es la muerte y el sufrimiento, en los que tantas veces se refugia, sino la angustia que genera la necesidad de ponerse en cuestión, de combinar el entusiasmo y la crítica, el amor y el respeto.

Estanislao Zuleta, Elogio de la dificultad.

This chapter aims to address **RQ-1** and accomplish the related research objectives **RO-1** and **RO-2**. To address these objectives, we first tackle **RO-1** by introducing a definition of Successor Features (SFs) grounded in a shared latent transition dynamics. Specifically, we assume the existence of a latent transition dynamics to which the dynamics of any environment can be transformed. Next, for **RO-2**, we describe a mechanism to learn the optimal SFs and their derived policies that solve each task within this latent transition dynamics. Finally, in **RO-3**, we propose an ensemble model to enable the transfer of these learnt representations to target tasks.

3.1 Background and Motivation

Reinforcement Learning (RL) has demonstrated remarkable performance in various complex games and in robotic control tasks (Barreto et al., 2018; Mnih et al., 2015; J. Zhang et al., 2017). However, the classical RL has low sample efficiency and the solutions are task-specific. An example of this is an agent controlling a robot arm. The agent needs millions of interactions with the environment to solve a simple task, such as reaching objects. Interestingly, any modification to the robot or a change in the task to perform requires a new start of the learning process anew (J. Zhang et al., 2017).

Intuitively, one can expect the agent to reuse previously learnt knowledge in the new conditions. Transfer Reinforcement Learning (TRL) aims to increase sample efficiency and reduce task specificity (Zhu et al., 2023a). Typically, TRL involves two sets of tasks: a set of *source tasks* to learn *what* knowledge to reuse and a set of *target tasks* applying *how* to reuse that knowledge. To answer these questions, many works have proposed reusing knowledge representations (Schaul et al., 2015). In this vein, Barreto et al. (2017) proposed a transferable representation that generalises the successor representation proposed by Dayan (1993) and makes it possible to evaluate a policy across tasks.

Successor features have shown impressive results in transferring knowledge between tasks that differ in reward functions but share transition probability distributions or *dynamics* (Barreto et al., 2017, 2018; Barreto et al., 2020; J. Zhang et al., 2017). This condition is unusual in real-world problems. For instance, if an agent controls robotic arms with different physical specifications (e.g., degrees of freedom or arm dimensions), this implies handling diverse dynamics.

To manage diverse dynamics in successor features, J. Zhang et al. (2017) proposed a linear relationship between the source and target feature functions. Under the assumption that one knows that the source and target dynamics are similar before-hand. Recent work proposed using Gaussian processes to represent target successor features as noised source successor features (Abdolshah et al., 2021), but the implementation of the Gaussian process is complicated, and its cubic complexity makes it difficult to scale in high-dimensional environments (Goumiri et al., 2020).

To overcome complexity and the assumption of similarities in the tasks, we propose a novel mechanism based on *successor features*. Our method assumes the existence of a latent base dynamics and an invertible and differentiable function that can map any dynamics to these base dynamics. In other words, we assume that a given dynamics is a transformation of the base dynamics. We can use the transformed dynamics to build a new

successor feature, *transformed successor feature*. This extends the ability of the original successor feature to handle diverse dynamics. To reuse the successor feature in the target tasks, we propose an ensemble method (Ganaie et al., 2022; van der Laan et al., 2007).

In the *Reacher* environment, our approach shows that it is capable of outperforming the original successor feature sharing dynamics. In a diverse dynamics setting, our approach achieves at least the same performance as the baseline. In *Cartpole* with different pole lengths and *Hopper* with different reward dynamics, our approach shows that it is capable of transferring previously acquired knowledge, outperforming the baseline in a diverse dynamics setting.

Finally, our approach makes the following contributions:

- i) A novel mechanism to predict state features within a base dynamics. It involves projecting the TSF into the base dynamics using probability density transformation techniques, enabling the capture of both similar and diverse dynamics.
- ii) A simple but powerful approach using ensemble models to transfer between source and target tasks. This is an expressive and straightforward solution for the successor features to manage diverse dynamics.

3.2 Preliminaries and Problem Formulation

RL problems are commonly modelled by a Markov Decision Process (MDP). An MDP is defined by the tuple $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{M}$ is the MDP; $\mathcal{S}$ and $\mathcal{A}$ are the states and action spaces; $p(\cdot|s, a)$ is the transition probability distribution (dynamics) that defines the distribution of the next state given a state, s , taking action, a ; r is a reward function that can be defined by $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$; and $\gamma \in [0, 1)$ corresponds to the discount factor on future rewards. The main objective of RL is to discover the optimal policy $\pi^* : \mathcal{S} \mapsto \mathcal{A}$, which returns an action given a state to maximise the expected accumulated return $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

There is a category of RL algorithms which finds π^* from the state-action value function (also known as the Q function), which is defined as

$$Q^\pi(s, a) = \mathbb{E}_{\pi, s_{t+1} \sim p} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a \right]. \quad (3.1)$$

Many dynamic programming-based RL algorithms alternate between two processes: *policy evaluation* to compute Q^π and *policy improvement*, which obtains policies taking

greedy actions upon the Q function, $\pi'(s) \in \arg \max_a Q^\pi(s, a)$. Constant alternation of both processes will converge in π^* . Therefore, obtaining π^* for one task requires intensive interactions. In this chapter, we focus on scenarios where knowledge can be reused in new tasks without restarting the learning process.

TRL is a compelling set of techniques to reuse previously learnt knowledge and commonly defines two sets of tasks. A set of *source tasks*, $\mathcal{M}^s \equiv \{M_i\}_{i=1}^n$, is used by the agent to extract knowledge, and a set of target tasks, to which knowledge is transferred, is defined by $\mathcal{M}^\tau \equiv \{M_j\}_{j=1}^m$. That is, the aim of TRL is to transfer the knowledge learnt in $\mathcal{M}^s$ to $\mathcal{M}^\tau$.

Successor feature is a solution to achieve TRL. In the successor feature (Barreto et al., 2017) approach, the reward function is decomposed into a simple product of a feature function and a weight, as $r_i(s_t, a_t, s_{t+1}) = \phi(s_t, a_t, s_{t+1})^\top \mathbf{w}_i$, where $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ denotes the feature function of dimension d and $\mathbf{w}_i$ is the i -th task-specific vector.

The basic idea is that the feature function can be shared by all tasks, which facilitates knowledge sharing among them. This new representation of the reward function allows us to rewrite the Q -value function as

$$\begin{aligned} Q_i^\pi(s, a) &= \mathbb{E}_{\substack{a_{t+1} \sim \pi \\ s_{t+1} \sim p}} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a \right]^\top \mathbf{w}_i \\ &= \psi^\pi(s, a)^\top \mathbf{w}_i, \end{aligned} \quad (3.2)$$

where ψ^π is named *successor feature* and encapsulates the dynamic of the environment. This factorisation of the rewards and Q function grants the agent the possibility of reusing the successor features in the target tasks. Furthermore, this enables fast computation of Q values in tasks with the same dynamics $Q_j^\pi(s, a) = \psi^\pi(s, a)^\top \mathbf{w}_j$ and only requires fitting the preferences over the features for the task-specific $\mathbf{w}_j$. Nevertheless, Barreto et al. (2017) approach is limited to scenarios with fixed dynamics and tasks that differ in reward functions, which is unusual in RL settings.

3.3 Method

Our idea lies in the intuition that the successor features can be extracted from a base probability distribution. Instead of assuming a fixed dynamic for all tasks, we assume the existence of a latent base transition dynamic and that all task-specific transition dynamics are just different transformations of this base dynamic, as shown in Fig. 3.1. Such a transformation could not only support diverse dynamics across different tasks but also establish

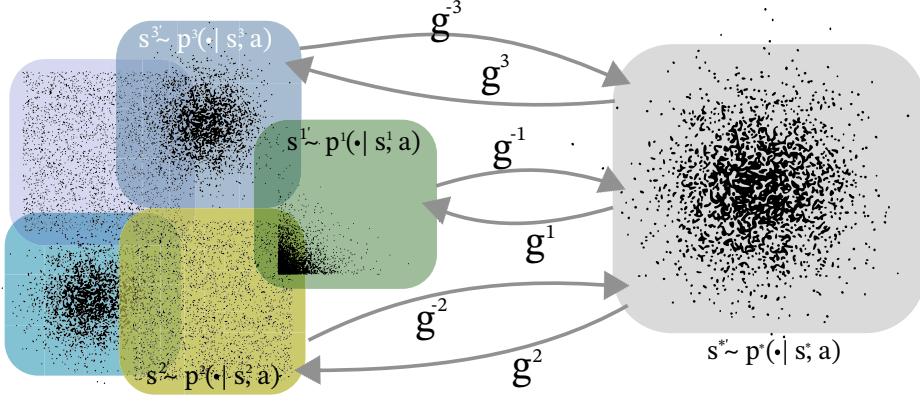


Figure 3.1: Transformations between base distribution probability, $p^*(\cdot|\cdot, \cdot)$, and source task distribution probabilities for source tasks, i.e., $p^1, p^2, p^3, \dots$.

links between them to facilitate knowledge sharing. Building on this idea, we derive a new definition of successor features based on the latent transition dynamics, which we term transformed successor features. Finally, we explain how these features can be used to enable transfer reinforcement learning.

3.3.1 Transformed Successor Features

We first define a base dynamic shared across all tasks as the state transition probability distribution at time step t ,

$$p^*(s_{t+1}|s_t, a_t), \quad (3.3)$$

where $s_{t+1}, s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. With this base dynamic, we assume that the transition dynamic of each task $p^i(s_{t+1}|s_t, a_t)$ is a transformation such that,

$$p^i = \mathcal{T}^i(p^*), \quad (3.4)$$

where $\mathcal{T}^i$ denotes a transformation. More specifically, we assume that there exists an invertible and differentiable function g^i mapping the state sampled from the base dynamic and the task-specific dynamic as,

$$s_t^* = g^{-i}(s_t^i), \quad s_{t+1}^* = g^{-i}(s_{t+1}^i), \quad (3.5)$$

where g^{-i} is the inverse function of g . Note that throughout this chapter we use a linear form, $g(x) = b \cdot x + c$, and leave the extension to more complex transformations for future work, such as normalising flow-based transformations (Kobyzev et al., 2021; Papamakarios et al., 2021; Rezende & Mohamed, 2015). Because g is linear, the task distributions shown in Fig. 3.1 maintain the same functional shape as the base distribution;

the transformation merely shifts and scales the transition dynamics without warping the underlying probability density.

The requirement that g^{-i} be invertible and differentiable constrains the class of transition dynamics that can be represented. Under this modelling assumption, the transition dynamics of all tasks are described as transformations of a common underlying probability density. This is viable in many control problems where transition dynamics depend on parametric values that define different behaviours, such as physical dimensions, weight, or environmental conditions (Ghugare & Eysenbach, 2025). In such cases, differences across tasks can be interpreted as structured variations of a shared dynamical process rather than entirely unrelated transition dynamics.

Recent work has empirically demonstrated the feasibility of learning optimal policies using normalising flow-based transformations (Ghugare & Eysenbach, 2025). Moreover, normalising flow models continue to evolve, either by relaxing strict bijectivity constraints while retaining the advantages of invertible mappings (Cornish et al., 2020), or by improving the expressivity of invertible transformations (Draxler et al., 2024). These advances suggest that, within the class of smooth invertible mappings, the practical limitations imposed by invertibility are progressively being reduced, making the proposed approximation increasingly flexible in structured dynamical settings.

Following the Jacobian transformation, we derive a *task-specific dynamic* p^i as a transformation of p^* as,

$$\begin{aligned}
 p^i(s_{t+1}^i, s_t^i, a_t) &= p^*(s_{t+1}^*, s_t^*, a_t) \left| \det \begin{pmatrix} \frac{\partial s_t^*}{\partial s_t^i} & \frac{\partial s_{t+1}^*}{\partial s_t^i} \\ \frac{\partial s_t^*}{\partial s_{t+1}^i} & \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \end{pmatrix} \right| \\
 &= p^*(s_{t+1}^*, s_t^*, a_t) \left| \det \begin{pmatrix} \frac{\partial g^{-i}(s_t^i)}{\partial s_t^i} & \frac{\partial g^{-i}(s_{t+1}^i)}{\partial s_t^i} \\ \frac{\partial g^{-i}(s_t^i)}{\partial s_{t+1}^i} & \frac{\partial g^{-i}(s_{t+1}^i)}{\partial s_{t+1}^i} \end{pmatrix} \right| \\
 &= p^*(s_{t+1}^*, s_t^*, a_t) \left| \det \begin{pmatrix} \frac{\partial g^{-i}(s_t^i)}{\partial s_t^i} & 0 \\ 0 & \frac{\partial g^{-i}(s_{t+1}^i)}{\partial s_{t+1}^i} \end{pmatrix} \right|,
 \end{aligned} \tag{3.6}$$

and the conditional probability can be derived as

$$\begin{aligned}
p^i(s_{t+1}^i \mid s_t^i, a_t) &= \frac{p^i(s_{t+1}^i, s_t^i, a_t)}{p^i(s_t^i, a_t)} \\
&= \frac{p^*(s_{t+1}^*, s_t^*, a_t) \left| \det \left(\begin{bmatrix} \frac{\partial s_t^*}{\partial s_t^i} & 0 \\ 0 & \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \end{bmatrix} \right) \right|}{p^*(s_t^*, a_t) \left| \frac{\partial s_t^*}{\partial s_t^i} \right|} \\
&= \frac{p^*(s_{t+1}^*, s_t^*, a_t) \left| \frac{\partial s_t^*}{\partial s_t^i} \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right|}{p^*(s_t^*, a_t) \left| \frac{\partial s_t^*}{\partial s_t^i} \right|} \\
&= \frac{p^*(s_{t+1}^*, s_t^*, a_t)}{p^*(s_t^*, a_t)} \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right| \\
&= p^*(s_{t+1}^* \mid s_t^*, a_t) \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right| \\
&= p^*(g^{-i}(s_{t+1}^i) \mid g^{-i}(s_t^i), a_t) \left| \frac{\partial g^{-i}(s_{t+1}^i)}{\partial s_{t+1}^i} \right|.
\end{aligned} \tag{3.7}$$

Now, having a change of variable as

$$\int_a f(x) dx = \int_\alpha f(g(u)) \frac{\partial x}{\partial u} du, \text{ where } \alpha = g(a) \text{ and } x = g(u),$$

we can now rewrite the integral with respect to s_{t+1}^i to s_{t+1}^* as

$$\begin{aligned}
\int_{s_{t+1}^i} \phi(s_t, a_t, s_{t+1}^i) ds_{t+1}^i &= \int_{s_{t+1}^*} \phi(s_t, a_t, g^i(s_{t+1}^*)) \left| \frac{\partial g^i(s_{t+1}^*)}{\partial s_{t+1}^*} \right| ds_{t+1}^* \\
&= \int_{s_{t+1}^*} \phi(s_t, a_t, g^i(s_{t+1}^*)) \left| \frac{\partial s_{t+1}^i}{\partial s_{t+1}^*} \right| ds_{t+1}^*.
\end{aligned} \tag{3.8}$$

We define the Q function following the policy π with continuous state spaces and dis-

crete timesteps as follows,

$$\begin{aligned}
\psi^{\pi_i}(s, a) &= \mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} \left[\sum_{k=t}^{\infty} \gamma^{k-t} \phi(s_k, a_k, s_{k+1}^i) \mid s_t = s, a_t = a \right] \\
&= \mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} [\phi(s_t, a_t, s_{t+1}^i) \mid s_t = s, a_t = a] \\
&\quad + \mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} \left[\sum_{k=t+1}^{\infty} \gamma^{k-t} \phi(s_k^i, a_k, s_{k+1}^i) \mid s_t = s, a_t = a \right] \\
&= \underbrace{\mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} [\phi(s_t, a_t, s_{t+1}^i) \mid s_t = s, a_t = a]}_{\text{Expression at } t} \\
&\quad + \underbrace{\mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} [\phi(s_{t+1}^i, a_{t+1}, s_{t+2}^i) \mid s_t = s, a_t = a]}_{\text{Expression at } t+1} \\
&\quad + \mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} \left[\sum_{k=t+2}^{\infty} \gamma^{k-t} \phi(s_k^i, a_k, s_{k+1}^i) \mid s_t = s, a_t = a \right].
\end{aligned} \tag{3.9}$$

We first expand the *Expression at t*.

$$\begin{aligned}
\mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} [\phi(s_t, a_t, s_{t+1}^i) \mid s_t = s, a_t = a] &= \\
&\mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^i} \phi(s_t, a_t, s_{t+1}^i) p^i(s_{t+1}^i \mid s_t, a_t) ds_{t+1}^i \right] \\
&= \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^i} \phi(s_t, a_t, s_{t+1}^i) \underbrace{p^*(s_{t+1}^* \mid s_t, a_t) \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right|}_{\text{Jacobian transformation Eq. (3.6)}} ds_{t+1}^i \right] \\
&= \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^*} \phi(s_t, a_t, g^i(s_{t+1}^*)) \underbrace{\left| \frac{\partial s_{t+1}^i}{\partial s_{t+1}^*} \right|}_{\text{Change of variable Eq. (3.8)}} p^*(s_{t+1}^* \mid s_t, a_t) \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right| \underbrace{ds_{t+1}^*}_{\text{Change of variable Eq. (3.8)}} \right] \\
&= \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^*} \phi(s_t, a_t, g^i(s_{t+1}^*)) \cancel{\left| \frac{\partial s_{t+1}^i}{\partial s_{t+1}^*} \right|} p^*(s_{t+1}^* \mid s_t, a_t) \cancel{\left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right|} ds_{t+1}^* \right] \\
&= \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^*} \phi(s_t, a_t, g^i(s_{t+1}^*)) p^*(s_{t+1}^* \mid s_t, a_t) ds_{t+1}^* \right] \\
&= \mathbb{E}_{\pi_i, s_{t+1}^* \sim p^*} [\phi(s_t, a_t, g^i(s_{t+1}^*))].
\end{aligned}$$

Now the *Expression at t + 1*.

$$\begin{aligned}
& \mathbb{E}_{\pi_i, s_{t+1}^i \sim p^i} [\phi(s_{t+1}^i, a_{t+1}, s_{t+2}^i) \mid s_t = s, a_t = a] = \\
& \mathbb{E}_{\pi_i} \left[\int_{s_{t+2}^i} \int_{s_{t+1}^i} \phi(s_{t+1}^i, a_{t+1}, s_{t+2}^i) p^i(s_{t+1}^i \mid s_t, a_t) p^i(s_{t+2}^i \mid s_{t+1}^i, a_{t+1}) ds_{t+1}^i ds_{t+2}^i \right] \\
& = \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^i} \int_{s_{t+2}^i} \phi(s_{t+1}^i, a_{t+1}, s_{t+2}^i) \right. \\
& \quad \times \underbrace{p^*(s_{t+2}^* \mid s_{t+1}^*, a_{t+1}) \left| \frac{\partial s_{t+2}^*}{\partial s_{t+2}^i} \right| p^*(s_{t+1}^* \mid s_t, a_t) \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right|}_{\text{Jacobian transformation (3.6)}} ds_{t+1}^i ds_{t+2}^i \left. \right] \\
& = \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^*} \int_{s_{t+2}^*} \underbrace{\phi(g^i(s_{t+1}^*), a_{t+1}, g^i(s_{t+2}^*)) \left| \frac{\partial s_{t+1}^i}{\partial s_{t+1}^*} \frac{\partial s_{t+2}^i}{\partial s_{t+2}^*} \right|}_{\text{Change of variable (3.8)}} \right. \\
& \quad \times p^*(s_{t+2}^* \mid s_{t+1}^*, a_{t+1}) \left| \frac{\partial s_{t+2}^*}{\partial s_{t+2}^i} \right| p^*(s_{t+1}^* \mid s_t, a_t) \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right| \underbrace{ds_{t+1}^* ds_{t+2}^*}_{\text{Jacobian transformation (3.6)}} \left. \right] \\
& = \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^*} \int_{s_{t+2}^*} \phi(g^i(s_{t+1}^*), a_{t+1}, g^i(s_{t+2}^*)) \left| \frac{\partial s_{t+1}^i}{\partial s_{t+1}^*} \frac{\partial s_{t+2}^i}{\partial s_{t+2}^*} \right| \right. \\
& \quad \times p^*(s_{t+2}^* \mid s_{t+1}^*, a_{t+1}) \left| \frac{\partial s_{t+2}^*}{\partial s_{t+2}^i} \right| p^*(s_{t+1}^* \mid s_t, a_t) \left| \frac{\partial s_{t+1}^*}{\partial s_{t+1}^i} \right| ds_{t+1}^* ds_{t+2}^* \left. \right] \\
& = \mathbb{E}_{\pi_i} \left[\int_{s_{t+1}^*} \int_{s_{t+2}^*} \phi(g^i(s_{t+1}^*), a_{t+1}, g^i(s_{t+2}^*)) p^*(s_{t+2}^* \mid s_{t+1}^*, a_{t+1}) p^*(s_{t+1}^* \mid s_t, a_t) ds_{t+1}^* ds_{t+2}^* \right] \\
& = \mathbb{E}_{\pi_i, s^* \sim p^*} \left[\phi(g^i(s_{t+1}^*), a_{t+1}, g^i(s_{t+2}^*)) \right].
\end{aligned}$$

We can proceed with the next timesteps, applying the Jacobian transformation and the change of variable equation. For the change of variable equation, in further steps we must compute the Jacobian matrix to apply the change of variable.

Putting both equations together, we then have,

$$\begin{aligned}
\psi^{\pi_i}(s, a) &= \mathbb{E}_{\pi_i, s_{t+1}^* \sim p^*} \left[\phi(s_t, a_t, g^i(s_{t+1}^*)) \right] \\
&+ \mathbb{E}_{\pi_i, s^* \sim p^*} \left[\sum_{k=t+1} \gamma^{k-t} \phi(g^i(s_k^*), a_k, g^i(s_{k+1}^*)) \right].
\end{aligned} \tag{3.10}$$

The above derived Q -value function bears an expectation of future state-action features with respect to a base dynamics. That is, these computed Q values can be viewed as gener-

alised values that take advantage of the particularities of the base dynamics and therefore hope to capture different dynamics. Another merit is that it transforms Q functions for all tasks into expectations under the same base dynamic rather than task-specific dynamics, making subsequent knowledge sharing possible. To achieve knowledge sharing, we define

$$\phi(g^i(s_t^*), a_t, g^i(s_{t+1}^*)) = \phi(s_t^i, a_t, s_{t+1}^i) \odot [h(g^{-i}(s_{t+1}^i))], \quad (3.11)$$

where h is an affine function shared by all tasks. This decouples the base feature function ϕ from the task-specific transformation g^i . h enhances the flexible relationship between the feature function and the dynamics transformation. Similar to the original successor features, our derived Q function expresses the TSF,

$$\hat{\psi}_i^\pi(s^i, a) = \mathbb{E}_{\pi, P^*} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t^i, a_t, s_{t+1}^i) \odot [h(g^{-i}(s_{t+1}^i))] \right], \quad (3.12)$$

where i denotes a task. With respect to this TSF, we make the following remarks:

Remark 3.1. *This new representation retains the same capabilities as the original successor feature and decouples the dynamics from the rewards.*

Remark 3.2. *The transition dynamics of TSF is the base probability distribution, thus TSF predict the future features visitation in the latent base dynamics.*

Remark 3.3. *There is a shared base feature function ϕ across tasks that facilitates knowledge sharing. Furthermore, the task-specific transformations also provides a way to express the similarity or relationship between tasks.*

Remark 3.4. *The term $h(g^{-i}(s_{t+1}^i))$ adds the particularities of the current state of each task to the shared base feature function. We can recover the shared base features when the outputs of h are 1. Different values suggest more dissimilarities among tasks. Therefore, this definition enables the handling of diverse dynamics.*

3.3.2 Transfer via Transformed Successor Features

As a TRL setting, our approach consists of two steps. The knowledge extraction in source tasks, *source Transformed Successor Feature (TSF) learning*, and the transference to target tasks, *target TSF learning*.

3.3.3 Source Transformed Successor Feature Learning

To extract knowledge, the agent minimises the total loss function, $L = L_{\hat{\psi}_i^\pi} + \beta L_w$, where β is an hyperparameter, $L_{\hat{\psi}_i^\pi}$ is the Temporal Difference (TD) error to fit source TSF defined by,

$$L_{\hat{\psi}_i^\pi} = \|\phi_i(s_t^i, a_t, s_{t+1}^i) + \gamma \hat{\psi}_i^\pi(s_{t+1}^i, a_{t+1}) - \hat{\psi}_i^\pi(s_t^i, a_t)\|, \quad (3.13)$$

and reward fitting using

$$L_w = \|R - \phi_i(s_t^i, a_t, s_{t+1}^i)^\top \mathbf{w}_i\|, \quad (3.14)$$

where R is the reward obtained. To do so, we use a combination of the total loss

$$L = L_{\hat{\psi}_i^\pi} + \beta L_w. \quad (3.15)$$

Furthermore, to minimise total loss, the agent learns all components required for the source TSF, that is, the invertible function g^i , the shared affine function h , and the reward mapper $\mathbf{w}_i$.

It is noteworthy that this formulation differs from the original successor feature objective proposed by Barreto et al. (2018). In Barreto et al. (2018), the loss optimises only the temporal-difference error over a fixed feature representation, assuming shared transition dynamics across tasks. In contrast, our formulation jointly learns a task-specific invertible transformation g^i and a shared affine transformation h , which map task-dependent dynamics into a shared latent representation. Consequently, the temporal-difference loss is applied over transformed features ϕ_i rather than fixed environment features, enabling transfer across environments with differing transition dynamics.

To clarify the complete building and learning process for the source TSF we propose Algorithm 1. The Algorithm 1 describes the process for extracting transformed successor features from the source tasks. This algorithm has the feature function as an input. The algorithm uses ε -greedy for exploration (Lines 5 – 6). Compute the transformed feature function, line 14, and fit the models that minimise accumulated loss, line 17.

3.3.4 Target Transformed Successor Feature Learning

To construct the target TSF to be reused in the target task set, We compute the target TSF as a weighted source TSF using the super-learner ensemble model (van der Laan et al., 2007). The weights express the relevance of each source TSF in the new tasks, and they generate the following definition.

Algorithm 1 Transformed Successor Feature Algorithm.

Require:

$M_{i \in \{1, \dots, n\}}, \mathcal{A}, \mathcal{S}$ ▷ Source tasks, Action and state spaces
 $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ ▷ Feature function with d features dimension
 $iters, timesteps, eps$ ▷ Learning cycles over all tasks, timesteps on each task, Max length for episodes
 $\varepsilon, \gamma, \alpha_w, \alpha_\psi, \alpha_\phi, \alpha_g, \alpha_h$ ▷ Exploration rate, Discount factor, Learning rates
 $n_batch, freq, \beta$ ▷ Batch number used in replay buffer, Update models frequency, Loss coefficient

Ensure:

$h : \mathbb{R}^d \rightarrow \mathbb{R}^d$ ▷ Affine function, with d features dimension
 $\Psi^s = \{\psi_i | i \in \{1, \dots, n\}\}$ ▷ Transformed successor features for n source tasks
 $W^s = \{w_i | i \in \{1, \dots, n\}\}$ ▷ Reward mappers for n source tasks
 $D^s = \{D_i | i \in \{1, \dots, n\}\}$ ▷ Experience buffers per task
 $G^s = \{g^i | i \in \{1, \dots, n\}\}$ ▷ Transformation function for n source tasks
1: Initialise Ψ^s, W^s, D^s, G^s, h
2: **for all** $iters$ and $M_{i \in \{1, \dots, n\}}$ **do** ▷ For each source task
3: Sample the initial State randomly $s \in \mathcal{S}$
4: **for all** $timesteps$ **do**
5: **if** $Bernulli(\varepsilon) = 1$ **then** $a \leftarrow Uniform(\mathcal{A})$ ▷ ϵ -greedy
6: **else** $a \leftarrow \arg \max_{b \in \mathcal{A}} \psi_i(s, b)^\top w_i$
7: **end if**
8: Take action a in M_i and observe s' , and r .
9: Append obtained transition (s, a, s', r) into buffer D_i
10: **if** $freq$ **then**
11: Retrieve n_batch transitions from buffer D_i
12: $(s, a, s', r) \leftarrow transitions$
13: $a' \leftarrow \arg \max_{b \in \mathcal{A}} \psi_i(s', b)^\top w_i$
14: $\phi_i \leftarrow \phi(s, a, s') \odot (h(g^{-i}(s')))$ ▷ Eq. (3.11)
15: $L_{\psi_i} \leftarrow \|\phi_i + \gamma \psi_i(s', a') - \psi_i(s, a)\|$.
16: $L_{w_i} \leftarrow \|r - (\phi_i)^\top w_i\|$.
17: $Loss \leftarrow L_{\psi_i} + \beta L_{w_i}$
18: Update ψ_i, w_i, g^i, h using $Loss$ and learning rates
19: **end if**
20: $s \leftarrow s'$
21: **end for**
22: **end for**

Given weights $\Omega = \{\omega_{ji} | \{i\}_{i=1}^n, \{j\}_{j=1}^m\}$, a *target TSF*, $\hat{\psi}_j$, is computed as the weighted

learnt *source* TSF,

$$\begin{aligned}
\hat{\psi}_j(s^j, a) &= \sum_i \omega_{ji} \hat{\psi}_i^\pi(s^j, a) \\
&= \sum_i \omega_{ji} \mathbb{E}_{\pi, p^*} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t^j, a_t, s_{t+1}^j) \odot \left(h \left(g^{-i}(s_{t+1}^j) \right) \right) \right] \\
&= \mathbb{E}_{\pi, p^*} \left[\sum_{t=0}^{\infty} \gamma^t \underbrace{\phi(s_t^j, a_t, s_{t+1}^j) \odot \left(h \left(\sum_i \omega_{ji} g^{-i}(s_{t+1}^j) \right) \right)}_{\text{target transformed feature function}} \right].
\end{aligned} \tag{3.16}$$

Moreover, we expand the source transformed successor feature and acquire a new target transformed feature function.

$$\phi_j(s_t^j, a_t, s_{t+1}^j) = \phi(s_t^j, a_t, s_{t+1}^j) \odot \left(h \left(\sum_i \omega_{ji} g^{-i}(s_{t+1}^j) \right) \right). \tag{3.17}$$

To learn the weights Ω , we use the same total loss with the new definitions of the target transformed feature function and the target transformed successor feature. Thus, our proposed algorithm only needs to fit the reward mappers and Ω for the target task set that can be learnt simultaneously. The algorithm for the target transformed successor feature is shown in Algorithm 2. This algorithm uses ensemble models to derive a successor feature useful for transferring knowledge to target tasks. In evaluating target tasks, our agents also use the ε -greedy algorithm for exploration/exploitation, Lines 5–7. The target feature function is computed in Line 10, and the accumulated loss is computed in Line 13.

3.4 Experiments and Analysis

3.4.1 Experiment Setup

In this section, we explain the design of the experiment, the environment set up, the metrics chosen to evaluate performance, and the baseline for comparison purposes.

Metrics To analyse our agent and the baseline agent, we consider the work of Zhu et al. (2023a). Who describes an *Asymptotic Performance* metric to assess TRL settings. *Asymptotic Performance*, one of the most common metrics, is defined as the agent’s ultimate performance (returns). In TRL, the use of this metric is for comparison purposes in assessing the impact of knowledge transfer between tasks. Specifically, the metric selected for this chapter is *mastery* which measures how well an agent performs in the target tasks given a source task set.

Algorithm 2 Target Transformed Successor Feature Algorithm.

Require:

$\psi_{i \in \{1, \dots, n\}}, g^{i \in \{1, \dots, n\}}, M_{j \in \{1, \dots, m\}}$ $\triangleright$ Successor features, Transformation functions for n source tasks and target tasks
 $iters, timesteps, \mathcal{A}, \mathcal{S}$ $\triangleright$ Learning cycles over all tasks, timesteps on each task, Action and State spaces
 $\epsilon, \gamma, \alpha_w, \alpha_\omega, \beta$ $\triangleright$ Exploration rate, Discount factor, Learning rates, Loss coefficient

Ensure:

$\Omega^\tau : \{\omega_{ji} | j \in \{1, \dots, m\}, i \in \{1, \dots, n\}\}$ $\triangleright$ omega values per target task
 $W^\tau : \{\mathbf{w}_j | j \in \{1, \dots, m\}\}$ $\triangleright$ Reward Mappers for target tasks

- 1: Initialise random values for Ω^τ, W^τ
- 2: **for all** $iters$ and $M_{j \in \{1, \dots, m\}}$ **do** $\triangleright$ For each target task
- 3: Sample the initial State randomly $s \in \mathcal{S}$
- 4: **for all** timesteps **do**
- 5: **if** $Bernulli(\epsilon) = 1$ **then** $a \leftarrow Uniform(\mathcal{A})$ $\triangleright$ ϵ -greedy
- 6: **else** $a \leftarrow \arg \max_{b \in \mathcal{A}} (\sum_i \omega_{ji} \psi_i(s, b))^\top \mathbf{w}_j$
- 7: **end if**
- 8: Take action a in M_j and obtain s' , and r
- 9: $a' \leftarrow \arg \max_{b \in \mathcal{A}} (\sum_i \omega_{ji} \psi_i(s', b))^\top \mathbf{w}_j$
- 10: $\phi_j \leftarrow \phi(s, a, s') \odot (h(\sum_i \omega_{ji} g^{-i}(s')))$
- 11: $L_{\omega_j} \leftarrow \|\phi_j + \gamma \sum_i \omega_{ji} \psi_i(s', a') - \sum_i \omega_{ji} \psi_i(s, a)\|$
- 12: $L_{\mathbf{w}_j} \leftarrow \|r - \phi_j^\top \mathbf{w}_j\|$
- 13: $Loss \leftarrow L_{\mathbf{w}_j} + \beta L_{\omega_j}$
- 14: Update ω_{ji} and $\mathbf{w}_j$ minimising $Loss$
- 15: $s \leftarrow s'$
- 16: **end for**
- 17: **end for**

Agents For the baseline agent, we implemented the successor feature based on Barreto et al. (2017), SFDQN. Additional information of the hyperparameters for the successor feature models can be found in Table 3.1.

For the TSFDQN agent, we first select the hyperparameters for the functions g and h . To build invertible functions g , we constructed a linear model and proceeded with a grid search for the most desirable dimension. We found that a linear transformation, $g : \mathbb{R}^{|s|} \rightarrow \mathbb{R}^{|g|}$, with $|s|$ as the dimension of the state space and $|g| = 100$ as the dimension performs decently in the Reacher environment.

To find a combination of hyperparameters for our approach, we fixed the values related to the successor feature to make fair comparisons and modify the dimension values for functions g and h (10, 50, 100), β (1, 3, 30, 100) for the combined loss and α (10^{-3} , 10^{-5} ,

Table 3.1: Hyperparameters used for SFDQN agent.

Setting	Value
γ (discount factor)	0.9
ϵ (exploration-exploitation) target tasks	0.1
ϵ (exploration-exploitation) source tasks	0.3
Max episode length Reacher	500
Max episode length Cartpole and Hopper	1000
α (learning rate) Successor Feature ψ	10^{-3}
α (learning rate) Reward mappers (w)	10^{-3}
Successor Features number of layers	2
Successor Features number of neurons per layer	256
Mini-batch size	32
Replay Buffer storage capacity	10^6
Discount Factor (γ)	0.9
Feature dimension, d , in Reacher environment	12
Feature dimension, d , in Cartpole environment	20
Feature dimension, d , in Hopper environment	50

g_function_dim	omegas_lr	betas	ridge_lambda	lasso_lambda	mean_Average_Reward/timesteps	mean_Accumulative_Reward/timesteps
10	0.001000	1	0	0	203.435340	198448.629163
10	0.000010	1	0	0	181.529200	180306.640661
50	0.001000	1	0	0	241.027867	236395.574666
50	0.000010	1	0	0	249.323944	245649.738251
100	0.001000	1	0	0	252.998077	252612.494407
100	0.000010	1	0	0	273.565987	267560.575703
100	1e-5	3	1e-3	0	315.912948	279492.330584
100	1e-5	3	0	1	261.298160	253492.204330
100	1e-5	30	0	0	326.070874	311318.214128
100	1e-5	100	0	0	362.302774	351924.046092

Figure 3.2: Grid search for g and h dimensions.

10^{-6}) for the learning rate for Ω , as well as regularisation coefficients such as the ridge and lasso for Ω .

We ran three experiments using each set-up and chose the best run. Our best setup was $\beta = 100$, g and h dimension 100, and $\alpha = 10^{-5}$. The results of the grid search is shown in Fig. 3.2.

The final list of hyperparameters used is shown in Table 3.2. For the code implementation, we extended a base *successor features* framework¹. We kept the project structure and configuration management but rewrote the rest to use the PyTorch library. Our code

¹<https://github.com/mike-gimelfarb/deep-successor-features-for-transfer>

Table 3.2: Hyperparameters used for TSFDQN agent.

Setting	Value
γ (discount factor)	0.9
ϵ (exploration-exploitation) target tasks	0.1
ϵ (exploration-exploitation) source tasks	0.3
Max episode length Reacher	500
Max episode length Cartpole and Hopper	1000
α (learning rate) Successor Feature ψ	10^{-3}
α (learning rate) Reward mappers ($\mathbf{w}$)	10^{-3}
α (learning rate) Affine function (h)	10^{-3}
α (learning rate) invertible function (g)	10^{-3}
α (learning rate) ω values	10^{-5}
Successor Features number of layers	2
Successor Features number of neurons per layer	256
g and h dimension	100
β value for combined losses	100
Mini-batch size	32
Replay Buffer storage capacity	10^6
Discount Factor (γ)	0.9
Feature dimension, d , in Reacher environment	12
Feature dimension, d , in Cartpole environment	20
Feature dimension, d , in Hopper environment	50

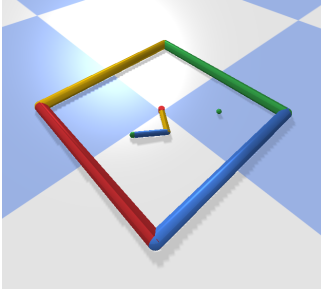
is made public².

