Composing Policies in Deep Reinforcement Learning

by

Jae Yoon Kim

Supervisor: Prof. Farookh Khadeer Hussain
Co-supervisor: Dr. Christy Jie Liang
Dr. Junyu Xuan

A thesis submitted for the degree of
Doctor of Philosophy in AI

at the
Australian Artificial Intelligence Institute
Faculty of Engineering and Information Technology
University of Technology Sydney

June 2025

Certificate of Original Authorship

I, Jae Yoon Kim, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

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 is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 4 Nov 2024

Abstract

Reinforcement learning (RL) has been developed with an atomic agent on various environment. As RL researchers adopt many agents on much complicated domains such as vision, language and robotics, the cooperation of those agents is required to achieve an optimal performance. In case of lifelong RL, there is a challenge on how to compose multiple agents to get an optimal design for the domain. Like this, the importance of agents' cooperation has been highlighted. This thesis endeavors the cooperation of several agents in hierarchical RL (HRL) and transfer RL (TRL) to find an optimal design.

In HRL, there are two researches for the optimal design of multiple agents. The first one is to research the optimal level synchronization based on flow-based deep generative model in HRL. The research focuses on how to find an off-policy correction which reflects the exact goal for the current lower-level policy without an indirect estimation. It adopts the flow-based deep generative model (FDGM) to support a direct off-policy correction. However, FDGM has its own chronic issue regarding a biased log-density estimate. It considers an inverse model to reflect the lower-level policy with FDGM when the higher-level policy is being trained, considering overcoming the chronic issue of FDGM. The second research is to study the autonomous non-monolithic exploration in options framework, especially HRL. HRL can support autonomous function with a mode switching controller of its own unique structure.

In TRL, this thesis shows the unsupervised pre-training RL with Successor Features (SFs) using non-monolithic exploration scheme. An existing research, APS, has a combined intrinsic reward for pre-training. The combined intrinsic reward causes the performance of fine-tuning to be deteriorated. The research splits the combined intrinsic reward based on non-monolithic exploration scheme. It helps the original intention of SFs by decoupling the dynamics of the environment from the rewards. The second one is to research an offline-to-online RL having non-monolithic exploration scheme to support an online policy without destroying an offline policy. The research can modulate how much an offline policy or an online policy will be utilized during online training. The research focuses on balancing the pro of the offline policy, which is termed exploitation, with that of the online policy, which is referred to as exploration, without modifying the offline policy.

In conclusion, this thesis contributes to the composition of agent design in deep reinforcement learning for optimization. These studies advanced the off-policy correction and anonymous exploration in HRL as well as the performance of pre-training an agent with SFs and offline-to-online RL. These findings have potential to be applied to various real-world applications.

Acknowledgements

I would firstly like to express my earnest thanks to my principal supervisor Prof. Farookh Khadeer Hussain, my co-supervisor Dr. Christy Liang and my co-supervisor Dr. Junyu Xuan. Their comprehensive guidance has covered all aspects of my PhD study, including research topic selection, research methodology experiments, academic writing skills and thesis writing, even sentence structure and formulas. Their step-by-step approach taught me how to do scientific research. Their academic rigor and respectful personalities have benefited my master's research and will be a treasure in my life. Without their excellent guidance and constant encouragement, this research would not have been completed a semester early. Once again, I would like to thank my three supervisors for guiding me against numerous difficulties during the special time of the epidemic.

I am grateful to all members of the Australian Artificial Intelligence Institute (AAII) for their careful participation in my presentation and valuable comments on my research. I am especially grateful to my family for helping and supporting me with my papers and thesis.

Last but not least, I would also like to thank my parents for their generous support of my PhD, both financially and spiritually. I would also like to thank my friends for their support.

Contents

Declaration of Authorship **ii**

Abstract **iii**

Acknowledgements **v**

List of Figures **xi**

List of Tables **xv**

1 Introduction **1**
1.1 Optimizing agent design in HRL 4
1.2 Optimizing agent design in TRL 5
1.3 Thesis Outline 6
1.4 Publications 7

2 Review of Related Work **9**
2.1 Off-policy correction in HRL 9
2.1.1 Hierarchical RL 9
2.1.2 Deep generative models 11
2.2 Non-monolithic exploration in HRL 14
2.2.1 Options framework 14
2.2.2 Exploration 14
2.2.3 Hierarchical RL 15
2.3 Pre-training RL with SFs 16
2.4 Offline-to-online RL with an unmodified offline agent 17
2.4.1 Non-monolithic exploration 17
2.4.2 Pre-training 18
2.4.3 Offline RL 18
2.4.4 Offline-to-online RL 18

3 Hierarchical reinforcement learning with optimal level synchronization based on flow-based deep generative model **21**

3.1 Introduction . . . 21
3.2 Preliminary Knowledge . . . 25
3.2.1 Hierarchical reinforcement learning . . . 25
3.2.2 Off-policy correction . . . 25
3.3 Our model . . . 27
3.3.1 Forward part μ_z . . . 30
3.3.2 Inverse part μ_{lo}^{-1} . . . 31
3.3.2.1 Conditional part $\mu_{z-\text{state}}$ . . . 31
3.3.2.2 FDGM part μ_{rnvp} . . . 32
3.3.3 Off-policy correction, convergence and algorithm . . . 33
3.4 Experiments . . . 35
3.4.1 Comparative analysis with the optimal goal space . . . 39
3.4.2 Comparative analysis with a non-optimal goal space . . . 40
3.4.3 Comparative analysis between our model and LSP . . . 42
3.4.4 Ablation study . . . 43
3.5 Discussion . . . 44
3.5.1 The performance of the higher-level policy based on our model in all tasks . . . 44
3.5.2 The inference from the performance of our model compared with LSP 44
3.5.3 The inverse operation and layer dim. in Algorithm 1 and Algorithm 2 44
3.5.4 The performance difference between the low reward shape and the high reward shape . . . 45
3.5.5 The further research for an environment with a small action space and a big feature space . . . 45
3.6 Appendices . . . 46
3.6.1 The training parameters for our model . . . 46
3.6.2 The key points for experiments . . . 48

4 An Autonomous Non-monolithic Agent with Multi-mode Exploration based on Options Framework 49
4.1 Introduction . . . 49
4.2 Our model . . . 52
4.2.1 The inherent switching mode decision of a policy itself . . . 54
4.2.2 Empowering more entropy choice for exploration . . . 54
4.2.3 Guided exploration . . . 55
4.2.4 Evaluation for robustness . . . 56
4.3 Experiments . . . 58
4.3.1 Comparison with the reference paper and pure off-policy . . . 60
4.3.1.1 Ant Push . . . 60
4.3.1.2 Ant Fall . . . 61
4.3.2 Ablation study . . . 61
4.3.2.1 Without the reward modification . . . 62
4.3.2.2 Without the loss modification . . . 62

4.3.2.3 Without both the reward modification and the loss modification . . . 62
4.4 Discussion . . . 62
4.4.1 The effect of on-policy for exploration . . . 62
4.4.2 The effect of reward modification . . . 62
4.4.3 The effect of loss modification . . . 63

5 Decoupling Exploration and Exploitation for Unsupervised Pre-training with Successor Features 65
5.1 Introduction . . . 65
5.2 PRELIMINARY KNOWLEDGE . . . 69
5.2.1 Successor features . . . 69
5.2.2 Non-monolithic exploration . . . 69
5.3 Our methodology . . . 70
5.3.1 The model with a non-monolithic exploration . . . 72
5.3.2 The optimized training method of separate agents . . . 72
5.3.3 The greater flexibility and generalization of discriminator . . . 73
5.4 Experiments . . . 75
5.4.1 Using the intrinsic reward of APS and DIAYN . . . 76
5.4.2 The performance of all NMPSs against that of APS . . . 78
5.4.3 The analysis of four factors of NMPS . . . 80
5.5 Discussion . . . 80
5.5.1 Decoupling a monolithic exploration agent with SFs . . . 80
5.5.2 The factors to maximize the performance of NMPS . . . 80
5.5.3 The vulnerable point of competence-based approach . . . 81