Dynamics Settings We designed two settings to test our approach. A *shared dynamics setting* in which the dynamics remain fixed among the different tasks. In the second setting, *diverse dynamics setting*, we introduce specific settings to the environment to simulate diverse dynamics.

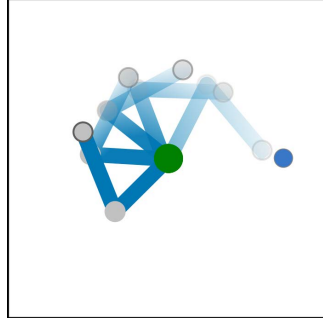
3.4.2 Experiment Results and Analysis

To present our results, we organised the discussion into two parts, namely the *shared dynamics setting* and the *diverse dynamics setting*.

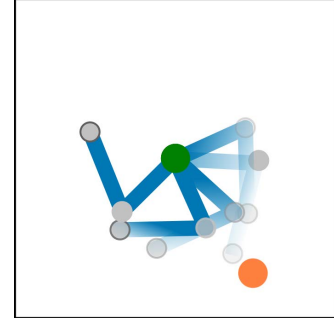
²<https://github.com/okgaces/deep-successor-features-for-transfer>



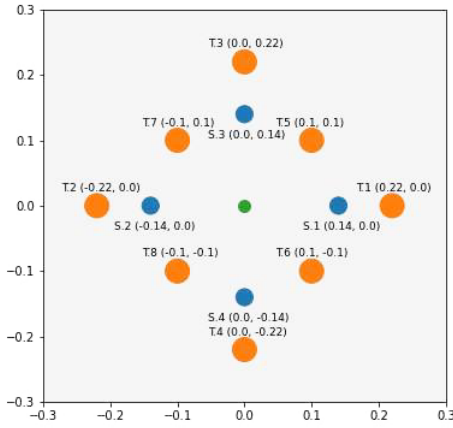
(a) Robot arm Reacher environment.



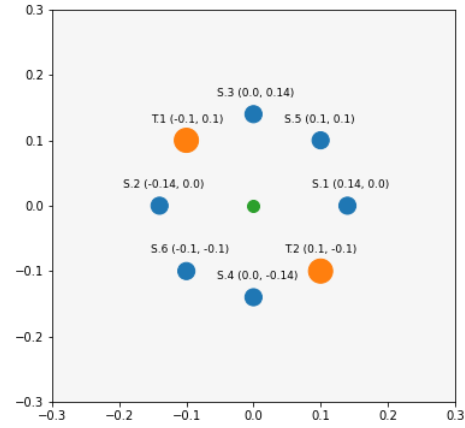
(b) Example of robot reacher solving a source task.



(c) Example of robot reacher solving a target task.



(d) Scenario 1: 4 source and 8 target tasks.



(e) Scenario 2: 6 source and 2 target tasks.

Figure 3.3: Robot Reacher setup. The blue dots are the coordinates for the source tasks. The orange dots are the coordinates for the target tasks. The green dot is the centre of the reacher arm.

3.4.2.1 Shared Dynamics Setting

In the first setting, *shared dynamics*, we used the Reacher environment. We took the initial configuration of the Reacher environment in Barreto et al. (2017).

Reacher Environment The Reacher environment is a 2D link reacher. The agent learns to manipulate robot arms with the objective of reaching different objects in fixed coordinates. We set up the environment using the metre, kilogramme, second (MKS) base unit system. The environment is defined as a square bounded by the four points: $(-0.3, -0.3)$, $(-0.3, 0.3)$, $(0.3, 0.3)$, and $(0.3, -0.3)$. An arm with two linkages, two hinges, and one end effector was placed inside this area, as shown in Fig. 3.3(a). The agent learns to reach dif-

Table 3.3: Scenario 1: Target coordinates for the Reacher environment and torque multiplier assigned to each task.

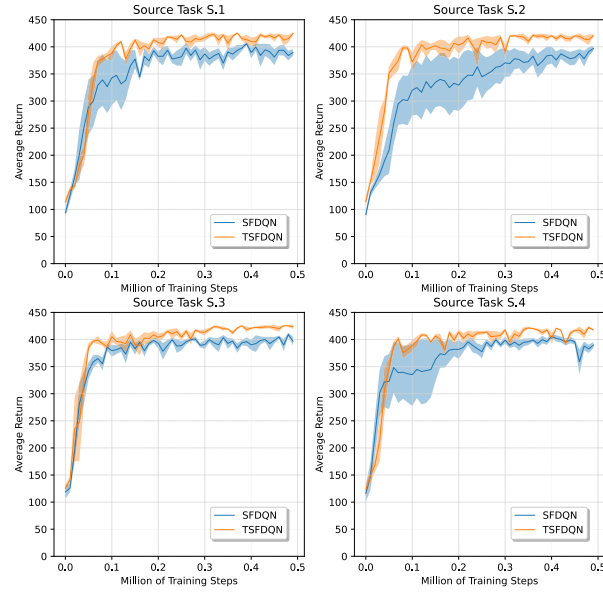
Type of Task	Task	Coordinates	Torque multiplier	
			<i>shared dynamics</i>	<i>diverse dynamics</i>
Source Tasks	S.1	(0.14, 0.0)	0.05	0.05
	S.2	(-0.14, 0.0)	0.05	0.05
	S.3	(0.0, 0.14)	0.05	0.1
	S.4	(0.0, -0.14)	0.05	1.0
Target Tasks	T.1	(0.22, 0.0)	0.05	0.0001
	T.2	(-0.22, 0.0)	0.05	0.001
	T.3	(0.0, 0.22)	0.05	0.01
	T.4	(0.0, -0.22)	0.05	0.1
	T.5	(0.1, 0.1)	0.05	0.5
	T.6	(0.1, -0.1)	0.05	1.0
	T.7	(-0.1, 0.1)	0.05	5.0
	T.8	(-0.1, -0.1)	0.05	10.0

ferent object coordinates by sequentially solving each source task. An example of how the robot completes a source task is shown in Fig. 3.3(b). Then, the agent is evaluated on unseen target coordinates. An example of the robot solving a specific target task is shown in Fig. 3.3(c).

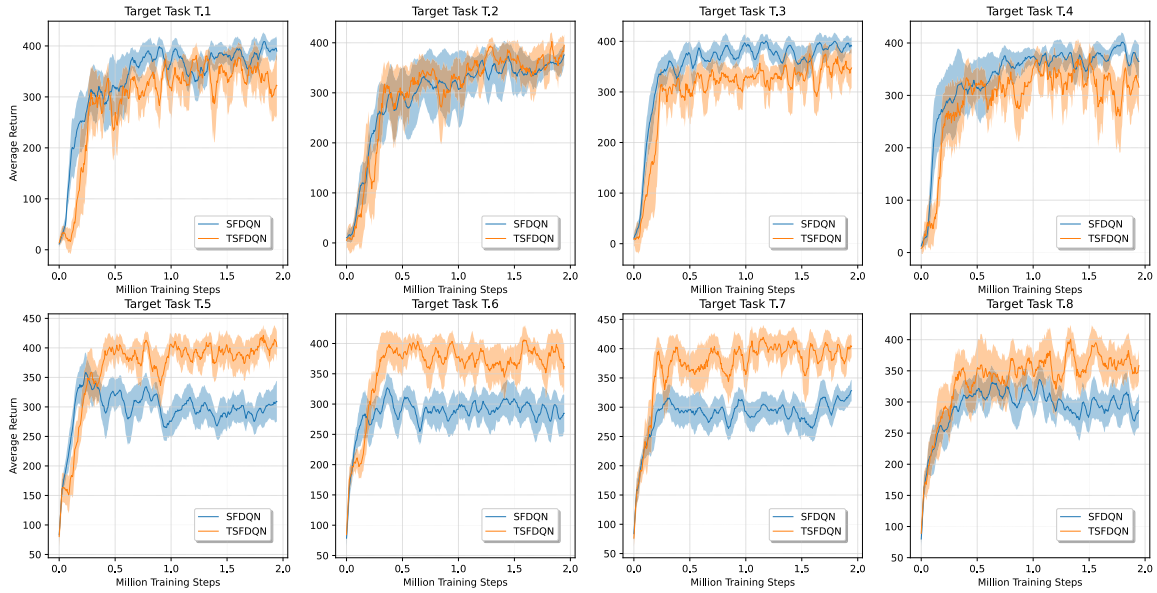
We designed two scenarios, each involving a different set of source and target tasks, and therefore different coordinates. The coordinates are shown in Fig. 3.3. The objectives can be located anywhere within the range $[-0.22, 0.22]$ for both x and y coordinates. Thus, we define the tasks as points of the form $\{(x, y) | -0.22 \leq x \leq 0.22, -0.22 \leq y \leq 0.22\}$. At each timestep, the agent receives a reward according to the 2-*norm* distance between the object coordinate position and the arm end effector; this means the agent receives a numerical reward within the range $[-1.5, 1]$.

Scenario 1 was designed based on Barreto et al. (2017). It consisted of 4 source tasks and 8 target tasks. The coordinates are shown in Fig. 3.3(d). To model the shared transition dynamics, we set the *torque multiplier* to 0.05 for all tasks, as shown in Table 3.3 under the *shared dynamics* column for the *torque multiplier*.

Each agent interacted with each source task in 10,000 timesteps, where every 1,000 timesteps the agent was moved to perform target tasks for 500 timesteps. To avoid catastrophic forgetting, agents revisit tasks a total of 50 times. That represents a total of 2×10^6 training steps. The same number of training steps was used for the SFDQN agent.



(a) Average return in source tasks.



(b) Average return in target tasks.

Figure 3.4: Scenario 1: Average return in source and target tasks by total training steps. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.

Table 3.4: Scenario 2: Target coordinates and torque multipliers for the Reacher Environment.

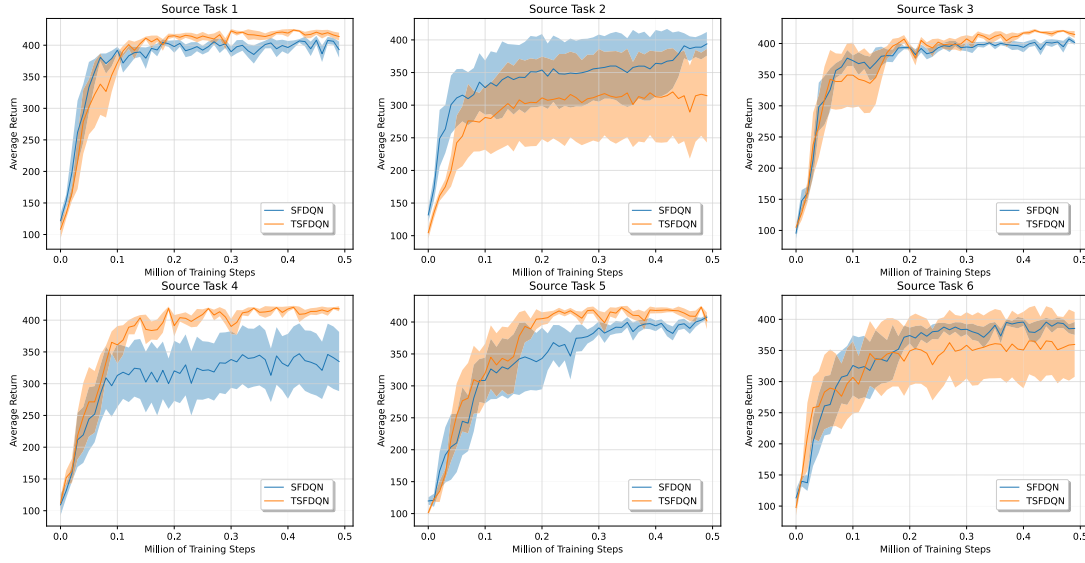
Type of Task	Task	Coordinates	Torque multiplier	
			<i>shared dynamics</i>	<i>diverse dynamics</i>
Source Tasks	S.1	(0.14, 0.0)	0.05	0.05
	S.2	(-0.14, 0.0)	0.05	0.5
	S.3	(0.0, 0.14)	0.05	5.0
	S.4	(0.0, -0.14)	0.05	0.005
	S.5	(0.1, 0.1)	0.05	50.0
	S.6	(-0.1, -0.1)	0.05	0.0005
Target Tasks	T.1	(-0.1, 0.1)	0.05	0.0001
	T.2	(0.1, -0.1)	0.05	10.0

To build the TSF, we hand-made the shared base feature function. In spite of RL settings lacking the privilege of this hand-made function, our definition consists of two components: the shared-based feature function and the transformation functions, g and h , Eq. (3.12). With this scenario, we can measure the effect of the transformation functions.

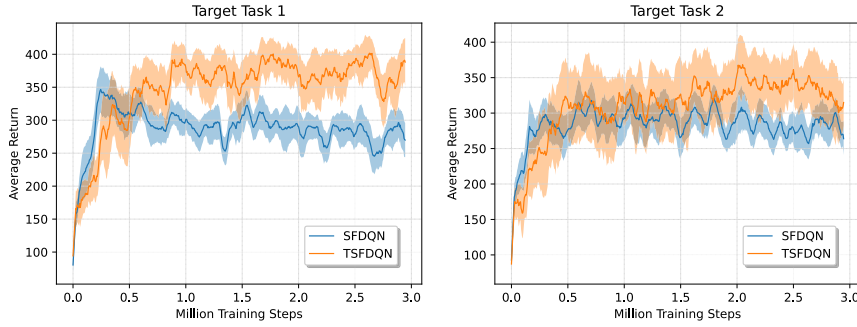
The output of the feature function depicts the desired tasks, that is, the dimensionality of the feature function depends on the number of tasks, that is, $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$, for d the number of tasks.

The asymptotic performance for the shared dynamics setting is shown in Fig. 3.4. In the source tasks, Fig. 3.4(a), we notice that TSF reuses the optimal policies discovered in the source tasks, generating better performance. The results suggest that the functions h and g , extract more information from the dynamics of the environment, thus generating better representations to transfer. In summary, the performance improvement in source tasks after 2×10^5 in which both agents saturate is 20% better than the baseline. The metrics on target tasks bring the effect of the ensemble model into the transfer process. In the target tasks, the agent fits to the most desirable combination of policies for each target task. However, in this experiment, it is noticeable that in target tasks (T.1, T.2, T.3, T.4) the performance is similar to or below the baseline. From Fig. 3.4(b), it can be seen that the TSFDQN algorithm outperforms the target tasks (T.5, T.6, T.7, T.8), which corresponds to the most dissimilar tasks according to their coordinates, as shown in Fig. 3.3.

From the results, we derived two hypotheses. First, in Scenario 1, Fig. 3.3(d), there are four target tasks similar to the source tasks. Second, we wonder about the existence of better mechanisms to transfer apart from the ensemble model, although this hypothesis is outside the scope of this chapter.



(a) Average return in source tasks.



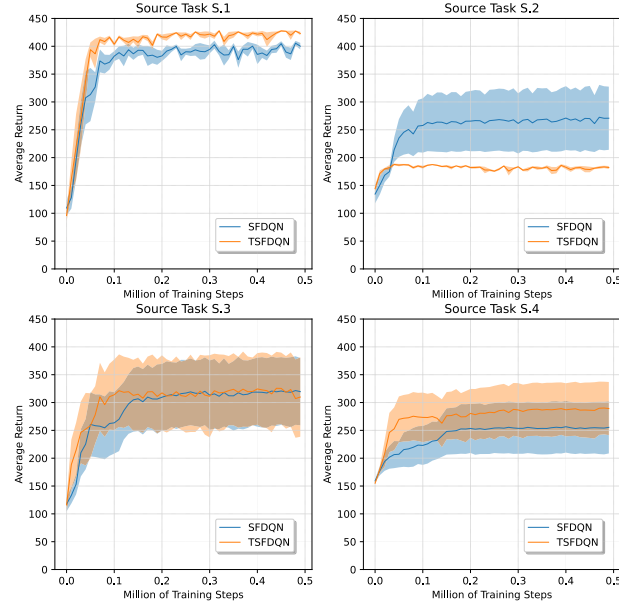
(b) Average return in target tasks.

Figure 3.5: Scenario 2: Average return in source and target tasks by total training steps. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.

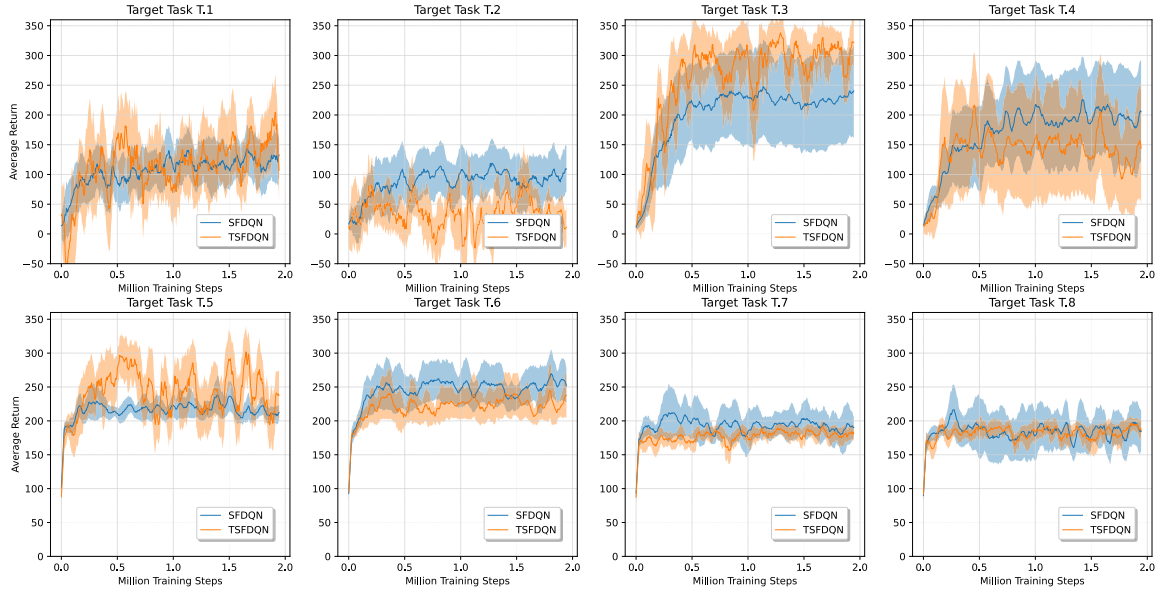
For *Scenario 2*, we used the best hyperparameters identified in Scenario 1. The task coordinates and the corresponding *torque multiplier* are given in Fig. 3.3(e) and Table 3.4. To execute this scenario, we exposed each source task to the same strategy of 10^4 training steps and 50 cycles. In this scenario, we train the agent with the 6 source tasks and 50 cycles, for a total of 3×10^6 training steps.

The results for this set of experiments are shown in Fig. 3.5. In this scenario, TSFDQN in 4 of the 6 source tasks outperformed the baseline. Our hypothesis is that transformation functions could affect the optimal policies in source tasks but leverage the knowledge in target tasks. The TSFDQN agent outperforms in the 2 target tasks.

CHAPTER 3. ADAPTIVE SUCCESSOR FEATURES VIA TRANSITION DYNAMICS TRANSFORMATION



(a) Average return in source tasks.



(b) Average return in target tasks.

Figure 3.6: Scenario 1: Average return in source and target tasks by total training steps for diverse dynamics setting. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.

3.4.2.2 Diverse Dynamics Setting

For this setting, we test the agents in the Reacher, Cartpole, and Hopper environments.

Reacher Environment. In this setting, in addition to learning to achieve objects, our agent now learns to manipulate the robot arm under different physical conditions, specifically motor torques. We introduced a different condition to change the environment dynamics. We modified the torque of the robot arm to make it slower or faster to reach a specific task. In this experiment, we added the torque multiplier between 0.0001 and 10 to the tasks in Fig. 3.3.

For Scenario 1, we plot the results in Fig. 3.6. In the source tasks in Fig. 3.6(a), our agent is able to induce improved policies. Our agent could not find an optimal policy for source task S.2. Despite the shared torque multiplier of source tasks S.1 and S.2, the agent receives more accumulated rewards for source task S.1 than for source task S.2. Furthermore, when analysing the final values of the Ω values in the supplementary material, the agent is not biased by sharing similarities among tasks, but only in tasks that represent a better induced policy. To give an example, the target task T.5 is more relevant to the source task S.2 than to S.1 given the similarity in torque. Intuition suggests that similar dynamics represent similar relevance in the tasks.

The analysis of target tasks, illustrated in Fig. 3.6(b), shows that our agent is able to perform similarly or better on the most diverse tasks. For example, Tasks T.1 and T.8 represent the tasks with more dynamics differences compared to the source tasks. Our agent achieves a performance improvement of 12% in target task T.1 and 5% in task T.8, which are the slowest and fastest robot arms with a torque multiplier set in 10^{-4} and 10, respectively. For tasks T.3 and T.5, our agent achieves a performance improvement of 30% and 15%, respectively.

We set a *diverse dynamic setting* for Scenario 1 according to Table 3.3. It is noticeable that source tasks S.1 and S.2 have the same torque multiplier and are also on the same axis; that is, these two tasks are similar in dynamics. In this vein, we wanted to understand the relevance assigned to the different tasks by the proposed model.

Further analysis of Ω values for this Scenario reveals that the ensemble model assigns relevance to successor features based on the target task, and it is not biased by the policies or performance in source tasks. For instance, the TSFDQN agent in T.3 outperforms the baseline, splitting the relevance into three source tasks: S.1, S.2, and S.3. However, the agent in S.2 did not find an optimal policy.

Table 3.5: Ω values after 2×10^6 training steps for Scenario 1, diverse dynamics setting.

Target Tasks	T.1	T.2	T.3	T.4	T.5	T.6	T.7	T.8
S.1	0.37	0.29	0.30	0.25	0.17	0.11	0.07	0.08
S.2	0.34	0.31	0.36	0.30	0.52	0.42	0.66	0.74
S.3	0.11	0.11	0.27	0.08	0.12	0.03	0.10	0.03
S.4	0.18	0.29	0.07	0.37	0.19	0.44	0.17	0.15

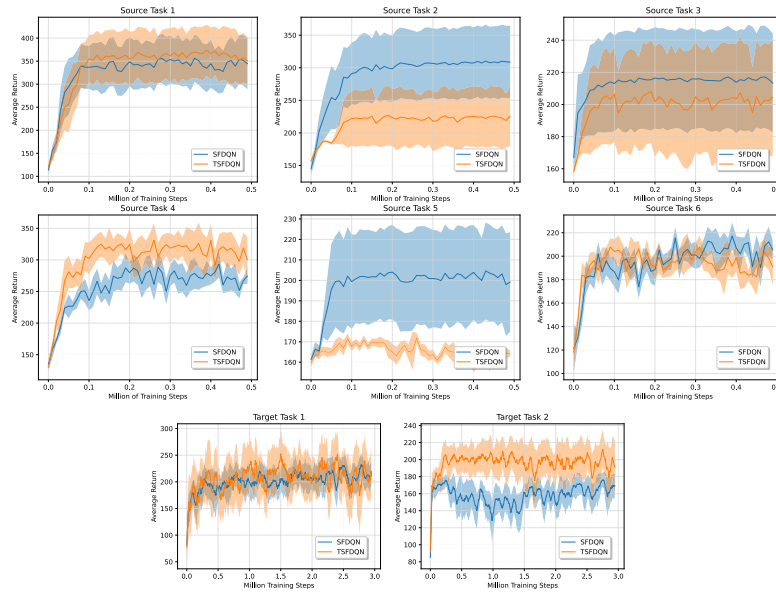


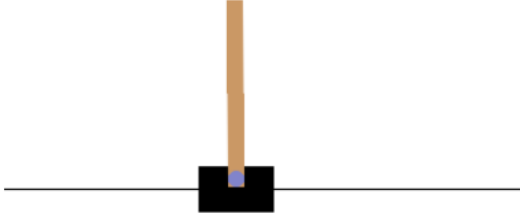
Figure 3.7: Scenario 2: Average return in source (top) and target (bottom) tasks. The shading shows one-half of a standard deviation over 5 runs with independent random seeds.

In the case of Scenario 2. The results are shown in Fig. 3.7. In this scenario, the source tasks S.2 and S.5 did not induce optimal policies.

For Scenario 2, contrary to the *shared dynamics setting*, in the source tasks, the agent could not find the optimal successor features to induce the optimal policies for source tasks S.2 and S.5. However, these suboptimal successor features help to induce a near-optimal policy in target tasks. Our analysis of Ω , detailed in Table 3.6, indicates that differences in the dynamics characteristics and diverse rewards enhance the ensemble model’s prediction of a better successor feature. This can be seen in the Ω values larger than 0 for more than 1 source tasks. Our hypothesis is that larger differences in dynamics among source tasks leverage better source transformed successor features, hence, better target transformed successor features as well as better transformed states into the base probability distribution.

Table 3.6: Ω values after 3×10^6 training steps for Scenario 2, diverse dynamics setting.

	Target Task 1	Target Task 2
Source Task 1	0.03	0.00
Source Task 2	0.01	0.42
Source Task 3	0.00	0.24
Source Task 4	0.36	0.02
Source Task 5	0.05	0.31
Source Task 6	0.55	0.00

(a) Cartpole. Source task with pole of length $0.5m$

(b) Source and target tasks in Cartpole environment.

	Pole length (m)
Source Task 1	0.5
Source Task 2	0.8
Source Task 3	1.0
Target Task 1	0.1
Target Task 2	3.0

Figure 3.8: Cartpole environment. Set of source and target tasks defined by the pole lengths in meters.

To summarise, across both scenarios and settings (shared and diverse transition dynamics), results in the Reacher environment indicate that TSFDQN generally outperforms SFDQN, with improvements more pronounced in the shared dynamics setting. Notably, in Scenario 1 under diverse dynamics, task T.3 shows a torque multiplier similar to the source tasks, and this target task exhibits the greatest relative improvement compared to the other target tasks. We hypothesise that when transition dynamics are similar, TSFDQN effectively leverages structural similarities to derive improved successor features. On the other hand, in Scenario 2, where transfer occurs across highly dissimilar transition dynamics, TSFDQN demonstrates more substantial performance gains over SFDQN. Importantly, the source tasks in Scenario 2 exhibit greater diversity in transition dynamics, suggesting that TSFDQN benefits from exposure to more varied source dynamics. This increased diversity appears to provide a richer basis for knowledge transfer to dissimilar target tasks.

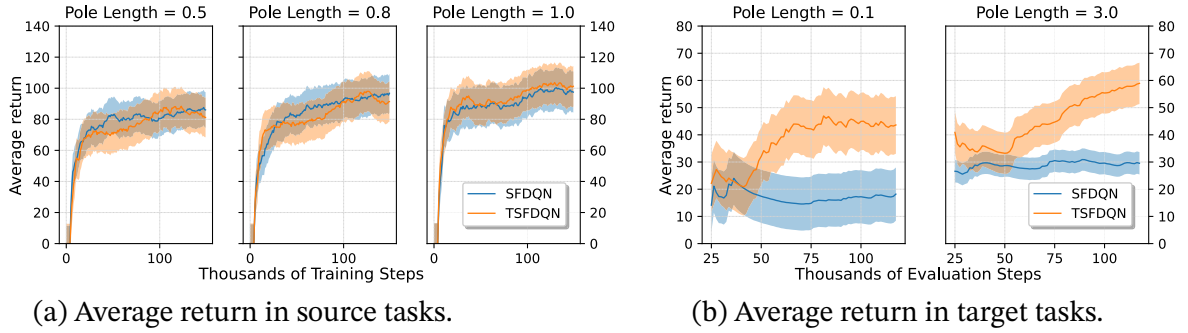


Figure 3.9: Average return in source and target tasks by total training steps. Shades show half a standard deviation over 10 runs with independent random seeds.

Cartpole environment. The second environment is the classical Cartpole environment with *diverse dynamics* setting. In this environment, an agent controls a pole and receives reward for each timestep it remains balanced. An example of the Cartpole is shown in Fig. 3.8(a). We extended the experiment proposed by Abdolshah et al. (2021) to balance poles of different lengths. The agents first learn to balance three different poles $\{0.5m, 0.8m, 1.0m\}$, and then use the acquired knowledge to balance poles of different lengths $\{0.1m, 3.0m\}$, as depicted in Table 3.8(b).

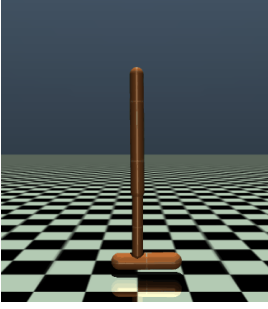
In this case, we learn the feature function using the process described in (Barreto et al., 2020). In a pre-training stage, the agent uses a random policy in the source tasks to fit the most suitable feature function and weights according to Eq. (3.14) with a fixed feature dimension, d , as 20. After the pre-training stage, the reward mappers are discarded and the agent proceeds with the regular training/evaluation already described. For TSFDQN, we use the same learning rate for the Ω values and the output of the g dimension found in the Reacher environment.

The results are shown in Fig. 3.9. In source tasks, Fig. 3.9(a), the TSFDQN agent achieved similar performance to that of the SFDQN agent. In target tasks, our agent performs better than SFDQN with a mean improvement of 49.03%. It is noticeable that the performance on target tasks of the TSFDQN agent on target tasks is approximately a 40% of the performance on source tasks, while the SFDQN deteriorates by about 60%. This suggests that our proposed model acquires similar policies in source tasks but improves those in target tasks.

We analysed the averaged Ω over the 10 runs in the Cartpole environment. The final values are depicted in Table 3.7. As mentioned earlier, TSFDQN tends to assign greater relevance to tasks with similar dynamics. In this regard, the Ω for the *Target Task 1* give more importance to the *Source Task 1*. On the other hand, *Target task 1* did not differentiate

Table 3.7: Ω for target tasks in Cartpole environment.

Target Tasks	Target Task 1	Target Task 2
Source Task 1	0.654	0.323
Source Task 2	0.243	0.322
Source Task 3	0.103	0.355



(a) Hopper environment.

(b) Source and target tasks in Hopper environment.

	healthy z range
Source Task 1	$(0.03, \infty)$
Source Task 2	$(0.1, \infty)$
Source Task 3	$(0.7, \infty)$
Target Task 1	$(0.0, \infty)$

Figure 3.10: Hopper environment. Set of source and target tasks defined by the healthy ranges.

among source tasks.

Hopper environment. The third environment is based on the hopper environment.

The agent aims to control a robotic leg to jump forward. The agent receives rewards for every timestep for healthy conditions, such as when the leg height is below a threshold, and an additional reward for moving forward. Our version has 27 discrete actions. Each action is a 3-tuple with values $\{-0.1, 0, 0.1\}$.

To add dissimilarities in the reward function and dynamics, we set different values for healthy states; thus, this environment represents a *diverse dynamics* setting. Despite its simplicity, a robot in a state with a higher value within a *healthy z range* can take an action that leads to an unhealthy state and a lower reward. In other words, this scenario covers dissimilarities in rewards, and transition dynamics also tests the agent’s ability to provide control to different robots and reward conditions.

We set the TRL where agent learnt from source tasks with different *healthy z range* values and then reused the knowledge in two target tasks, as shown in Table 3.10(b). We learn the feature function using the same strategy as in the Cartpole environment and the dimensions d as 50. With the same learning rate for the Ω values and the g dimension as before.

The results are shown in Fig. 3.11. Results in source tasks indicate that the transforma-

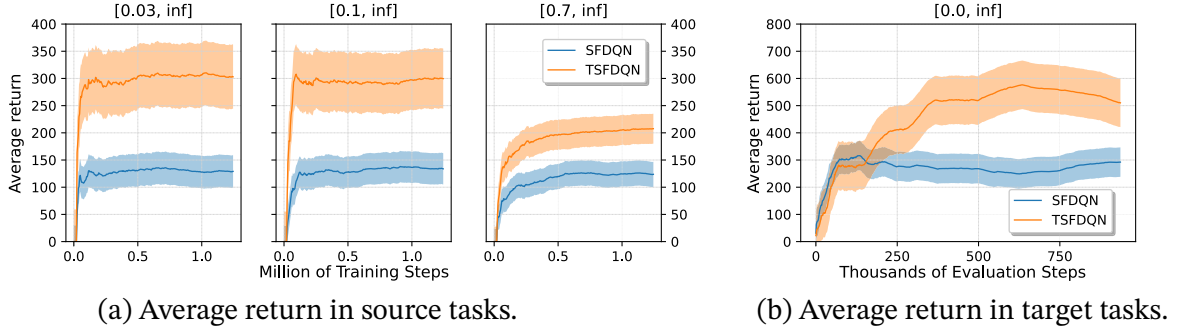


Figure 3.11: Average return in source and target tasks by total training steps. Shades show half a standard deviation over 5 runs with independent random seeds.

Table 3.8: Ω for target tasks in Hopper environment.

Target Tasks	Target Task 1
Source Task 1	0.38
Source Task 2	0.60
Source Task 3	0.02

tion model in the transformed feature function provides enhanced features to be predicted by the successor feature, hence inducing better policies. Similarly, in the target task, the ensemble model induces a better policy with an average gain of 25% compared to the average return in source tasks.

We present the final averaged omegas in Table 3.8. Ω values reveal that the TSFDQN agent transfers knowledge according to the discovered similarities. In this vein, the agent found *Target Task 1* has more similarities with *Source Task 1* and *Source Task 2* rather than *Source Task 3*.

3.5 Summary

In this chapter, we introduced a novel method for handling shared and diverse transition dynamics within the successor feature framework. Our results suggest that the proposed transformed feature function effectively captures the structural characteristics of both shared and diverse dynamics. The ensemble model estimates the relative importance of each source task, improving successor feature prediction in target tasks. Furthermore, by applying probability density transformation techniques, successor features can be extended to operate under heterogeneous transition dynamics.

Across environments, the empirical findings reveal consistent advantages of the pro-

posed approach. In the Reacher environment, TSF-based agents generally outperformed standard successor feature baselines, with improvements being more evident under shared dynamics and in scenarios where source-task diversity supported transfer. However, performance gains were comparatively moderate. In contrast, in the Cartpole and Hopper environments, the improvements were substantially more noticeable. In these domains, the proposed method achieved significantly better transfer performance under both shared and diverse dynamics settings, demonstrating a solution to transition variation. These results suggest that the transformed successor feature framework is effective in particular environments, where differences in dynamics are structurally meaningful and transferable through latent transformations.

Experiments under diverse dynamics further demonstrate that the approach can learn suboptimal but transferable successor features in source tasks, which can then support policy transfer to target tasks. By modelling diverse dynamics through probability density transformations into a shared distribution using invertible functions, the framework enables structured knowledge reuse across heterogeneous environments.

We also acknowledge the current limitations of our work. The agents learning the feature functions do not generate optimal policies. To address this, our approach could be combined with feature-extraction methods designed for the successor feature framework, such as Carvalho et al. (2023). We also plan to draw on the ideas of Hunt et al. (2019) to extend our work to stochastic policies and continuous action spaces.

The assumption of invertible and differentiable functions limits the class of transition dynamics that can be modelled. However, recent work has empirically demonstrated the ability to learn optimal policies using normalising flow-based transformations (Ghugare & Eysenbach, 2025), while progressively relaxing strict constraints in normalising flows (Cornish et al., 2020; Draxler et al., 2024). These advances suggest that, within the class of smooth invertible mappings, the practical limitations imposed by invertibility are being reduced, making the proposed approximation increasingly flexible in structured dynamical settings.

Finally, using more expressive invertible and differentiable functions could capture more intricate dynamics and potentially improve the overall performance of transformed successor feature-based agents. In particular, normalising flows provide a principled way to apply change-of-variable techniques with invertible mappings, enabling the modelling of complex probability distributions (Cornish et al., 2020; Draxler et al., 2024; Rezende & Mohamed, 2015).

ADAPTIVE SUCCESSOR FEATURES VIA GAUSSIAN PROCESS KERNEL SPECIFICATION

You don't develop courage by being happy in your relationships everyday. You develop it by surviving difficult times and challenging adversity.

Epicurus

In this chapter, we aim to address **RQ-2** by accomplishing **RO-3** and **RO-4**. To do so, we leverage the intuition of transferring similarities and dissimilarities between source and target tasks while explicitly capturing epistemic uncertainty. To facilitate this transfer, we design a novel and valid Successor Feature (SF)-based Dynamics Transfer Kernel within a Gaussian Process (GP) framework, which uses target observations to adapt source SFs for solving target tasks. Additionally, we exploit the properties of GPs to model uncertainty and incorporate it into policy selection using the Generalised policy improvement (GPI) mechanism.

4.1 Background and Motivation

Reinforcement Learning (RL) has proven to be a powerful technique for solving problems through trial and error. Although RL has achieved human-level performance in complex tasks such as games and robotic control (Mnih et al., 2015; Schrittwieser et al., 2020), it typically requires large amounts of data and computational resources to solve a single task.

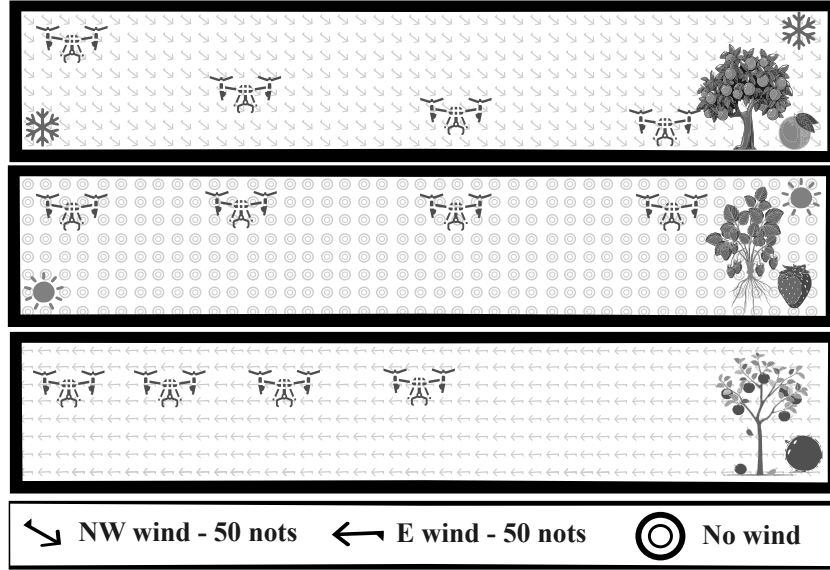


Figure 4.1: Seasonal drone picker. Different wind conditions occur each season, resulting in varying transition dynamics.