6 A Non-Monolithic Policy Approach of Offline-to-Online Reinforcement Learning 83
6.1 Introduction . . . 83
6.2 Our methodology . . . 85
6.2.1 Offline-to-online RL with a mode-switching controller . . . 85
6.2.2 The ability to leverage flexibility and generalization on a downstream task . . . 88
6.3 Experiments . . . 89
6.3.1 The comparison between our model and PEX . . . 91
6.3.1.1 Normalized return . . . 94
6.3.1.2 Execution count . . . 94
6.3.2 The comparison between our model and others . . . 95
6.4 Discussion . . . 96
6.4.1 Will π^{on} be also suboptimal and even unsafe? . . . 96
6.4.2 The effect of non-monolithic exploration methodology for offline-to-online RL . . . 96
6.4.3 The efficient usage of offline policy during online fine-tuning . . . 97
6.4.4 How can the method be translated to a wider real-world problems? . . . 97
6.4.5 The considerations of further research . . . 97

7 Conclusion **99**

Bibliography **103**

List of Figures

3.1 An example of several tasks in the Ant environment used in this research. The target location (yellow arrow) is the reward point for the approach of the agent, ant (green arrow). A trained policy requires the combination of various actions of the agent and its interplay with an object in the environment (pink blocks) in each task: the series of several directional movements (third from right), clearing away the obstacle (second from right) and making a bridge with an object (right). A higher-level policy guides its lower-level policy to be rewarded with an abstract action (orange arrow) which includes the desired positions and orientation of the ant and its limbs. 24

3.2 The model architecture of HIRO (left), LSP (center) and Ma et al. (right). 26

3.3 The architecture of our suggested model using FDGM. (A) A low-level policy of HIRO is used only for the forward part to Environment. (B) The observation embedding of LSP, which is an encoder, is replaced by a low-level policy of HIRO as an independent policy. (C) The RealNVP of LSP, which is a neual network and belongs to a policy, is used as an independent policy. (D) The combination of a forward part and an inverse part is adopted from the model architecture of [1]. 27

3.4 Communication graph representing the dependence of our model. 33

3.5 For the task of the higher-level policy, the average rewards of our model (green) are compared with that of HIRO (salmon) with the same NN size on condition of the optimal goal space. 36

3.6 The average rewards of our model (green, blue) for the task of the higher-level policy are compared with that of HIRO (salmon, violet) on condition of a non-optimal goal space. The numbers of (our model or HIRO)_8_(15 or 8) are the goal dimension of higher-level policy and that of lower-level policy respectively. 37

3.7 For the task of the higher-level policy, the average rewards of our model (green, blue, orange, salmon, violet and navy for Ant Push Single and green, blue, orange, cyan and red for Ant Fall Single) are compared with that of LSP (pink for Ant Push Single and violet for Ant Fall Single). They are based on the change of higher-level goal dimension of a) our model_8_15, (b) our model_8_8 (c) our model_8_15 (d) our model_8_8 used in the corresponding task of Fig. 3.6. 38

3.8 Average result of our method (green) compared with that of the variant model (salmon) in Ant Fall Multi. 43

3.9 The example of how to create a non-optimal goal space of a higher-level policy and lower-level policy of our model from the default optimal goal space tuples of a higher-level policy and lower-level policy of HIRO. Each level's default tuple in (a) is represented with color, which is blue for higher-level policy or red for lower-level policy, and the number of tuple element's position. Only the tuple's elements of destination (higher-level's goal) are replaced by the corresponding tuple's elements of source (lower-level's goal). In the case of (Our model, HIRO, or LSP)_8_8 for the lower-level's goal, the last 7 tuple elements of the lower-level goal are removed. (a) The default tuple, which is optimal tuple, of a higher-level policy and lower-level policy of HIRO. Appendix 3.6.1 shows the definition of each level's tupe. (b) Our model_8_15 (c) Our model_8_8 (d) Our model_3_8. The goal space of the higher-level policy of 'Our model_2_8' and 'Our model_3_8' is constrained to be the same as the default goal space of Ant Push Single and Ant Fall Single of HIRO, respectively. (d) Our model_6_15 (e) Our model_4_8. 46

4.1 An example of noise-based monolithic exploration (left) and non-monolithic exploration (right). The final action, which is a scalar in this example for the well understanding explanation, denotes the action of an agent represented with a solid circle at each step. The solid line denotes the exploitation, an original action of a behaviour policy. The solid circle in the noise-based monolithic exploration is a final action which combines the original action of a behaviour policy and a sampled bounded noise at each step. However, the solid circle in the non-monolithic exploration is defined according to the mode of each step, i.e. exploitation which is an an original action of a behaviour policy or exploration which is a random noise or a policy. 51

4.2 The architecture of our suggested model (right) compared with that of the reference paper using a homeostasis [2] (left). 52

4.3 The count of exploration modes and exploitation and the reward and success rate of higher level policy for our model, Ref:Uniform random, Ref:PPO and HIRO in Ant Push. 56

4.4 The count of exploration modes and exploitation and the reward and success rate of higher level policy for our model, Ref:Uniform random, Ref:PPO and HIRO in Ant Fall. 57

4.5 Three types of ablation study against our normal model in Ant Push. 59

5.1 An example of noise-based monolithic exploration (left) and non-monolithic exploration (right). In the noise-based monolithic exploration, noise and the agent's action play roles in exploration and exploitation, respectively. The final action, 'Action', taken by the agent in the environment is the result of adding the action with a noise. In the non-monolithic exploration, the exploitation agent and the exploration agent work for their own purposes with the help of a mode-switching controller. The mode-switching controller considers the state of one or both agents. 66

5.2 The architecture of our proposed pre-training model, NMPS, (right) compared with Active Pre-training with Successor Features (left). 69

5.3 The comparison result of fine-tuning of NMPS (the best one), APS, DIAYN and SMM on Walker (a), Jaco Arm (b) and Quadruped (c) by using the intrinsic reward of APS or DIAYN for Explor of NMPS. 78
5.4 The comparison result of fine-tuning of all $NMPS_X$s of NMPS and APS on Walker (a) and Jaco Arm (b, smoothed line) and most NMPS variants and APS on Quadruped (c) by using the intrinsic reward of APS or DIAYN for Explor of NMPS. 78

6.1 Illustration of offline-to-online RL training schemes employed in our model with an unmodified offline policy. 86
6.2 Normalized Return Curves of different methods, which are our model, PEX, Offline and Buffer, on benchmark tasks from D4RL. IQL is used for all methods as the backbone. 92
6.3 Execution count of our model and PEX on benchmark tasks from D4RL. The execution counts of offline policy and online policy of PEX or our model are referred to as $PEX\_Offline$ and PEX_Online or $OurModel\_Offline$ and $OurModel\_Online$, respectively. 93

List of Tables

3.1 Key Notations. 28

4.1 Key Notations. 53

5.1 Key Notations. 68

5.2 Comparing all variants of NMPS based on 4 factors that are 'Reward', '$D_R^{\#}$', 'The train of feature or skill ('T. f or s' marked in the table)' and 'Action' on the model notation of $NMPS\_\{X\ or\ D\}\_\{sep,\ exploit\ or\ explor\}\_\{e,\ x\ or\ *\}$. 70

6.1 The comparison between our model and PEX. 90

Chapter 1

Introduction

In recent years, reinforcement learning (RL) has made significant advancements across various domains such as vision, language, and robotics. Particularly, practical tasks in open environments have become more complex, requiring RL agents to adapt to changing conditions. The collaboration of several agents, such as in hierarchical RL (HRL), has shown improved performance on these realistic tasks compared to an atomic RL. Additionally, transfer RL has emerged as a crucial area of research for addressing complex tasks with limited resources.

There are several unresolved issues in the cooperation, exploration, and transfer learning of several agents, aimed at finding optimal relationships among them. This thesis aims to investigate the design of cooperation and exploration at a hierarchical level, as well as the pre-training and offline-to-online schemes, to derive an abstract optimal policy for the interaction of multiple agents.

This thesis investigates two key areas—HRL and transfer RL—concerning the cooperation of multiple agents: (1) optimal level synchronization and non-monolithic exploration in HRL, and (2) pre-training RL and offline-to-online RL using the non-monolithic exploration scheme for transfer RL. In one of my research projects, which focuses on the optimal synchronization between levels of hierarchical RL (HRL), the non-monolithic exploration scheme is not utilized. However, other research works consider it a primary mechanism.

Meanwhile, the research in these two areas demonstrates that the capabilities of multiple agents compared to an atomic agent can be significantly enhanced.

In our research on HRL, we explored the use of off-policy correction as employed in HIRO [3]. We identified an issue where the algorithm induced a corrected goal that did not accurately reflect the correction between high-level and low-level hierarchical structures. To address this, we investigated how to precisely reflect the value of a lower-level policy to a higher-level policy as a goal. We discovered that using a flow-based deep generative model (FDGM) was suitable for this purpose. We also looked into a chronic issue with FDGM and developed an alternative model to overcome this critical problem.

Furthermore, we explored how to leverage autonomous non-monolithic exploration within an option framework. A significant challenge in this context is developing an autonomous mode-switching controller for non-monolithic exploration in a hierarchical agent. To tackle this challenge, we assigned roles to the hierarchical levels, incorporating a mode-switching controller and non-monolithic exploration. Additionally, we focused on maximizing the advantages of a hierarchical agent to establish an autonomous non-monolithic exploration agent. We formulated the model with a distinct purpose for each hierarchical level, aiming to achieve natural non-monolithic exploration utilizing hierarchical RL.

In my investigation of transfer RL, we focused on the intrinsic reward of Active Pre-training with Successor features(APS) [4], which utilizes a combined intrinsic reward for pre-training. However, this intrinsic reward can deteriorate the performance of APS by hindering the original intention of its dynamic model. To address this, we integrated a non-monolithic exploration scheme into the APS model, which splits the combined intrinsic reward into the non-monolithic exploration scheme.

We also delved into a critical issue with PEX [5], which leverages split agents for offline-to-online RL. Policy EXpansion(PEX) includes a mode-switching controller that selects an active agent at each training step during online training. However, we identified a critical problem with the mode-switching controller, which impacts the performance of online training. To mitigate this issue, we propose replacing the mode-switching controller with the one used in the non-monolithic exploration scheme, which offers a different control viewpoint.

The contributions of the thesis are represented as follows.

- We explore leveraging a hierarchical RL model to enhance off-policy correction. We propose a novel HRL model architecture that defines an inverse model of the model-free HRL. This architecture takes advantage of the inverse operation of an FDGM to improve off-policy correction.
- We explore the utilization of an autonomous non-monolithic exploration scheme within an option framework. We propose an option framework model that supports autonomous non-monolithic multi-mode exploration.
- We explore the utilization of a non-monolithic exploration scheme for a monolithic agent using successor features during unsupervised pre-training. We propose a decoupled exploration methodology based on the non-monolithic exploration scheme to enhance the performance of a monolithic agent using successor features during unsupervised pre-training.
- We explore the utilization of a non-monolithic exploration scheme for offline-to-online RL. We propose the development of a model for offline-to-online RL that leverages a non-monolithic exploration methodology.

Multiple agents with an optimized composition can outperform an atomic agent when efficiently coordinated on practical tasks. This thesis demonstrates how to optimize their cooperation to achieve optimal performance across various domains. In HRL, cooperation among multiple agents is explored using their own beneficial schemes. In Transfer RL, multiple agents function as an abstract combined agent by splitting the monolithic exploration of an agent into exploitation and exploration. This research showcases the superior composition of multiple agents compared to an atomic agent, as accepted in current RL design. The following shows the simplified concept and achievement for understanding the devised algorithm in HRL and transfer RL.

1.1 Optimizing agent design in HRL

In Chapter 3, the high-dimensional state and action space, or sparse reward tasks in a Reinforcement Learning (RL) environment, often require a superior controller such as Hierarchical Reinforcement Learning (HRL) over atomic RL.

HRL can handle the complexity of commands to achieve task objectives through its hierarchical structure. One of the challenges in HRL is efficiently training each level policy with optimal data collection from its experience. Off-policy correction is a critical technique for facilitating sample-efficient off-policy training in HRL, as it addresses the nonstationary issue of higher-level policy training.

However, existing methods use indirect probabilistic approaches, which may not fully reflect the current knowledge of the lower-level policy, thereby limiting the effectiveness of training the higher-level policy. In this thesis, we propose a novel HRL model that supports direct off-policy correction based on a Flow-based Deep Generative Model (FDGM).

This leverages the inverse operation of FDGM to achieve goals corresponding to the current knowledge of the lower-level policy. Additionally, our model addresses the disadvantages of FDGM to effectively use it in HRL. Our model demonstrates higher performance than an existing model through comparative experiment results on benchmark environments. The code for the comparison is readily available.[0]

In Chapter 4, most exploration research in reinforcement learning (RL) has concentrated on *the way of exploration*, specifically, *how to explore*. However, another important aspect, *when to explore*, has not received as much attention in RL exploration research.

The issue of *when* in monolithic exploration in typical RL exploration behavior often ties an exploratory action to an exploitative action of an agent. Recently, non-monolithic exploration research has emerged to study the mode-switching exploration behavior observed in humans and animals.

The primary goal of our research is to enable an agent to autonomously decide when to explore or exploit. We present initial research on autonomous multi-mode exploration of

[0] https://github.com/jangikim2/Hierarchical_Reinforcement_Learning

non-monolithic behavior within an option framework. Through comparative experimental results, we demonstrate the superior performance of our method compared to existing non-monolithic exploration methods.

1.2 Optimizing agent design in TRL

In Chapter 5, unsupervised pre-training has been on the lookout for the virtue of a value function representation referred to as successor features (SFs), which decouples the dynamics of the environment from the rewards. It has a significant impact on the process of task-specific fine-tuning due to the decomposition.

However, existing approaches struggle with local optima due to the unified intrinsic reward of exploration and exploitation without considering the linear regression problem and the discriminator supporting a small skill sapce. We propose a novel unsupervised pre-training model with SFs based on a non-monolithic exploration methodology.

Our approach pursues the decomposition of exploitation and exploration of an agent built on SFs, which requires separate agents for the respective purpose. The idea will leverage not only the inherent characteristics of SFs such as a quick adaptation to new tasks but also the exploratory and task-agnostic capabilities.

Our suggested model is termed Non-Monolithic unsupervised Pre-training with Successor features (NMPS), which improves the performance of the original monolithic exploration method of pre-training with SFs. NMPS outperforms Active Pre-training with Successor Features (APS) in a comparative experiment.

In Chapter 6, Offline-to-online reinforcement learning (RL) leverages both pre-trained offline policies and online policies trained for downstream tasks, aiming to improve data efficiency and accelerate performance enhancement.

An existing approach, Policy Expansion (PEX), utilizes a policy set composed of both policies without modifying the offline policy for exploration and learning. However, this approach fails to ensure sufficient learning of the online policy due to an excessive focus on exploration with both policies.

Since the pre-trained offline policy can assist the online policy in exploiting a downstream task based on its prior experience, it should be executed effectively and tailored to the specific requirements of the downstream task.

In contrast, the online policy, with its immature behavioral strategy, has the potential for exploration during the training phase. Therefore, our research focuses on harmonizing the advantages of the offline policy, termed exploitation, with those of the online policy, referred to as exploration, without modifying the offline policy.

In this study, we propose an innovative offline-to-online RL method that employs a non-monolithic exploration approach. Our methodology demonstrates superior performance compared to PEX. The code for this comparison is readily available. [0]

1.3 Thesis Outline

This thesis is organized as follows:

In Chapter 2, the related work regarding relevant researches is reviewed to understand the history and category of related researches and discuss the pros and cons of them.

Chapters 3 to 4 represent the research in HRL. Chapters 3 focuses on a novel HRL model supporting direct off-policy correction based on FDGM in order to overcome the shortcoming of existing research, HIRO. Chapter 4 exhibits a noble HRL model using a non-monolithic exploration scheme. I maximize the advantage of HRL structure in order to draw optimal cooperation of multiple agents against an atomic agent in the above two researches.

In Chapters 5 and 6, I research the optimal multiple agents' structure in transfer RL, pre-training RL and offline-to-online RL. Chapter 5 shows the model of a decoupled exploration methodology based on non-monolithic exploration methodology to improve the performance of a monolithic agent using successor features during unsupervised pre-training. Chapter 6 represents the model development of offline-to-online RL taking advantage of a non-monolithic exploration methodology.