Thus, RL is impractical for many real-world scenarios involving multiple tasks. Fig. 4.1 shows a seasonal harvesting robot that picks apples in autumn, mandarins in winter, and berries in summer. An RL agent would need to learn to harvest for each season from scratch, due to the different fruit types and the control challenges caused by the varying weather conditions impacting flyability (Gao et al., 2021).

Transfer Reinforcement Learning (TRL) addresses this limitation by reusing knowledge from previously learnt tasks to solve new, unseen ones. Following the harvesting robot example, the robot could leverage knowledge gained during autumn and winter seasons to facilitate berry picking in summer. To do this, TRL requires addressing two key questions: *what* knowledge should be transferred, and *how* should it be applied?

SFs offer a structured approach to answering both questions. They are typically used where tasks share the same transition probability distribution (dynamics), and rewards are linear functions of feature vectors. This means the expected effect of an action in a state remains consistent across tasks. Under this framework, *what* to transfer is answered by reusing learnt representations, and *how* to transfer is reduced to estimating the new reward weights. However, the fixed dynamics assumption is often unrealistic in practice, as seen with the seasonal harvesting robot. This limits SFs’ applicability, preventing effective knowledge transfer with significantly varying dynamics.

Recent work has explored enabling transfer across dissimilar environments. We proposed the existence of a shared latent transition dynamics, where each source task’s dy-

namics is a transformation of this latent model (Garces, Kiyoshige et al., 2023). However, our approach was limited to linear transformations. J. Zhang et al. (2017) enabled transference by assuming that target task feature functions are linearly related to those of the source tasks. Abdolshah et al. (2021) used Gaussian Process (GP) to model the target task SFs as noisy versions of source SFs, thereby allowing knowledge transfer despite the absence of shared transition dynamics. Yet, their approach assumes that the noise is fixed and known in advance and does not leverage the full capability of GPs to model uncertainty. These works rely on predefined relationships between transition dynamics or feature functions, requiring prior knowledge of the environment’s structure. Hence, effective SF transfer across diverse dynamics, without a priori assumptions, remains a critical gap.

To overcome this, we propose a novel mechanism that leverages SFs via GP from source to target tasks with differing dynamics. Our approach introduces a new SF-based Dynamics Transfer Kernel, employing Bayesian inference to approximate target SFs based on immediate observed feature values. This method quantifies uncertainty and captures dynamic differences by comparing feature values, enabling SF transfer without requiring shared, fixed dynamics, or prior knowledge of their relationship. Overall, our contributions are:

1. A novel SF-based Dynamics Transfer Kernel for transferring SFs across tasks with dissimilar dynamics, supported by theoretical evidence of its validity;
2. an uncertainty-based policy selection method; and
3. empirical evidence demonstrating our approach’s ability to adapt knowledge to new tasks with differing dynamics, using control and navigation environments widely used in the SF literature.

4.2 Preliminaries and Problem Formulation

We begin this section by introducing the foundations for the rest of the chapter. First, we recall the definition of Gaussian Process (GP). For a given dataset D consisting of n training inputs $\{x_i\}_{i=1}^n$ for $x_i \in \mathbb{R}^d$, and corresponding real-valued targets $\mathbf{y} = [y_1, \dots, y_n]^\top$, a Gaussian Process (GP) defines a distribution over functions $f : \mathbb{R}^d \rightarrow \mathbb{R}$. This distribution is characterised by its mean, $m(\cdot)$, and kernel, $k(\cdot, \cdot)$, functions, formally written as $f \sim \text{GP}(m(\cdot), k(\cdot, \cdot))$ (Rasmussen & Williams, 2005). Observed targets $\mathbf{y}$ are noisy evaluations of f , with $y_i = f(x_i) + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$. Typically, $m(\cdot)$ is assumed to be zero, and the chosen $k(\cdot, \cdot)$ is the Squared Exponential (SE) kernel, $k(x, x') = \sigma^2 \exp[-\frac{1}{2} \frac{\|x - x'\|^2}{l^2}]$,

where σ^2 represents variance or the average distance of the function from its mean, and l is the lengthscale, representing how quickly the function’s output varies with respect to changes in the input space. GPs are well-suited for low-data regimes because they provide closed-form predictions, enable Bayesian inference, incorporate prior beliefs about the function, and model epistemic uncertainty (Tighineanu et al., 2022; Wei et al., 2023).

Second, we restate that RL problems are commonly modelled as Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. We are interested in TRL, which defines two sets of tasks: source (**S**) and target (**T**), and aims to construct a policy for a task in **T** by reusing knowledge learnt from tasks in **S**. Each task is defined by its own MDP. In particular, the source task set is given by $\mathbf{S} = \{(\mathcal{S}, \mathcal{A}, r_i = \phi^\top \mathbf{w}_i, p_i, \gamma)\}_{i=1}^n$, and the set of m target tasks is $\mathbf{T} = \{(\mathcal{S}, \mathcal{A}, r_j = \phi^\top \mathbf{w}_j, p_j, \gamma)\}_{j=1}^m$. Tasks differ in their transition dynamics p as well as in reward structures r , where $r(s, a, s') = \phi(s, a, s')^\top \mathbf{w}$ is represented as linear combinations of a shared ϕ function and task-specific weight vectors $\mathbf{w}$.

Our problem setting considers scenarios with different dynamics. Therefore, for a source task $S_i \in \mathbf{S}$ and target task $T_j \in \mathbf{T}$, their dynamics are denoted by p_{S_i} and p_{T_j} , respectively. We adopt the standard SF assumption of a shared, known feature function $\phi(s, a, s') \in \mathbb{R}^d$, which characterises the reward functions. In addition, we adopted the mild assumption of the existence of a collection of relevant experiences from the training on the source tasks. That is, for each source task $S_i \in \mathbf{S}$, a dataset D_i of k observations is stored. This dataset is given by $D_i = \{(x_\ell^{S_i}, \psi^{\pi_{S_i}}(x_\ell^{S_i}), \phi(x_\ell^{S_i}, s_{\ell+1}^{S_i}))\}_{\ell=1}^k$, where $x_\ell^{S_i} \equiv (s_\ell^{S_i}, a_\ell^{S_i})$ represents a state-action pair, $\psi^{\pi_{S_i}}(x_\ell^{S_i})$ its SF representation, and $\phi(x_\ell^{S_i}, s_{\ell+1}^{S_i})$ the feature vector for the observed transition to the next state $s_{\ell+1}^{S_i}$. Our interest is to enable the transfer of SF across tasks with different dynamics.

4.3 Method

Our approach enables SF transfer by incorporating dynamic differences directly into a novel SF representation. A common alternative is to assume a fixed relationship between dynamics; however, this typically depends on a priori assumptions and expert knowledge. In contrast, our method leverages the ability of GPs to infer from limited data, quantify uncertainty, and exploit their covariance structure to relate inputs. By tailoring this covariance, we enable SFs to transfer effectively across environments with varying dynamics.

4.3.1 Transition Dynamics Comparison via Feature Observations

Consistent with the problem setting and the canonical state-action SF function, we extend this function to obtain a one-step observation for a task $S_i \in \mathbf{S}$ as follows,

$$\begin{aligned}
\psi^{\pi_{S_i}}(s, a) &= \mathbb{E}_{\pi, s_{t+1} \sim p_{S_i}} \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t, s_{t+1}) \right] \\
&= \mathbb{E}_{s_1 \sim p_{S_i}} [\phi(s_0, a_0, s_1)] \\
&\quad + \mathbb{E}_{\pi, s_{t+1} \sim p_{S_i}} \left[\sum_{t=1}^{\infty} \gamma^t \phi(s_t, a_t, s_{t+1}) \right] \\
&= \phi_{S_i}(s, a) + \mathbb{E}_{\pi, s_{t+1} \sim p_{S_i}} \left[\sum_{t=1}^{\infty} \gamma^t \phi(s_t, a_t, s_{t+1}) \right],
\end{aligned}$$

where $\phi_{S_i}(s, a) = \mathbb{E}_{s' \sim p_{S_i}} [\phi(s, a, s')]$ expresses the immediate observed features induced by p_{S_i} dynamics.

Similarly, for a target task $T_j \in \mathbf{T}$, the immediate observed features are given by $\phi_{T_j}(s, a) = \mathbb{E}_{s' \sim p_{T_j}} [\phi(s, a, s')]$. These immediate features capture underlying dynamics independently of policy-induced behaviour (unlike SFs, which capture both), preventing policy and transition dynamics from intertwining during comparison. By jointly exploiting the shared feature function, the linearity of expectation, and a shared state space assumption, we can indirectly compare source and target dynamics by examining the differences between their expected feature functions, as follows,

$$\begin{aligned}
\| \phi_{S_i}(s, a) - \phi_{T_j}(s, a) \| &= \| \mathbb{E}_{s' \sim p_{S_i}} [\phi(s, a, s')] - \mathbb{E}_{s' \sim p_{T_j}} [\phi(s, a, s')] \| \\
&= \| \int_{s'} \phi(s, a, s') p_{S_i}(s' | s, a) ds' - \int_{s'} \phi(s, a, s') p_{T_j}(s' | s, a) ds' \| \\
&= \| \int_{s'} \phi(s, a, s') [p_{S_i}(s' | s, a) - p_{T_j}(s' | s, a)] ds' \| \\
&= \| \int_{s'} \phi(s, a, s') \Delta P ds' \|.
\end{aligned}$$

As SF settings require a shared feature function rather than state spaces, we distinguish source and target state spaces as $\mathcal{S}_{S_i}$ and $\mathcal{S}_{T_j}$, respectively. Assuming the ϕ function is differentiable with respect to state variables, we apply a change of variables using a Jacobian transformation. The dynamics for the source task are then expressed as, $p_{S_i}(s'_{S_i} | s_{S_i}, a) = p_{\phi_{S_i}}(\phi(s_{S_i}, a, s'_{S_i}) | s_{S_i}, a) \left| \frac{d\phi(s_{S_i}, a, s'_{S_i})}{ds'_{S_i}} \right|$ for $s_{S_i} \in \mathcal{S}_{S_i}$. Similarly, for the target task we derive, $p_{T_j}(s'_{T_j} | s_{T_j}, a) = p_{\phi_{T_j}}(\phi(s_{T_j}, a, s'_{T_j}) | s_{T_j}, a) \left| \frac{d\phi(s_{T_j}, a, s'_{T_j})}{ds'_{T_j}} \right|$ for

$s_{T_j} \in \mathcal{S}_{T_j}$. We rewrite the features comparison as,

$$\begin{aligned}
& \| \phi_{S_i}(s_{S_i}, a) - \phi_{T_j}(s_{T_j}, a) \| \\
&= \| \mathbb{E}_{s'_{S_i} \sim p_{S_i}} [\phi(s_{S_i}, a, s'_{S_i})] - \mathbb{E}_{s'_{T_j} \sim p_{T_j}} [\phi(s_{T_j}, a, s'_{T_j})] \| \\
&= \| \int_{s'_{S_i}} \phi(s_{S_i}, a, s'_{S_i}) p_{S_i}(s'_{S_i} | s_{S_i}, a) ds'_{S_i} \\
&\quad - \int_{s'_{T_j}} \phi(s_{T_j}, a, s'_{T_j}) p_{T_j}(s'_{T_j} | s_{T_j}, a) ds'_{T_j} \| \\
&= \| \int_{\phi=\phi(s_{S_i}, a, s'_{S_i})} \phi p_{\phi_{S_i}}(\phi | s_{S_i}, a) \left| \frac{d\phi}{ds'_{S_i}} \right| ds'_{S_i} \quad (4.1) \\
&\quad - \int_{\phi=\phi(s_{T_j}, a, s'_{T_j})} \phi p_{\phi_{T_j}}(\phi | s_{T_j}, a) \left| \frac{d\phi}{ds'_{T_j}} \right| ds'_{T_j} \| \\
&= \| \int_{\phi=\phi(s_{S_i}, a, s'_{S_i})} \phi p_{\phi_{S_i}}(\phi | s_{S_i}, a) d\phi \\
&\quad - \int_{\phi=\phi(s_{T_j}, a, s'_{T_j})} \phi p_{\phi_{T_j}}(\phi | s_{T_j}, a) d\phi \| \\
&= \| \int_{\phi} \phi [p_{\phi_{S_i}}(\phi | s_{S_i}, a) - p_{\phi_{T_j}}(\phi | s_{T_j}, a)] d\phi \| .
\end{aligned}$$

The analysis of transitions within feature space in the SF context has been studied by Lehnert and Littman (2020). While (4.1) enables indirect dynamics comparison, it requires access to the ϕ_{S_i} and ϕ_{T_j} functions. When these are unavailable, we approximate this by computing the divergence between observed features as the average distance between each of the n source and m target feature observations,

$$\| \phi_{S_i} - \phi_{T_j} \| = \frac{1}{nm} \sum_{k=0}^m \sum_{\ell=0}^n \| \phi_{S_i, \ell} - \phi_{T_j, k} \| .$$

This allows a straightforward comparison between tasks without requiring additional models. We propose incorporating this as a metric to model task dissimilarities. From now on, when referring to S we mean any task in the source task set $\mathbf{S}$, that is, $S \in \mathbf{S}$. Similarly, T refer to any task in $\mathbf{T}$ such that $T \in \mathbf{T}$.

4.3.2 Successor Feature-based Transition Dynamics Transfer Kernel

To leverage our metric, the overall idea is summarised in Fig. 4.2. We hypothesise that, in the feature function space, different immediate features are observed due to the influence of the underlying transition dynamics. By using the proposed metric to compare

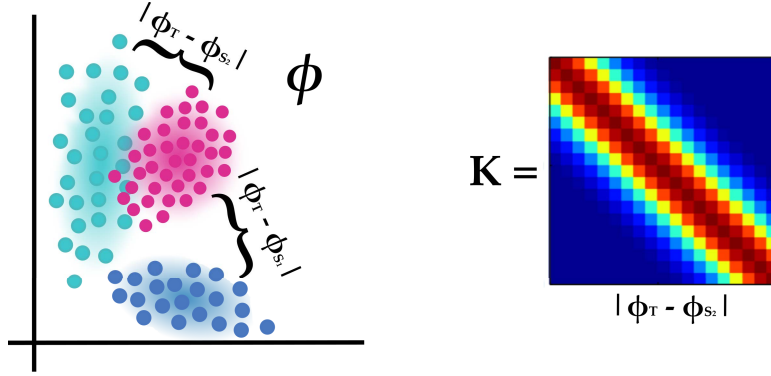


Figure 4.2: Left, each colour group in the observed feature space represents a different task with distinct transition dynamics. Right, covariance matrix using the immediate observed features metric.

these observed features, we can effectively quantify differences across varying transition dynamics.

We introduce this metric into an extended Squared Exponential (SE) kernel, given its ability to approximate arbitrary functions (Micchelli et al., 2006). The proposed kernel captures dissimilarities per feature dimension, d , as

$$k_\tau(S, T) = \alpha_{ST}^2 \exp \left[-\frac{1}{2} \frac{\|\phi_{S,d} - \phi_{T,d}\|^2}{l_{ST}^2} \right], \quad (4.2)$$

where α_{ST}^2 and l_{ST}^2 represent the overall amplitude and smoothness, given the feature observations.

Notably, the proposed kernel depends only on observed features from both tasks, but approximating the SF also requires conditioning on the state-action pair. For simplicity, all (s, a) inputs are rewritten as x . Therefore, the kernel function for inputs $x \equiv (s, a)$ and $x' \equiv (s', a')$ is,

$$k_\chi(x, x') = \alpha_x^2 \exp \left[-\frac{1}{2} \frac{\|x - x'\|^2}{l_x^2} \right], \quad (4.3)$$

where α_x^2 and l_x^2 control variance and smoothness given the input observations. To effectively combine tasks and inputs, we propose the following kernel,

$$k((x, S), (x', T)) = k_\chi(x, x')(\delta_{S,T} + (1 - \delta_{S,T})k_\tau(S, T)), \quad (4.4)$$

where $\delta_{S,T}$ is the Kronecker delta.

To ensure a tailored kernel is valid, it must produce a symmetric, positive semi-definite (PSD) covariance matrix (Rasmussen & Williams, 2005). We begin by defining a general

covariance matrix relating source and target task observations,

$$K_* = \begin{bmatrix} K_{SS} & K_{ST} \\ K_{ST}^\top & K_{TT} \end{bmatrix}, \quad (4.5)$$

where K_{SS} is the self-covariance of source task observations, K_{ST} the covariance block of observations of source and target tasks, and K_{TT} the self-covariance of target task observations. To include both the task and input kernels, we exploit the off-diagonal block of this covariance matrix.

As shown in the kernel in (4.4) and covariance matrix in (4.5), the off-diagonal block relates the n source and m target observations, incorporating this relationship into inference. The diagonal blocks, K_{SS} and K_{TT} , compare inputs within their respective tasks using the kernel from (4.3). The covariance matrix for K_{SS} is given by,

$$K_{SS} = \begin{bmatrix} k_\chi(x_1^S, x_1^S) & \dots & k_\chi(x_1^S, x_n^S) \\ \vdots & \ddots & \vdots \\ k_\chi(x_n^S, x_1^S) & \dots & k_\chi(x_n^S, x_n^S) \end{bmatrix},$$

while for K_{TT} is,

$$K_{TT} = \begin{bmatrix} k_\chi(x_1^T, x_1^T) & \dots & k_\chi(x_1^T, x_m^T) \\ \vdots & \ddots & \vdots \\ k_\chi(x_m^T, x_1^T) & \dots & k_\chi(x_m^T, x_m^T) \end{bmatrix}.$$

To capture task dissimilarities, we reformulate K_{ST} to integrate source and target distance metrics as,

$$K_{ST} = \begin{bmatrix} k_\tau(S, T)k_\chi(x_1^S, x_1^T) & \dots & k_\tau(S, T)k_\chi(x_1^S, x_m^T) \\ \vdots & \ddots & \vdots \\ k_\tau(S, T)k_\chi(x_n^S, x_1^T) & \dots & k_\tau(S, T)k_\chi(x_n^S, x_m^T) \end{bmatrix}.$$

As stated, for K_* to be a valid covariance matrix, it must be symmetric and positive semi-definite (PSD). Therefore, we derive the following lemma and proof.

Lemma 4.1. *Let $k(\cdot, \cdot)$ be the kernel function defined by (4.4). The matrix K_* , formed by the blocks K_{SS} , K_{ST} , and K_{TT} , is symmetric and PSD.*

Proof. To prove symmetry, we proceed by cases, considering n source and m target task observations.

Case $n > 0$ and $m = 0$. In this case, $K_* = K_{SS}$. Let x_i^S and x_j^S be arbitrary source task inputs. The (i, j) -th element of K_{SS} is given by $[K_{SS}]_{ij} = k_\chi(x_i^S, x_j^S)$. Since $k_\chi(\cdot, \cdot)$ is

the symmetric SE kernel, by definition, $k_\chi(x_i^S, x_j^S) = k_\chi(x_j^S, x_i^S)$. It therefore follows that $[K_{SS}]_{ij} = [K_{SS}]_{ji}$. Thus, K_* is a symmetric matrix.

Case $n = 0$ and $m > 0$. In this case, $K_* = K_{TT}$. Let x_i^T and x_j^T be arbitrary target task inputs. The (i, j) -th element of K_{TT} is given by $[K_{TT}]_{ij} = k_\chi(x_i^T, x_j^T)$. Since $k_\chi(\cdot, \cdot)$ is the symmetric SE kernel, by definition, $k_\chi(x_i^T, x_j^T) = k_\chi(x_j^T, x_i^T)$. It therefore follows that $[K_{TT}]_{ij} = [K_{TT}]_{ji}$. Thus, K_* is a symmetric matrix.

Case $n > 0$ and $m > 0$. In this case, K_* is a block matrix as described in Eq. (4.5). By previous cases, K_{SS} and K_{TT} are symmetric. Consider the off-diagonal block K_{ST} . Let x_i^S and x_j^T be arbitrary source and target task inputs. The (i, j) -th element of K_{ST} is given by $[K_{ST}]_{ij} = k_\chi(x_i^S, x_j^T)k_\tau(S, T)$. Since both $k_\chi(\cdot, \cdot)$ and $k_\tau(S, T)$ are symmetric SE kernels, we have that $k_\chi(x_i^S, x_j^T) = k_\chi(x_j^T, x_i^S)$ and $k_\tau(S, T) = k_\tau(T, S)$. The corresponding element in the transpose block is given by $[K_{ST}]_{ji}^\top = k_\chi(x_j^T, x_i^S)k_\tau(T, S)$. Thus, based on the symmetry of k_χ and k_τ , it follows $[K_{ST}]_{ij} = [K_{ST}]_{ji}^\top$. Since K_{SS} , K_{TT} are symmetric and $[K_{ST}]_{ij} = [K_{ST}]_{ji}^\top$, K_* is a symmetric matrix. $\square$

Proof. To prove that K_* is PSD, we must show that for any vector $\mathbf{v} \in \mathbb{R}^{n+m}$ $\mathbf{v}^\top K_* \mathbf{v} \geq 0$.

$$\begin{aligned}
\mathbf{v}^\top K_* \mathbf{v} &= \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} v_i [K_*]_{ij} v_j \\
&= \left[\sum_{j=1}^{n+m} v_1 [K_*]_{1j} v_j \right] + \sum_{i=2}^{n+m} \left[\sum_{j=1}^{n+m} v_i [K_*]_{ij} v_j \right] \\
&= \left[\sum_{j=1}^n v_1 k_\chi(x_1, x_j) v_j + \sum_{j=n+1}^{n+m} v_1 k_\chi(x_1, x_j) k_\tau(S, T) v_j \right] \text{ (by definition)} \\
&\vdots \\
&\quad + \left[\sum_{j=1}^n v_n k_\chi(x_n, x_j) v_j + \sum_{j=n+1}^{n+m} v_n k_\chi(x_n, x_j) k_\tau(S, T) v_j \right] \\
&\quad + \left[\sum_{j=1}^n v_{n+1} k_\chi(x_{n+1}, x_j) k_\tau(S, T) v_j + \sum_{j=n+1}^{n+m} v_{n+1} k_\chi(x_{n+1}, x_j) v_j \right] \\
&\vdots \\
&\quad + \left[\sum_{j=1}^n v_{n+m} k_\chi(x_{n+m}, x_j) k_\tau(S, T) v_j + \sum_{j=n+1}^{n+m} v_{n+m} k_\chi(x_{n+m}, x_j) v_j \right] \\
&= \sum_{i=1}^n \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j + \sum_{i=1}^n \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) k_\tau(S, T) v_j
\end{aligned}$$

$$\begin{aligned}
& + \sum_{i=n+1}^{n+m} \sum_{j=1}^n v_i k_\chi(x_i, x_j) k_\tau(S, T) v_j + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \\
& = \mathbf{v}_{1:n}^\top K_{SS} \mathbf{v}_{1:n} + k_\tau(S, T) \left[\sum_{i=1}^n \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \right. \\
& \quad \left. + \sum_{i=n+1}^{n+m} \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j \right] + \mathbf{v}_{n+1:n+m}^\top K_{TT} \mathbf{v}_{n+1:n+m} \\
& = \underbrace{\mathbf{v}_{1:n}^\top K_{SS} \mathbf{v}_{1:n}}_{\geq 0} + 2k_\tau(S, T) \left[\sum_{i=1}^n \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \right] \\
& \quad + \underbrace{\mathbf{v}_{n+1:n+m}^\top K_{TT} \mathbf{v}_{n+1:n+m}}_{\geq 0} \text{ (by PSD)}
\end{aligned}$$

Next, we must prove the second term on the RHS is non-negative. To do this, we proceed by cases.

Case $\mathbf{0} \in \mathbb{R}^{n+m}$, $\forall_i v_i = 0$. In this case, $\mathbf{0}^\top K_* \mathbf{0} = 0$

Case $\mathbf{v} \in \mathbb{R}^{n+m}$, $\forall_i v_i \geq 0$. Since kernel functions are SE kernels then $k_\chi(\cdot, \cdot) \in [0, 1]$ and $k_\tau(\cdot, \cdot) \in [0, 1]$, addition and multiplication of non-negative values remains non-negative, thus $\mathbf{v}^\top K_* \mathbf{v} \geq 0$

Case $\mathbf{v} \in \mathbb{R}^{n+m}$, $\forall_i v_i < 0$. Let $\mathbf{v} = [-v_1, \dots, -v_{n+m}]$

$$\begin{aligned}
\mathbf{v}^\top K_* \mathbf{v} & = \sum_{i=1}^n \sum_{j=1}^n (-v_i) k_\chi(x_i, x_j) (-v_j) \\
& \quad + 2k_\tau(S, T) \left[\sum_{i=1}^n \sum_{j=n+1}^{n+m} (-v_i) k_\chi(x_i, x_j) (-v_j) \right] \\
& \quad + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} (-v_i) k_\chi(x_i, x_j) (-v_j) \\
& = \sum_{i=1}^n \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j \\
& \quad + 2k_\tau(S, T) \left[\sum_{i=1}^n \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \right] \\
& \quad + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \\
& \geq 0 \text{ (Reduced to case } \forall_i v_i \geq 0)
\end{aligned}$$

Case $\mathbf{v} \in \mathbb{R}^{n+m}$, $\exists_i v_i < 0$. We start by noticing that the first and third terms are ex-

pressed in the quadratic form and the kernel function $k_\chi(\cdot, \cdot)$ is PSD, thus the only term that could add negative values is the second term. Moreover, $k_\tau(\cdot, \cdot) \in [0, 1]$. This yields 3 cases:

Case $\mathbf{v} \in \mathbb{R}^{n+m}$, $\exists_i v_i < 0$ and $k_\tau(\cdot, \cdot) = 1$. The quadratic form is given by,

$$\begin{aligned} \mathbf{v}^\top K_* \mathbf{v} &= \sum_{i=1}^n \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j \\ &\quad + 2 \left[\sum_{i=1}^n \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \right] \\ &\quad + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \\ &\geq 0 \text{ (by quadratic form and } k_\chi(\cdot, \cdot) \text{ is PSD.)} \end{aligned}$$

Case $\mathbf{v} \in \mathbb{R}^{n+m}$, $\exists_i v_i < 0$ and $k_\tau(\cdot, \cdot) = 0$. The quadratic form is given by,

$$\begin{aligned} \mathbf{v}^\top K_* \mathbf{v} &= \sum_{i=1}^n \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j \\ &\quad + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \\ &\geq 0 \text{ (by } k_\chi(\cdot, \cdot) \text{ is PSD.)} \end{aligned}$$

Case $\mathbf{v} \in \mathbb{R}^{n+m}$, $\exists_i v_i < 0$ and $k_\tau(\cdot, \cdot) \in (0, 1)$. Building on previous cases, $\mathbf{v}^\top K_* \mathbf{v}$ is bounded by,

$$\begin{aligned} &\sum_{i=1}^n \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \\ &\leq \mathbf{v}^\top K_* \mathbf{v} \leq \\ &\sum_{i=1}^n \sum_{j=1}^n v_i k_\chi(x_i, x_j) v_j + 2 \left[\sum_{i=1}^n \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \right] \\ &\quad + \sum_{i=n+1}^{n+m} \sum_{j=n+1}^{n+m} v_i k_\chi(x_i, x_j) v_j \\ &\text{(by bounds and previous cases } \mathbf{v}^\top K_* \mathbf{v} \text{ must be } \geq 0) \end{aligned}$$

□

4.3.3 Successor Features Approximation via Gaussian Process

We introduce a GP approximation for SFs in the target task, leveraging source task knowledge through the defined kernel function. Without loss of generality, we model each d dimension of the SF (*i.e.*, ψ_d^π , for a given state-action $x \equiv (s, a)$), with a zero-mean GP $\text{GP}(0, k)^1$, where $k(\cdot, \cdot)$ is the kernel from (4.4). The posterior mean and variance for a new x^T are,

$$\boldsymbol{\mu}_d(x^T) = k_\chi(x^T, \mathbf{x})^\top K_*^{-1} \mathbf{y}_d, \quad (4.6)$$

$$\sigma_d^2(x^T) = k_\chi(x^T, x^T) - k_\chi(x^T, \mathbf{x})^\top K_*^{-1} k_\chi(x^T, \mathbf{x}), \quad (4.7)$$

where $\mathbf{x}$ is a vector with all source and target inputs, and $\mathbf{y}_d$ is the d -th dimension of SFs from source and target tasks.

The posterior mean yields the predicted SF for each dimension, which are concatenated for decision-making. Unlike Abdolshah et al. (2021), our approach does not assume a fixed relationship between target SF and source observations, requiring access to true target SFs for posterior updates. We approximate these SFs using the Bellman equation and currently observed features. At time t , given target observed features ϕ_t^T and new state s_{t+1}^T , the SF used to update the posterior is,

$$\psi^T(x_t^T) \approx \phi_t^T + \gamma \boldsymbol{\mu}(x_{t+1}^T), \quad (4.8)$$

where $x_{t+1}^T \equiv (s_{t+1}^T, a_{t+1})$, a_{t+1} is the greedy action such that $\arg \max_{a_{t+1} \in \mathcal{A}} \boldsymbol{\mu}(x_{t+1}^T)^\top \mathbf{w}^T$, $\mathbf{w}^T$ the target task weights, and $\boldsymbol{\mu}$ is the vector comprised of posterior means for all D dimensions, $\boldsymbol{\mu}(s_{t+1}^T, a_{t+1}) = [\mu_1(x_{t+1}^T), \dots, \mu_D(x_{t+1}^T)]^\top$.

Our approach is summarised in Algorithm 4, which outlines the full procedure for computing SFs for each target task dimension by leveraging all source tasks. Lines 2-10 cover model fitting and SF reconstruction using GP models. Lines 21-22 describe how current observed features influence target SFs. To reduce Bellman backup overestimation bias, we use an ε -greedy algorithm.

We acknowledge that Algorithm 4 can be optimised. For instance, instead of performing the computations in Lines 14-20, the approximation of the target SF can be deferred to the next timestep. This allows the computed SF to be reused, eliminating redundant computation.

Additionally, Algorithm 3 describes the strategy for fitting the GP. This approach was used to accelerate convergence and minimizes the number of optimization steps by early stopping optimisation steps. This is critical in RL settings where the GP must be fitted at every timestep.

¹To ensure zero mean, we subtract the mean when fitting the GP, and restore it when doing the inference.

Algorithm 3 Gaussian Process Fitting with Gradient Descent.

Require: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, kernel hyperparameters (variance, lengthscale) $\theta = (\sigma_f^2, l)$, learning rate η , convergence threshold ϵ , max iterations T , MLL changes $\Delta = \{\}$ of length 10.

Ensure: Optimised hyperparameters θ

```

1: for  $t = 1$  to  $T$  do
2:   Compute initial  $K_*$ , using  $k_\theta(\mathbf{X}, \mathbf{X})$   $\triangleright \mathbf{X} = [x_1, \dots, x_N]^\top$ 
3:   Compute Marginal Log-likelihood (MLL)  $\mathcal{L}(\theta)$  using:

$$\mathcal{L}(\theta) = \left[ \frac{1}{2} \mathbf{y}^\top K_*^{-1} \mathbf{y} + \frac{1}{2} \log \det(K_*) + \frac{N}{2} \log(2\pi) \right]$$

 $\triangleright \mathbf{y} = [y_1, \dots, y_N]^\top$ 
4:   Update hyperparameters:  $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}(\theta)$ 
5:   if  $|\Delta| > 0$  then
6:     Compute change in MLL:  $\delta = |[\Delta]_{-1} - \mathcal{L}(\theta)|$   $\triangleright$  Using the last MLL change.
7:     Append  $\delta$  to  $\Delta$ 
8:   end if
9:   if  $|\Delta| = 10$  AND  $\text{mean}(\Delta) \leq \epsilon$  then
10:    break  $\triangleright$  Convergence criteria met
11:   end if
12: end for

```

4.3.4 Uncertainty-driven Generalised Policy Improvement

Beyond supporting tailored kernel functions for knowledge transfer, GP inherently quantifies model uncertainty. In particular, GP can predict epistemic uncertainty for a given input. This is achieved by comparing the input with previously observed data with the proposed kernel function. The kernel function encodes the similarity between the new input and the data used by the GP. As a result, the predicted epistemic uncertainty reflects how confident is the GP about the prediction of that input.

We harness this uncertainty for policy selection using a *pessimistic* interpretation of GPI, which maximises the lower confidence bound (Dimitrakakis & Ortner, 2022; Eysenbach & Levine, 2022; Jeong et al., 2023; Rashidinejad et al., 2021). This approach explicitly manages epistemic uncertainty within the GPI process. Thus, for a given target task weights $\mathbf{w}^T$, the GPI policy is defined as follows,

$$\pi^{\text{GPI}}(s) \in \arg \max_{a \in \mathcal{A}} \max_{S \in \mathcal{S}} (\psi^{\pi_S}(s, a) - \lambda \sigma^{\pi_S}(s, a))^\top \mathbf{w}^T, \quad (4.9)$$

Consequently, we propose that this guides the policy selection towards more robust and certain policy value estimations as implemented in Algorithm 4 Line 11.

Algorithm 4 Successor Features via GP and SF-based Dynamics Transfer Kernel.

Require: Source environments $\mathcal{D}^{S_1 \dots S_N}$ and target observations $\mathcal{D}^T = \{\}$, feature function $\phi(\cdot, \cdot) \in \mathbb{R}^D$, initial reward mapper weight for target environment $\mathbf{w}^T$, standard deviation bound λ .

Ensure: Optimal policy $\pi^*(s)$ for the target environment.

```

1: while  $t$  steps not terminated do
2:   for each  $S \in \{S_1, \dots, S_N\}$  do
3:     for each  $a' \in A$  do
4:        $x_t^T \leftarrow (s_t, a')$ 
5:        $\text{GP}_d^S(\{\mathcal{D}^S, \mathcal{D}^T\}), \forall d \in \{1, \dots, D\}$  ▷ Fit the GPs using Algorithm. 3
6:       Compute  $K_*, \mathbf{k}$  ▷ By (4.5)
7:        $\tilde{\psi}^{\pi^S}(x_t^T) = \mu_d^S(x_t^T), \forall d \in \{1, \dots, D\}$  ▷ By (4.6)
8:        $\sigma^{\pi^S}(x_t^T) = \sigma_d^S(x_t^T), \forall d \in \{1, \dots, D\}$  ▷ By (4.7)
9:     end for
10:   end for
11:   Compute  $\tilde{Q}^{\pi^{S_i}}(s_t, a') = (\tilde{\psi}^{\pi^{S_i}}(s_t, a') - \lambda \sigma)^\top \mathbf{w}^T, \forall a' \in A, \forall i \in \{1, \dots, N\}$  ▷ By (4.9)
12:    $\pi^*(s) = \arg \max_{a' \in A} \max_{S_i \in \{S_1, \dots, S_N\}} \tilde{Q}^{\pi^{S_i}}(x_t^T)$  ▷ GPI
13:   Observe immediate reward  $r(s, a)$ , next state  $s_{t+1}^T$ , and immediate features  $\phi_T$ , after following  $\pi^*(s)$ 
14:   for each  $S \in \{S_1, \dots, S_N\}$  do
15:     for each  $a' \in A$  do
16:        $x_{t+1}^T \leftarrow (s_{t+1}^T, a')$ 
17:       Compute  $K_*, \mathbf{k}$  ▷ Using (4.5)
18:        $\tilde{\psi}_d^{\pi^S}(x_{t+1}^T) = \mu_d^S(x_{t+1}^T), \forall d \in \{1, \dots, D\}$ 
19:     end for
20:   end for
21:    $\boldsymbol{\mu}(x_{t+1}^T) = \max_{a \in \mathcal{A}} \max_{S_i \in \{S_1, \dots, S_N\}} \tilde{\psi}^{S_i}(s_{t+1}^T, a)^\top \mathbf{w}^T$ 
22:    $\tilde{\psi}^T(x_t^T) = \phi_T + \gamma \boldsymbol{\mu}(x_{t+1}^T), \forall d \in \{1, \dots, D\}$  ▷ By (4.8)
23:   Add new observation  $r(s, a), \phi_T, \tilde{\psi}^T(x_t^T)$  to  $\mathcal{D}^T$ 
24:   Update  $\mathbf{w}^T, \min_{\mathbf{w}^T} \mathbb{E}_{\forall (s, a) \in \mathcal{D}^T} [(r(s, a) - \phi(s, a)^\top \mathbf{w}^T)^2]$ 
25: end while

```

4.4 Experiments and Analysis

In this section, we describe the metrics, environments, implemented baselines, and experimental results.

4.4.1 Experiment Setup

4.4.1.1 Metrics

We evaluated TRL algorithm performance using two established metrics: *time-to-threshold* and *mastery* (Zhu et al., 2023b). *Time-to-threshold* measures timesteps needed to reach a performance criterion; lower values indicate greater data efficiency and faster learning. *Mastery* quantifies peak performance on target tasks upon convergence, assessing how well prior knowledge is exploited.

4.4.1.2 Agents

We evaluated four distinct agents: SFDQN&GPI (Barreto et al., 2017), a foundational SF approach; GP (Abdolshah et al., 2021), which uses GP for SF transfer across dissimilar dynamics; h -GPI (Alegre et al., 2023a), an extension of GPI for multi-step planning via approximated dynamics; and finally, our proposed Off-diagonal agent.

Our experimental setup followed four phases:

1. **Feature function fitting:** A random policy collected full transition observations from source tasks. The feature function ϕ and weight vectors $\mathbf{w}$ were fitted by minimising the loss $\mathcal{L}_{\mathbf{w}}$. The ϕ function was then fixed for source and target tasks, with architectures detailed per environment. The weight vectors were discarded, consistent with Barreto et al. (2017).

2. **Source task optimal policies extraction:** With the ϕ function fixed, SF representations and the optimal policy were learnt for each source task.

Source task sample collection: After learning SFs in source tasks, 10^5 observations (states, actions, ϕ values, and SF values) were collected using an ϵ -greedy strategy.

3. **Target task adaptation and evaluation:** Agents adapted to target task observations for 10^3 timesteps. This phase balanced exploration-exploitation using an ϵ -greedy strategy, with ϵ of 0.5 linearly decaying to 0. Agents adapted their models accordingly: SFDQN&GPI updated SF representations; the GP agent the source SFs as noisy target observations; the h -GPI agent approximated dynamics and SF representations; and our agent employed its knowledge transfer mechanisms. After 10^3 timesteps, all agents transitioned to the **Evaluation** phase, stopping new observation incorporation.