[0] https://github.com/jangikim2/Offline-to-online-RL-with-non-monolithic-exploration-methodology/tree/main

Finally, Chapter 7 suggests a summary of the thesis and discusses a prospective further research.

1.4 Publications

I published and submitted my papers to an international journal and conference during my PhD research period as follows:

- **Jae Yoon Kim**, Junyu Xuan, Christy Liang, Farookh Hussain, *An Autonomous Non-monolithic Agent with Multi-mode Exploration based on Options Framework*, 2023 International Joint Conference on Neural Networks (IJCNN), Gold Coast, Australia, June 18-23, 2023. (ERA rank: B)
- **Jae Yoon Kim**, Junyu Xuan, Christy Liang, Farookh Hussain, *Decoupling Exploration and Exploitation for Unsupervised Pre-training with Successor Features*, 2024 The IEEE World Congress on Computational Intelligence (IEEE WCCI 2024), Yokohama, Japan, June 30- July 5, 2024. (ERA rank: B)
- **Jae Yoon Kim**, Junyu Xuan, Christy Liang, Farookh Hussain, *A Non-Monolithic Policy Approach of Offline-to-Online Reinforcement Learning*, 2024 International Conference on Neural Information Processing (ICONIP 2024), Auckland, New Zealand, Dec 2-6, 2024. (ERA rank: B)
- **Jae Yoon Kim**, Junyu Xuan, Christy Liang, Farookh Hussain, *Hierarchical reinforcement learning with optimal level synchronization based on flow-based deep generative model*, Neurocomputing Journal. (ERA rank: B) (Under review)

Chapter 2

Review of Related Work

2.1 Off-policy correction in HRL

2.1.1 Hierarchical RL

A large body of research has focused on developing an optimal method to connect adjacent level policies in the HRL framework from an action point of view. In option-critic architecture, the policy over options chooses an option to follow its intra policy which continues to work until the condition of option termination is reached [6]. The strategic attentive writer with the structure of HRL has an advantage over the sequential decision making domain due to the macro-action which is partially organized depending on environment information [7]. Feudal reinforcement learning (FeUdal) networks for HRL acknowledges semantically meaningful sub-goals in an environment followed by different policies of the manager [8]. [9] suggests an efficient and general method to discover a sub-goal with two unsupervised learning methods, anomaly outlier detection and K-means clustering. The hierarchical Actor-Critic has DDPG with an actor-critic network and Hindsight Experience Replay (HER) [10] on every policy [11]. [12] introduces a nested and goal-conditioned HRL framework which can divide a task into several sub tasks with two classes of hindsight transition. [13] develops a method to utilize representation learning through sub-optimality which is defined as the difference between the value function of an optimal HRL which does not utilize the representation learning and the value function of an HRL which utilizes the

representation learning. However, the above mentioned researches do not consider the optimal training for the level synchronization between adjacent level policies.

The main focus of HIRO is to find an effective training method for each HRL policy regarding the dynamics of training between adjacent level policies to synchronize each other. In order to find an optimal goal $g_t \in R^{d_s}$ for an off-policy correction in HRL, HIRO finds 10 candidates which are eight candidate goals sampled randomly from a Gaussian centred at the distance of state $s_{t+c} - s_t$ between the horizon of the higher-level policy 'c', an original goal g_t and a goal equivalent to the distance of state $s_{t+c} - s_t$. Ten candidates are evaluated in an induced log probability of a lower-level policy,

$$\begin{aligned} &\log \mu_{lo}(a_{t:t+c-1}|s_{t:t+c-1}, \tilde{g}_{t:t+c-1}) \\ &\propto -\frac{1}{2}\sum_{i=t}^{t+c-1} \|a_i - \mu_{lo}(s_i, \tilde{g}_i)\|_2^2 + \text{const} \end{aligned} \tag{2.1}$$

where μ_{lo} is the lower-level policy, $x_{t:t+c-1}$ is the collected sequence $x_t, ..., x_{t:t+c-1}$ by higher-level policy, a_t is the action of μ_{lo} and $\tilde{g}_t$ is the relabelled g_t.

After finding a best candidate, in the end, the best goal is used for training its higher-level policy. The research considers three other methods detailed in its Appendix. Yet, it still does not propose a proper method for off-policy correction of the model-free HRL since it relies on an indirect estimation. Our research extends the work begun in HIRO on the off-policy correction which is the most interesting part of our research.

LSP suggests that each policy in HRL can be trained based on a bottom-up layerwise way through a latent variable which is incorporated in the maximum entropy objective. Furthermore, for higher layers to retain full expressivity, where latent variable and action must be invertible to each other, the research takes advantage of a FDGM. In Fig. 3.2, LSP illustrates how a latent goal from a higher-level policy and an observation from the environment are provided to its lower-level policy. Two invertible coupling layers (orange), which represent FDGM in LSP, receive the latent goal from the higher-level and condition on an observation embedding. The observation embedding, implemented with two fully connected layers (green), is responsible for changing the dimension of the observation. However, this is not appropriate for the off-policy correction of HRL since it cannot support

a flexible goal space due to the intrinsic characteristic of FDGM, which is the issue of invertible bijective transformation function of FDGM.

For our purposes, our research embraces the use of a FDGM in HRL similar to LSP though.

2.1.2 Deep generative models

We have investigated the following deep generative models for an inverse model in HRL. In the end, we have decided that using a FDGM in our model ensures our model to be a valid inverse model in HRL. **GAN and VAE** [14] proposes an innovative deep generative model, Generative Adversarial Nets (GAN), which has two models, a discriminator and a generator. A Variational AutoEncoder (VAE) [15] has a probabilistic encoder which generates a latent variable for a decoder to overcome the shortcomings of the vanilla Autoencoder [16]. Lots of variants of these two generative models have been developed because of their outstanding unsupervised learning framework [17–23]. But GAN has the biggest defect leading to a training divergence because it cannot find a Nash equilibrium easily [24]. Although VAE supports precise control over the latent variable, it also has a blurry output due to a drawback of imperfect reconstruction which results in difficulty training the decoder. Because of the innate disadvantages of GAN and VAE, they are avoided in our research.

Normalizing Flow A simple distribution can be transformed into a complex distribution by repeatedly using an invertible mapping function. The change of the variable theorem makes the transformation from a variable to a new one possible and leads to the final distribution of the target variable as follows. Suppose a probability density function $z \sim p(z)$ for a random variable z. If an invertible bijective transformation function f exists between a new variable x and z, $x = f(z)$ and $z = f^{-1}(x)$. Again, if the change of the variable theorem is applied to $\mathbf{z}$ and a final target variable $\mathbf{x}$ in the multivariate version through the operation of successive distribution $p(\mathbf{z})$ and successive transformation function $f(\mathbf{z})$,

$$\mathbf{z}_{i-1} \sim \ p_{i-1}(\mathbf{z}_{i-1}),\ \mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i). \tag{2.2}$$

Then

$$\begin{aligned} p_i(\mathbf{z}_i) =& p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| det \frac{df_i^{-1}}{d\mathbf{z}_i} \right| \\ =& p_{i-1}(\mathbf{z}_{i-1}) \left| det \frac{df_i}{d\mathbf{z}_{i-1}} \right|^{-1}, \text{thus} \end{aligned} \tag{2.3}$$

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \tag{2.4}$$

where $det \frac{\partial f_i}{\partial \mathbf{z}_i}$ is the Jacobian determinant of the function f.

Finally, the chain of K transformations of probability density function f_i, which is easily inverted and whose Jacobian determinant can be easily computed, from the initial distribution $\mathbf{z}_0$ yields the final target variable $\mathbf{x}$,

$$\begin{aligned} \mathbf{x} = \mathbf{z}_\mathrm{K} &= f_\mathrm{K} \circ f_{\mathrm{K}\text{-}1} \circ \cdots \circ f_1(\mathbf{z}_0) \\ \log p(\mathbf{x}) = \log \pi_\mathrm{K}(\mathbf{z}_\mathrm{K}) &= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{\mathrm{K}} \log \left| det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \end{aligned} \tag{2.5}$$

The flow results from the path travelled by the random variables $\mathbf{z}_i = f_i(\mathbf{z}_{i-1})$. A normalizing flow is called the full chain configured by the successive distribution π_i.

In our research, we focus on the advantages of a normalizing flow, which are model flexibility and generation speed, even though it also has drawbacks. A Real-valued Non-Volume Preserving algorithm (RealNVP) makes use of a normalizing flow which is implemented with an invertible bijective transformation function. Each bijection called an affine coupling layer, which is $f : \mathbf{x} \mapsto \mathbf{y}$, decomposes an input dimension into two sections. The intrinsic transformation property using the affine coupling layer causes the input dimension to be unchanged with the alternate modification of the two split input sections in each coupling layer. Based on this property, the inverse operation is attained without difficulty. In addition, the inverse operation easily computes its Jacobian determinant since its Jacobian is a lower triangular matrix. RealNVP uses a multi-scale architecture as well as a batch normalization for better performance. To support a local correlation structure of an image, there are two masked convolution methods: the spatial checkerboard pattern mask and channel-wise mask [25]. Non-linear Independent Components Estimation (NICE) which is

a previous model of RealNVP uses an additive coupling layer which does not use the scale term of an affine coupling layer [26]. Generative flow with 1×1 convolutions (Glow) is a method to simplify the architecture regarding a reverse operation of channel ordering of NICE and RealNVP [27].

Several studies have tried to overcome the chronic drawback of a normalizing flow, biased log-density estimation [28, 29]. [1] (see Fig. 3.2) utilizes the intrinsic characteristic of FDGM with an inductive bias based on a model architecture. Hence, the research takes advantage of a biased log-density estimation of FDGM itself. The research suggests a model architecture using a VAE which extracts a global representation of an image and a FDGM, which depends on a local representation, with a conditional input of the global representation of the image. Finally, an unbiased log-density estimation of an image can be expected from the FDGM using the output of the VAE.

The compression encoder $q_\phi(z|x)$ in the VAE framework compresses the image x with a high dimension to the global latent representation z with a low dimension as follows:

$$q_\phi(z|x) = N(z; \mu(x), \sigma^2(x)) \tag{2.6}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are neural networks for a learned mean and variance and a variational posterior distribution $q_\phi(z|x)$ as a diagonal Gaussian models a latent variable Z. Then to address the biased log-density estimation of FDGM regarding the reconstruction of image x, $p_\theta(x|z)$ is considered by using z. Hence, an invertible flow-based decoder $f_\theta(x; z)$ having the main input of image x with a conditional input of global latent representation z is used for finding a local representation v as follows:

$$v = f_\theta(x; z) \tag{2.7}$$

where v has a standard normal distribution $v \sim N(0, I)$. Finally, the image x is reconstructed by using the inverse function of flow-based decoder as follows:

$$x = f_\theta^{-1}(v; z). \tag{2.8}$$

We adopt this architecture since it can resolve the flexibility of the goal dimension between policies and the biased log-density estimation for a FDGM of our model.

Auto-regressive flow An auto-regressive flow provides tractable likelihoods due to the observable probability of a full sequence given by the product of conditional past probabilities so that it has a better density estimation than non-autoregressive flow-based models [30–34]. Yet, the restoration speed of the original data which results from its operation countervails its merit.

2.2 Non-monolithic exploration in HRL

2.2.1 Options framework

The set of option, which is a generalized concept of action, over an MDP is comprised of a semi-Markov decision process (SMDP). Semi-MDPs are defined to deal with the different levels of an abstract action based on the variable period. HRL is a representative generalization of reinforcement learning where the environment is modelled as a semi-MDP [35].

Each action in non-monolithic exploration mode which adopts a multi-mode exploration has different effect during the different period. Thus the sequence of action is defined by taking advantage of an option framework for a multi-mode exploration.

2.2.2 Exploration

The various events for triggers have been considered whether they are acquired from uncertainty or not [36], [37], [38].

The experiment of [39] shows the efficacy of robot behaviour learning from self-exploration and a socially guided exploration supported by a human partner. [40] claims about the Bayesian framework which supports changing dynamics online and prevents conservativeness by using a variance bonus uncovering the level of transition of adversity. [41] claims Tactical Optimistic and Pessimistic (TOP) estimation for the value estimation strategy of optimistic and pessimistic online by using a quantile approximation [42]. Hence, the belief

distribution is constructed by following quantile estimation:

$$q_{\tilde{Z}^{\pi}(s,a)} = q_{\bar{Z}(s,a)} + \beta q_{\sigma(s,a)} \tag{2.9}$$

where $q_{\bar{Z}(s,a)}$ and $q_{\sigma(s,a)}$ are the mean and standard deviation of quantile estimation respectively and Z(s,a) is a return random variable. The belief distribution is optimistic and pessimistic when $\beta \geq 0$ and $\beta < 0$ respectively.

[2] claims the importance of a non-monolithic exploration against a monolithic exploration. Its representative non-monolithic exploration method utilizes 'homeostasis' based on the difference of value function between k steps which is referred to as the following 'value promise discrepancy',

$$D_{promise}(t-k,t) := \left|V(s_t - k) - \sum_{i=0}^{k-1} \gamma^i R_{t-i} - \gamma^k V(s_t)\right| \tag{2.10}$$

where $V(s)$ is the agent's value estimate at state s, γ is a discount factor and R is the reward.

The above-mentioned researches regarding the guidance-exploration framework, robust-MDP research and the adaptive optimism inspired our research on the basis of [2].

2.2.3 Hierarchical RL

The Semi-Markov Decision Process(SMDP) takes advantage of options defining a domain knowledge and can reuse the solutions to sub-goals [35]. [43] claims that an Adaptive Skills, Adaptive Partitions framework supports learning near-optimal skills which are composed automatically and concurrently with skill partitions, in which an initial misspecified model is corrected. [44] proposes an algorithm to solve tasks with sparse reward in which the research suggests an algorithm to accelerate exploration with the construction of options minimizing the cover time. [45] also deals with a sparse reward environment. Thus, it formalizes the concept of fast and slow curiosity for the purpose of stimulating a long-time horizon exploration. The option-critic architecture has the intra policy, which follows the option chosen by the policy over options until the end of the condition of option

termination [46]. [47] claims that each policy in HRL, which utilizes a flow-based deep generative model for retaining a full expressivity, is trained through a latent variable with a bottom-up layer-wise method. HIRO claims the method to synchronize adjacent levels of hierarchical reinforcement learning to efficiently train the higher level policy.

Our model makes use of HIRO for our exploration research because it is a traditional goal-conditioned HRL.

2.3 Pre-training RL with SFs

The research in [48] makes use of SFs to transfer previous knowledge to similar environments in simulated and real experiments. As the importance of SFs is magnified, [49] researches visual semantic planning to predict a sequence of actions for a desired goal state by using a deep successor representation. [50] shows that although SFs in transfer learning promote learning speed, there is a restriction in transferring knowledge between tasks when an SF representation relies on the task's optimal policy. UVFAs, which is $V(s, g, \theta)$ using parameters θ, claims the importance of value function approximation of adjacent states as well as adjacent goals [51]. USFA [52] combines UVFAs, SFs and GPI to support multi-task deep reinforcement learning.

In order to overcome 'linear regression problem', VISR takes advantage of SFs and the behavioral mutual information (BMI) [53, 54], which constructs a reward function as [55] to support fast task inference as follows:

$$\log q(w|s) = \phi(s)^T \mathbf{w} \tag{2.11}$$

where q is a BMI discriminator.

Since APS makes use of the strong points of both VISR and APT, the intractable conditional entropy $-H(s|z)$ can be lower bounded. For the variational lower bound of APS, the

intrinsic reward for the pre-training is the sum of $\log q(w|s)$ of VISR and $r_t^{APT}(s)$ of APT.

$$\begin{aligned} r_{APS}(s,a,s') &= r_{APS}^{exploitation}(s,a,s') + r_{APS}^{exploration}(s,a,s') \\ &= \phi(s)^T w + \log\left(1 + \frac{1}{k}\sum_{h^{(j)} \in N_k(h)} \left\|h - h^{(j)}\right\|_{n_h}^{n_h}\right) \\ &= \log q(w|s) + r_t^{APT}(s) \end{aligned} \tag{2.12}$$

where $h = \phi(s')$ and $N_k(\cdot)$ denotes the k nearest neighbors.