We used a pre-defined shared feature function ϕ and optimal SF representations. This allowed us to isolate the SF transfer mechanism and assess its effectiveness on dissimilar tasks, without evaluating the quality of the source policies.

The hyperparameters used for the GP-based agents are reported in Table 4.1. The hyperparameters for the h -GPI agent are listed in Table 4.2. Finally, Table 4.3 presents the hyperparameters for the ϕ function, the SF models, and the SF-based agents.

Table 4.1: Gaussian process agents hyperparameters for Foraging and CartPole environments.

Hyperparameter	Value
Samples from source tasks	500
Foraging - GP Learning rate	10^{-2}
Cartpole - GP Learning rate	10^{-1}
Number of gradient updates	200
Source Noise value for GP baseline (Abdolshah et al., 2021) (σ_S^2)	0.1
Target Noise value for GP baseline (σ) (Abdolshah et al., 2021)	0.01

Table 4.2: h -GPI agent hyperparameters for Foraging environment.

Hyperparameter	Value
Dynamics train frequency (timesteps)	80
Dynamics rollout frequency (timesteps)	80
Dynamics rollout Len	1
Dynamics real samples ratio	0.5
Number of ensemble models	7
Holdout ratio	0.1
Ensemble architecture	[256,256], [256,256]
Number of elite ensemble models	5

Table 4.3: Successor features and ϕ model hyperparameters for Foraging and CartPole environments.

Hyperparameter	Value
ϕ and SF network architecture	[input $\times$ 4, input $\times$ 8], [input $\times$ 8, input $\times$ 4]
SF target network update freq. (timesteps)	10^3
Gamma (γ)	0.95
Batch size	64
Replay buffer size	10^5
Learning rate	10^{-3}
learning rate weight vectors	10^{-2}
Number of steps for adaptation phase	10^3
Initial ε exploration/exploitation	0.5
Final ε exploration/exploitation	0.0
Foraging - features dimension	2
CartPole - features dimension	32
CartPole - Early Stop at (episodes)	100

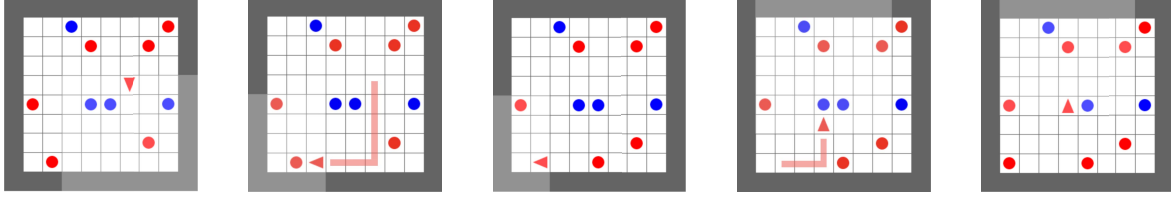


Figure 4.3: Foraging environment. Sequence of agent’s navigation across a 10×10 grid with blue and red objects. The agent (red triangle) navigates through the grid gathering objects. Each object represents different reward to the agent. The reward is given by the weight vector.

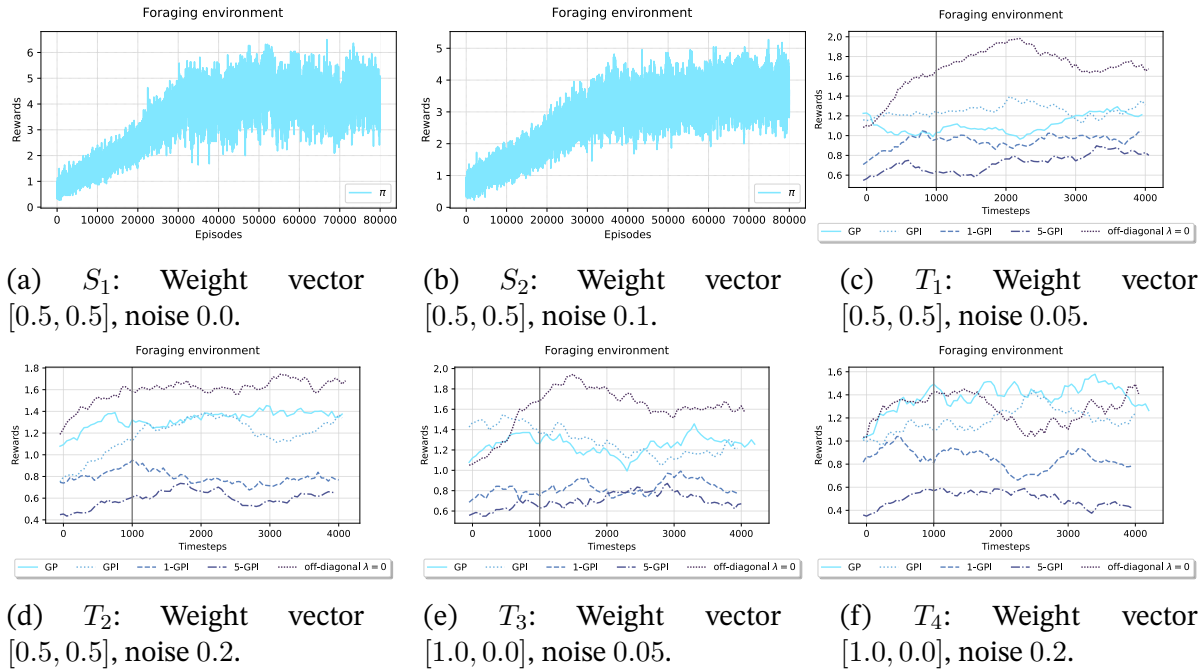


Figure 4.4: Foraging environment. Average reward over 10 random seeds. Figs. 4.4(a)-4.4(b) represent source tasks in foraging environment. Figs. 4.4(c)-4.4(f) represent the target tasks.

4.4.2 Experiment Results and Analysis

Agents were tested in *CartPole* and *foraging* environments, which are extensively used in SF literature to evaluate TRL performance across different dynamics.

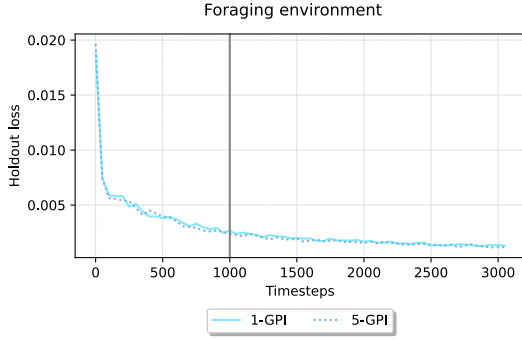
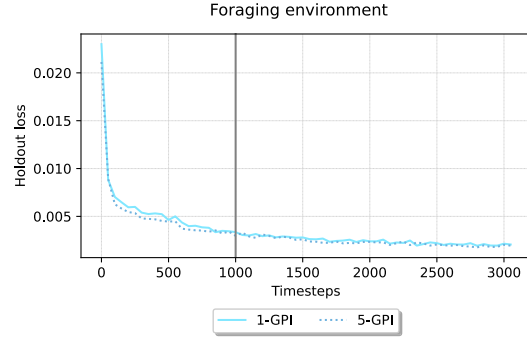
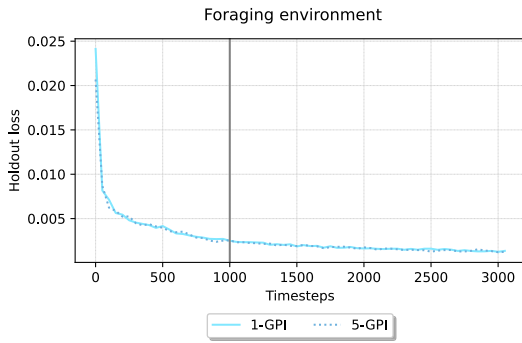
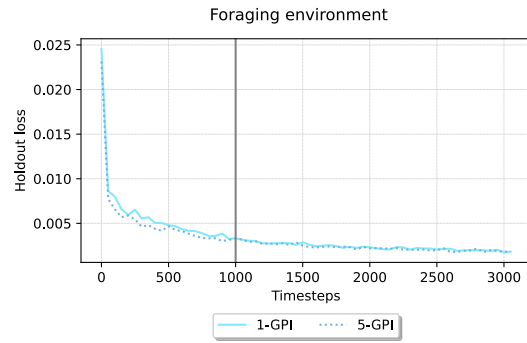
Foraging Originally introduced to SF literature by Barreto et al. (2020), it has been extensively used within the context of SFs (Borsa, Barreto, Quan, Mankowitz, et al., 2019; Carvalho et al., 2023). In this 10×10 grid-based task, an agent has three discrete actions (*rotate right*, *rotate left*, *move forward*) to move and gather two types of objects. The

agent receives an egocentric, image-like input formed by stacking 2, $7 \times 7 \times 3$ observations (Fig. 4.3) depicting object information. With respect to the observation space, the agents received a partial observation of the environment. This poses a challenge for DQN agents. To overcome this partial observability, we stacked observations and used them as input to a Convolutional Neural Network to produce the agent’s state, as implemented by Mnih et al. (2015). We used a stack size of 2, which worked well for our experiment. Each episode lasts 50 timesteps, with rewards granted upon gathering objects (agent occupies the same location as an object); tasks are defined by their reward weight vectors. New objects respawn at uniformly sampled locations and types.

To model dynamics dissimilarities, we introduced noise into the observations. Specifically, with a certain probability, the environment executed a random action, thereby introducing noise into the subsequent transition. It is crucial to note that this noise probability is distinct from the ε value used for the agent’s exploration-exploitation strategy, as the environment continuously introduced noisy observations irrespective of the agent’s policy. For the TRL setting, two source tasks were defined for the agent to learn to solve the problem. In both source tasks, the reward weight vector was set to $[0.5, 0.5]$, *i.e.*, the agent received 0.5 for either object type. For the first source task, no noise was added to the observations, whereas for the second, a noise of 0.1 was applied. For the target tasks, four settings were designed with the following reward weight vectors and noise probabilities: $T_1 = ([0.5, 0.5], 0.05)$, $T_2 = ([0.5, 0.5], 0.2)$, $T_3 = ([1.0, 0.0], 0.05)$, $T_4 = ([1.0, 0.0], 0.2)$.

For **feature function fitting**, 10^4 observations were collected per source task. A shared Convolutional Neural Network (CNN) with 3 hidden layers (16, 32, and 64 filters) transformed observations into a 128-dimensional state vector. All convolutional layers used a (2, 2) kernel and stride of 1. This state function was shared among source and target tasks, and its output fed the ϕ , SF, and GP models. The ϕ function was implemented as an Multi-layer Perceptron (MLP) with 2 hidden layers (512 and 1024 units), using Leaky Rectified Linear Unit (*LeakyReLU*) activation functions, and outputting a 2-dimensional feature vector.

For the **source task optimal policies extraction** phase, agents trained in the source tasks, fitting the SFs for 4×10^6 timesteps (80000 episodes), Fig. 4.4(a)-4.4(b). For exploration-exploitation, ε started at 1 and linearly decayed to 0 over 1.6×10^6 timesteps. The agent fitted both SFs and the source weight vector by minimising $\mathcal{L} = \mathcal{L}_\psi + \mathcal{L}_\phi$. Agents used $\gamma = 0.95$, a replay buffer of 10^4 observations and batch size of 64, a learning rate of 10^{-3} , the target network was updated every 10^3 steps. To fit the vector weight a learning rate of 10^{-2} was used. The SF-based agents used a Double DQN variant with experience replay.

(a) T_1 : Weight vector $[0.5, 0.5]$, noise 0.05.(b) T_2 : Weight vector $[0.5, 0.5]$, noise 0.2.(c) T_3 : Weight vector $[1.0, 0.0]$, noise 0.05.(d) T_4 : Weight vector $[1.0, 0.0]$, noise 0.2.Figure 4.5: Foraging environment. Holdout losses in target tasks for h -GPI agent.

The average rewards for the optimal source tasks are shown in Figs. 4.4(a) and 4.4(b). Then, the agent transitioned to the **source task sample collection** phase.

In the **target task adaptation and evaluation** phase, SF-based agents used the same hyperparameters as in source task fitting. The h -GPI agent approximated dynamics with an ensemble probabilistic model of 7 MLPs (Alegre et al., 2023a; K. Chua et al., 2018), planned with horizons of 1 (1-GPI) and 5 (5-GPI), and combined real and sampled observations. The ensemble model was fitted by minimising the negative log likelihood every 80 timesteps, using all available transition samples, and a learning rate of 3×10^{-4} . Convergence was monitored on 10% validation set, with early stopping if the improvement fell below 10^2 over 5 consecutive optimisation steps. The fitting process was monitored using the holdout loss, which is reported in Figs. 4.5(a), 4.5(b), 4.5(c), and 4.5(d). Each MLP within the ensemble consisted of 2 hidden layers, with 256 units, and LeakyReLU activation functions. The GPs were fitted (as detailed in Algorithm 3) using a learning rate of 10^{-2} , chosen after evaluating 10^{-1} and 10^{-2} for improved convergence and stable losses in this environment.

Evaluation results for agents on T_1 , T_2 , and T_3 , are shown in Figs. 4.4(c)-4.4(e). Our

approach significantly outperformed the other agents in the chosen metrics; that is, superior performance upon convergence (mastery) and fewer samples to reach it (time-to-threshold). While GP achieved stable performance in T_4 , Fig. 4.4(f), results indicate that our agent maintains competitive performance across all tasks except the most extreme dynamics changes, suggesting that the proposed kernel-based transfer mechanism is robust but may require additional modelling capacity under highly dissimilar dynamics.

Across all target tasks, our proposed Off-diagonal agent consistently outperformed classical SF-based baselines under dissimilar dynamics settings. In contrast, the h -GPI agent, despite incorporating planning and dynamics approximation, did not surpass GPI and showed degraded performance as the planning horizon increased (e.g., $h = 5$ vs. $h = 1$). We hypothesise that h -GPI suffered from compound error during multi-step planning, likely due to inaccuracies in approximating feature functions and weight vectors rather than errors in the learnt dynamics model itself, as supported by the holdout losses (Figs. 4.5(a)-4.5(d)). Unlike h -GPI, our approach does not rely on explicit environment model approximation. Instead, it incorporates transition dissimilarities directly within the successor feature representation through the kernel-based GP update mechanism. While standard SF-based agents adapted with approximately 10^3 samples in similar environments, this adaptation proved insufficient under stronger dynamics variations. In contrast, our method maintained robust transfer performance, demonstrating its ability to leverage knowledge across dissimilar dynamics and reward structures.

CartPole This environment, widely used for classical control (Sutton & Barto, 2018) and evaluating SF transfer across dissimilar dynamics (Abdolshah et al., 2021; Garces, Kiyoshige et al., 2023), involves an agent balancing a pole using 2 discrete actions (*left* or *right*). The agent receives a reward each timestep the pole remains upright, with the task solved after 200 consecutive steps. In our TRL setup, the source task used a $0.5m$ pole and the target a $3m$ pole (Fig. 4.6(a)). Pole length differences introduce dissimilar transition dynamics due to distinct physical properties. Agents received a 4-value observation vector (cart position, cart velocity, pole angle, pole velocity) as state. That is, the agent received a 4-dimensional vector instead of an image-like observation. The feature function dimension was 32. For the SF-based agents, the Double DQN agent presented undesired behaviours and did not converge in the selected seeds. Using a single DQN, the source policies converged as shown in Fig. 4.6(b).

An example of the source and target tasks is shown in Fig. 4.6(a). The performance of an optimal policy for the source task is presented in Fig. 4.6(b). We selected the optimal

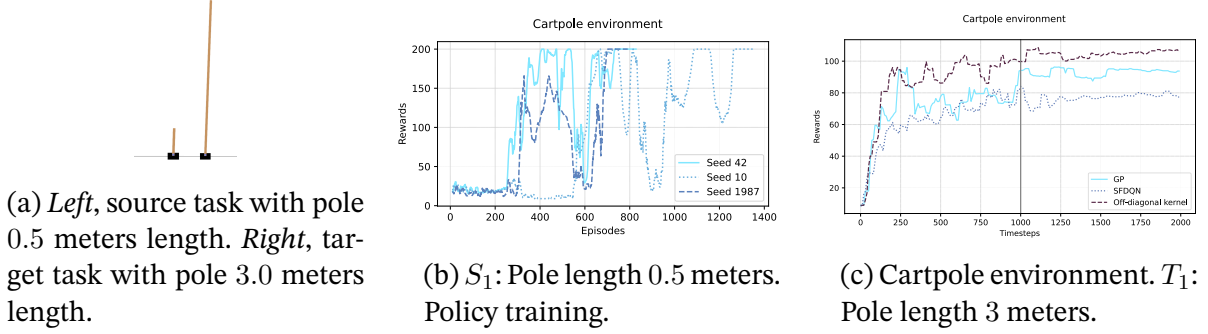


Figure 4.6: CartPole environment. Results are averaged over 10 random seeds, and a vertical line indicates the end of the adaptation phase.

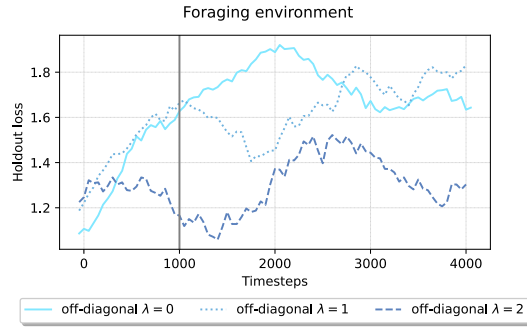
policy found from seed 1987. This seed was chosen because our experimental setup collected samples from the optimal policy, and this specific policy had less noisy suboptimal data compared to the others.

In the **Feature function fitting** phase, ϕ was implemented as an MLP with two hidden layers using LeakyReLU activations. The input layer had dimension 10 (concatenated states and one-hot encoded actions), followed by hidden layers with 16 and 32 units. The output layer matched the 32-dimensional feature vector.

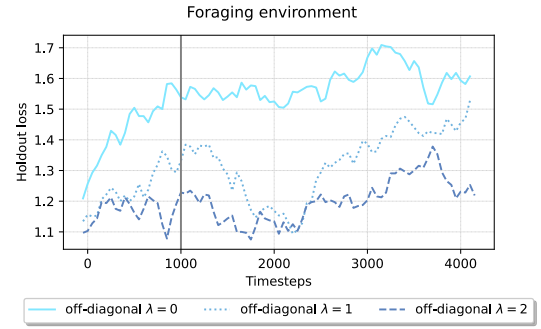
For **source task optimal policies extraction**, the SF model was implemented as an MLP. This network had a 4-dimensional input layer, 2 hidden layers of 16 and 32 units respectively, and LeakyReLU activation functions. The output layer had a dimension of 32×2 representing the feature dimension multiplied by the number of actions. Agent stopped training once it had solved the task for 100 consecutive episodes. The agent then transitioned to the **source task sample collection** phase. The remaining hyperparameters used were similar to those of the foraging environment.

For **target task adaptation and evaluation**, all agents were provided with the source SF model, the ϕ function, and source observations. They then collected target observations using an ϵ -greedy strategy. SF-based agents used the same hyperparameters as in source task fitting. GP and our approach used GP fitting with a learning rate of 0.1. Each agent was executed 10 times using different random seeds.

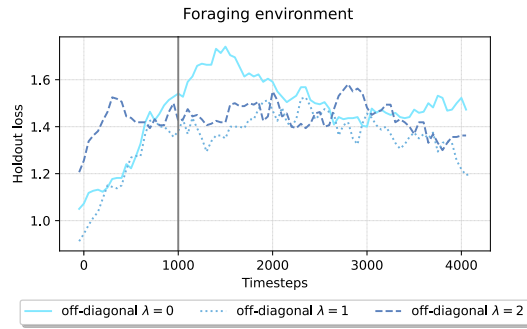
Results in Fig. 4.6(c) show that our off-diagonal kernel-based agent outperforms all baseline agents in both time-to-threshold and mastery metrics in the CartPole environment. In contrast, the SFDQN agent with adapted observations achieves substantially lower performance, consistent with findings reported by Abdolshah et al. (2021), where GP-based approaches outperformed SFDQN under diverse dynamics. These results highlight the advantage of modelling cross-task correlations through the structured covari-



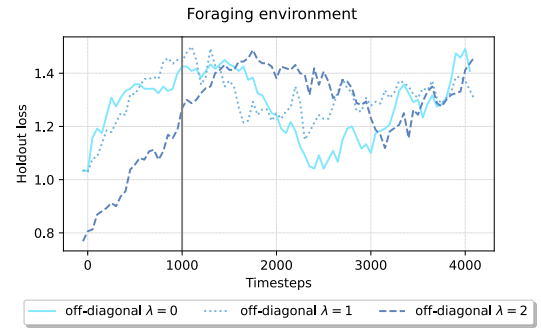
(a) Foraging environment. T_1 : Weight vector $[0.5, 0.5]$, noise 0.05.



(b) Foraging environment. T_2 : Weight vector $[0.5, 0.5]$, noise 0.2.



(c) Foraging environment. T_3 : Weight vector $[1.0, 0.0]$, noise 0.05.



(d) Foraging environment. T_4 : Weight vector $[1.0, 0.0]$, noise 0.2.

Figure 4.7: Evaluation of uncertainty-based policy selection. Results are averaged over 10 random seeds, and a vertical line indicates the end of the adaptation phase.

ance matrix. By explicitly capturing relationships between source and target tasks, our approach enables more effective knowledge transfer than methods relying solely on feature adaptation. The flexibility of the kernel design, combined with the GP update mechanism, provides a principled way to leverage prior knowledge and generalise to new tasks under diverse transition dynamics.

Uncertainty-driven Policy Selection Study In addition to comparing against other SF-based agents, we evaluated our uncertainty-driven GPI approach using the same foraging environment setting. Figs. 4.7(a) and 4.7(b) show the results, suggesting that this novel uncertainty-driven GPI adds instability to the transfer process, as evidenced by the increased instability with rising λ . While this initial method attempts to manage epistemic uncertainty, the mixed experimental results suggest that further research is needed.

4.5 Summary

In this chapter, we introduced a novel Gaussian Process (GP)-based framework with a tailored kernel, designed to capture input relationships, incorporate epistemic uncertainty into policy selection, and enable successor feature (SF) transfer across dissimilar dynamics.

Our SF-based Dynamics Transfer Kernel extends the squared exponential kernel, although theoretical results suggest that the approach is applicable to any valid kernel. Future extensions could explore alternatives such as the Wasserstein distance, for which positive semi-definite kernel conditions are known (Candelieri et al., 2025; Plaen et al., 2020). While our current formulation employs a scalar GP, we hypothesise that the task kernel in (4.2) could support a multi-output GP (Álvarez et al., 2010), thereby eliminating the need for separate models per feature and leveraging inter-feature dependencies. Another key direction is the development of scalable approximations to reduce the inherent cubic complexity of GPs, enabling application to larger datasets and more complex problems.

The proposed kernel and the method for approximating target SFs from the Bellman optimality equation present two main limitations: (i) susceptibility to overestimation bias, and (ii) vulnerability to error compounding caused by GP approximation, as described in (4.8). Despite these limitations, our empirical results demonstrate that agent performance converges.

A notable advantage of GPs is their capacity to handle noisy measurements during training, a property embedded in the kernel functions and the covariance matrix diagonal. From this perspective, the prediction errors in GPs can be interpreted as noisy measurements. Nonetheless, an important avenue for future research is to establish a theoretical foundation explaining why such noise does not propagate through error compounding to degrade agent performance. In our framework, the approximation of the feature function introduces errors that can affect the agent’s ability to recover optimal policies. Because our kernel and metric compare observed features, errors in the feature representation can influence the kernel behaviour and, consequently, the SF approximation. However, this limitation is not unique to our method, as inaccuracies in feature representations also affect other algorithms based on SF. Therefore, a central challenge in SFs remains the design or learning of expressive feature functions that accurately capture the underlying dynamics (Ollivier, 2025; Touati et al., 2023).

Our uncertainty-driven GPI achieved mixed transfer performance. Although this initial integration of epistemic uncertainty into GPI proved challenging, it highlights an im-

portant research direction for more effective uncertainty modelling in this framework.

In conclusion, the proposed GP-based framework advances the transfer of SF to scenarios with dissimilar dynamics, addressing a fundamental limitation of existing SF-based methods.

ADAPTIVE SUCCESSOR FEATURES VIA HIERARCHICAL BEHAVIOUR COMPOSITION

*Divide each difficulty into as many parts as is
feasible and necessary to resolve it.*

René Descartes, Discourse on Method

This chapter addresses **RQ-3** and **RO-5**. We interpret Successor Features (SFs) both as representations that encapsulate specific behaviours, which can be combined or composed to generate new ones, and as a knowledge representation mechanism that captures both current and potential successor states. We leverage these perspectives to extract and compose behaviours for solving novel, unseen tasks. Building on these concepts, we propose a two-level Markov Decision Process (MDP) that enables the composition of SFs. Within this framework, SFs act as both states and actions, while an accompanying algorithm orchestrates their composition to solve new target tasks.

5.1 Background and Motivation

Reinforcement Learning (RL) has demonstrated remarkable results in domains that challenge human cognitive skills, such as complex games and decision-making problems (Kober et al., 2013; Mnih et al., 2015). However, classical RL requires a substantial number of samples to achieve desirable behaviour for a single task (Lazaric, 2012; Taylor & Stone, 2009). Consider a robot that must pick apples, oranges, or both depending on the season, with

each season being a different task. Using classical RL forces the robot to relearn each task, requiring millions of samples and making this approach resource-intensive and impractical. Ideally, we expect the robot to use previously acquired skills to solve new unseen tasks more efficiently.

To address sample inefficiency when solving new tasks, a significant body of work has introduced multi-task learning in RL (Caruana, 1997). Multi-task RL aims to solve a finite set of tasks simultaneously, thereby reducing the resources required to train each task independently. In RL, these solutions have demonstrated outstanding results in complex problems such as control and scheduling (Cheng et al., 2023; D’Eramo et al., 2020; Y. T. Liu & Ahmad, 2024; Sodhani et al., 2021; J. Zhang, Guo, et al., 2024). However, multi-task RL is beneficial only when all the tasks are simultaneously accessible and fixed. On the other hand, another line of research, meta-RL, assumes that tasks are drawn from a probability distribution, aiming to learn how to quickly adapt to or solve tasks within that distribution (Beck et al., 2025; Duan et al., 2016; Finn et al., 2017).

To formalise the problem of reusing previous knowledge, Taylor and Stone (2009) defined a framework for *Transfer Reinforcement Learning (TRL)*. The aim of this more general framework is to answer *what* knowledge and *how* to transfer it between different sets of tasks (Wulfmeier et al., 2023; Zhu et al., 2023b).

Transfer Reinforcement Learning (TRL) uses the human-like ability to reuse previously learnt knowledge to solve new tasks (Zhu et al., 2023b). TRL defines two sets of tasks. The first, known as *source tasks*, is used to find optimal behaviours, which are then reused in the second set, or *target tasks*, to improve performance or reduce sample requirements. For example, source tasks might involve picking only apples or oranges in different seasons, while the target task might require picking both. To answer *what* knowledge to transfer, widely studied approaches fall under *representation transfer*, with a significant body of work achieving this through successor representations (Dayan, 1993). Based on these, recent research has addressed *how* to reuse knowledge through *knowledge composition* (Wulfmeier et al., 2023). This chapter delves into a specific and powerful framework that combines these aspects: *successor features (SFs)* and *generalised policy improvement (GPI)*.

SFs and Generalised policy improvement (GPI) form a framework for TRL on tasks with different reward functions. SF is a behaviour representation that generalises successor representations. Furthermore, it assumes that reward functions have a linear relationship between a common feature function and task-specific vector weights. These vector weights can be seen as preferences over features and help distinguish tasks. GPI

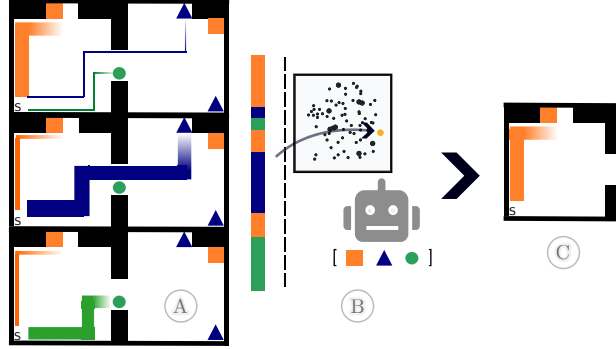


Figure 5.1: Overview of proposed SF composition. (A) Source task policies. (B) High-level state space. (C) Composed policy.

enables updating and composing SFs; *e.g.*, in the fruit-harvesting task, different seasons can be modelled by varying preferences for apples or oranges, allowing SF-GPI to learn behaviours per season and combine them to handle new ones without retraining. GPI composes SFs using a two-level decision process, which first selects an SF and then follows its behaviours at each timestep. However, it has been proven that the performance of SF-GPI’s is related to the similarity between the vector weights of the source and target tasks, resulting in a tight upper bound on performance (Barreto et al., 2017; Nemecek & Parr, 2021). To address this, recent works have focused on dynamically constructing a set of basis behaviours that, when composed by GPI, ensure optimal performance on target tasks (Alegre et al., 2022; Alver & Precup, 2022; Nemecek & Parr, 2021; Zahavy et al., 2022). However, these methods require substantial computational resources and additional sample complexity. Recent works have also been built upon GPI to exploit the compositionality properties of SF-GPI (Barreto et al., 2019; Carvalho et al., 2023; Hunt et al., 2019; Y. T. Liu & Ahmad, 2024). In contrast, our approach focuses on learning a fixed number of SFs and composing them to solve new unseen tasks, without relying on GPI.

We propose a method for deriving and composing SFs within a two-level decision framework, modelled by a high-level Markov decision process with novel SF-based states and actions. As illustrated in Fig. 5.1, the agent evaluates from a set of possible behaviours (A), analyses in SF-based high-level state, and adaptively selects the proper SF (B) to commit to for an extended period, with the aim of solving the task (C). This approach leverages source task behaviours to solve unseen target tasks, showing steady performance across varying task distances and consistently outperforming baselines in three extensively used environments in SF literature: *Four-room*, *Reacher*, *Foraging*.

To summarise, our work makes the following contributions:

1. A novel high-level Markov decision process with high-level state and action spaces using SFs to support decision-making and leverage prior knowledge for unseen tasks.
2. A novel algorithm for learning and composing SFs from source tasks, considering them as composable behaviours. 3) Empirical results showing that while the SF-GPI performance degrades, our method remains steady across tasks with varying distances.

5.2 Preliminaries and Problem Formulation

We begin by restating that Reinforcement learning (RL) problems are commonly modelled by a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. We consider settings where tasks share transition dynamics and a feature function $\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$. The reward for the i -th task is defined as $r_{\mathbf{w}_i}(s_t, a_t, s_{t+1}) = \phi(s_t, a_t, s_{t+1})^\top \mathbf{w}_i$, where $\mathbf{w}_i \in \mathbb{R}^d$ is a preference vector in the simplex (*i.e.*, its entries sum up to 1). This setting has been widely studied by Alegre et al. (2022), Nemecek and Parr (2021), and Zahavy et al. (2021, 2022).

In TRL, a set of n source tasks is defined as $\mathbf{W}_s = \{\mathbf{w}_i\}_{i=0}^{n-1}$ and m target tasks as $\mathbf{W}_\tau = \{\mathbf{w}_j\}_{j=0}^{m-1}$, with each task's reward function given by $r(s_t, a_t, s_{t+1}) = \phi(s_t, a_t, s_{t+1})^\top \mathbf{w}$. We are interested in TRL problems in which the objective is to compose source policies to solve target tasks without additional training (Zhu et al., 2023b). In this chapter, we refer to tasks directly using their associated weight vectors. Additionally, the terms *behaviours* and *successor features (SFs)* are used interchangeably, as SFs represent behaviours induced by a *policy*.

We denote the SFs learnt in the i -th source task as $\psi^{\pi_{\mathbf{w}_i}}$, where $\pi_{\mathbf{w}_i}$ is the policy optimised for task $\mathbf{w}_i$. This forms a set of task-specific SFs for all source tasks $\Psi = \{\psi^{\pi_{\mathbf{w}_i}}\}_{i=0}^{n-1}$. To take advantage of these SFs, Barreto et al. (2017) propose generalised policy improvement, which enables their composition to derive policies for the target tasks.

As mentioned previously, GPI is a generalisation of the processes of *policy improvement* and *policy evaluation*. Instead of evaluating one policy at a time, GPI considers multiple policies. It assumes access to a set of source tasks, $\mathbf{W}_s$, and their corresponding SF, $\Psi = \{\psi^{\pi_{\mathbf{w}_i}}\}_{i=0}^{n-1}$. GPI enables the computation of an improved policy as,

$$\pi^{\text{GPI}}(s; w) \in \arg \max_{a \in \mathcal{A}} \max_{\psi \in \Psi} \psi(s, a)^\top \mathbf{w}. \quad (5.1)$$

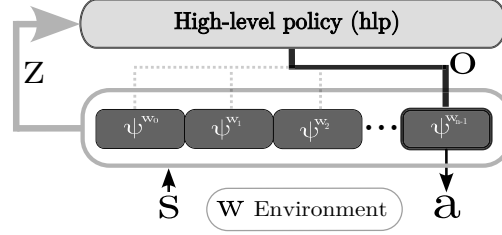


Figure 5.2: High-level SF composition for TRL. Z and o denote the high-level state and action, s and a the low-level state and action, and w the current task.

SF-GPI serves as two functions: it updates multiple SFs simultaneously using task-specific samples and the GPI-selected policy in (5.1), and acts as a hierarchical solution by composing SFs at each timestep (Barreto et al., 2020).

Importantly, Barreto et al. (2017) derived a theorem demonstrating that $\pi^{\text{GPI}}(s; w)$ performs at least as well as any other policy in the set. The theorem states that the performance gap between the optimal policy for a source task and the GPI policy for a target task is bounded by the difference in task weights and the feature function: $Q_{\mathbf{w}_j}^{\pi^{\text{GPI}}}(s, a) - Q_{\mathbf{w}_j}^{\pi^*}(s, a) \leq \frac{2}{1-\gamma}(\max_{s,a} \|\phi(s, a)\| \min_i \|\mathbf{w}_j - \mathbf{w}_i\|)$. This implies that GPI performs better when the source and target weights are closer. Moreover, Nemecek and Parr (2021) and S. Zhang et al. (2024) have demonstrated the tightness between this relationship.

5.3 Method

Our goal is to achieve TRL through adaptive composition of SFs by learning optimal SFs for source tasks $\mathbf{W}_s$ and composing them to create behaviours for target tasks $\mathbf{W}_\tau$ without additional training. One solution is a two-level decision hierarchy. At the first level, a *high-level policy* selects an optimal policy from a set of source policies, and at the second level, the *low-level policy* chooses the low-level action (Shah et al., 2022). GPI is a known mechanism for composing SFs that uses a two-level decision hierarchy at each timestep (Barreto et al., 2017). Rather than making high-level decisions every timestep, we present a novel high-level policy capable of acting over extended intervals. This policy, shown in Fig. 5.2, uses SFs from source tasks to form a high-level state Z and selects high-level action o . Once a high-level action is chosen, it decides the low-level action a (Fig. 5.2) to use in the environment. Building on this, we propose a high-level Markov decision process that allows solving source tasks and composing them to achieve TRL.

5.3.1 High-Level Markov Decision Process

By using SFs as actions or as we named high-level actions, we model our problem with the following MDP. This high-level MDP is defined by the tuple $(\mathcal{X}, \mathcal{O}, p_{\text{hlp}}, r_{\text{hlp}}, \gamma_{\text{hlp}})$, where $\mathcal{X}$ represents the set of high-level states, $\mathcal{O}$ the set of high-level actions, and $p_{\text{hlp}}(z_{t+1}|z_t, o_t)$ the probability distribution of reaching the high-level state z_{t+1} given a high-level state $z_t \in \mathcal{X}$, after following the high-level action, $o_t \in \mathcal{O}$. The high-level reward, r_{hlp} , is computed based on the shared function ϕ and $\mathbf{W}_s$, and γ_{hlp} is the high-level discount factor. Solving the proposed MDP enables learning SFs from the source tasks and adapting those behaviours to the target tasks.

5.3.2 High-Level Actions with Successor Features

The proposed MDP naturally aligns with the *options* framework (Sutton et al., 1999) and hierarchical RL (Hutsebaut-Buysse et al., 2022; Nachum et al., 2018), where an *option* or high-level action includes a termination function to decide whether the option should continue. Building on this concept, we propose a termination function β that depends on the observed features and a predefined maximum horizon H , representing the number of timesteps following a policy. Formally, β is defined as:

Definition 5.1 (Termination function β). Given a horizon for a high-level action H ; the observed features of a transition, $\phi(s_t, a_t, s_{t+1}) \in \mathbb{R}^d$; the current number of timesteps executed by a low-level policy, h ; and a threshold value φ to determine whether there are any observed features of interest, the function $\beta \in \{0, 1\}$ is defined by,

$$\beta(\phi(s_t, a_t, s_{t+1}), h) = \begin{cases} 1 & \text{if } \max(\phi(s_t, a_t, s_{t+1})) \geq \varphi \\ & \text{or } h > H \\ 0 & \text{otherwise} \end{cases}. \quad (5.2)$$

This function β serves for the most commonly used feature functions in the SFs literature (Alegre et al., 2022; Barreto et al., 2017, 2018; Barreto et al., 2020; Zahavy et al., 2021, 2022). The feature function $\phi_{\text{goal}}(s_t, a_t, s_{t+1}) \in \mathbb{R}^d$, represents d object types, setting 1 in the i -th dimension when object i is observed. In contrast, $\phi_{\text{distance}}(s_t, a_t, s_{t+1}) \in \mathbb{R}^d$, encodes inverse distances to the d objects, approaching 1 when the agent is near the object. Our β function works with both and introduces a tolerance parameter φ . With $\varphi = 1$, β returns 1 when an object is detected (in ϕ_{goal}), or closed (in ϕ_{distance}), or if the horizon H is reached. Lowering φ (e.g., to 0.98) adds tolerance to ϕ_{distance} without affecting ϕ_{goal} . This

flexibility enables the β function to generalise to other scenarios by linking the values of the feature function to the policy steps used to derive high-level actions.

Definition 5.2 (High-level action). Given an SF for the i -th source task $\psi^{\pi_{\mathbf{w}_i}}$ and β function, the high-level action o_i is defined by the tuple $o_i \equiv (\psi^{\pi_{\mathbf{w}_i}}, \beta)$

This builds the set of high-level actions $\mathcal{O} = \{o_i \mid o_i = (\psi^{\pi_{\mathbf{w}_i}}, \beta)\}_{i=0}^{n-1}$. We now define high-level states from SFs.

5.3.3 High-Level States with Successor Features

Although SFs are defined as action-state functions, the representation of high-level states requires rewriting action-state as state functions. This transformation is straightforward due to the relationship between the SF and the value functions. Specifically, the SFs state function is defined as

Definition 5.3 (SFs state function). Given a state s_t and an action a_t , the SF for a policy $\pi_{\mathbf{w}_i}$ is computed as $\psi^{\pi_{\mathbf{w}_i}}(s_t, a_t)$. Then, the SF state function is defined by,

$$\psi^{\pi_{\mathbf{w}_i}}(s_t) = \mathbb{E}_{a_t \sim \pi_{\mathbf{w}_i}(s_t)} [\psi^{\pi_{\mathbf{w}_i}}(s_t, a_t)]$$

Alternative approaches can be used to estimate the SFs state function and the action-state value function, such as the duelling architecture (Z. Wang et al., 2016). Instead, for discrete actions, we propose to approximate the SFs state function as

$$\psi^{\pi_{\mathbf{w}_i}}(s_t) \approx \frac{1}{|\mathcal{A}|} \sum_{a_t \in \mathcal{A}} \psi^{\pi_{\mathbf{w}_i}}(s_t, a_t). \quad (5.3)$$

The SF state function provides relevant information about expected behaviours by representing the expected discounted accumulated features for a given state s_t under policy $\pi_{\mathbf{w}_i}$. Our idea is to use that information as part of the high-level state. Therefore, we define high-level states as vectors that capture the expected behaviours of the source tasks.

Definition 5.4 (High-Level State). Given the n source tasks $\mathbf{W}_s = \{\mathbf{w}_i\}_{i=0}^{n-1}$ and their corresponding SFs $\{\psi^{\pi_{\mathbf{w}_i}}\}_{i=0}^{n-1}$, for each $\psi^{\pi_{\mathbf{w}_i}}(s_t) \in \mathbb{R}^d$, we build $Z_t \in \mathbb{R}^{d \times n}$ for state s_t as,

$$Z_t = [\psi^{\pi_{\mathbf{w}_0}}(s_t)^\top, \psi^{\pi_{\mathbf{w}_1}}(s_t)^\top, \dots, \psi^{\pi_{\mathbf{w}_{n-1}}}(s_t)^\top]^\top$$

5.3.4 High-Level Reward

For the high-level reward, r_{hlp} , there are two possible definitions. A hand-designed reward that serves as an intrinsic reward (Kulkarni, Narasimhan, et al., 2016), or based on the rewards of the source tasks. We opted for the latter to avoid making additional assumptions, such as additional source tasks, making it more suitable for TRL. Given that the high-level action is executed until a maximum horizon, H , and taking into account the source tasks, we propose that the high-level reward for a horizon $h \leq H$ is computed as $r_{\text{hlp}}^h = \mathbb{E}_{w \sim \mathcal{U}(\mathbf{W}_s)} \left[\sum_{k=t}^h \gamma^k \phi(s_k, a_k, s_{k+1})^\top \mathbf{w} \right]$, where t is the initial timestep of the high-level action, h the timestep when it ends, and $\mathbf{w}$ is uniformly sampled from the set of source tasks $\mathbf{W}_s$. This reward effectively relates the source tasks to multi-step execution of the high-level action.

With the concepts of high-level action, state, and reward, we now turn to explain how to solve the high-level MDP.

5.3.5 High-Level Policy Learning and Knowledge Transfer

To solve the high-level MDP. We propose learning a Q -value function to derive the high-level policy, $Q^{\text{hlp}}(Z_t, o_t)$, where Z_t is the high-level state introduced in Definition 5.4, and $o_t \in \mathbb{O}$ the high-level action, Definition 5.2. Therefore, $Q^{\text{hlp}}(Z_t, o_t)$ can be learnt by the multi-step temporal difference defined by,

$$\mathcal{L}_{Q^{\text{hlp}}} = || r_{\text{hlp}}^h + \gamma_{\text{hlp}}^h \max_{o' \in \mathbb{O}} Q^{\text{hlp}}(Z_{t+h}, o') - Q^{\text{hlp}}(Z_t, o_t) || . \quad (5.4)$$

This high-level policy adaptively selects high-level actions for given high-level states and implicitly composes SFs. To summarise our proposed solution, we detailed Algorithm 5, a value-based algorithm which employs an ε -greedy strategy. The high-level policy computes the high-level state and selects a greedy high-level action with probability ε_o , that is, $\varepsilon_o \in [0, 1]$, (lines 6-8). Once the high-level action is selected, this encapsulates the SFs for the source task $\mathbf{w}$ (lines 8-9), which can be used to derive a low-level policy for the task $\mathbf{w}_{\text{hlp}}$ by choosing actions greedily with a probability $\varepsilon_a \in [0, 1]$, (lines 11-12). For the high-level policy, we use the multi-step temporal difference, *i.e.*, we use the initial and the last state after executing the low-level policy (lines 18-22).

It should be noted that while interacting with the environments, the agent collects a history of transitions with a maximum length of H , which is defined in β , resulting in a sequence $(s_0, a_0, s_1, a_1, s_2, \dots, s_{h \leq H})$. These experiences are useful in fitting all models, as

Algorithm 5 Extract and compose SF via high-level policy.

Require: Source tasks $\mathbf{W}_s$, timesteps, Low-level actions $\mathcal{A}$, horizon H , β function, epsilon for high-level policy ε_o , epsilon for low-level policies ε_a , swap_every_n_episodes, Uniform distribution $\mathcal{U}$.

Ensure: High-level policy Q^{hlp} and high-level actions $\mathcal{G} = \{\psi^{\pi_{\mathbf{w}_i}}, \beta\}_{i=0}^{|\mathbf{W}_s|-1}$.

- 1: Initialize Q^{hlp} , high-level actions $\mathcal{G}$ models, high-level policy buffer, B , temporal buffer with length H , B_H , low-level policy buffers for $\forall \psi^{\pi_{\mathbf{w}_i}} \in \mathcal{G}$, $B_{\mathbf{w}_i}$
- 2: Select $\mathbf{w}_{\text{hlp}}$ from $\mathbf{W}_s$ and set its environment to env_{hlp} .
- 3: $s_0 \leftarrow$ Initial state from env_{hlp}
- 4: **for** t from 0 to timesteps **do**
- 5: **if** $\psi^{\pi_{\mathbf{w}}}$ is unset **then**
- 6: $Z_t \leftarrow [\psi^{\pi_{\mathbf{w}_0}}(s_t)^\top, \dots, \psi^{\pi_{\mathbf{w}_{n-1}}}(s_t)^\top]^\top$ ▷ Def. 5.4
- 7: **if** Bernulli(ε_o) **then** $(\psi^{\pi_{\mathbf{w}}}, \beta) \leftarrow \mathcal{U}(\mathcal{G})$ ▷ ε -greedy
- 8: **else** $(\psi^{\pi_{\mathbf{w}}}, \beta) \leftarrow \arg \max_{o \in \mathcal{G}} Q^{\text{hlp}}(Z_t, o)$
- 9: $h_{\mathbf{w}} \leftarrow 0$, $\mathbf{w} \leftarrow$ task associated to $\psi^{\pi_{\mathbf{w}}}$
- 10: **end if**
- 11: **if** Bernulli(ε_a) **then** $a_t \leftarrow \mathcal{U}(\mathcal{A})$
- 12: **else** $a_t \leftarrow \arg \max_a \psi^{\pi_{\mathbf{w}}}(s_t, a)^\top \mathbf{w}_{\text{hlp}}$ ▷ ε -greedy
- 13: Execute a in env_{hlp} and observe s_{t+1} and $\phi(s_t, a_t, s_{t+1})$
- 14: Store $s_t, \mathbf{w}, s_{t+1}, \phi(s_t, a_t, s_{t+1})$ to B_h .
- 15: Store $s_t, a_t, s_{t+1}, \phi(s_t, a_t, s_{t+1})$ to $B_{\mathbf{w}}$.
- 16: $h_{\mathbf{w}} \leftarrow h_{\mathbf{w}} + 1$ ▷ Increase by one timestep
- 17: Fit $\psi^{\pi_{\mathbf{w}}}$ with new experience. Using Temporal Difference (TD) Eq. (2.4).
- 18: **if** $\beta(\phi_{t+1}, h_{\mathbf{w}}) = 1$ **then** ▷ Def. 5.1
- 19: Compute discounted rewards from B_h with initial r_0 and final $r_{h_{\mathbf{w}}}$, as $r = \gamma^0 \phi(s_0, a_0, s_1)^\top \mathbf{w}_{\text{hlp}} + \dots + \gamma^{h_{\mathbf{w}}} \phi(s_{h-1}, a_{h-1}, s_h)^\top \mathbf{w}_{\text{hlp}}$
- 20: Store in B : first state in B_h , s_0 , latest state, $s_{h_{\mathbf{w}}}$, and r .
- 21: $\mathbf{w} \leftarrow$ unset
- 22: **end if**
- 23: **if** episode ends **then**
- 24: **if** swap_every_n_episodes **then** $\mathbf{w}_{\text{hlp}} \leftarrow \mathcal{U}(\mathbf{W}_s)$ and set new env to env_{hlp}
- 25: $\mathbf{w} \leftarrow$ unset
- 26: $s_{t+1} \leftarrow$ initial observation from env_{hlp}
- 27: **end if**
- 28: Fit Q^{hlp} minimising loss in (5.4) using samples from B .
- 29: **end for**

shown in lines 17 and 28. Lastly, the strategy to select the next source task to be used in order to collect experiences is outlined in the Algorithm. 5 in lines 24-27.

We posit that the high-level policy not only composes SFs from source tasks but also guides the agent towards meaningful experiences through the high-level reward, aligning with GPI as a mechanism for generalised policy updates.

To achieve TRL through the composition of SF, Algorithm 6 outlines the process for a given target task. It first exploits the relationship between high-level states and actions (lines 6–8), then derives a low-level policy using SFs. This mechanism composes behaviours from source tasks (line 9), enabling efficient adaptation to new tasks without prior experience.

Algorithm 6 Knowledge Transfer via high-level policy.

Require: Target task $\mathbf{w}_j \in \mathbf{W}_\tau$, environment env_j , High-level policy Q^{hlp} , high-level actions $\mathcal{O} = \{(\psi^{\pi_{\mathbf{w}_i}}(s, a), \beta)\}_{i=0}^{|\mathbf{W}_s|-1}$

Ensure: Accumulated reward r

- 1: $\psi^{\pi_{\mathbf{w}}} \leftarrow \text{unset}$
- 2: $s_t \leftarrow \text{initial state from } \text{env}_j, t \leftarrow 0, r \leftarrow 0$
- 3: **while not done do**
- 4: **if** $\psi^{\pi_{\mathbf{w}}}$ is unset **then**
- 5: $Z_t \leftarrow [\psi^{\pi_{\mathbf{w}_0}}(s_t)^\top, \dots, \psi^{\pi_{\mathbf{w}_{n-1}}}(s_t)^\top]^\top$ ▷ By (5.3)
- 6: $h_{\mathbf{w}} \leftarrow 0$
- 7: $(\psi^{\pi_{\mathbf{w}}}, \beta) \leftarrow \arg \max_{o \in \mathcal{O}} Q^{\text{hlp}}(Z_t, o)$
- 8: **end if**
- 9: $a_t \leftarrow \arg \max_a \psi^{\pi_{\mathbf{w}}}(s_t, a)^\top \mathbf{w}_j$
- 10: Execute a_t in env_j and observe $s_{t+1}, \phi(s_t, a_t, s_{t+1})$
- 11: **if** $\beta(\phi(s_t, a_t, s_{t+1}), h_{\mathbf{w}}) = 1$ **then** $\psi^{\pi_{\mathbf{w}}} \leftarrow \text{unset}$ ▷ Definition 5.1
- 12: $h_{\mathbf{w}} \leftarrow h_{\mathbf{w}} + 1, s_t \leftarrow s_{t+1}, t \leftarrow t + 1, r \leftarrow r + \phi(s, a, s')^\top \mathbf{w}_j$
- 13: **end while**

5.4 Experiments and Analysis

This section describes the metrics, provides an overview of the agents and environments, and assesses the experiment results.

5.4.1 Experiment Setup

Metrics We evaluate TRL using the common metric of *asymptotic performance* to assess its impact on agents’ final performance (Zhu et al., 2023b). Additionally, we analyse the relative improvement across tasks distributed within the simplex.

Agents Despite SF-GPI having been extended to settings for continuous action spaces (Hunt et al., 2019; Y. T. Liu & Ahmad, 2024), by implementing policy-gradient algorithms such as soft-actor critic algorithms (Haarnoja, Zhou, et al., 2018), this chapter is focused on discrete action spaces and specifically in the implementation of agents based on Deep Q-Network (DQN) (Mnih et al., 2015). To build the baselines, we implemented SF-based agents. Each source task has its own SF, and by training the source tasks sequentially, we use GPI to update SFs concurrently. We implemented 2 variantes of the SFs algorithm from Barreto et al. (2017). The first, *GPI- π* , follows the original idea, updating only the SFs selected by GPI. The second, *GPI-all- π* , updates all SFs concurrently with collected experiences. This adds more computational complexity, making it unfeasible for a larger number of source tasks. The process used for both baselines is summarised in Algorithm 7. This illustrates the sequential training of source tasks using GPI, along with policy updates

Table 5.1: Hypparameter settings for low-level policies using tabular successor features.

Hypparameter	Value
α	0.3
γ	0.95
Initial_epsilon	1
Final_epsilon	0.05
Epsilon decay steps	600.000
Use replay buffer	True
Prioritised Experience Replay (PER)	True
Minimum priority	0.01
Batch size	5
Buffer size	10^6

Table 5.2: Hypparameter settings for low-level policies using function approximation.

Parameter	Value
α	1×10^{-3}
γ	0.9
Network architecture	[256, 256]
Batch size	256
Initial_epsilon	0.05
Final_epsilon	0.05
Prioritised Experience Replay (PER)	True
Minimum priority	0.01
Buffer size	4×10^6
Target network update frequency (timesteps)	500

to learn and compute successor features. To fit SFs, we use prioritised experience replay and a ε -greedy algorithm with a linear decaying epsilon. Our approach is influenced by Hierarchical-DQN (Kulkarni, Narasimhan, et al., 2016), both high-level states and high-level actions are implemented using SF with the same hyperparameters as GPI. However, we do not rely on GPI for updating SFs or for high-level decisions. This design avoids confounding effects and ensures a direct comparison between independently trained agents.

We report the hyperparameters used in our experiments to ensure reproducibility. Table 5.1 summarises those used to train the tabular SF models, while Table 5.2 presents the function approximation setting. Finally, the hyperparameters employed to train the high-level policy, shared across both environments, are also reported in Table 5.3. These hyperparameters were used for both the baselines and our proposed approach.

Algorithm 7 Sequential source task training with GPI and concurrent update of tabular successor features for source tasks.

Require: set of source tasks, $\mathbf{W}_s = \{\mathbf{w}_0, \dots, \mathbf{w}_{n-1}\}$, SFs $\Psi = \{\psi^{\pi_{\mathbf{w}_0}}, \dots, \psi^{\pi_{\mathbf{w}_{n-1}}}\}$, learning rate α , exploration probability ε , timesteps, update_all_ π , batch number n_batch.
Ensure: Successor feature for new source task $\mathbf{w}_n, \psi^{\mathbf{w}_n}$

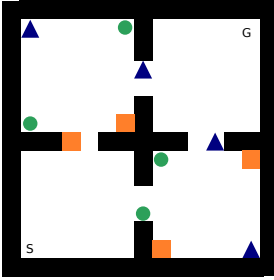
```

1: Initialise all SFs in  $\Psi$ 
2: new_episode  $\leftarrow$  TRUE
3: for  $i = 0 \dots |\mathbf{W}_s|$  do
4:    $\mathbf{w}_i \leftarrow \mathbf{W}_s[i]$  ▷ Get the current training source task
5:   Initialise replay buffer  $B_i$ 
6:   end_episode  $\leftarrow$  FALSE
7:   for  $t = 0 \dots \text{timesteps}$  do
8:     if end_episode then
9:        $s_t \leftarrow$  initial state sampled from environment
10:    end if
11:     $c \leftarrow \arg \max_{b \in \{0, \dots, |\Psi|\}} \max_a \psi^{\pi_{\mathbf{w}_b}}(s_t, a)^\top \mathbf{w}_i$  ▷ Index of max SF
12:    if Bernoulli( $\varepsilon$ ) then then
13:       $a_t \leftarrow \text{Uniform}(A)$  ▷  $\varepsilon$ -greedy exploration
14:    else
15:       $a_t \leftarrow \arg \max_a \psi^{\pi_{\mathbf{w}_c}}(s_t, a)^\top \mathbf{w}_i$  ▷ GPI
16:    end if
17:    Execute  $a_t$ , observe  $\phi_t, s_{t+1}$ , end_episode from environment
18:    Store transition  $s_t, a_t, s_{t+1}$ , and  $\phi_t$  in  $B$ .
19:    Sample n_batch number of experiences from  $B_i$ 
20:    if  $c \neq i$  ▷ GPI- $\pi$  baseline then
21:      sf_to_update  $\leftarrow \{c, i\}$  ▷ Update current SF  $i$  and GPI SF  $c$ 
22:    else
23:      sf_to_update  $\leftarrow \{i\}$  ▷  $i$  current task index
24:    end if
25:    if update_all_ $\pi$  ▷ GPI-all- $\pi$  baseline then
26:      sf_to_update  $\leftarrow \{0, \dots, n-1\}$  ▷ All SFs including the current SF
27:    end if
28:    for  $k = 0 \dots \text{n\_batch}$  do
29:      for  $l \in \text{sf\_to\_update}$  do
30:         $\mathbf{w} \leftarrow \mathbf{W}_s[l]$  ▷ Select vector weight
31:         $\psi^{\mathbf{w}} \leftarrow \Psi[l]$  ▷ Select SF
32:         $a' \leftarrow \arg \max_{a'} \max_{\psi \in \Psi} \psi(s_{k+1}, a')^\top \mathbf{w}$  ▷ Greedy action using GPI
33:         $\delta_t = \phi_k + \gamma \psi^{\mathbf{w}}(s_{k+1}, a') - \psi^{\mathbf{w}}(s_k, a_k)$ 
34:         $\psi^{\pi_{\mathbf{w}}}(s_k, a_k) \leftarrow \psi^{\pi_{\mathbf{w}}}(s_k, a_k) + \alpha \delta_t$  ▷ Update SFs
35:      end for
36:    end for
37:  end for
38: end for

```

Table 5.3: Hypparameter settings for high-level policy.

Parameter	Value
α	3×10^{-4}
γ	0.95
Network architecture	[32, 32]
Batch size	32
Initial epsilon	1
Final epsilon	0.05
epsilon_decay_steps	2.7×10^6
Prioritised Experience Replay (PER)	True
Minimum priority	0.01
Buffer size	4×10^6
Target network update frequency (timesteps)	500
H (max. horizon)	50



(a) 3 kind of objects and 4 instances of each kind.

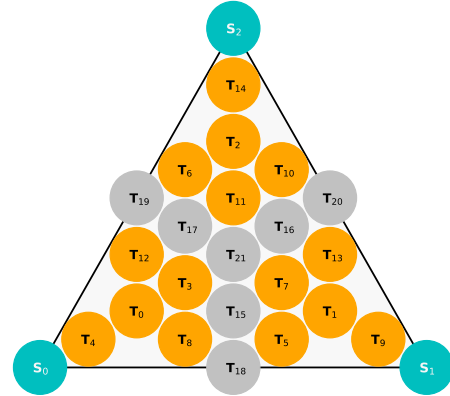
(b) Distributed tasks over simples of $\mathbb{R}^3$.

Figure 5.3: Four-room environment. *Cyan* dots are source tasks, with $S_0 = [1.0, 0.0, 0.0]$, $S_1 = [0.0, 1.0, 0.0]$, $S_2 = [0.0, 0.0, 1.0]$. *Orange* and *grey* dots are target tasks. *Grey* dots we considered *GPI-challenging tasks* due to they represent the farthest tasks.

5.4.2 Experiment Results and Analysis

Four-room Environment Inspired by Barreto et al. (2017), the four-room environment is widely used in SF literature (Alegre et al., 2022; Alver & Precup, 2022; Barreto et al., 2017; Nemecek & Parr, 2021). It consists of a 12×12 grid divided into four rooms. The agent navigates from a starting point S to a goal G , while collecting 12 objects (3 types, 4 instances each), as shown in Fig. 5.3(a). The agent has four deterministic actions: left, down, right, and up. The agent cannot cross walls and will remain in place if blocked. The state is

Table 5.4: Target tasks values for the Four-room environment.

T_0	[0.66,0.17,0.17]	T_7	[0.25,0.50,0.25]	T_{14}	[0.08,0.08,0.84]
T_1	[0.17,0.66,0.17]	T_8	[0.59,0.33,0.08]	T_{15}	[0.42,0.42,0.17]
T_2	[0.17,0.17,0.66]	T_9	[0.08,0.84,0.08]	T_{16}	[0.17,0.42,0.42]
T_3	[0.50,0.25,0.25]	T_{10}	[0.08,0.33,0.59]	T_{17}	[0.42,0.17,0.42]
T_4	[0.84,0.08,0.08]	T_{11}	[0.25,0.25,0.50]	T_{18}	[0.50,0.50,0.00]
T_5	[0.33,0.59,0.08]	T_{12}	[0.59,0.08,0.33]	T_{19}	[0.50,0.00,0.50]
T_6	[0.33,0.08,0.59]	T_{13}	[0.08,0.59,0.33]	T_{20}	[0.00,0.50,0.50]
				T_{21}	[0.33,0.33,0.33]

represented as a vector in $\mathbb{R}^{14}$, where the first 2 components are the agent’s coordinates, and the remaining 12 indicates the collected objects. The fixed, hand-designed feature function $\phi(s_t, a_t, s_{t+1}) \in [0, 1]^3$ encodes the type of objects as a one-hot encoded vector. The total reward is determined by task-specific weight vectors that assign values to object types. For example, if the one-hot encoding indicates triangles in the first component, squares in the second, and circles in the last, the task $[1, 0, 0]$ focuses only on triangles. Hence, the maximum reward per episode is 4, reflecting the number of triangles plus a reward for reaching the goal G .

The experiment involved 22 target tasks and 3 source tasks. Source tasks are represented as *cyan* dots at the extrema of the simplex in $\mathbb{R}^3$, and target tasks are distributed within the simplex, implying different distances from source tasks, as shown in Fig. 5.3(b) and with their respective values in Table 5.4. Based on these varying distances and following the GPI theorem from Barreto et al. (2017), we identified $T_3, T_7, T_{11}, T_{15}, T_{16}, T_{17}$, and T_{21} as GPI-challenging tasks, with T_{21} being the most so due to its location at the barycentre of the simplex. These tasks are shown in *grey* in Fig. 5.3(b). The impact of source-target task distances on agent performance is analysed in detail in the following section.

The agents were trained sequentially on each source task for 10^6 steps, with target task evaluations every 10^4 steps. Agents based on GPI began with randomly initialised Q-values for the first source task, then reused experience replay and fitted Q-values from the previously trained task to initialise the next.

As in Figs. 5.4(a) to 5.4(c), performance on the first source task is initially low, but improves in later tasks due to reuse of experience. Our approach does not benefit from this. Although it fails to derive an optimal policy for S_2 , this does not hinder the ability of the composition mechanism to solve target tasks.

We first focus on tasks T_4, T_9 , and T_{14} , which are closest to at least one of the source

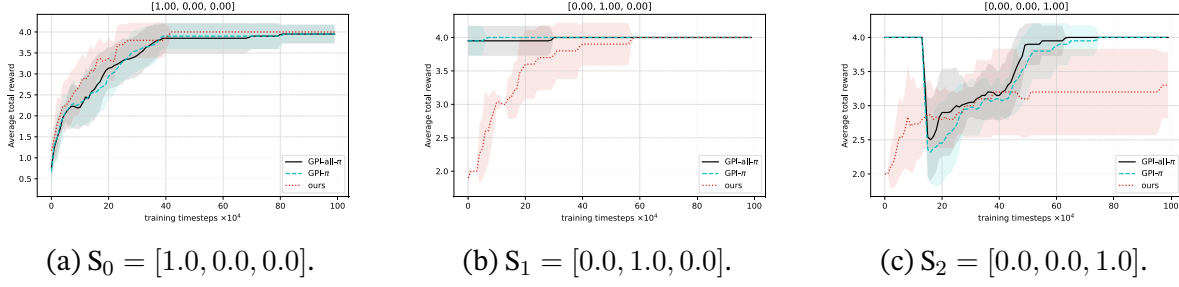


Figure 5.4: Four-room Environment. Source tasks average rewards. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.

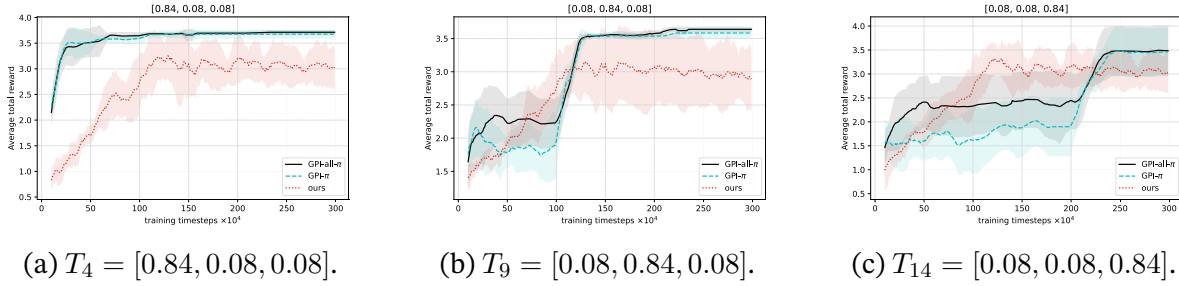


Figure 5.5: Four-room Environment. Target tasks average rewards. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.

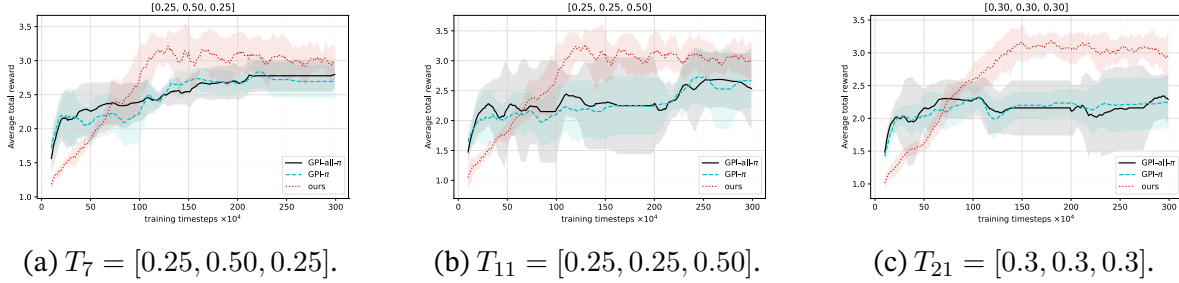


Figure 5.6: Four-room Environment. Target tasks average rewards. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.

tasks. The results demonstrate expected performance for agents based on GPI, in accordance with the GPI theorem. As shown in Figs. 5.5(a), 5.5(b), and 5.5(c), once the closest source task policy is learnt, target task improves. This becomes clearer in Fig. 5.5(c), agent’s performance on T_{14} increases sharply after training on S_2 , its closest source task.

For GPI-challenging tasks T_7 , T_{11} , and especially T_{21} in the barycentre, GPI-based agents show performance degradation in Fig. 5.6(c), while our approach maintains steady performance over varying task distances. For T_{21} , it significantly outperforms baselines, achieving superior asymptotic performance.

The results for the remaining 16 target are shown in Fig. 5.7, with the summarised

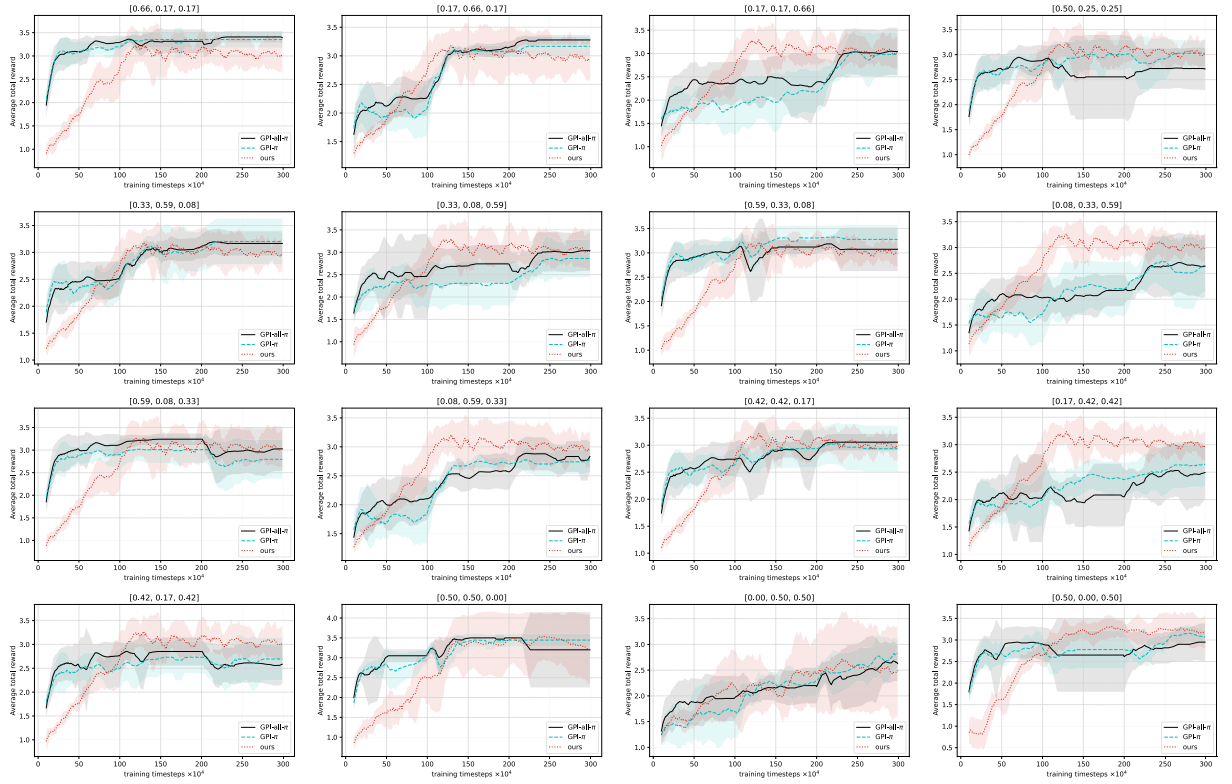
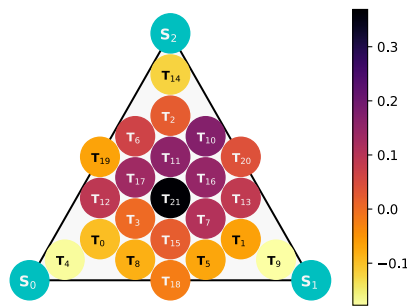
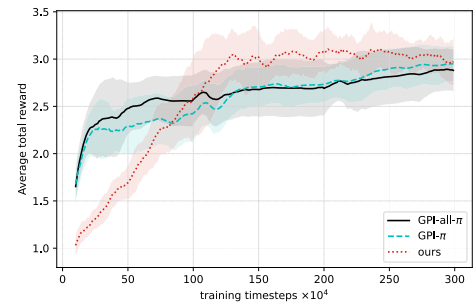


Figure 5.7: Results for the 16 target tasks in the simplex of $\mathbb{R}^3$ for the Four-room Environment per target tasks. Average of total reward over 10 random seeds with 1 standard deviation.

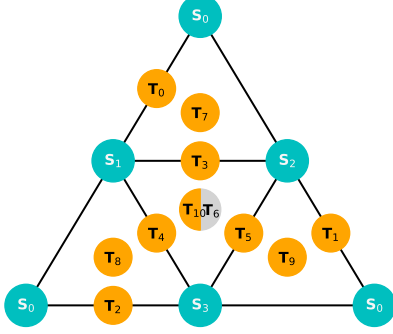


(a) Relative change, *ours* vs GPI- π . Average after 2.5 million steps.



(b) Average reward over 22 tasks.

Figure 5.8: Four-room Environment. *Cyan* dots represent source tasks. $S_0 = [1.0, 0.0, 0.0]$, $S_1 = [0.0, 1.0, 0.0]$, $S_2 = [0.0, 0.0, 1.0]$. $T_3, T_7, T_{11}, T_{15}, T_{16}, T_{17}$, and T_{21} are *GPI-challenging tasks*. Average reward over 10 random seeds with shaded area indicating 1 standard deviation.



(a) Distributed tasks over simplex of $\mathbb{R}^4$ showed as graph.

T_0	$[0.50, 0.50, 0.00, 0.00]$
T_1	$[0.50, 0.00, 0.50, 0.00]$
T_2	$[0.50, 0.00, 0.00, 0.05]$
T_3	$[0.00, 0.50, 0.50, 0.00]$
T_4	$[0.00, 0.50, 0.00, 0.50]$
T_5	$[0.00, 0.00, 0.50, 0.50]$
T_6	$[0.25, 0.25, 0.25, 0.25]$
T_7	$[0.33, 0.33, 0.33, 0.00]$
T_8	$[0.33, 0.33, 0.00, 0.33]$
T_9	$[0.33, 0.00, 0.33, 0.33]$
T_{10}	$[0.00, 0.33, 0.33, 0.33]$

(b) Target tasks values.

Figure 5.9: Reacher environment. *Cyan* dots are source tasks, with $S_0 = [1.0, 0.0, 0.0, 0.0]$, $S_1 = [0.0, 1.0, 0.0, 0.0]$, $S_2 = [0.0, 0.0, 1.0, 0.0]$, and $S_3 = [0.0, 0.0, 0.0, 1.0]$. *Orange* and *grey* dots are target tasks. *Grey* dot we considered *GPI-challenging* task. T_6 is the barycenter projected to one of the tetrahedron faces.

outcomes described as average total reward (Fig. 5.8(b)) and relative improvement (Fig. 5.8(a)). Fig. 5.8(b) presents the asymptotic performance of the three proposed agents across all tasks. The results of the 10 runs reveal that *GPI-all- π* and *GPI- π* achieve similar asymptotic performance, with *GPI-all- π* converging faster but showing greater variance. Our approach achieves higher average performance than the baselines, although the overlapping variance during the later training stages results in overall comparable outcomes. Once SFs are properly learnt, the composition process becomes more effective. We hypothesise that the delayed convergence stems from concurrently learning SFs and their composition. Moreover, the training order of the source tasks may influence the quality of the composed policies in GPI-based methods.

Lastly, Fig. 5.8(a) compares the relative performance of our approach and *GPI- π* across 10 runs. GPI performs better on tasks closer to the source, but degrades as distances increase. Conversely, our approach presents consistent performance across all target tasks, achieving improvements of 10% to 30% on GPI-challenging tasks, with the highest improvement observed in the barycentre. This improvement is attributed to our high-level policy’s ability to reason over high-level states and adaptively select corresponding high-level actions, ensuring steady performance across a wider range of tasks.

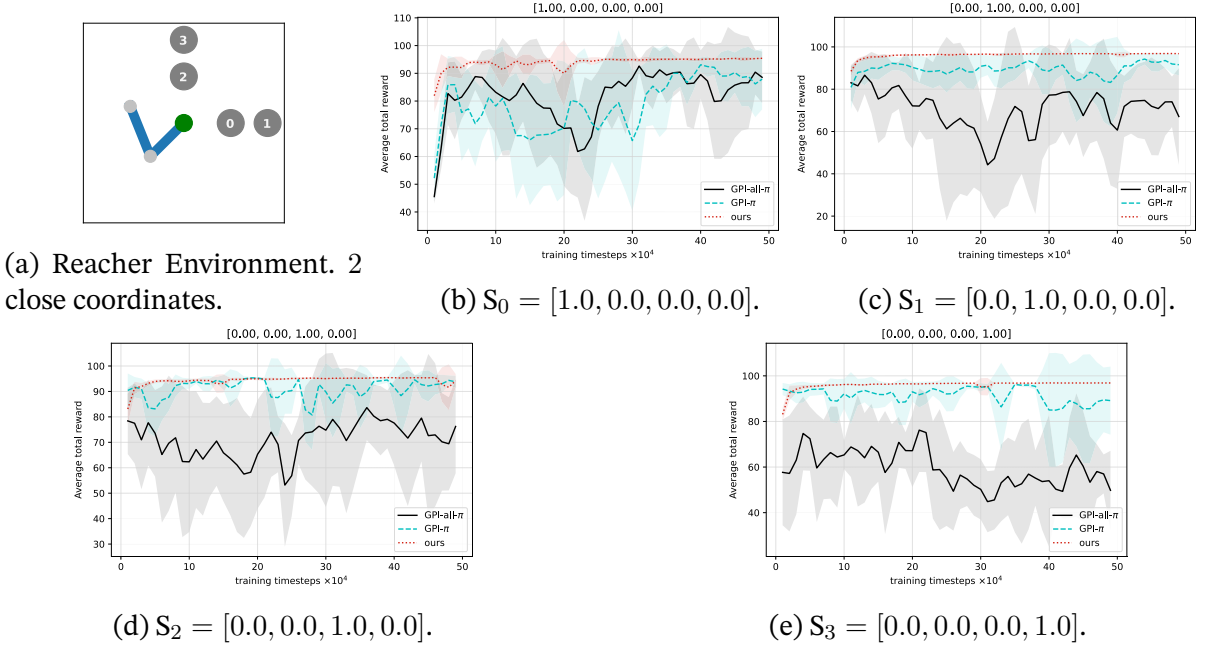


Figure 5.10: Reacher Environment using 2 close coordinates. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.