2.4 Offline-to-online RL with an unmodified offline agent

2.4.1 Non-monolithic exploration

A noise-based monolithic exploration RL agent capitalizes on a final action by adding its policy's action to random noise for exploration. In contrast, the non-monolithic exploration methodology comprises discrete agents—one for exploitation and another for exploration—managed by a mode-switching controller that selects the appropriate agent at the right time. Fig. 1, shown in 'supplementary document.pdf' on the GitHub mentioned in the Abstract, illustrates the diagrams for both the noise-based monolithic exploration RL agent and the non-monolithic exploration RL agent.

The significance of 'when' to explore, a primary consideration in the non-monolithic exploration methodology, has been emphasized in [56–58], contrasting with the traditional focus on 'how' to explore, typically associated with monolithic exploration approaches. In [58], the non-monolithic exploration methodology incorporates 'Homeostasis' [59] as a mode-switching controller, abbreviated as 'Homeo' (see 'A.5 Homeostasis' in 'supplementary document.pdf'), to select an active agent between exploitation and exploration. 'Homeo' leverages the 'value promise discrepancy', $D_{promise}(t-k,t)$, which represents the difference in the value function over k steps:

$$D_{promise}(t-k,t) := \left|V(s_{t-k}) - \sum_{i=0}^{k-1} \gamma^i R_{t-i} - \gamma^k V(s_t)\right| \tag{2.13}$$

where $V(s)$ is an agent's value estimate at state s, R is a reward and γ is a discount factor.

2.4.2 Pre-training

Pre-training is designed to enhance sample efficiency and transfer prior knowledge to a downstream task [60]. The main types include online or offline pre-training, supervised or unsupervised pre-training [61], and representation [62] or policy pre-training [63].

Our research focuses on offline pre-training in RL, which involves utilizing an off-policy approach, as seen in offline RL methods such as CQL [64], IQL [65], AWAC [66], and COMBO [67].

2.4.3 Offline RL

Online RL presents significant challenges due to the high costs and risks associated with interactions in real environments. Consequently, the RL research community has shown interest in Offline RL, which leverages existing pre-used datasets. However, a persistent issue is the vulnerability to distributional shifts between the offline dataset and the distribution of the downstream task. Offline RL has been explored in various areas, including policy constraints, importance sampling, regularization, uncertainty estimation, model-based methods, one-step methods, imitation learning, and trajectory optimization [68, 69].

In the evaluation and improvement of an offline policy, the policy loss term typically includes loss terms related to policy constraints [70, 71], uncertainty [72, 73], and regularization [64, 74].

2.4.4 Offline-to-online RL

Prior work in offline-to-online RL has focused on consolidating an acquired online policy based on an offline policy [75–77].

PEX was designed to deviate from the traditional unified framework, thereby avoiding compromises to a pre-trained offline policy. In PEX, π_β is kept frozen, while Π consists of both π_β and a learnable policy π_θ.

$$\Pi = [\pi_\beta, \pi_\theta] \tag{2.14}$$

A proposal actions set, $\mathbb{A} = \{a_i \sim \pi_i(s) | \pi_i \in \Pi\}$, is formed from Π at the current state s. One of them will be selected from this set using a categorical distribution, $P_{\mathbf{w}}$, which is associated with a value function $\mathbf{Q}_\phi = \left[Q_\phi(s, a_i) | a_i \in \mathbb{A}\right] \in \mathbb{R}^K$ where K is cardinality of Π (here $K = 2$).

$$P_{\mathbf{w}}[i] = \frac{\exp(Q_\phi(s, a_i)/\alpha)}{\sum_j \exp(Q_\phi(s, a_j)/\alpha)}, \quad \forall i \in [1, \cdots K] \tag{2.15}$$

where α is temparature. Then $\mathbf{w}$ is selected through $\mathbf{w} \sim P_{\mathbf{w}}$. Finally, the composite policy $\tilde{\pi}$ is used for a final action.

$$\tilde{\pi}(a|s) = [\delta_{a \sim \pi_\beta(s)}, \delta_{a \sim \pi_\theta(s)}]\mathbf{w}, \quad \mathbf{w} \sim P_{\mathbf{w}} \tag{2.16}$$

where $\mathbf{w} \in \mathbb{R}^K$ is a one-hot vector and $\delta_{a \sim \pi}$ is the Dirac delta distribution centered at an action $a \sim \pi$.

Our research does not utilize the categorical distribution $P_{\mathbf{w}}$ for exploration and learning.

Chapter 3

Hierarchical reinforcement learning with optimal level synchronization based on flow-based deep generative model

3.1 Introduction

Among many reinforcement learning (RL) algorithms, hierarchical reinforcement learning (HRL) stands out by addressing the limitations of atomic RL algorithms such as Deep Q-Network (DQN) [78], Deep Deterministic Policy Gradient algorithm (DDPG) [79], Soft Actor-Critic (SAC) [80], and Twin Delayed Deep Deterministic algorithm (TD3) [81]. This hierarchical approach effectively mitigates the exploration challenges and computational inefficiencies inherent in atomic RL frameworks. HRL exhibits a strong capability to efficiently seek optimal solutions in high-dimensional state and action spaces or sparse reward environments. This efficiency is achieved through temporal abstraction goals between the policies of HRL. As HRL is composed of hierarchical policies, effective communication between higher-level and lower-level policies is crucial for its performance.

In a two-level hierarchical structure, the higher-level policy focuses on abstract and generalized features, while the lower-level policy handles primitive actions in conjunction with the environment. For example, in the MuJoCo Ant environment (see Fig. 3.1), the higher-level policy learns high-level movement directions (indicated by arrows), and the lower-level policy generates actions at each step based on these higher-level directions, also referred to as higher-level actions or goals.

In goal-conditioned HRL, a policy is conditioned on an additional input called a goal, which is sent from its higher-level policy. The communication role between each HRL policy is crucial, and there are several ways to handle this in an HRL algorithm. For example, an HRL algorithm can control a goal using an atomic RL algorithm [3, 82] or various machine learning algorithms [9, 83, 84].

Meanwhile, the goal is also used for training its own policy. One of the key challenges in hierarchical RL, typically trained in a bottom-up layer-wise method, is how to train each policy with optimal data collection for the task. The off-policy correction in HRL, as suggested in HIerarchical Reinforcement learning with Off-policy correction (HIRO) [3](see Fig. 3.2), serves as an alternative solution to the optimal training issue in HRL.

HIRO combines off-policy and correction to address how to train a higher-level policy from the viewpoint of the current lower-level policy. Off-policy method has been highlighted for its ability to overcome the local minima of on-policy method. However, off-policy also faces a non-stationary issue when training a higher-level policy of HRL due to its inherent algorithm characteristics. The correction component requires careful examination.

HIRO has attempted to address this issue through various indirect estimations, proposing an off-policy correction to mitigate the challenges of off-policy learning by relabeling past experiences, specifically the actions of the higher-level policy, as goals. However, the estimation, which relies on a indirect probabilistic approach, has proven to be insufficient or imprecise due to its inherent limitations. The method's inability not to accurately reflect the current knowledge of the lower-level policy stems from its reliance on indirect estimation.

Our research emphasizes the importance of a direct estimation that accurately reflects the current knowledge of a low-level policy. This direct approach improves the effectiveness of training the higher-level policy. In this research, we propose a novel HRL model with a direct off-policy correction strategy. This strategy leverages the current knowledge of a lower-level policy using a Flow-Based Deep Generative Model (FDGM), avoiding the need for superficial data using indirect estimations.

To reflect the current knowledge of a lower-level policy in the off-policy correction after the policy is trained, we utilize the current internal parameters of its Neural Network (NN) to find an exact goal for training its higher-level policy. Using the inverse of a policy allows us to compute the goal directly, effectively relabeling a previous goal of a higher-level policy.

When the result of an inverse operation of a lower-level policy is evaluated as a goal of a higher-level policy, reflecting the current knowledge of the lower-level policy, the estimation utilizing the inverse model of a policy can become an actual and accurate off-policy correction method for HRL. In this context, finding the inverse value of a policy in a model-free HRL is defined as an inverse model of the policy.