Reacher environment The reacher environment consists of a robot arm with 2 joints and 2 segments. This environment has been extensively used in SF-GPI works by Alegre et al. (2022, 2023b) and Barreto et al. (2017). The agent learns to manipulate the arm by applying torque to each joint, aiming to reach 4 different objects based on a preference vector. The action space is discretised into 9 pairs of values (a, b) , where $a, b \in -1, 0, 1$, representing the torque applied to each joint. The agent controls the arm for a total of 100 timesteps. The hand-designed feature function is a vector of Euclidean distances, computed as $1 - d(r, o)$, where r is the endpoint of the arm and o denotes each of the fixed objects, $\phi(s_t, a_t, s_{t+1}) \in \mathbb{R}^4$. Tasks are defined by weight vectors that prioritise objects, specifying their relative preferences. For example, a weight vector $[1, 0, 0, 0]$ prioritises the first object, rewarding the agent only for reaching it, while $[0.5, 0.5, 0, 0]$ rewards the agent for reaching either of the first 2 objects. The agent’s state is represented as a vector containing the angles of the first and second joints, enabling informed decisions about torque application to reach desired objects.

The experiment was conducted with 4 source tasks and 11 target tasks distributed within the simplex of $\mathbb{R}^4$, as shown in Fig. 5.9(a) and their respective values in Table 5.9(b). The agents were tested in two settings, averaging the results in 5 independent runs, as described below.

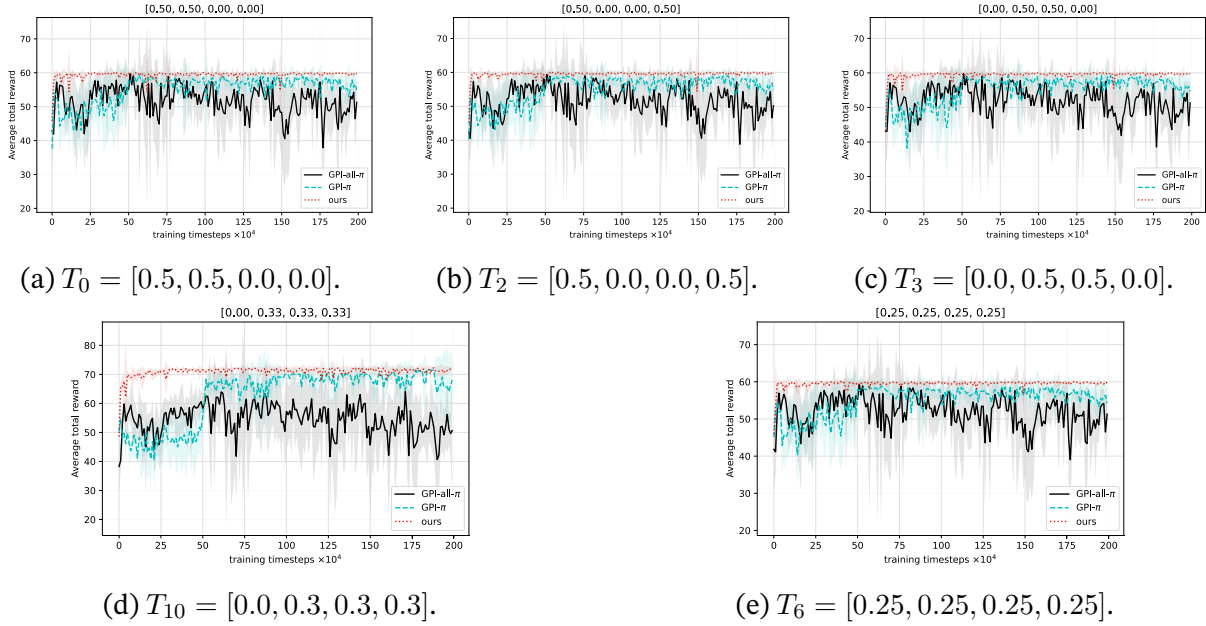


Figure 5.11: Target tasks results for the Reacher Environment using 2 close coordinates. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.

Simple setting. In the first, 2 of the 4 objects shared similar coordinates, as illustrated in Fig. 5.10(a). While learning the optimal policy for a source task, the agents reused policies and experiences from previously learnt source task, as evidenced in Figs. 5.10(b) through 5.10(e). However, for this environment, the GPI-all- π agent was negatively impacted by concurrent reuse and updating of all policies, whereas the GPI- π agent benefited from this strategy. Notably, our approach exhibited stable learning for all source tasks, with consistently low variance.

For the proposed target tasks, the stability and low variance of our approach are evident. These tasks posed challenges for GPI-based agents, as shown in Figs. 5.11(a) to 5.11(e), where our agent maintained steady performance across all GPI-challenging tasks. It achieved a general improvement of 4% in 7 out of 11 target tasks. The relative modest improvement compared to GPI-based agents when varying source-task distances is primarily due to the feature function space, the closeness of the objects, and its impact on source policies. For example, the optimal policy for S_0 can be effective for S_1 , as objects are close. Fig. 5.12(b) suggests that our approach achieved small relative improvements across 11 target tasks. Nevertheless, in the most GPI-challenging task (Fig. 5.11(e)), our approach demonstrated better asymptotic performance compared to the baselines. The remaining results are shown in Fig. 5.13, which highlight the stable performance and the low variance when transferring SFs to target tasks.

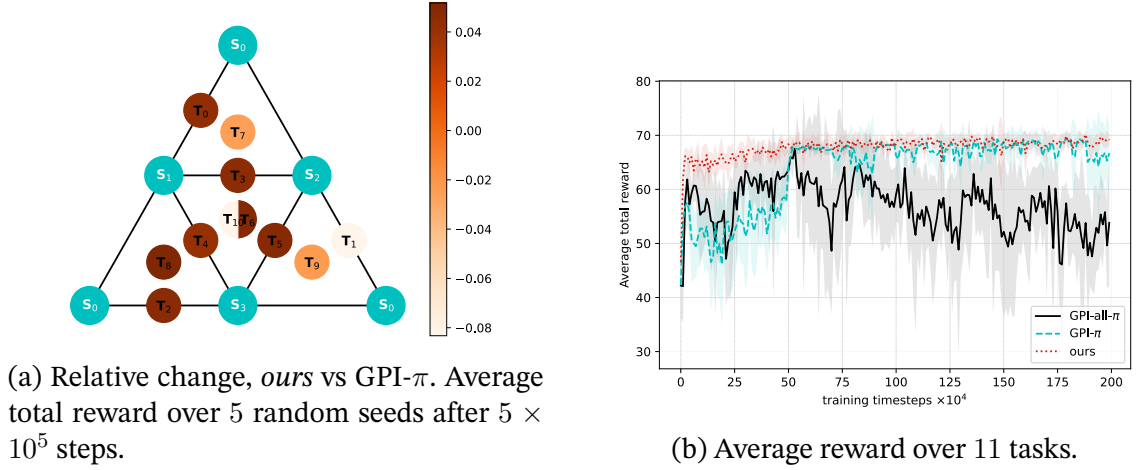


Figure 5.12: Reacher Environment using 2 close coordinates. Cyan dots are source tasks. T_6 is the barycentre projected to one of the tetrahedron faces. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.

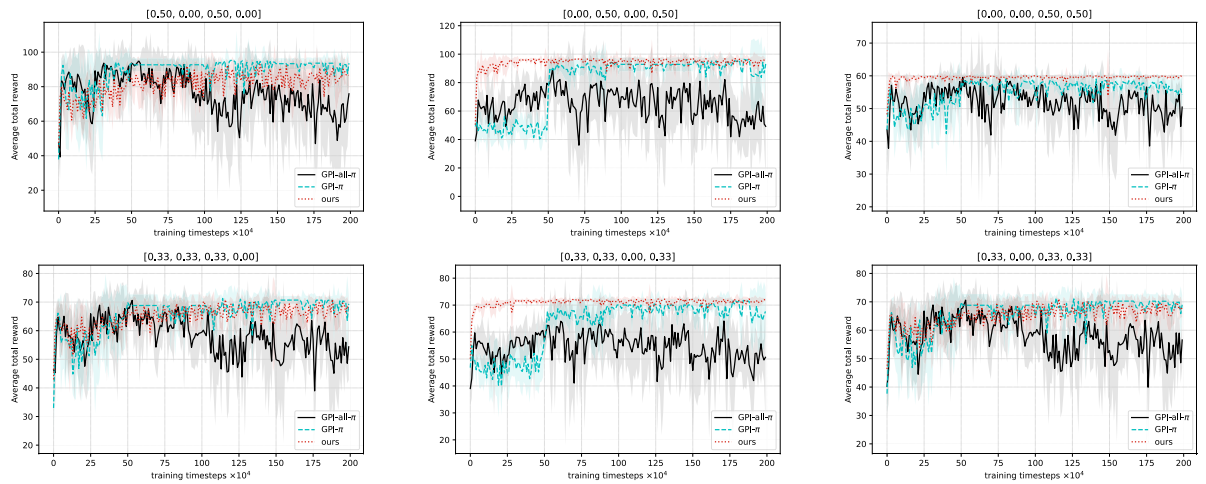


Figure 5.13: Results for the remaining 6 target tasks in the simplex of $\mathbb{R}^4$ for the Reacher environment per target tasks using 2 close coordinates Fig. 5.10. Average of the 11 target tasks. Average of total reward over 5 random seeds with 1 standard deviation.

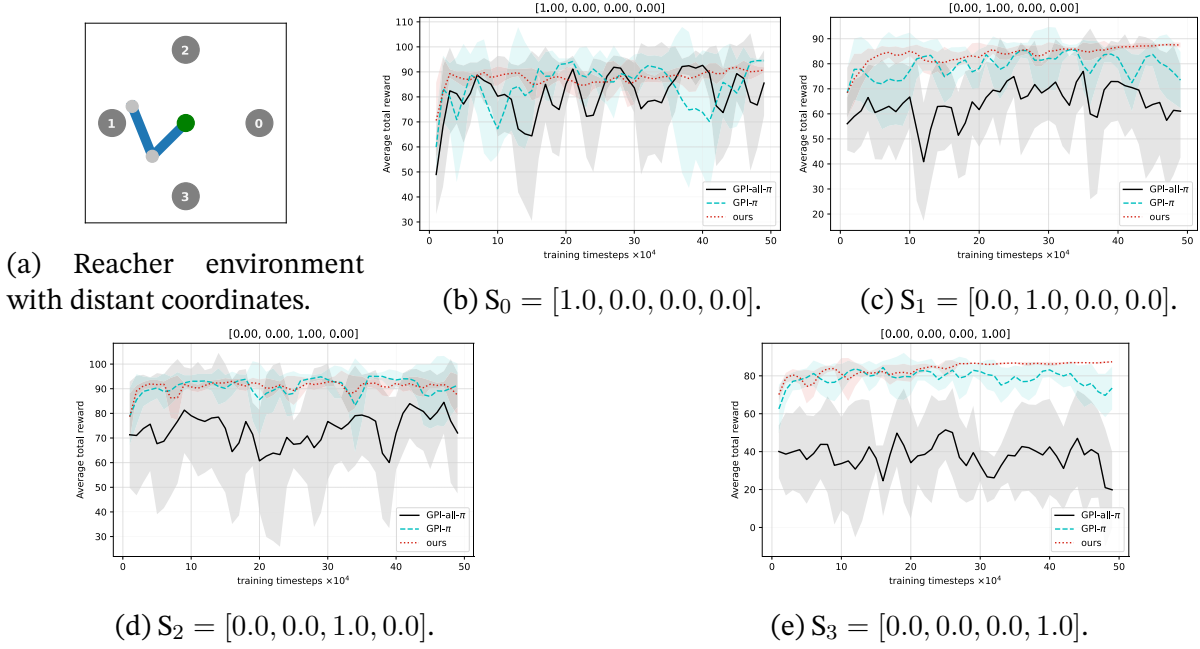
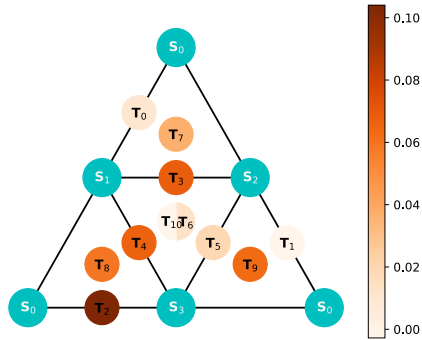
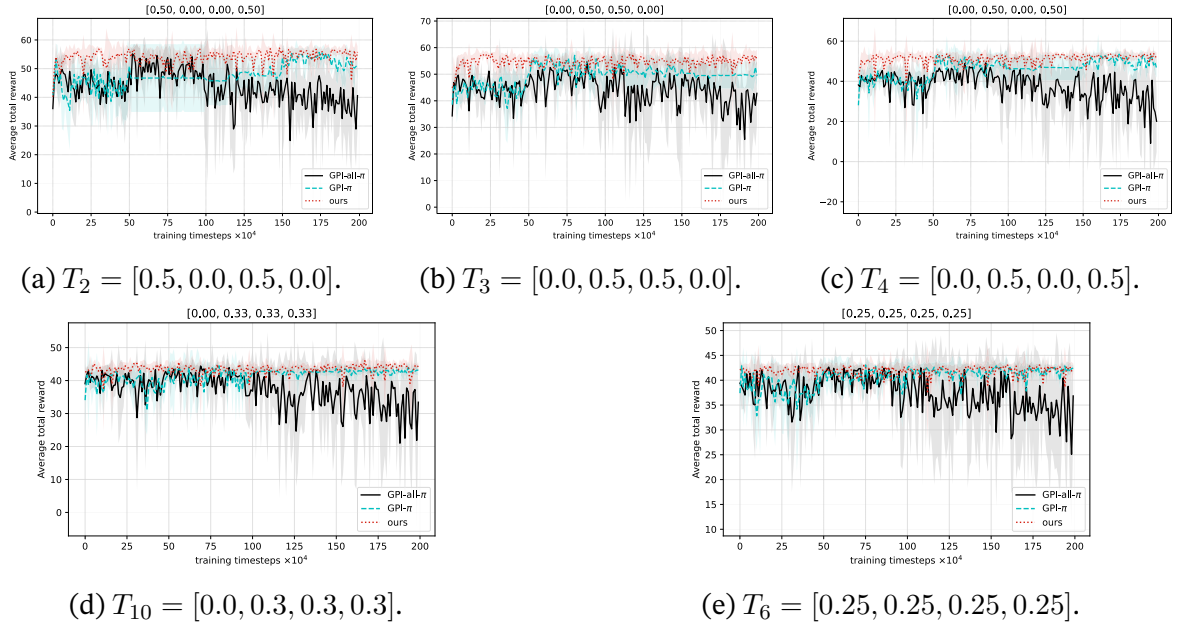


Figure 5.14: Reacher Environment using 4 distant coordinates. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.

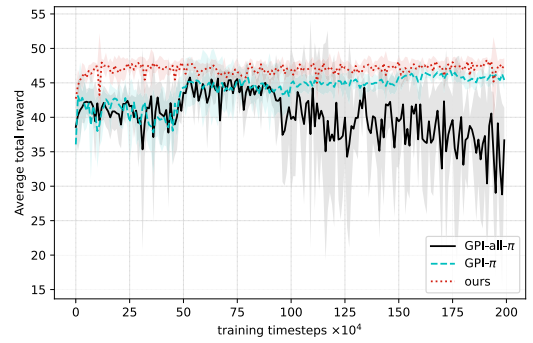
Challenging setting. In a more challenging setting (Fig. 5.14(a)), the 4 objects were positioned at distant coordinates, with target tasks distributed as shown in Fig. 5.16(a). Similar to the simple setting, source tasks (Figs. 5.14(b) to 5.14(e)) demonstrated that our approach maintained low-variance results while learning optimal policies for source tasks. For the target tasks (Figs. 5.15(a) to 5.15(e)), our agents outperformed the baselines in 82% of cases.

The asymptotic results in Fig. 5.16(b) show that our approach effectively combines source behaviours, achieving an overall improvement of 4.2% compared to the GPI agent while exhibiting lower variance across 5 random seeds. This suggests reduced noise in SF models compared to GPI when using function approximations. In contrast to tabular scenarios, the asymptotic performance of GPI-all- π did not converge, likely due to its reuse strategy and noisy experiences destabilising SF representations and average performance. We attribute this to the sequential learning process in GPI, where combining experiences from multiple source tasks can introduce noisy samples. This effect is particularly evident in GPI-all- π , which, while aiding rapid convergence, can also produce noisier results.

The performances on the remaining 6 target tasks are shown in Fig. 5.17. Our method achieved relative improvements of 2% to 10%, outperforming GPI in 9 of 11 tasks (82%) and matching performance in the remaining 2. Notably, even the modest 2% gain in the



(a) Relative change, *ours* vs GPI- π . Average reward after 5×10^5 steps.



(b) Average reward over 11 tasks.

Figure 5.16: Reacher Environment using 4 distant coordinates. *Cyan* dots are source tasks. T_6 is the barycentre projected to one of the tetrahedron faces. Average reward over 5 random seeds with shaded area indicating 1 standard deviation.

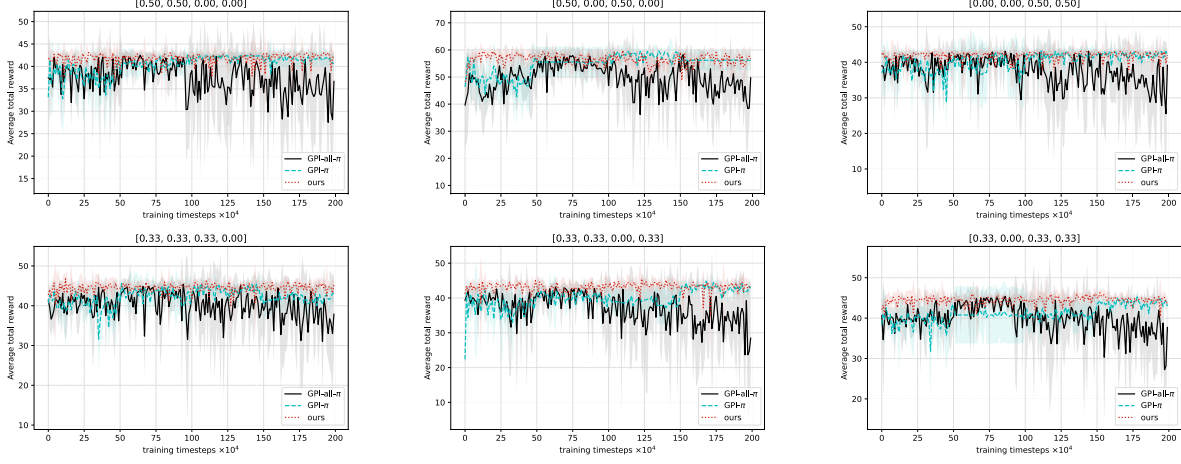


Figure 5.17: Results for the 6 remaining target tasks in the simplex of $\mathbb{R}^4$ for the Reacher Environment per target tasks using 4 distant coordinates. Average of total reward over 5 random seeds with 1 standard deviation.

most GPI-challenging task (Fig. 5.15(e)) demonstrates the method’s ability to compose behaviours across significant source-target differences.

Foraging environment It has been extensively used in SF-GPI works (Barreto et al., 2020; Borsa, Barreto, Quan, Mankowitz, et al., 2019; Carvalho et al., 2023). It consists of an 8×8 grid containing 10 objects of 2 types. For 200 steps, the agent collects objects by navigating to their coordinates, as shown in Fig. 5.18(a). Upon collection, a new object of random type respawns at a random location. Reward depend on object type, with a feature function similar to that used in the four-room domain. Agents receive egocentric $7 \times 7 \times 3$ images encoding object types. To capture temporal patterns, the last 5 frames are stacked and passed as input. This input is processed through a CNN with 3 convolutional layers, with 16, 32, and 64 filters, using 2×2 kernels, stride 3, ReLU activations, and Batch Normalisation. The CNN output feeds into the ψ -network as described.

We compare our method with the Option Keyboard (OK), introduced by Barreto et al. (2019), which composes behaviours in the cumulant space using predefined vector weights as options. A Q-function selects among these options. Since OK typically requires domain-specific rewards, we adapted our high-level reward and termination function β from the proposed high-level MDP, and aligned its vector weights with our source tasks to ensure a fair comparison. This allows us to compare GPI-based composition in cumulant space (OK) and our approach, which reasons over high-level states.

To obtain optimal SFs, we trained agents in source tasks 2 source tasks, $[1, 0]$ and $[0, 1]$, running each 3 times and selecting the best-performing SF-derived policies. These SFs

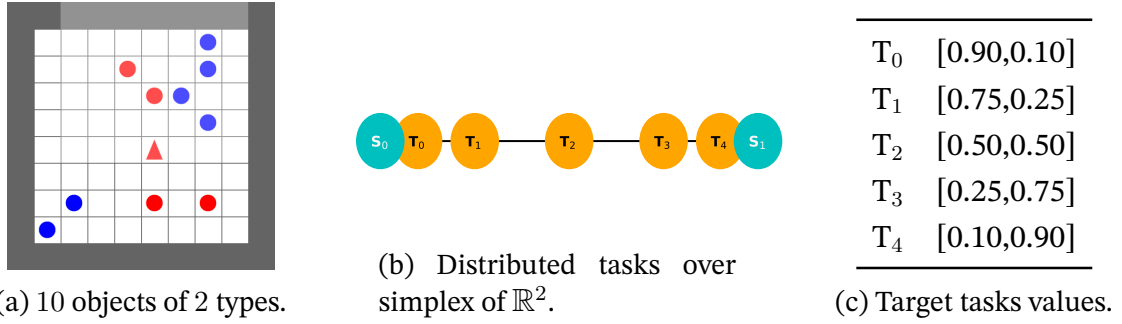


Figure 5.18: Foraging environment. Cyan dots are source tasks, with $S_0 = [1.0, 0.0]$, $S_1 = [0.0, 1.0]$. Orange dots are target tasks.

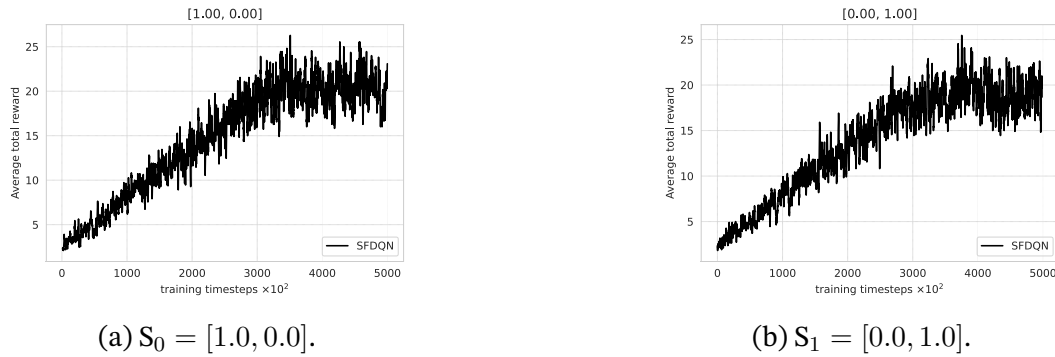


Figure 5.19: Foraging environment. Source task average performance.

were fixed during composition learning, *i.e.*, when fitting high-level Q-functions. Transfer performance was evaluated across 5 target tasks distributed over the $\mathbb{R}^2$ simplex, as shown in Fig. 5.21(a).

The average performance across the target tasks, along with the results per-task, is shown in Fig. 5.21(b). OK showed steady performance in all tasks, supported by our adapted high-level reward. This highlights OK’s ability to generalise via GPI. However, our method achieved a stronger overall performance, proving the benefit of reasoning over high-level states when composing SFs for TRL, as the relative change reveals in Fig. 5.21(a).

5.4.2.1 Ablation Study

In addition to baseline comparisons, we examine the impact of our proposed high-level state within the high-level MDP. Specifically, we compare states based on successor features (*ours*) versus those from low-level observations (*low-level observations*). Both were evaluated across all target tasks in the four-room environment, focusing on asymptotic performance and variance for average performance across target tasks.

The average total reward for all tasks is presented in Fig. 5.22(b). Here, the average

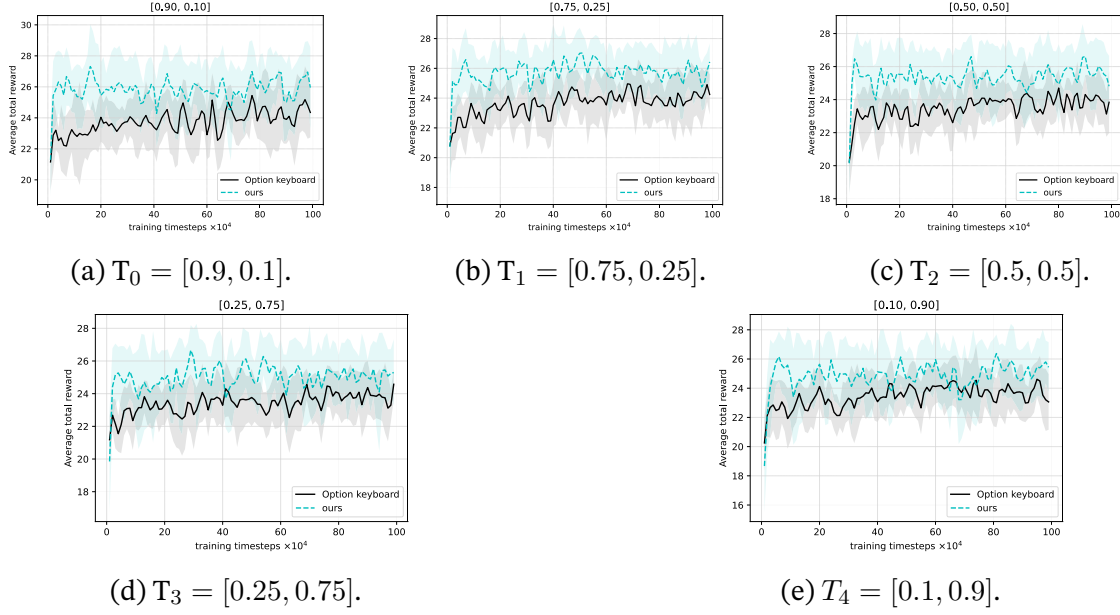


Figure 5.20: Foraging environment. Average of total reward over 10 random seeds with shaded area indicating 1 standard deviation.

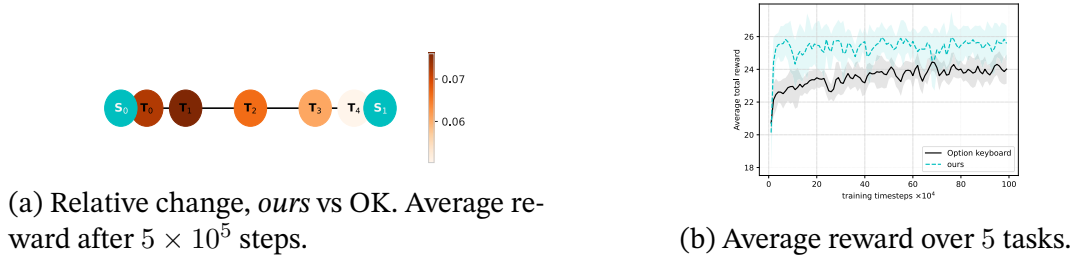
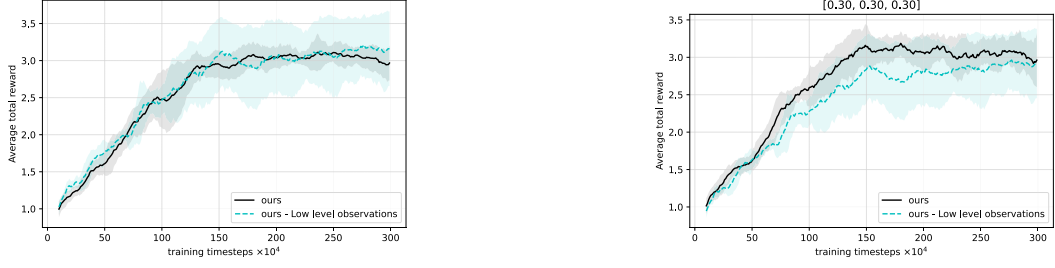


Figure 5.21: Foraging environment. *Cyan* dots are source tasks, with $S_0 = [1.0, 0.0]$, $S_1 = [0.0, 1.0]$. Average of total reward over 10 random seeds with shaded area indicating 1 standard deviation.

rewards achieved by both agents are comparable, with the primary distinction being a reduction in variance for the proposed high-level state. This suggests a positive influence of the high-level state on policy decisions. For the most GPI-challenging task, the proposed high-level state significantly improves variance and asymptotic performance (Fig. 5.22(b)). The SF-based state forms a more meaningful representation, enhancing decision-making for unseen tasks.

5.4.2.2 Impact Analysis of Source and Target Task Distances

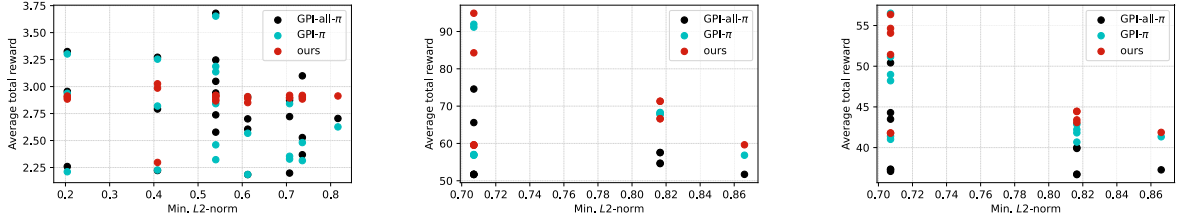
We empirically examine how the distance between source and target tasks affects the final performance. We analyze the average performance of agents on 10 random seeds after



(a) Average total reward over 22 target tasks.

(b) Average reward in GPI-challenging task.

Figure 5.22: Ablation and task distance analysis in the four-room environment. Average total reward over 10 random seeds with shaded area indicating 1 standard deviation.



(a) Distances vs average total reward after 2.5×10^6 steps over 10 random seeds in the Four-room environment.

(b) Distances vs average total reward over 5 random seeds after 5×10^4 timesteps. Reacher Environment for 2 close coordinates scenario.

(c) Distances vs average total reward over 5 random seeds after 5×10^4 million steps. Reacher Environment 4 distant coordinates.

Figure 5.23: Impact analysis of source and target task distances. Average total reward over random seeds with shaded area indicating 1 standard deviation.

saturation, *i.e.*, 2.5 million timesteps. We computed the distance between the i -th target task $T_i \in \mathbf{W}_\tau$ to the closest source task, as $\min_{S \in \mathbf{W}_s} (\|S - T_i\|_2)$.

We conducted this analysis in both the Four-room and Reacher environments. For the Reacher setting, the comparisons are shown in Figs. 5.23(b) and 5.23(c). In these scenarios, while increasing the source-target distance negatively impacts performance, our approach consistently outperforms the GPI baseline. In contrast, as shown in Fig. 5.23(a), the performance of GPI-based agents degrades with distance, consistent with theoretical bounds. However, our method maintains stable performance across tasks, demonstrating greater robustness and adaptability.

Unlike the Reacher environment, the Four-room environment consists of a larger number of tasks with varying distances. In this setting, the advantages of our approach are more evident: it consistently achieves higher average performance on target tasks while sustaining steady results across different distances.

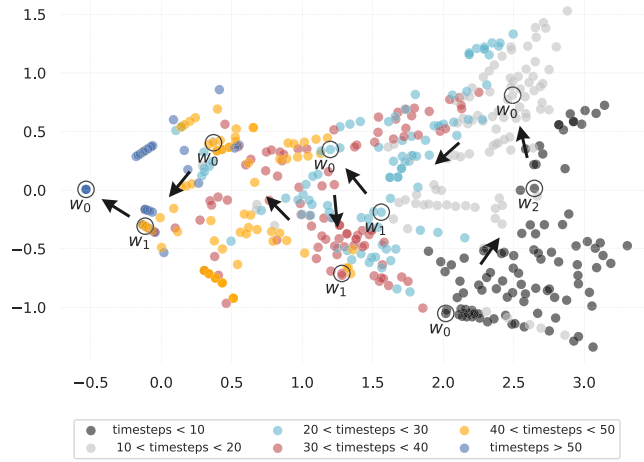


Figure 5.24: Principal Component Analysis (PCA) of high-level states for task $[0.33, 0.33, 0.33]$ in four-room.

5.4.2.3 High-level States and SFs Composition Method

As extensively discussed, we propose using high-level states for high-level Markov decision processes. To explore this space, we collected high-level states from multiple random seeds in a four-room environment. We then applied principal component analysis (PCA) to reduce the original 9-dimensional values to 2 dimensions, as shown in Fig. 5.24. We manually added 6 clusters to analyse temporal and policy composition patterns. We observed that the reduced 2-dimension values change over time. Specifically, for timesteps under 20, the values are widely spread, but over time the values converge to a single point where one of the principal components is 0. These changes are expected, as early timesteps involve exploration and diverse feature observations. As the agent makes optimal decisions, the number of observable features decreases. In addition, we plotted the policy composition of one trial and noticed 2 key findings. First, the agent uses multiple behaviours to navigate the environment. Second, this composition helped guide the agent through the state space toward a specific region, as suggested by Fig. 5.24.

5.4.2.4 Computational Complexity Analysis

To assess the impact of our high-level state, we analysed computational complexity of a single high-level decision step, *i.e.*, the agent selects the SF to follow and chooses the best action. Let the SF be modelled by a single-layer MLP with input dimension s and output dimension of $d \times a$, where d is the feature function dimension and a the number of actions. The high-level policy has an input size $d \times n$ and an output size n , where n is the number of

tasks. Thus, the complexity of a high-level decision step in our approach is $\mathcal{O}(dn^2 + sda)$, compared to GPI’s $\mathcal{O}(nsda)$. The quadratic term in our method reflects the worst-case, where all n SFs are evaluated to select the best action. In optimistic scenarios, where high-level decisions are deferred due to extended option execution, our approach becomes more efficient than GPI, which recomputes at every timestep. Strategies to address quadratic complexity are discussed in the next section.

5.5 Summary

In this chapter, we introduced a hierarchical mechanism for adaptive successor feature (SF) composition in Transfer Reinforcement Learning (TRL). By constructing high-level state and action spaces from source-task SFs, our method enables behaviour adaptation to unseen tasks. Principal component analysis (PCA) revealed that the high-level state space evolves meaningfully over time, reflecting reduced complexity and improved decision efficiency. The algorithm jointly learns SFs the source policies and a high-level policy to compose these behaviours, outperforming baselines in the Four-room, Reacher, and Foraging environments extensively used in the SF literature.

Despite these contributions, we identified four areas for improvement. First, joint learning of SFs and high-level policies is computationally intensive, affecting efficiency metrics such as time to threshold (Zhu et al., 2023b), although addressing this lies beyond our scope. Second, our β function is tailored to goal- or distance-based tasks; however, since feature values often represent salient events (Barreto et al., 2018), it can be adapted to generalise across a broader range of scenarios. In such cases, termination functions could be informed by domain knowledge (e.g., thresholds for risk features) or learnt directly as binary decisions from reward signals. Third, scaling the high-level state space becomes challenging with many tasks or high-dimensional SFs. We suggest two potential directions: (1) clustering tasks to form equivalence classes for high-level states, and (2) applying dimensionality reduction to preserve only the most informative representations. Finally, while our empirical results demonstrate strong transfer robustness, the method currently lacks formal performance guarantees. Future work could address this by exploring hierarchical RL regret analysis.

Overall, while our method provides an alternative to GPI, it also creates opportunities to extend frameworks such as Option Keyboard with high-level state reasoning. In particular, GPI could serve as a low-level strategy—retaining its strengths for similar tasks—while our approach contributes adaptability across diverse scenarios. Together, these elements

support the development of more scalable TRL solutions.

ADAPTIVE SUCCESSOR FEATURES FOR HUMAN LEARNING SUPPORT

*The Scientific Revolution has not been a revolution
of knowledge so much as a revolution of ignorance.*

*Yuval Noah Harari, Sapiens: A Brief History of
Humankind*

For the last objective, RO-6, which contributes to answering RQ-4, we identified a relevant real-world case for applying Transfer Reinforcement Learning (TRL) as a practical solution. Learning Analytics contains the study and development of technological methods designed to support human learning activities. Within this field, Self-regulated learning (SRL) is recognised as a complex skill that plays a crucial role in academic success (Bannert, 2007; Winne, 2022; Winne & Hadwin, 1998). To foster SRL in online environments, existing platforms have primarily relied on recommender systems, while recent studies have proposed the use of Reinforcement Learning (RL) (Osakwe et al., 2023, 2024). However, we argue that such RL-based approaches are difficult to scale and integrate into current solutions. To address this challenge, we investigate TRL and Successor Features as a promising alternative capable of overcoming these limitations and effectively supporting SRL in online environments.

6.1 Background and Motivation

Self-regulated learning (SRL) involves multiple cognitive, metacognitive, motivational, and emotional learning processes that learners consciously and strategically enact to accomplish their learning tasks (Butler & Winne, 1995; Zimmerman, 1989). SRL skills have been deemed crucial not only in boosting learners' academic success, but also in developing skilful lifelong learners that can adapt to different learning contexts (Fan, Lim, et al., 2022). For this reason, measuring and supporting student SRL has been considered one of the key foci in the educational technology research over the past decades (Raković et al., 2024).

Personalised SRL support provided to learners in online learning environments is, however, often limited compared to personalised SRL support provided to learners in traditional educational settings (Wong et al., 2017). One of the major limitations was associated with detecting learners' evolving needs during a learning session and tailoring just-in-time SRL feedback personalised to those needs (Raković et al., 2024). The provision of such feedback requires an understanding of learning strategies (*i.e.*, goal-oriented sequences of learning processes) that each individual learner enacts throughout the session (N. Srivastava et al., 2022).