The FDGM, capable of supporting the exact inverse value of a policy, is a promising candidate for defining an inverse model in HRL. However, FDGM has an inherent restriction between input and output dimensions, as well as a chronic defect—a biased log-density estimate. The former issue makes it challenging to define an inverse model that uses a general method to support the flexible choice of a goal dimension. Meanwhile, ongoing research is addressing the latter issue in FDGM. Our research considers an architecture inspired by [1] to leverage the advantages of FDGM in HRL despite their inherent disadvantages.

Our model is compared to HIRO regarding the methodology for off-policy correction and the issue of biased log-density estimation related to FDGM. Our model is also compared to Latent Space Policies (LSP) [82](see Fig. 3.2), which uses FDGM for a HRL, regarding the issue of invertible bijective transformation function of FDGM mentioned above.

In utilizing optimal off-policy correction to synchronize each level of HRL, several key questions arise. How can off-policy correction in HRL be effectively operated using the current internal knowledge of a newly trained lower-level policy to optimize training of its

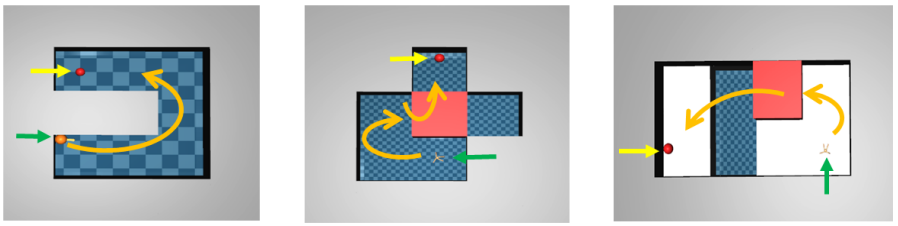


FIGURE 3.1: An example of several tasks in the Ant environment used in this research. The target location (yellow arrow) is the reward point for the approach of the agent, ant (green arrow). A trained policy requires the combination of various actions of the agent and its interplay with an object in the environment (pink blocks) in each task: the series of several directional movements (third from right), clearing away the obstacle (second from right) and making a bridge with an object (right). A higher-level policy guides its lower-level policy to be rewarded with an abstract action (orange arrow) which includes the desired positions and orientation of the ant and its limbs.

higher-level policy? Additionally, how can one address the disadvantages of FDGM when considering it for a lower-level policy to utilize its inverse operation for relabeling a goal for off-policy correction?

Our experimental results on benchmark MuJoCo environments demonstrate that our approach outperforms HIRO and LSP in terms of both speed and accuracy of performance. In summary, the contributions of this research are as follows:

- We propose a novel hierarchical RL model architecture that defines an inverse model of the model-free HRL.
- Our model adopts a direct method based on FDGM to identify an exact goal that reflects the current knowledge of a recently updated lower-level policy. This approach is crucial for effective off-policy correction in HRL.
- Despite utilizing FDGM in the lower-level policy, our model effectively addresses the drawbacks associated with FDGM used in HRL.

The rest of this paper is organized as follows. Section 3.2 introduces the preliminaries of HRL and off-policy correction. Section 2.1 analyzes related research. Section 3.3 describes our proposed HRL model using a FDGM. Section 3.4 presents the experimental results comparing the performance of our HRL model with HIRO and LSP. Section 3.5 discusses the insights gained from the experiments. Finally, Section 7 concludes this paper and outlines future works.

3.2 Preliminary Knowledge

3.2.1 Hierarchical reinforcement learning

The structure of HRL is composed of several unit agents which are stacked hierarchically. A policy of goal-conditioned HRL usually has two inputs which are the state of an environment s_t and a goal g_t, which is a temporal abstraction, sent from its higher-level policy μ_{hi}. A unit agent is usually constructed on an atomic RL since it acts as a policy in a level of HRL. The lowest-level policy communicates with an environment by its action $a_t \sim \mu_{lo}(s_t, g_t)$ based on the goal g_t of its higher-level policy which is $g_t \sim \mu_{hi}$ every c time steps (c is the horizon of μ_{hi}) or of a fixed goal transition function $h(s_{t-1}, g_{t-1}, s_t)$. The training period of a higher-level policy μ_{hi}, c, is longer than that of its lower-level policy μ_{lo} due to the hierarchical structure. The higher a policy is located in a HRL, the more abstract the policy is. An environment reacts to the action of a lowest-level policy, a_t, of a HRL with a reward R_t and a next state s_{t+1} sampled from its reward function $R(s_t, a_t)$ and its transition function $f(s_t, a_t)$ respectively. The intrinsic reward $r_t = r(s_t, g_t, a_t, s_{t+1})$ in which r is a fixed parameterized reward function for a lower-level policy in a HRL, except for the highest-level policy which uses a reward R_t of an environment, is usually supplied by a higher-level policy. Specifically, in Fig. 3.1, the higher-level policy devises an abstract strategy which is orange arrow, whereas the lower-level policy does a physical action according to the command of its higher-level policy and a current state. The model architectures such as Fig. 3.2 can show the above outline of HRL. The key notations used in this research are summarised in Table 3.1.

3.2.2 Off-policy correction

It is crucial to devise an optimal training solution to enhance the performance of HRL. Off-policy methods in HRL are proposed to address the local minimum issue of on-policy methods. In off-policy policy gradient methods, experience replay involves storing sampled trajectory data from past episodes in a replay buffer, leading to improved sample efficiency. This approach allows for better exploration by collecting samples using a behavior policy different from the target policy. The objective function of the off-policy policy gradient is

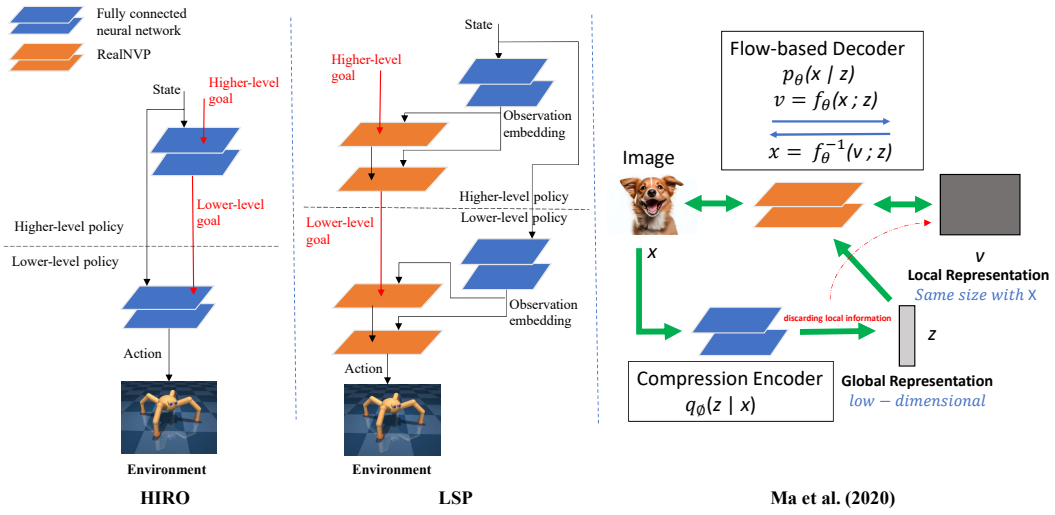


FIGURE 3.2: The model architecture of HIRO (left), LSP (center) and Ma et al. (right).

as follows,

$$J(\theta) = \mathbb{E}_{S\sim d^\beta}\left[\sum_{a\in A} Q^\pi(s,a)\pi_\theta(a|s)\right] \tag{3.1}$$

where $d^\beta(s)$ is the stationary distribution of the behaviour policy $\beta(a|s)$, $\pi_\theta(a|s)$ is the target policy and $Q^\pi(s,a)$ is the action-value function estimated regarding the target policy $\pi_\theta(a|s)$ [85].