More recently, researchers have shown trace data collected in online learning environments can be used to detect SRL strategies as they emerge throughout a learning session and thus inform SRL support (*e.g.*, (Fan, van der Graaf, et al., 2022; Li et al., 2023)). However, many learners still struggle to productively self-regulate their learning due to the lack of timely and personalised feedback. This can further negatively affect learning experiences and outcomes (Azevedo et al., 2008; Heikkinen et al., 2023; Li et al., 2024). In recent studies, researchers have emphasised the need for developing SRL recommender systems that can provide continuous, adaptive and personalised scaffolding tailored to real-time learning dynamics of each individual learner (Du & Hew, 2022).

Given the complexity of real-time learning dynamics, an effective SRL recommender system requires methods capable of modelling sequential decision-making and adapting to individual learners' evolving needs. RL, with its strength in optimising sequential processes, has shown promise in non-education domains and is now being explored for its potential in SRL. In non-educational contexts, classical RL approaches have achieved remarkable success, reaching human-level performance in video games (Mnih et al., 2015) and optimising industrial manufacturing processes (Luo, 2020a). These advancements highlight RL's potential for understanding real-time sequences and personalisation. Lever-

aging this potential, Osakwe et al. (2023) applied RL algorithms to analyse learners' SRL behaviours. The authors used simulations to optimise learning process sequences in SRL and compared the results with those obtained from other machine learning methods. Similarly, Osakwe et al. (2024) trained an RL agent with a Long Short-Term Memory network (LSTM)-based reward function to optimise SRL sequences. These studies demonstrate RL's potential for providing personalised SRL support, particularly in offline and simulated environments.

Several limitations may prevent classical RL methods from effectively study and support SRL processes. First, classical RL methods rely on large amount of data samples, making sampling inefficient and scaling difficult (Lazaric, 2012; Zhu et al., 2023b). Second, the one-policy-per-reward framework restricts adaptability to new rewards or individual learner SRL behaviours, requiring substantial data to train policies for each reward (e.g., task scores or learning gains) (Osakwe et al., 2023). Third, modelling each learner as an RL agent, optimising for a specific reward, as reported by Osakwe et al. (2023), requires unique policies per learner, making real-time personalisation resource-intensive and impractical. Therefore, developing scalable and data-efficient methods for automating and personalising SRL strategy recommendations remains a critical challenge in educational technology research.

To address these limitations, we examined TRL, a scalable alternative to classical RL for detecting and supporting SRL strategies. Unlike RL, which requires restart the policy training for each learner, TRL agent leverages previously learnt knowledge (*i.e.*, source contexts) to provide personalised recommendations with minimal data (*i.e.*, target contexts) (Zhu et al., 2023b). A widely studied TRL approach is Successor Feature Representation, which enable knowledge reuse across contexts with differing rewards (Barreto et al., 2017; Dayan, 1993). While Successor Features have shown success in traffic control (Szoke et al., 2023), navigation (J. Zhang et al., 2017), text generation (Xu et al., 2020), human RL (Momennejad et al., 2017), and control tasks (J. Zhang, Ren, et al., 2024), their application in the context of real-time SRL strategy detection and support remains unexplored. Addressing this gap is directly aligned with **RO-6** and contributes to answering research question **RQ-4** of this thesis. Building on this motivation, this chapter poses the following research question (RQ): *To what extent can TRL enable scalable detection of, and inform support for, learner SRL strategies?*

This chapter explores the use of Successor Features in Self-Regulated Learning to personalise scaffolding with less data. Building on Osakwe et al. (2024)'s work, our approach defines a single reward function with a reusable policy across contexts. When a learner un-

dertakes a task (*e.g.*, essay writing), our method reuses previously optimised SRL strategies documented to maximise learning outcomes to generate personalised, real-time recommendations based on the current state of learners' SRL strategies (Osakwe et al., 2024). This reduces data requirements and adapts to individual outcomes. Therefore, our contributions are twofold:

1. We formalised the problem of scaffolding SRL with RL as a Transfer Reinforcement Learning setting and demonstrated the feasibility of transferring learnt SRL strategies to new study sessions with different rewards via Successor Features, thus enabling scalable transfer of optimal policies.
2. We presented the performance results of a TRL method that outperformed a RL baseline. To this end, we used SRL strategies collected from 92 students in an online learning session that involved writing from multiple source documents.

6.2 Preliminaries and Problem Formulation

Self-regulated learning (SRL) is a multi-dimensional construct that encompasses cognitive, metacognitive, motivational, and emotional dimensions enabling learners to engage in human learning tasks (Butler & Winne, 1995; Panadero, 2017; Zimmerman, 1989). Researchers have proposed a variety of theoretical and methodological frameworks to conceptualise SRL, which broadly fall into two categories: macro-level frameworks that describe general learning phases and constructs (*e.g.*, (Fan, van der Graaf, et al., 2022; Winne & Hadwin, 1998)), and micro-level frameworks that examine SRL processes at a finer granularity (*e.g.*, (Azevedo et al., 2008)). In this chapter, we focus on the framework that categorises processes into Metacognition (MC), Low Cognition (LC), and High Cognition (HC). This SRL classification is summarised in Table 6.1 based on Bannert (2007)'s work. For clarity, we group the SRL processes of First-reading and Re-reading into a single Reading process. Accordingly, this study considers six SRL processes rather than seven.

These frameworks have increasingly been used to interpret raw trace data and infer different SRL processes, strategies, and tactics (N. Srivastava et al., 2022; van der Graaf et al., 2022). More recently, educational researchers have moved beyond simply observing learner behaviour to providing personalised feedback tailored to learners' evolving needs (*e.g.*, (Li et al., 2023; Lim et al., 2023)).

Despite its importance, many learners struggle to engage in productive SRL in online environments (*i.e.*, digital learning settings delivered through web-based platforms

Table 6.1: SRL Processes and their definitions across different cognitive categories.

Main Categories	SRL processes	Definitions
Metacognition (MC)	Orientation (MC.O)	Orientation on the learning-related activities; on prior knowledge; on the task and feeling about the task. Reading of general instruction and rubric of essay.
	Planning (MC.P)	Planning of the reading and writing process by arranging activities and determining strategies. Proceeding to the next topic.
	Monitoring (MC.M)	Monitoring and checking the reading and writing process; checking of progress according to instruction or plan.
	Evaluation (MC.E)	Evaluation of the learning process; checking of content-wise correctness (e.g., the essay content) of learning activities.
Low Cognition (LC)	First-reading (LC.R)	Reading information from the materials and superficial describing of pictorial representations for the first time.
	Re-reading (LC.R)	Rereading of information in the text or figures.
High Cognition (HC)	Elaboration/ Organisation (HCEO)	Elaborate by connecting content-related comments and concepts during reading or writing; Organising of content by creating an overview; adding information generated by oneself; and editing information by rephrasing or integrating information with prior knowledge.

and technologies (Ellis & Bliuc, 2019)), as such environments often demand high levels of independence and autonomy (Du & Hew, 2022). These environments frequently lack timely and personalised feedback to guide SRL processes effectively. While educators seek to support SRL, the absence of real-time, individualised feedback can discourage learners, hinder outcomes, and reduce the overall effectiveness of SRL (Azevedo et al., 2008; Heikkinen et al., 2023; Li et al., 2024). This highlights the need for solutions that promote SRL by balancing learner autonomy with adaptive guidance (Lim et al., 2023, 2024). An SRL recommender system that provides continuous, personalised scaffolding could address this challenge by adapting to individual progress and fostering SRL skills across online and blended learning environments.

Reinforcement Learning (RL) is a machine learning technique that enables an agent to learn through trial and error, using rewards as feedback to distinguish between positive and negative behaviours. In each episode, the agent seeks to maximise its total cumulative reward by learning an optimal sequence of actions from a given state, guided by a policy—the function that determines the best action based on the current state (Doroudi et al., 2019; Sutton & Barto, 2018). As the agent explores the environment, it refines its policy by receiving information in the form of states and rewards, making RL adaptable for real-time applications (Luo, 2020b; Nguyen & La, 2019).

RL problems are commonly modelled by a Markov Decision Process (MDP) (Sutton & Barto, 2018), formally defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. In education, RL has been used to provide personalised recommendations (Doroudi et al., 2019). To support SRL, Osakwe et al. (2023) proposed modelling the SRL context as an RL problem, mapping RL terms to SRL processes: i) the **agent** represents the student, ii) an **action** corresponds to an SRL process, iii) a **state** is the sequence of SRL processes in a study session, iv) a **policy** represents the learner’s SRL decision-making process, v) **rewards** are assessment scores or equivalent measures, and vi) an **episode** corresponds to a study session. In this framework, the policy value, $Q^\pi(s, a)$, depicts the learner’s expected assessment score when following policy π in state s , current SRL strategies taken, and taking the action a (the next SRL strategy). Osakwe et al. (2024) introduced a reward function combining assessment score and learning gain, derived from pre- and post-assessment scores based on Marx and Cummings (2007).

In classical RL, agents aim to discover an optimal policy to maximise rewards, requiring substantial data and computational resources (Zhu et al., 2023b). For instance, Osakwe et al. (2023)’s approach suggests that for personalised recommendations across multiple learners, multiple reward functions must be modelled to account for different learning outcomes (e.g., learning gain). This dramatically increases agents to be trained for each

reward, escalating data and resource demands, making classical RL impractical for personalised SRL recommendations.

Transfer Reinforcement Learning (TRL) addresses the problem of sample inefficiency in RL by leveraging the human-like ability to reuse previously learnt knowledge to solve new problems (Lazaric, 2012; Zhu et al., 2023b). For example, a person who knows how to ride a bicycle can relatively quickly apply that knowledge to ride a motorcycle. TRL is particularly effective for sequential and dynamic tasks (Zhu et al., 2023b). Similarly, TRL agents can reuse prior knowledge for adjusting to new reward structures, more efficiently. To apply previously acquired knowledge in new contexts, TRL defines a set of *source* contexts to discover and select relevant knowledge, and a set of *target* contexts where this knowledge can be applied without retraining models (Zhu et al., 2023b). One TRL approach used to identify what knowledge to reuse and how to apply is Successor Features (Barreto et al., 2017; Dayan, 1993).

The Successor Features approach is particularly well-suited for contexts that differ only in reward structures. Dayan (1993) and later Barreto et al. (2017) decouple the environment’s dynamics from the reward function by assuming the reward can be approximated as a linear combination of the discounted states occupancy. Thus, the reward function can be rewritten as, $r(s, a) \approx \mathbb{I}_s^\top \mathbf{w}$, where $\mathbb{I}_s$ is the one-hot vector representation for state s or a latent representation of the state as $r(s, a) \approx \phi(s)^\top \mathbf{w}$ (Borsa, Barreto, Quan, Mankowitz, et al., 2019; Kulkarni, Saeedi, et al., 2016). This means that the agent receives a reward based on the state s . Using this reward function, we can express the value function as a linear combination of the expected discounted frequency of states following a policy π ,

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_{\pi, s_{t+1} \sim p} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \\ &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \right]^\top \mathbf{w} \\ &= \boldsymbol{\psi}(s, a)^\top \mathbf{w} \end{aligned}$$

where $\boldsymbol{\psi}^\pi \in \mathbb{R}^{|S|}$ has been named the *Successor Feature Representation*.

In this thesis, we consider a family of SRL tasks modelled as MDPs $(\mathcal{S}, \mathcal{A}, p, r_i, \gamma)$ that share the same state space $\mathcal{S}$ (sequences of SRL processes), action space $\mathcal{A}$ (possible next SRL actions), and transition dynamics p , but differ in their reward functions r_i . Each reward function is assumed to admit a linear decomposition $r_i(s, a) = \phi(s)^\top \mathbf{w}_i$, where $\mathbf{w}_i$ captures task-specific learning objectives (e.g., assessment score, learning gain, or composite outcomes). Given a set of source tasks $\mathcal{T}_1, \dots, \mathcal{T}_n$ with learnt successor features,

the objective is to efficiently adapt to a new target task $\mathcal{T}_T$ with reward weights w_T without retraining a new agent from scratch. The central problem is therefore to determine whether the SF-based TRL framework can enable scalable, real-time personalised SRL recommendations by reusing learnt representations across heterogeneous educational reward structures.

While Successor Feature (SF)-based TRL has shown strong performance in robotics, planning, and control domains (Lazaric, 2012; Zhu et al., 2023b), its applicability to other domains remains underexplored. In particular, the use of TRL for modelling and supporting SRL processes has not been investigated. This thesis addresses this gap by adapting the SF-based TRL framework to educational settings and evaluating whether it can provide a scalable solution for personalised SRL support.

6.3 Method

This section outlines the experimental methodology. It presents the evaluation metrics, details the data collection process used for training, and specifies the environment and reward function employed for training the RL agents.

6.3.1 Research design and context

Participants The study involved 92 learners (age mean=14.18 y/o, SD=1.48) from two secondary schools in Colombia. Students who participated in the study obtained parental consents and also signed consents themselves.

Task design The learning task was divided into three parts. The *first part* involved a pre-test to assess students' prior domain knowledge (15 minutes to answer 15 multiple-choice questions). The *second part* provided training on the learning platform and tools used in the study (5 min). The *third part* was the main task, in which the students were asked to write a 300-400 word essay in Spanish, using the provided tools and using reading material provided about artificial intelligence and its applications in medicine within a 45-minute time limit. The design encouraged participants to self-regulate, focusing on relevant material and time management, rather than reading everything, to align with task instructions and the marking rubric. The essays were scored (*ES*) by two human markers using the following criteria: explanation of the AI concept and examples (9 points), AI current and future uses (6 points), and word count (1 point). Finally, the *fourth part* asked students to complete a post-test to assess student's learning gain (15 minutes to answer the same

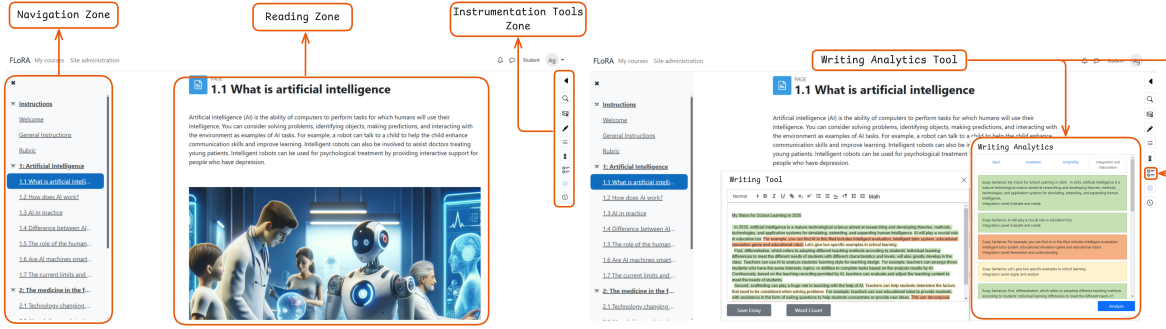


Figure 6.1: Learning platform that supports SRL.

questions is in the pre-test). The results of pre- and post- test were used to calculate the learning gain using the Normalised Learning Gain (NLG) following Marx and Cummings (2007) and Osakwe et al. (2024) as

$$NLG = \begin{cases} \frac{\text{post} - \text{pre}}{100 - \text{pre}} & \text{if Post} > \text{Pre}, \\ 0 & \text{if Post} = \text{Pre}, \\ \frac{\text{post} - \text{pre}}{\text{pre}} & \text{if post} < \text{pre} \end{cases}$$

Learning Platform To complete the task, students used an online platform designed to support SRL (Lämsä et al., 2025) and facilitate the analysis of granular trace data, which has been previously studied and validated (Fan, Lim, et al., 2022; Lämsä et al., 2025). The platform provided reading materials with a navigation bar for students to explore different pages. These included general instructions and a marking rubric, enabling students to familiarise themselves with the task requirements and assessment criteria. Besides, the platform offered tools for text annotation, note-taking, searching, essay writing, planning, and time management. An overview of the platform¹ is shown in Fig. 6.1.

Data collection and processing Of the 92 participants, we excluded the data from one participant from our analysis because the essay was unavailable for marking (no ES) and 27 due to incomplete learning gain data (missing pre- or post-test scores required for calculation). This left us with a total of 64 learners for data analysis. The trace data collected from the online tool included three main time-stamped components: 1) navigation logs, 2) mouse traces, including clicks, movements, and scrolls, and 3) keyboard strokes. The trace data was collected from the main task in the study. The average duration of task completion was 42 minutes, with a standard deviation of 4 minutes and 3 seconds.

¹<https://www.floraengine.org/home>

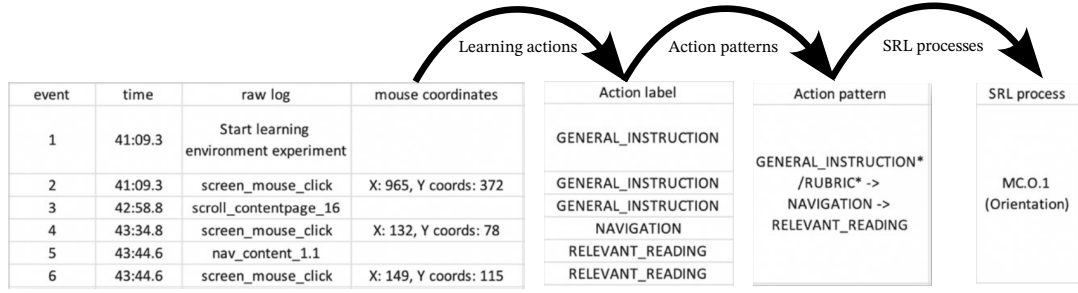


Figure 6.2: Parser from raw data to SRL processes.

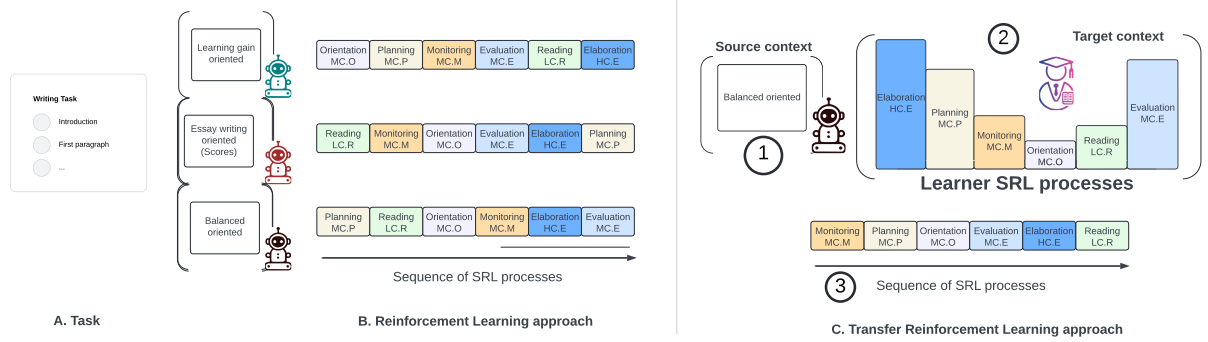


Figure 6.3: A. Task (e.g., essay writing). B. Overview of Osakwe et al. (2024) approach to address model SRL using RL. C. Propose approach to scale up implementations of TRL for SRL.

Using a SRL process trace parser, informed by the SRL framework of Bannert et al. (2014), we automatically categorised SRL processes into cognition and metacognition. The parser consisted of (i) an action library – which converted raw trace data to learning actions and (ii) a process library – which labelled learning actions into SRL processes. The validity of employing this specific trace parser and trace data has been tested and proven using thought-aloud data (Fan, Lim, et al., 2022; Fan, van der Graaf, et al., 2022), the overview of the parser is shown in Fig. 6.2. On average, these were the distribution of SRL processes detected for each student MC.O mean=1 minute and 41 seconds, SD=2 minutes and 4 seconds; MC.P mean=29 seconds, SD=30 seconds; MC.M mean=37 seconds, SD=51 seconds; MC.E mean=15 minutes and 2 seconds, SD=25 seconds; LC.R mean=13 minutes and 42 seconds, SD=05 minutes and 23 seconds; and HC.E/O mean=22 minutes, SD=6 minutes and 4 seconds.

6.3.2 Self-Regulated Learning Meets Transfer Reinforcement Learning and Successor Features

We propose using a TRL framework for an online task requiring students to self-regulate their learning, aiming to scaffold of SRL strategy recommendations. TRL enhances scalability by enabling the **transfer of SRL strategies** that maximise learning outcomes (optimal policies) from previous study sessions with specific rewards (**source contexts**) to new sessions with new reward structures (**target contexts**). This reduces the need to train multiple optimal policies, enhancing the efficiency and adaptability of SRL support.

TRL enables the adaptation of RL approaches, such as the one presented in Fig. 6.3-B. Osakwe et al. (2023) applied RL for SRL by modelling three agents, each tied to a reward function for a writing task. Their evaluation varied α values: $\alpha = 1$ focused entirely on **learning gain**, $\alpha = 0$ prioritised **assessment score**, and $\alpha = 0.5$ **balanced** the two.

While current RL for SRL requires training a new agent for each study session (target context), TRL enhances scalability by reusing optimal SRL strategies (policies) learnt in the source context. For SRL in particular, TRL allows only one or few agents to learn optimal policies for specific rewards (e.g., essay score) in an study session (source context), and reusing these optimal policies in new study sessions (target context). See our approach in Fig. 6.3-C.

In TRL for SRL, the *source context* corresponds to a collection of learners' SRL strategies used to learn optimal policies (sequences of SRL processes that optimise rewards) based on one or a few specific rewards, such as balanced rewards that prioritise essay score (ES) and normalised learning gain (NLG) to maximise learning outcomes (see Fig. 6.3-C 1). For the learning task in this study, the reward function could be defined based on the relative importance (α) of the essay score (ES) and the normalised learning gains (NLG). Setting $\alpha = 0.5$ results in a balanced reward function $R = 0.5 \times \text{NLG} + (1 - 0.5) \times \text{ES}$, which can define the source context. The relative importance can be adjusted according to the learning context and design.

The *target context* corresponds to any new learner attempting a writing task in a digital learning environment (Fig. 6.3-C 2). Learners engaging in new writing tasks are characterised by their reward functions, which may be defined either implicitly or explicitly. For instance, a target context where the learner focus solely on essay scores (ES) could be represented by the reward function $R = 0 \times \text{NLG} + (1 - 0) \times \text{ES}$, where $\alpha = 0$. When the rewards in the source and target contexts differ (e.g., when reward functions prioritise different learning outcomes), TRL mechanisms enable (i) the reuse of SRL strategies that

maximise learning outcomes (optimal policies) from the source context and (ii) the use of this prior knowledge along with the SRL processes performed by new learners to create an adapted policy that maximises the new reward, (Fig. 6.3-C 3).

Fundamentally, this distinction between source and target contexts constitutes a transfer learning setting, where knowledge is transferred across different reward functions. This aligns with the principles of TRL, particularly those fostered Successor Features, which enable efficient generalisation across tasks with varying reward structures.

6.4 Experiments and Analysis

6.4.1 Experiment Steup

Baseline The baseline in this study is a modified Deep Q-Learning (DQN) agent (Mnih et al., 2015). The DQN agent uses experience replay and mini-batch training, incorporating the Double Q-Network approach to enhance stability (Van Hasselt et al., 2016). As described in previous sections, the state representation is based on the frequency of SRL processes executed at each timestep. A Long Short-Term Memory (LSTM) network is integrated to model temporal dependencies in SRL process sequences. The final hidden state of the LSTM serves as the input to the Q-network, which then selects the optimal SRL process for the subsequent timestep.

Environmental Setting All required models and agents were implemented using the PyTorch library². We used data from 64 learners to train two LSTM neural networks. LSTMs were selected due to their effectiveness in handling sequential data, particularly for process prediction (e.g., (Camargo et al., 2019; Tax et al., 2017)). One network was trained to predict Normalised Learning Gains (NLG), while the other was trained to predict Essay Scores (ES).

For NLG prediction, we used a single-layer LSTM with 32 memory cells, achieving a mean squared error (MSE) of 0.01. For ES prediction, we employed a single-layer LSTM with 128 memory cells, obtaining a MSE of 0.02. These LSTM models then defined a reward function combining ES and the NLG, accounting their relative importance, as mentioned earlier.

For the *source context* in this experiment, we used a balanced reward function, as discussed throughout this chapter. To assess knowledge transfer, we evaluated the optimal

²<https://pytorch.org>

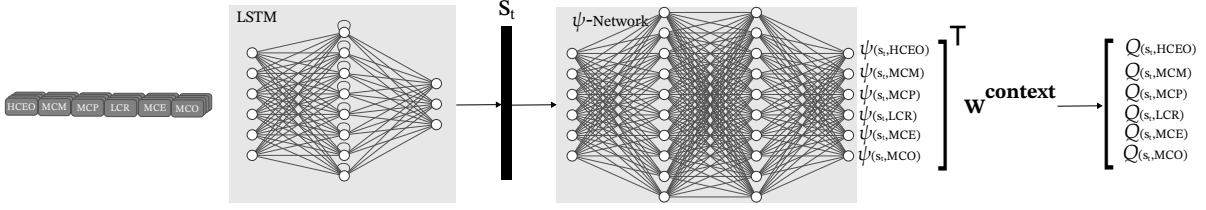


Figure 6.4: ψ -network architecture using LSTM network.

policies learnt in the *source context* under two different reward functions in the *target context*: one prioritising NLG and the other ES.

Agents trained exclusively in the source context for a total of 170 episodes, each episode consisted in 2700 timesteps, which represents the 45 minutes of the writing tasks by the learners. For training exploration, the ϵ -greedy strategy was initialised at 0.9 and gradually decayed to 0.1 over 4×10^5 training steps using linear decay. The replay buffer size was 4×10^5 , enabling learning from past experiences. The learnt policy is evaluated in the target tasks every 1×10^3 training steps, with an evaluation exploration $\epsilon = 0.1$. Additionally, Successor Feature Representation uses a separate learning rate of 1×10^{-2} to fit the weights ($\mathbf{w}$).

Architecture Similar to the baseline, our approach is adapted from DQN, particularly inspired by Universal Successor Feature Approximators (Borsa, Barreto, Quan, Mankowitz, et al., 2019), which integrates an LSTM to extract the dependencies between the sequence of SRL processes, to be used as the input for the Successor Features. Fig. 6.4 illustrates the Successor Feature Representation architecture used in our experiments. The ψ -Network outputs a vector of size $6 \times d$, where d is the dimension of the LSTM’s output state. The $\mathbf{w}^{\text{context}}$ is then used as dot product to compute the Q-values for each possible next SRL process.

For both DQN algorithms, we used the following hyperparameters. The learning rate for the value function networks was 1×10^{-3} , with a mini-batch size of 64. The discount factor (γ) was set to 1 to prioritise learning from late rewards, particularly those received at the end of an episode. The LSTM was a single-layer with 32 memory cells and latent feature state was 32. Target networks were updated every 100 training steps, ensuring stability during training.

Analysis To address this chapter’s RQ, we performed statistical comparisons and visualised SRL strategies distributions across source and target contexts to identify similarities and differences when reward functions vary. We segmented the data by context, select-

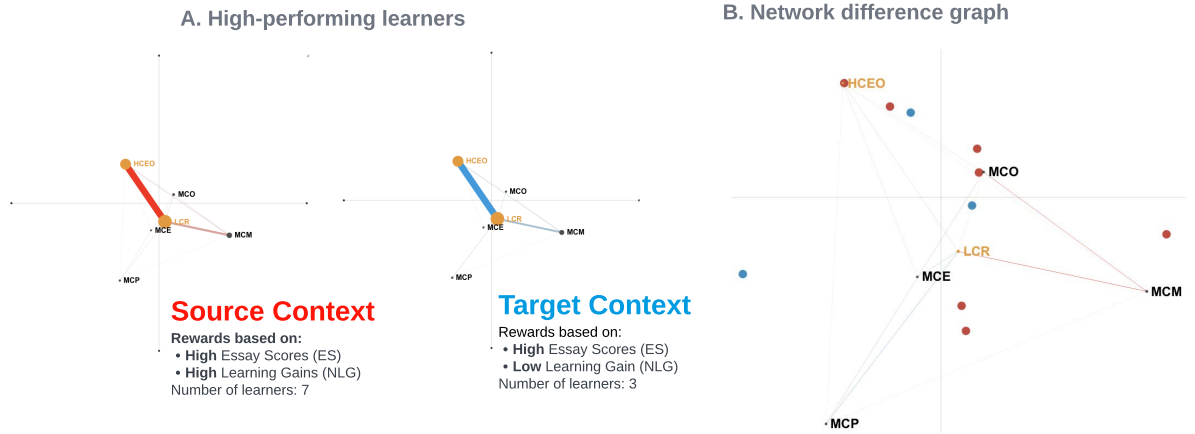


Figure 6.5: A. ENA representations of SRL behaviours for source and target task. B. Network difference graph.

ing high-performing learners whose SRL behaviours maximised both ES and NLG for the source and learners who maximised only ES for the target. We applied Epistemic Network Analysis (ENA), a widely used technique for modelling SRL strategies due to their temporal and sequential nature (Fan et al., 2023; Huang et al., 2023). ENA enabled us to compare SRL strategies distributions across contexts and examine pattern variations based on reward functions. We applied the ENA subtraction network graph algorithm to compute edge-weight differences between ENA representations, generating a network difference graph. To support statistical comparison, we extracted adjacency matrices from both ENA models and computed their Euclidean norm to quantify SRL strategy similarities across contexts.

To further address our **RQ**, we compared Successor Feature Representation, a TRL agent, with the baseline model to evaluate how well transferred SRL strategies improved task performance while considering computational resource demands. The Successor Feature Representation framework is ideal for implementing personalised recommendations to model individual learners’ SRL behaviours. We tested the Successor Feature Representation ability to generalise the learnt SRL strategies from the source to the target contexts. To assess the effectiveness of knowledge transfer, we measured *asymptotic performance*, a common reinforcement learning metric to evaluate transfer learning in unseen contexts (Zhu et al., 2023b).

6.4.2 Experiment Results and Analysis

Similarities and Differences across Contexts A key consideration when implementing TRL for SRL is the selection of the reward function for the source context. The reward function could be designed to represent generalities for the source and target context. Literal A in Fig. 6.5 presents the ENA representations for the source and target contexts in our study. In the source context, learners with high ES and high NLG exhibited more behaviours related to the link between High Cognitive – Elaboration/Organisation (HCEO) and Low Cognitive – Reading (LCR), with fewer transitions between LCR and Metacognitive - Monitoring (MCM). Other sequence transitions were less frequent. Likewise, learners in the target context, whose rewards focused primarily on ES with low NLG, frequently experienced transitions from HCEO to LCR and demonstrated behaviours from LCR to MCM.

The resultant network difference graph is shown in Fig. 6.5-B. This graph shows that both ENA representations are highly similar, with small differences found in the transitions between LCR, MCM, and MCO. The Euclidean norm resulted in a value of 0.062, further emphasising the high similarity between the two representations. *This similarity between the ENA representations of the source and target contexts suggests that rewards from the source context can be transferred to the target context*, even if the rewards differ, aligning with the foundations of TRL (Lazaric, 2012). While the rewards may vary, shared behaviour patterns indicate that training on a single reward in the source task remains applicable to the target task. This implies that, despite optimising for different reward structures, source and target behaviours are inherently similar.

Knowledge Transfer Effectiveness The implementation of the Successor Feature Representation allows a new learner (agent) to begin a study session, as shown in Fig. 6.3-C.2, where their SRL process behaviours were analysed to infer individual SRL behaviours and derive transferable rewards. Our approach demonstrated how knowledge can be transferred across learning contexts. This section presents the effectiveness metrics resulting from the computation of new optimal policies, which were derived based on the source context and the learner’s individual SRL behaviours in a target context.

Fig. 6.6 presents the learning curves for the source context (left) and the evaluation performance over evaluation episodes in both target contexts (middle and right). The left-most figure illustrates the training curve in the source context for both agents. As expected, both agents successfully learnt policies that maximise the reward, with a sharp improvement around the 100-th episode. The maximum reward achieved by the agents fell within episodes 120–140.

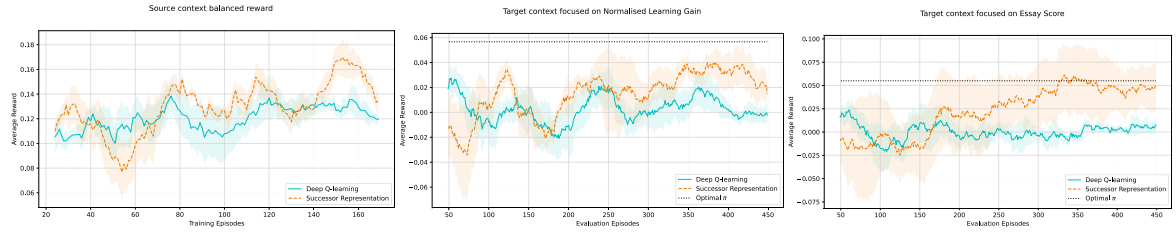


Figure 6.6: Left: Source context with balanced reward—learning steps vs. episode reward. Middle: Target context based on NLG. Right: Target context based on ES. Results are averaged over 6 random seeds, using a moving window of 50, and the shaded area representing 1 standard deviation.

The middle and right figures in Fig. 6.6 represent the target contexts. In the NLG-based context, reusing the baseline-derived policy did not yield a positive impact; instead, it resulted in slight negative transfer. This indicates that, although the policy improves in the source context, its performance deteriorates over time in the target context. Conversely, the Successor Feature Representation-based policy posits positive transfer. The baseline agent achieved an average reward of -0.002 over the last 50 evaluations and standard deviation of 0.002, whereas the Successor Feature Representation-based agent obtained a higher reward of 0.035, with a standard deviation of 0.0064. In the ES-based context, the baseline agent maintains a relatively stable performance over time. However, the Successor Feature Representation-based approach showed greater improvement. The average reward over the last 50 evaluations for the baseline agent was 0.005 with standard deviation 0.0023, whereas the Successor Feature Representation-based approach reached 0.05 with standard deviation 0.0032, demonstrating a clear advantage.

The target context in Fig. 6.6, also included the maximum reward (black dotted lines) achieved by an optimal policy. This maximum reward was computed by selecting the top 5% of the highest values per seed and averaging them to estimate the optimal policy’s peak performance per context. As shown in the Fig. 6.6, the Successor Feature Representation-based agent performs similarly to the optimal policy in the ES-based context. However, for the NLG-based context, its performance was 25% lower than the optimal policy. While the Successor Feature Representation approach achieved lower performance, it successfully leveraged previously learnt knowledge, avoiding the need to train from scratch, demonstrating a key advantage of TRL.

6.5 Summary

This chapter explored Transfer Reinforcement Learning (TRL) to provide scalable and personalised support for Self-Regulated Learning (SRL) by transferring optimised SRL strategies across different reward structures. Our findings indicate that strategies optimised for distinct rewards, such as Essay Score (ES) and Normalised Learning Gain (NLG), exhibit notable similarities, enabling policy reuse without retraining. The Successor Feature Representation-based approach further facilitates this transfer, demonstrating TRL's potential to overcome scalability limitations of traditional RL approaches, which rely on a one-policy-per-learner framework and require extensive computational resources (Osakwe et al., 2023, 2024). These results highlight TRL's capacity to generalise SRL strategies and provide personalised support throughout an entire task (*e.g.*, writing an essay).

However, two key limitations should be considered. First, reusing SRL strategies implies generalisation across learners, which may encourage behaviours that are effective for some learners but suboptimal for others. Future work could address this by incorporating learner-specific feedback into the reward function through reward shaping (Knox & Stone, 2009) or refining behavioural modelling using Imitation Learning (Torabi et al., 2018). Second, the analysis involved 92 learners completing a writing task. Expanding the dataset to a broader range of learners and SRL strategies could enhance robustness and improve NLG and ES prediction accuracy.

Future research on Successor Features in SRL should explore the latent space generated by LSTM networks and alternative methods for modelling the reward function. For instance, representing Successor Features as a vector in $\mathbb{R}^2$, where one dimension corresponds to NLG rewards and the other to ES, could improve knowledge transfer and generalisation across learning contexts. This approach aligns with the solution proposed by Barreto et al. (2020).

In conclusion, TRL mitigates the scalability challenges of traditional RL approaches and presents a promising framework for scaffolding SRL strategies in digital environments while fostering learner autonomy.

CONCLUSION

But as soon as I had finished my course of study, at which time it is usual to be admitted to the ranks of the well educated, I completely changed my opinion, for I found myself bogged down in so many doubts and errors, that it seemed to me that having set out to become learnt, I had derived no benefit from my studies, other than that of progressively revealing to myself how ignorant I was.

René Descartes, A Discourse on the Method

This chapter summarises the contributions of this thesis. It first revisits the findings according to each research question, then presents the key limitations of the proposed approaches, outlines directions for future work, and concludes with final remarks.

7.1 Findings

This thesis advances Successor Features (SFs) to make Transfer Reinforcement Learning (TRL) more suitable for real-world problem solving. The contributions form a coherent progression from generalisation across diverse dynamics, to uncertainty-aware adaptation, to alternative composition mechanisms, and finally to application in human learn-

ing. Together, they reposition SFs as a flexible and scalable transfer framework rather than a method constrained to fixed environments.

We began by addressing the challenge of generalising SFs across environments with diverse transition dynamics, **RQ-1**. Our results demonstrated that transformation-based methods can capture both shared and distinct dynamics, enabling SFs to extend beyond the restrictive setting of fixed environments. By applying probability density transformations, we showed that diverse dynamics can be represented in a common space, while an ensemble mechanism weighted source tasks to improve predictions in target tasks. This established the foundation for extending SFs beyond their original assumptions.

We then explored how SFs can be adapted across dissimilar transition dynamics and reward structures, **RQ-2**. To this end, we developed a Gaussian Process (GP)-based mechanism with tailored kernels that introduced epistemic uncertainty into the policy selection process. This approach enabled transfer in scenarios where classical methods fail, and theoretical analysis showed that the method generalises across a wide class of valid kernels. By modelling uncertainty directly in feature space, this stage advanced SFs from deterministic reuse to adaptive and uncertainty-aware transfer.

The next stage focused on composition, introducing a hierarchical mechanism for adaptive SF composition as an alternative to Generalised policy improvement (GPI), **RQ-3**. By constructing high-level state and action spaces from source-task SFs, the approach enabled behaviour adaptation to unseen tasks and outperformed baselines across benchmark environments such as four-room, reacher, and foraging. Principal component analysis confirmed that high-level state space evolves in meaningful ways, reducing complexity and improving decision efficiency. This contribution demonstrated that hierarchical composition can be an alternative for approaches based on GPI.

Finally, we extended TRL beyond control and robotics to the domain of human learning, **RQ-4**. We investigated Self-regulated learning (SRL) as a critical skill in education and demonstrated that TRL provides a scalable alternative where classical Reinforcement Learning (RL) struggles. Policies optimised for distinct outcomes, such as essay scores and normalised learning gain, exhibited a transferable structure, enabling the reuse of SRL strategies without retraining. This final stage showed that the proposed advances are not confined to benchmark domains but can support real-world, human-centred applications.

7.2 Limitations

In addressing **RQ-1** on generalising SFs across diverse transition dynamics, several constraints emerged. Learning stable transformed feature representations proved to be sensitive to the expressiveness of the chosen transformations, and the agents did not consistently converge to optimal policies. This suggests that while generalisation across diverse dynamics is feasible, achieving stable optimality requires more powerful feature-extraction mechanisms and richer transformation models, such as normalising flows, to better capture complex real-world dynamics.

RQ-2 exposed challenges related to uncertainty-aware adaptation. Although this work demonstrated the value of uncertainty-aware adaptation, it also revealed practical constraints. Gaussian Processes introduce cubic computational complexity, limiting scalability in high-dimensional settings or large task collections. Furthermore, integrating epistemic uncertainty within GPI yielded mixed empirical results, highlighting the challenge of balancing stability and exploration. These limitations indicate the need for approximation methods, multi-output extensions, and more scalable uncertainty modelling strategies.

In **RQ-3**, which explored hierarchical composition beyond GPI, computational overhead became more pronounced. Joint learning SFs and high-level policies increased training complexity, and scaling the approach to many tasks or high-dimensional feature spaces remains challenging. Moreover, formal performance guarantees comparable to those of GPI have not yet been established for the proposed mechanism. These constraints limit immediate deployment in large-scale settings.

Finally, **RQ-4** revealed domain-specific constraints when extending TRL to SRL. The datasets used were limited in size and diversity, and modelling human learning behaviours introduces ethical and fairness considerations. Additionally, reward design in educational contexts remains sensitive to the chosen performance metrics, increasing the risk of promoting strategies that benefit some learners while disadvantaging others. Broader empirical validation in diverse educational settings is required.

Across all stages, a common limitation lies in the absence of formal regret bounds and comprehensive theoretical guarantees under fully diverse dynamics. While empirical evidence supports the proposed methods, further theoretical analysis is required to fully characterise convergence properties and performance guarantees.

7.3 Future Work

The findings of this thesis open up several directions for further investigation, both to strengthen the proposed methods and to expand their applicability.

In relation to **RQ-1**, future research should explore more expressive transformation families to better model highly non-linear and stochastic transition dynamics. In particular, integrating normalising flows or invertible neural architectures may enable richer density transformations and improve stability when generalising across complex environments (Rezende & Mohamed, 2015). Extending the framework to partially observable settings would further strengthen its relevance in the real-world.

For **RQ-2**, improving scalability and uncertainty-aware adaptation remains central. Future research should investigate sparse and approximate GP methods (Álvarez et al., 2010; Lázaro-Gredilla & Figueiras-Vidal, 2009; Rasmussen & Williams, 2005), as well as alternative uncertainty-aware approaches such as Bayesian neural network models (Sendera et al., 2025). Extending the kernel design to multi-task and multi-output settings may allow for more efficient modelling of relationships among diverse transition dynamics (Álvarez et al., 2010). Further theoretical analysis of uncertainty propagation within policy selection remains an important direction.

With respect to **RQ-3**, the advancement of hierarchical composition requires mechanisms that reduce computational overhead while preserving behavioural flexibility. Task clustering (Madarasz and Behrens, 2019) and dimensionality reduction of successor features using techniques such as Principal Component Analysis or inducing-point methods such as in GP (Álvarez et al., 2010) may help improve scalability. Developing regret-based guarantees or performance bounds for the proposed composition strategy would strengthen its theoretical foundations. Hybrid approaches combining hierarchical composition with GPI may also offer improved efficiency.

Finally, extending **RQ-4** into educational settings requires large-scale empirical validation across diverse populations of learners and instructional contexts. Integrating student-specific reward shaping (Ng et al., 1999) and richer behavioural modelling, including imitation learning or inverse reinforcement learning approaches (Torabi et al., 2018). Large-scale empirical validation in authentic learning environments is necessary to assess the long-term impact, fairness, and applicability of TRL-based support solutions.

Together, these directions outline a pathway toward scalable, adaptable, uncertainty-aware, and human-centred transfer reinforcement learning frameworks that extend beyond current theoretical and empirical boundaries.

7.4 Concluding Remarks

Together, these contributions form a coherent research trajectory, beginning with the problem of generalisation across diverse transition dynamics, advancing through adaptation across diverse dynamics and uncertainty-aware mechanisms, moving to alternative composition strategies, and culminating in an application that bridges machine learning with human learning. Across these stages, this thesis demonstrates that SF-based TRL can move beyond the restrictive assumptions of fixed dynamics and deterministic reuse.

More broadly, this work repositions SFs from a representation transfer mechanism suited primarily to controlled environments to a more flexible framework capable of handling diverse dynamics, modelling epistemic uncertainty, and supporting real-world decision-making. By addressing generalisation, adaptation, composition, and application within a unified perspective, the thesis establishes conceptual and practical foundations for scalable TRL.

While challenges remain in scalability, optimality, and formal guarantees, the advances presented here indicate that SFs offer a practical direction for reinforcement learning beyond narrowly constrained settings. This opens opportunities for future research that extends transfer reinforcement learning beyond laboratory benchmarks and into dynamic, human-centred environments.

BIBLIOGRAPHY

- Abdolshah, M., Le, H., George, T. K., Gupta, S., Rana, S., & Venkatesh, S. (2021). A new representation of successor features for transfer across dissimilar environments. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (pp. 1–9, Vol. 139). PMLR.
- Adamczyk, J., Makarenko, V., Arriojas, A., Tiomkin, S., & Kulkarni, R. V. (2023, July). Bounding the optimal value function in compositional reinforcement learning. In R. J. Evans & I. Shpitser (Eds.), *Proceedings of the thirty-ninth conference on uncertainty in artificial intelligence* (pp. 22–32, Vol. 216). PMLR.
- Alegre, L. N., Bazzan, A., & Silva, B. C. D. (2022, July). Optimistic linear support and successor features as a basis for optimal policy transfer. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th international conference on machine learning* (pp. 394–413, Vol. 162). PMLR.
- Alegre, L. N., Bazzan, A. L. C., Nowe, A., & da Silva, B. C. (2023a). Multi-step generalized policy improvement by leveraging approximate models. *37th Conference on Neural Information Processing Systems*.
- Alegre, L. N., Bazzan, A. L. C., Nowe, A., & da Silva, B. C. (2023b). Multi-step generalized policy improvement by leveraging approximate models. *37th Conference on Neural Information Processing Systems*.
- Álvarez, M., Luengo, D., Titsias, M., & Lawrence, N. D. (2010, May). Efficient multioutput gaussian processes through variational inducing kernels. In Y. W. Teh & M. Titterton (Eds.), *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 25–32, Vol. 9). PMLR.
- Alver, S., & Precup, D. (2022). Constructing a good behavior basis for transfer using generalized policy updates. *International Conference on Learning Representations*.

- Azevedo, R., Moos, D. C., Greene, J. A., Winters, F. I., & Cromley, J. G. (2008). Why is externally-facilitated regulated learning more effective than self-regulated learning with hypermedia? *ETR&D*, 56, 45–72.
- Bannert, M. (2007). *Metakognition beim lernen mit hypermedien*. Waxmann.
- Bannert, M., Reimann, P., & Sonnenberg, C. (2014). Process mining techniques for analysing patterns and strategies in students' self-regulated learning. *Metacognition and learning*, 9, 161–185.
- Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., & Munos, R. (2018). Transfer in deep reinforcement learning using successor features and generalised policy improvement. *Proceedings of the 35th International Conference on Machine Learning*.
- Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., & Silver, D. (2017). Successor features for transfer in reinforcement learning. *Advances in Neural Information Processing Systems (NIPS)*, 30.
- Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Hunt, J., Mourad, S., Silver, D., & Precup, D. (2019). The option keyboard combining skills in reinforcement learning. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*.
- Barreto, A., Hou, S., Borsa, D., Silver, D., & Precup, D. (2020). Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48), 30079–30087. <https://doi.org/10.1073/pnas.1907370117>
- Beck, J., Vuorio, R., Liu, E. Z., Xiong, Z., Zintgraf, L., Finn, C., & Whiteson, S. (2025). A Tutorial on Meta-Reinforcement Learning. *Foundations and Trends® in Machine Learning*, 18(2-3), 224–384. <https://doi.org/10.1561/22000000080>
- Bonilla, E. V., Chai, K., & Williams, C. (2007). Multi-task gaussian process prediction. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 20). Curran Associates, Inc.

- Borsa, D., Barreto, A., Quan, J., Mankowitz, D. J., van Hasselt, H., Munos, R., Silver, D., & Schaul, T. (2019). Universal successor features approximators. *International Conference on Learning Representations*.
- Borsa, D., Barreto, A., Quan, J., Mankowitz, D., Munos, R., Van Hasselt, H., Silver, D., & Schaul, T. (2019). Universal successor features approximators. *ICRL 2019*.
- Brantley, K., Mehri, S., & Gordon, G. J. (2021). Successor feature sets: Generalizing successor representations across policies. *AAAI Conference on Artificial Intelligence*, 35, 11774–11781.
- Butler, D. L., & Winne, P. H. (1995). Feedback and self-regulated learning: A theoretical synthesis. *Review of Educational Research*, 65(3), 245–281.
- Camargo, M., Dumas, M., & González-Rojas, O. (2019). Learning accurate lstm models of business processes. *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*, 286–302.
- Candelieri, A., Ponti, A., & Archetti, F. (2025). Gaussian Process regression over discrete probability measures: On the non-stationarity relation between Euclidean and Wasserstein Squared Exponential Kernels. *Journal of Global Optimization*, 92(2), 253–278. <https://doi.org/10.1007/s10898-025-01463-y>
- Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1), 41–75. <https://doi.org/10.1023/A:1007379606734>
- Carvalho, W., Saraiva, A., Filos, A., Lampinen, A. K., Matthey, L., Lewis, R. L., Lee, H., Singh, S., Rezende, D. J., & Zoran, D. (2023). Combining behaviors with the successor features keyboard. *Proceedings of the 37th International Conference on Neural Information Processing Systems*.
- Cheng, G., Dong, L., Cai, W., & Sun, C. (2023). Multi-Task Reinforcement Learning With Attention-Based Mixture of Experts. *IEEE Robotics and Automation Letters*, 8(6), 3812–3819. <https://doi.org/10.1109/LRA.2023.3271445>
- Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 4759–4770.

- Chua, R., Ghosh, A., Kaplanis, C., Richards, B. A., & Precup, D. (2024). Learning successor features the simple way. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (pp. 49957–50030, Vol. 37). Curran Associates, Inc.
- Cornish, R., Caterini, A., Deligiannidis, G., & Doucet, A. (2020, July). Relaxing bijectivity constraints with continuously indexed normalising flows. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (pp. 2133–2143, Vol. 119). PMLR.
- Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4), 613–624.
<https://doi.org/10.1162/neco.1993.5.4.613>
- De Cothi, W., & Barry, C. (2020). Neurobiological successor features for spatial navigation. *Hippocampus*, 30(12), 1347–1355. <https://doi.org/10.1002/hipo.23246>
- D’Eramo, C., Tateo, D., Bonarini, A., Restelli, M., & Peters, J. (2020). Sharing knowledge in multi-task deep reinforcement learning. *International Conference on Learning Representations*.
- Dimitrakakis, C., & Ortner, R. (2022). *Decision Making Under Uncertainty and Reinforcement Learning: Theory and Algorithms*. Springer International Publishing. <https://doi.org/10.1007/978-3-031-07614-5>
- Doroudi, S., Aleven, V., & Brunskill, E. (2019). Where’s the reward? *Int J Artif Intell Educ*, 568–620.
- Draxler, F., Wahl, S., Schnörr, C., & Köthe, U. (2024). On the universality of volume-preserving and coupling-based normalizing flows. *Proceedings of the 41st International Conference on Machine Learning*.
- Du, J., & Hew, K. F. T. (2022). Using recommender systems to promote self-regulated learning in online education settings: Current knowledge gaps and suggestions for future research. *JRTE*, 54(4), 557–580.

- Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P. (2016). RL²: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.
- Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110(9), 2419–2468.
- Ellis, R. A., & Bliuc, A.-M. (2019). Exploring new elements of the student approaches to learning framework: The role of online learning technologies in student learning. *Active Learning in Higher Education*, 20(1), 11–24.
<https://doi.org/10.1177/1469787417721384>
- Eysenbach, B., & Levine, S. (2022). Maximum entropy RL (provably) solves some robust RL problems. *International Conference on Learning Representations*.
- Fan, Y., Lim, L., Van der Graaf, J., Kilgour, J., Raković, M., Moore, J., Molenaar, I., Bannert, M., & Gašević, D. (2022). Improving the measurement of self-regulated learning using multi-channel data. *Metacognition and Learning*, 17(3), 1025–1055.
- Fan, Y., Tan, Y., Raković, M., Wang, Y., Cai, Z., Shaffer, D. W., & Gašević, D. (2023). Dissecting learning tactics in mooc using ordered network analysis. *J. Comput. Assist. Learn.*, 39(1), 154–166.
- Fan, Y., van der Graaf, J., Lim, L., Raković, M., Singh, S., Kilgour, J., Moore, J., Molenaar, I., Bannert, M., & Gašević, D. (2022). Towards investigating the validity of measurement of self-regulated learning based on trace data. *Metacognition and Learning*, 17(3), 949–987.
- Filos, A., Lyle, C., Gal, Y., Levine, S., Jaques, N., & Farquhar, G. (2021, July). PsiPhi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (pp. 3305–3317, Vol. 139). PMLR.
- Finn, C., Abbeel, P., & Levine, S. (2017, August). Model-agnostic meta-learning for fast adaptation of deep networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 1126–1135, Vol. 70). PMLR.

- Ganaie, M., Hu, M., Malik, A., Tanveer, M., & Suganthan, P. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115, 105151.
- Gao, M., Hugenholtz, C. H., Fox, T. A., Kucharczyk, M., Barchyn, T. E., & Nesbit, P. R. (2021). Weather constraints on global drone flyability. *Scientific Reports*, 11(1). <https://doi.org/10.1038/s41598-021-91325-w>
- Ghugare, R., & Eysenbach, B. (2025). Normalizing flows are capable models for continuous control. *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=EJ34X5VWwu>
- Gimelfarb, M., Barreto, A., Sanner, S., & Lee, C.-G. (2021). Risk-aware transfer in reinforcement learning using successor features. <https://doi.org/10.48550/ARXIV.2105.14127>
- Goumiri, I. R., Priest, B. W., & Schneider, M. D. (2020). Reinforcement learning via gaussian processes with neural network dual kernels. *IEEE Conference on Games (CoG)*, 1–8.
- Gupta, A., Devin, C., Liu, Y., Abbeel, P., & Levine, S. (2017). Learning invariant feature spaces to transfer skills with reinforcement learning. *International Conference on Learning Representations*.
- Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., & Levine, S. (2018). Composable deep reinforcement learning for robotic manipulation. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6244–6251.
- Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, 1861–1870.
- Han, D., & Tschiatschek, S. (2022). Option Transfer and SMDP Abstraction with Successor Features. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 3036–3042. <https://doi.org/10.24963/ijcai.2022/421>
- Hansen, S., Dabney, W., Barreto, A., Van de Wiele, T., Warde-Farley, D., & Mnih, V. (2020). Fast task inference with variational intrinsic successor features. *International Conference on Learning Representations (ICLR)*.

- Heikkinen, S., Saqr, M., Malmberg, J., & Tedre, M. (2023). Supporting self-regulated learning with learning analytics interventions—a systematic literature review. *EAIT*, 28(3), 3059–3088.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Hu, Y., Tavakoli, M., & Jin, J. (2025). Pretraining using comparable human activities of daily living dataset in robotic surgical task learning. *IEEE Transactions on Medical Robotics and Bionics*, 7(3), 1111–1124. <https://doi.org/10.1109/TMRB.2025.3589768>
- Huang, X., Li, S., Wang, T., Pan, Z., & Lajoie, S. P. (2023). Exploring the co-occurrence of students’ learning behaviours and reasoning processes in an intelligent tutoring system: An epistemic network analysis. *J. Comput. Assist. Learn*, 39(5).
- Hunt, J., Barreto, A., Lillicrap, T., & Heess, N. (2019, June). Composing entropic policies using divergence correction. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 2911–2920, Vol. 97). PMLR.
- Hutsebaut-Buysse, M., Mets, K., & Latré, S. (2022). Hierarchical Reinforcement Learning: A Survey and Open Research Challenges. *Machine Learning and Knowledge Extraction*, 4(1), 172–221. <https://doi.org/10.3390/make4010009>
- Janner, M., Mordatch, I., & Levine, S. (2020). γ -models: Generative temporal difference learning for infinite-horizon prediction. *Advances in Neural Information Processing Systems (NIPS)*, 33, 1724–1735.
- Jeong, J., Wang, X., Gimelfarb, M., Kim, H., abdulhai, B., & Sanner, S. (2023). Conservative bayesian model-based value expansion for offline policy optimization. *The Eleventh International Conference on Learning Representations*.
- Ju, H., Juan, R., Gomez, R., Nakamura, K., & Li, G. (2022). Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, 4(12), 1077–1087. <https://doi.org/10.1038/s42256-022-00573-6>

- Kim, J., Xuan, J., Liang, C., & Hussain, F. (2024). Decoupling exploration and exploitation for unsupervised pre-training with successor features. *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–8.
<https://doi.org/10.1109/IJCNN60899.2024.10651424>
- Knox, W. B., & Stone, P. (2009). Interactively shaping agents via human reinforcement: The tamer framework. *K-CAP*, 9–16.
- Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 1238–1274.
<https://doi.org/10.1177/0278364913495721>
- Kobyzev, I., Prince, S. J., & Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11), 3964–3979.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90.
<https://doi.org/10.1145/3065386>
- Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., & Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Proceedings of the 30th International Conference on Neural Information Processing Systems*.
- Kulkarni, T. D., Saeedi, A., Gautam, S., & Gershman, S. J. (2016). Deep successor reinforcement learning.
- Kuric, D., Infante, G., Gómez, V., Jonsson, A., & van Hoof, H. (2024). Planning with a learned policy basis to optimally solve complex tasks. *34th International Conference on Automated Planning and Scheduling*.
- Lämsä, J., de Mooij, S., Aksela, O., Athavale, S., Bistolfi, I., Azevedo, R., Bannert, M., Gasevic, D., Molenaar, I., & Järvelä, S. (2025). Measuring secondary education students' self-regulated learning processes with digital trace data. *Learning and Individual Differences*, 118, 102625.

- Lazaric, A. (2012). Transfer in Reinforcement Learning: A Framework and a Survey. In M. Wiering & M. Van Otterlo (Eds.), *Reinforcement Learning* (pp. 143–173, Vol. 12). Springer Berlin Heidelberg.
https://doi.org/10.1007/978-3-642-27645-3_5
- Lázaro-Gredilla, M., & Figueiras-Vidal, A. (2009). Inter-domain gaussian processes for sparse inference using inducing features. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 22). Curran Associates, Inc.
- Le Lan, C., Tu, S., Oberman, A., Agarwal, R., & Bellemare, M. G. (2022). On the generalization of representations in reinforcement learning. *International Conference on Artificial Intelligence and Statistics*, 151, 4132–4157.
- Lee, J., Kim, R., Koh, Y., & Kang, J. (2019). Global stock market prediction based on stock chart images using deep q-network. *IEEE Access*, 7, 167260–167277.
- Lehnert, L., Frank, M. J., & Littman, M. L. (2022). Reward-Predictive Clustering.
<https://doi.org/10.48550/ARXIV.2211.03281>
- Lehnert, L., & Littman, M. L. (2020). Successor features combine elements of model-free and model-based reinforcement learning. *Journal of Machine Learning Research*, 21(196), 1–53. <http://jmlr.org/papers/v21/19-060.html>
- Li, T., Fan, Y., Srivastava, N., Zeng, Z., Li, X., Khosravi, H., Tsai, Y.-S., Swiecki, Z., & Gašević, D. (2024). Analytics of planning behaviours in self-regulated learning: Links with strategy use and prior knowledge. *LAK24*, 438–449.
- Li, T., Fan, Y., Tan, Y., Wang, Y., Singh, S., Li, X., Raković, M., Van Der Graaf, J., Lim, L., Yang, B., et al. (2023). Analytics of self-regulated learning scaffolding: Effects on learning processes. *Frontiers in psychology*, 14, 1206696.
- Lim, L., Bannert, M., van der Graaf, J., Fan, Y., Rakovic, M., Singh, S., Molenaar, I., & Gašević, D. (2024). How do students learn with real-time personalized scaffolds? *British Journal of Educational Technology*, 55(4), 1309–1327.
- Lim, L., Bannert, M., van der Graaf, J., Singh, S., Fan, Y., Surendrannair, S., Rakovic, M., Molenaar, I., Moore, J., & Gašević, D. (2023). Effects of real-time analytics-based

- personalized scaffolds on students' self-regulated learning. *Comput. Hum. Behav.*, 139.
- Lin, Y., Liu, Y., Lin, F., Zou, L., Wu, P., Zeng, W., Chen, H., & Miao, C. (2024). A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10), 13164–13184. <https://doi.org/10.1109/TNNLS.2023.3280161>
- Liu, H., & Abbeel, P. (2021a). Aps: Active pretraining with successor features. *International Conference on Machine Learning (ICML)*, 139, 6736–6747.
- Liu, H., & Abbeel, P. (2021b). Behavior from the void: Unsupervised active pre-training. In A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems*.
- Liu, Y. T., & Ahmad, A. (2023). Multi-task reinforcement learning in continuous control with successor feature-based concurrent composition.
- Liu, Y. T., & Ahmad, A. (2024). Multi-task reinforcement learning in continuous control with successor feature-based concurrent composition. *2024 European Control Conference (ECC)*, 3860–3867. <https://doi.org/10.23919/ECC64448.2024.10591301>
- Luo, S. (2020a). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing*, 91, 106208.
- Luo, S. (2020b). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing*, 91, 106208. <https://www.sciencedirect.com/science/article/pii/S1568494620301484>
- Ma, C., Ashley, D. R., Wen, J., & Bengio, Y. (2020). Universal successor features for transfer reinforcement learning.
- Ma, C., Wen, J., & Bengio, Y. (2018). Universal successor representations for transfer reinforcement learning. *International Conference on Learning Representations (ICLR)*.

- Machado, M. C., Barreto, A., Precup, D., & Bowling, M. (2023). Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80), 1–69. <http://jmlr.org/papers/v24/21-1213.html>
- Madarasz, T., & Behrens, T. (2019). Better transfer learning with inferred successor maps. *Advances in Neural Information Processing Systems (NIPS)*, 32.
- Malekzadeh, P., Salimibeni, M., Hou, M., Mohammadi, A., & Plataniotis, K. N. (2022). AKF-SR: Adaptive Kalman filtering-based successor representation. *Neurocomputing*, 467, 476–490. <https://doi.org/10.1016/j.neucom.2021.10.008>
- Marx, J. D., & Cummings, K. (2007). Normalized change. *American Journal of Physics*, 75(1), 87–91.
- Micchelli, C. A., Xu, Y., & Zhang, H. (2006). Universal Kernels. *Journal of Machine Learning Research*, 7(95), 2651–2667.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Momennejad, I., Russek, E. M., Cheong, J. H., Botvinick, M. M., Daw, N. D., & Gershman, S. J. (2017). The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9), 680–692.
- Mozifian, M., Fox, D., Meger, D., Ramos, F., & Garg, A. (2022). Learning successor feature representations to train robust policies for multi-task learning. *Deep Reinforcement Learning Workshop in Conference on Neural Information Processing Systems*.
- Mu, N., Luan, Y., Yang, Y., & Jia, Q.-s. (2024). S-EPOA: Overcoming the indivisibility of annotations with skill-driven preference-based reinforcement learning.
- Nachum, O., Gu, S., Lee, H., & Levine, S. (2018). Data-efficient hierarchical reinforcement learning. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 3307–3317.

- Neftci, E. O., & Averbeck, B. B. (2019). Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, 1(3), 133–143.
<https://doi.org/10.1038/s42256-019-0025-4>
- Nemecek, M., & Parr, R. (2021, July). Policy caches with successor features. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (pp. 8025–8033, Vol. 139). PMLR.
<https://proceedings.mlr.press/v139/nemecek21a.html>
- Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the Sixteenth International Conference on Machine Learning*, 278–287.
- Nguyen, H., & La, H. (2019). Review of deep reinforcement learning for robot manipulation. *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 590–595. <https://doi.org/10.1109/IRC.2019.00120>
- Ollivier, Y. (2025, February). Which Features are Best for Successor Features?
<https://doi.org/10.48550/arXiv.2502.10790>
- Osakwe, I., Chen, G., Fan, Y., Rakovic, M., Li, X., Singh, S., Molenaar, I., Bannert, M., & Gašević, D. (2023). Reinforcement learning for automatic detection of effective strategies for self-regulated learning. *CAEAI*, 5, 100181.
- Osakwe, I., Chen, G., Fan, Y., Rakovic, M., Singh, S., Lim, L., van der Graaf, J., Moore, J., Molenaar, I., Bannert, M., Whitelock-Wainwright, A., & Gašević, D. (2024). Towards prescriptive analytics of self-regulated learning strategies: A reinforcement learning approach. *BJET*, 55(4), 1747–1771.
- Panadero, E. (2017). A review of self-regulated learning: Six models and four directions for research. *Frontiers in psychology*, 8, 422.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57), 1–64.

- Park, S., Kreiman, T., & Levine, S. (2024). Foundation policies with Hilbert representations. *Proceedings of the 41st International Conference on Machine Learning*.
- Peng, X. B., Chang, M., Zhang, G., Abbeel, P., & Levine, S. (2019). MCP: Learning composable hierarchical control with multiplicative compositional policies. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc.
- Plaen, H. D., Fanuel, M., & Suykens, J. A. K. (2020). Wasserstein exponential kernels. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–6. <https://doi.org/10.1109/IJCNN48605.2020.9207630>
- Puterman, M. L. (1990). Chapter 8 markov decision processes. In *Stochastic models* (pp. 331–434, Vol. 2). Elsevier. <https://www.sciencedirect.com/science/article/pii/S0927050705801720>
- Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming* (1st). John Wiley & Sons, Inc.
- Raković, M., Bannert, M., Molenaar, I., Winne, P. H., & Gašević, D. (2024). In conversation: Bannert, molenaar, & winne—multiple perspectives on researching and supporting self-regulated learning via analytics. In *Theory informing and arising from learning analytics* (pp. 57–69). Springer.
- Ramesh, R., Tomar, M., & Ravindran, B. (2019). Successor options: An option discovery framework for reinforcement learning. *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 3304–3310.
- Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., & Russell, S. (2021). Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Proceedings of the 35th International Conference on Neural Information Processing Systems*.
- Rasmussen, C. E., & Williams, C. K. I. (2005, November). *Gaussian Processes for Machine Learning*. The MIT Press. <https://doi.org/10.7551/mitpress/3206.001.0001>

- Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. *International Conference on Machine Learning (ICML)*, 37, 1530–1538.
- Rosenblatt, F. (1957). *The perceptron: A perceiving and recognizing automaton* (tech. rep. No. Project Para Report No. 85-460-1). Cornell Aeronautical Laboratory.
- Russek, E. M., Momennejad, I., Botvinick, M. M., Gershman, S. J., & Daw, N. D. (2021). Neural evidence for the successor representation in choice evaluation.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3), 210–229.
<https://doi.org/10.1147/rd.33.0210>
- Scarpellini, G., Konyushkova, K., Fantacci, C., Paine, T., Chen, Y., & Denil, M. (2024). π^2 : Policy representation with successor features. *The Twelfth International Conference on Learning Representations*.
- Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. *International Conference on Machine Learning (ICML)*, 37, 1312–1320.
- Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., & Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839), 604–609. <https://doi.org/10.1038/s41586-020-03051-4>
- Seid, A. M., Boateng, G. O., Anokye, S., Kwantwi, T., Sun, G., & Liu, G. (2021). Collaborative computation offloading and resource allocation in multi-uav-assisted iot networks: A deep reinforcement learning approach. *IEEE Internet of Things Journal*, 8(15), 12203–12218.
- Sendera, M., Sorkhei, A., & Kuśmierczyk, T. (2025). Revisiting the equivalence of Bayesian neural networks and Gaussian processes: On the importance of learning activations. *Proceedings of the Forty-First Conference on Uncertainty in Artificial Intelligence*.

- Shah, D., Toshev, A. T., Levine, S., & brian ichter. (2022). Value function spaces: Skill-centric state abstractions for long-horizon reasoning. *International Conference on Learning Representations*.
- Sodhani, S., Zhang, A., & Pineau, J. (2021, July). Multi-task reinforcement learning with context-based representations. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (pp. 9767–9779, Vol. 139). PMLR.
- Srivastava, N., Fan, Y., Rakovic, M., Singh, S., Jovanovic, J., Van Der Graaf, J., Lim, L., Surendrannair, S., Kilgour, J., Molenaar, I., et al. (2022). Effects of internal and external conditions on strategies of self-regulated learning: A learning analytics study. *LAK22*, 392–403.
- Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, 2377–2385.
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second). The MIT Press.
- Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., & Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 761–768.
- Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 181–211. [https://doi.org/10.1016/S0004-3702\(99\)00052-1](https://doi.org/10.1016/S0004-3702(99)00052-1)
- Szoke, L., Aradi, S., Becsi, T., & Gaspar, P. (2022). Skills to drive: Successor features for autonomous highway pilot. *IEEE Transactions on Intelligent Transportation Systems*, 1–12.
- Szoke, L., Aradi, S., & Tettamanti, T. (2022). Investigating successor features in the domain of autonomous vehicle control. *Transportation Research Procedia*, 62, 181–188.

- Szoke, L., Aradi, S., & Bécsi, T. (2023). Traffic signal control with successor feature-based deep reinforcement learning agent. *Electronics*, 12(6).
- Tao, Y., Genc, S., Chung, J., Sun, T., & Mallya, S. (2021). Repaint: Knowledge transfer in deep reinforcement learning. In M. Meila & T. Zhang (Eds.), *International conference on machine learning (icml)* (pp. 10141–10152, Vol. 139).
- Tasfi, N., Santana, E., Liboni, L., & Capretz, M. (2023). Dynamic successor features for transfer learning and guided exploration. *Knowledge-Based Systems*, 267, 110401.
- Tasse, G. N., James, S., & Rosman, B. (2020). A boolean task algebra for reinforcement learning. *Proceedings of the 34th International Conference on Neural Information Processing Systems*.
- Tasse, G. N., James, S., & Rosman, B. (2022). Generalisation in lifelong reinforcement learning through logical composition. *International Conference on Learning Representations*.
- Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017). Predictive business process monitoring with lstm neural networks. In E. Dubois & K. Pohl (Eds.), *Advanced information systems engineering* (pp. 477–492). Springer International Publishing.
- Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10, 1633–1685.
- Garces, Kiyoshige**, Xuan, J., & Zuo, H. (2025). Adaptive successor features composition for transfer reinforcement learning. *IEEE Transactions on Artificial Intelligence*, 1–12. <https://doi.org/10.1109/TAI.2025.3610392>
- Tighineanu, P., Skubch, K., Baireuther, P., Reiss, A., Berkenkamp, F., & Vinogradska, J. (2022, March). Transfer learning with gaussian processes for bayesian optimization. In G. Camps-Valls, F. J. R. Ruiz, & I. Valera (Eds.), *Proceedings of the 25th international conference on artificial intelligence and statistics* (pp. 6152–6181, Vol. 151). PMLR.
- Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. *IJCAI-18*, 4950–4957.

- Touati, A., Rapin, J., & Ollivier, Y. (2023). Does zero-shot reinforcement learning exist? *The Eleventh International Conference on Learning Representations*.
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind; a quarterly review of psychology and philosophy*, 59(236), 433–460.
- van der Graaf, J., Lim, L., Fan, Y., Kilgour, J., Moore, J., Gašević, D., Bannert, M., & Molenaar, I. (2022). The dynamics between self-regulated learning and learning outcomes: An exploratory approach and implications. *Metacognition and Learning*, 17(3), 745–771.
- van der Laan, M. J., Polley, E. C., & Hubbard, A. E. (2007). Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6.
- Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proc Int AAAI Conf.*, 30(1).
- Van Niekerk, B., James, S., Earle, A., & Rosman, B. (2019, June). Composing value functions in reinforcement learning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 6401–6409, Vol. 97). PMLR.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–6010.
- Wang, C., Yu, X., Bai, C., Zhang, Q., & Wang, Z. (2024). Ensemble successor representations for task generalization in offline-to-online reinforcement learning. *Sci. China Inf. Sci.*, 67(7), 172203.
- Wang, J., Yan, Y., Hu, Y., Yang, X., & Zhang, L. (2025). A transfer reinforcement learning and digital-twin based task allocation method for human-robot collaboration assembly. *Engineering Applications of Artificial Intelligence*, 144, 110064. <https://doi.org/10.1016/j.engappai.2025.110064>
- Wang, L., Zhao, J., Du, Y., Adelson, E. H., & Tedrake, R. (2024). PoCo: Policy composition from and for heterogeneous robot learning.

- Wang, X., Zhong, H., Zhang, G., Ruan, G., He, Y., & Yu, Z. (2024). Adaptive look-ahead economic dispatch based on deep reinforcement learning. *Applied Energy*, 353, 122121. <https://doi.org/10.1016/j.apenergy.2023.122121>
- Wang, Z., & Pang, Z. (2025). Adaptive transfer reinforcement learning (TRL) for cooling water systems with uniform agent design and multi-agent coordination. *Energy and Buildings*, 345, 116071. <https://doi.org/10.1016/j.enbuild.2025.116071>
- Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. *Proceedings of the 33rd International Conference on International Conference on Machine Learning*.
- Wei, P., Ke, Y., Ong, Y.-S., & Ma, Z. (2023). Adaptive transfer kernel learning for transfer gaussian process regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6), 7142–7156. <https://doi.org/10.1109/TPAMI.2022.3219121>
- Winne, P. H. (2022). Modeling self-regulated learning as learners doing learning science: How trace data and learning analytics help develop skills for self-regulated learning. *Metacognition and Learning*, 17(3), 773–791. <https://doi.org/10.1007/s11409-022-09305-y>
- Winne, P. H., & Hadwin, A. F. (1998). Studying as self-regulated learning. In *Metacognition in educational theory and practice* (pp. 291–318). Routledge.
- Witzany, G. (2018). Memory and Learning as Key Competences of Living Organisms. In F. Baluska, M. Gagliano, & G. Witzany (Eds.), *Memory and Learning in Plants* (pp. 1–16). Springer International Publishing. https://doi.org/10.1007/978-3-319-75596-0_1
- Wong, P. C., Vuong, L. C., & Liu, K. (2017). Personalized learning: From neurogenetics of behaviors to designing optimal language training. *Neuropsychologia*, 98.
- Wulfmeier, M., Byravan, A., Bechtle, S., Hausman, K., & Heess, N. (2023, December). Foundations for Transfer in Reinforcement Learning: A Taxonomy of Knowledge Modalities. <https://doi.org/10.48550/arXiv.2312.01939>
- Xu, C., Li, Q., Zhang, D., Xie, Y., & Li, X. (2020). Deep successor feature learning for text generation. *Neurocomputing*, 396, 495–500.

- Zahavy, T., Barreto, A., Mankowitz, D. J., Hou, S., O'Donoghue, B., Kemaev, I., & Singh, S. (2021). Discovering a set of policies for the worst case reward. *International Conference on Learning Representations*.
- Zahavy, T., O'Donoghue, B., Barreto, A., Mnih, V., Flennerhag, S., & Singh, S. (2022). Discovering diverse nearly optimal policies with successor features. <https://doi.org/10.48550/ARXIV.2106.00669>
- Zeng, Y., Jiang, S., Konstantinou, G., Pou, J., Zou, G., & Zhang, X. (2025). Multi-Objective Controller Design for Grid-Following Converters With Easy Transfer Reinforcement Learning. *IEEE Transactions on Power Electronics*, 40(5), 6566–6577. <https://doi.org/10.1109/TPEL.2025.3525500>
- Zhang, J., Guo, B., Ding, X., Hu, D., Tang, J., Du, K., Tang, C., & Jiang, Y. (2024). An adaptive multi-objective multi-task scheduling method by hierarchical deep reinforcement learning. *Applied Soft Computing*, 154, 111342. <https://doi.org/10.1016/j.asoc.2024.111342>
- Zhang, J., Ren, J., Cui, Y., Fu, D., & Cong, J. (2024). Multi-USV task planning method based on improved deep reinforcement learning. *IEEE Internet of Things Journal*, 11(10), 18549–18567. <https://doi.org/10.1109/JIOT.2024.3363044>
- Zhang, J., Springenberg, J. T., Boedecker, J., & Burgard, W. (2017). Deep reinforcement learning with successor features for navigation across similar environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2371–2378.
- Zhang, J., Chen, Y., Pan, H., Cao, L., & Li, C. (2025). Constrained deep reinforcement transfer learning for short-term forecasting of wind discrepancies at ocean stations. *Neurocomputing*, 624, 129491. <https://doi.org/10.1016/j.neucom.2025.129491>
- Zhang, S., Fernando, H. D., Liu, M., Murugesan, K., Lu, S., Chen, P.-Y., Chen, T., & Wang, M. (2024). SF-DQN: Provable knowledge transfer using successor feature for deep reinforcement learning. *Proceedings of the 41st International Conference on Machine Learning*.

BIBLIOGRAPHY

- Zhu, Z., Lin, K., Jain, A. K., & Zhou, J. (2023a). Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–20.
- Zhu, Z., Lin, K., Jain, A. K., & Zhou, J. (2023b). Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–20. <https://doi.org/10.1109/TPAMI.2023.3292075>
- Zimmerman, B. (1989). Models of self-regulated learning and academic achievement. *Zimmerman and Schunk*.