However, the off-policy method in goal-conditioned HRL faces a non-stationary issue due to the bottom-up training approach in off-policy RL. Previously used outputs, such as goals from the higher-level policy, may not be suitable for training as they do not reflect the parameter value θ of the newly trained lower-level policy. Consider the operation of a lower policy μ_{lo} in HRL as follows:

$$a_t \sim \mu_{lo,\theta}(s_t, g_t), \text{ at time } t. \tag{3.2}$$

After $\mu_{lo,\theta}$ is trained after c steps,

$$\begin{gathered} a_{t+c} \sim \mu_{lo,\theta'}(s_{t+c}, g_t), \text{ assume } s_{t+c} = s_t \\ a_t \neq a_{t+c} \text{ or } a_t = a_{t+c} \end{gathered} \tag{3.3}$$

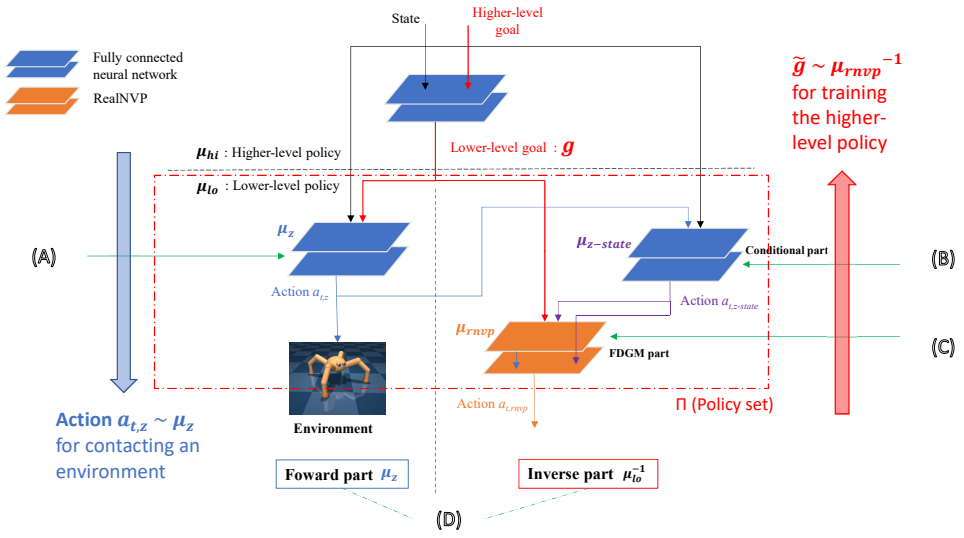


FIGURE 3.3: The architecture of our suggested model using FDGM. (A) A low-level policy of HIRO is used only for the forward part to Environment. (B) The observation embedding of LSP, which is an encoder, is replaced by a low-level policy of HIRO as an independent policy. (C) The RealNVP of LSP, which is a neual network and belongs to a policy, is used as an independent policy. (D) The combination of a forward part and an inverse part is adopted from the model architecture of [1].

where a_{t+c} and s_{t+c} are the action of $\mu_{lo,\theta'}$ and the state of the environment after c steps respectively. a_{t+c} may not be the same as a_t for the value of the $\mu_{lo,\theta}$ parameter which might be different from that of the $\mu_{lo,\theta'}$ parameter.

To resolve this problem, an off-policy correction is proposed in HIRO [3] through an indirect estimation strategy.

3.3 Our model

The aim of our research is not to use an indirect estimation like HIRO, but rather to employ a direct estimation that accurately reflects the current knowledge of lower-level policy to find an optimal goal for level synchronization in training the higher-level policy. The most direct approach is to find a goal, $\tilde{g}_t$, through the inverse operation of lower-level policy for training its higher-level policy μ_{hi}, expressed as:

$$\tilde{g}_t \sim \mu_{lo}^{-1}(a_t) \tag{3.4}$$

TABLE 3.1: Key Notations.

Symbol	Meaning
t	action step
μ_{hi}	higher-level policy
μ_{lo}	lower-level policy
μ_z	the policy of forward part
$\mu_{z-\text{state}}$	the policy of conditional part
μ_{rnvp}	the policy of FDGM part
μ_{rnvp}^{-1}	the inverse of the policy of FDGM part
Π	the lower-level policy set composing of $[\mu_z, \mu_{z-\text{state}}, \mu_{\text{rnvp}}]$
L_z	the loss of forward part
$L_{z-\text{state}}$	the loss of conditional part
L_{rnvp}	the loss of FDGM part
L_{common}	the averaged loss over the sum of loss of all three policies
g_t *or* g	the action of higher-level policy called a goal
$\tilde{g}_t$	a goal obtained through off-policy correction for training the higher-level policy μ_{hi}
a_t	the action of lower-level policy
R_t	the extrinsic reward of an environment
r_t	the intrinsic reward of lower-level policy
s_t *or* s	the state of the environment
$a_{t,z}$	the action of forward part
$a_{t,\text{rnvp}}$	the action of FDGM part of inverse part
$a_{t,z-\text{state}}$	the action of conditional part of inverse part
$[g, a]$	the concatenation of $g_{1:d}$ and an action $a_{t,z-\text{state}}$
$[a_{\text{rnvp}}, a_{z-\text{state}}]$	the concatenation of an action $a_{t,\text{rnvp}}$ and an action $a_{t,z-\text{state}}$
$x_{t:t+c-1}$	the sequence $x_t, \ldots, x_{t+c-1}$
c	the horizon of a higher-level policy
D_R	replay buffer

where a_t is the action of lower-level policy, μ_{lo}. Given that FDGM supports invertible algorithms and fast generation speeds, we adopt an FDGM-based lower-level policy in our research to relabel a goal for training a higher-level policy.

However, while FDGM is invertible, its bijective transformation property poses constraints on goal dimensions, particularly in incorporating an inverse model for model-free HRL. This constraint arises from the requirement for the input and output dimensions of FDGM to be the same, making it challenging to secure a model that allows for the freedom of goal dimensions between policies in HRL. In addition, FDGM suffers from the chronic drawback of biased log-density estimation.

To address those challenges, we define a lower-level policy set, denoted as Π, for the lower-level policy.

$$\Pi = [\mu_z,\ \mu_{z-\text{state}},\ \mu_{\text{rnvp}}] \tag{3.5}$$

We splits the lower-level policy, μ_{lo}, into two parts: a forward part composed of μ_z, which interacts with the environment, and an inverse part (μ_{lo}^{-1}) consisting of $\mu_{z-\text{state}}$ and μ_{rnvp} for the inverse model. The forward part is responsible for generating a real action in the environment. The action of the lower-level policy for the environment, denoted as $a_t \sim \mu_{lo}$, is determined by μ_z from Π at the current state s_t. The inverse part includes two components: the μ_{rnvp} part, which uses the output of the conditional part and a goal from the higher-level policy as inputs, and the conditional part ($\mu_{z-\text{state}}$), which supports the μ_{rnvp} part.

To obtain the local representation, the main inputs of the inverse part of our model are s_t and g_t for the FDGM part, and the output of the forward part, $a_{t,z}$, for the conditional part. However, the purpose of the inverse part is to find $\tilde{g_t}$ using the inverse operation of the FDGM part. Therefore, one of the inputs, s_t, of the FDGM part is transferred to an input of the conditional part. This design improves performance, distinguishing it from the observation embedding of LSP, which serves as an encoder.

To ensure the exact inverse operation of our model concerning the output of the corresponding level of the lower-level policy, the critical consideration is aligning the output of the inverse part, $a_{t,\text{rnvp}}$, with the output of the forward part, $a_{t,z}$. In essence, both outputs, $a_{t,z}$ and $a_{t,\text{rnvp}}$, should share a common internal factor under the common inputs, a goal g_t and a state s_t, despite their differing dimensions. Therefore, $a_{t,z-\text{state}}$ of $\mu_{z-\text{state}}$ serves as the refined global representation.

Through Π, our model ensures the freedom to choose the dimension of the goal at any level and addresses the issue of biased log-density estimation in FDGM. Since off-policy correction is not required at the highest level, the other levels, except for the highest level, can adopt the two parts of our model architecture: a forward part and an inverse part.

Algorithm 1 Find the goal using only μ_{rnvp}^{-1}

Inputs:
$a_{z-\text{state}}$, a_{rnvp}
Output:
$\tilde{g}$
Initialize:
Set the layer dimension of all policies including μ_{hi} and the critic of μ_{rnvp} to be the same except for the actor of μ_{rnvp}
Set the layer dimension of actor of μ_{rnvp}
Set the action dimension of $\mu_{z-\text{state}}$
for each c **do**
Sample $a_{z-\text{state}}$, $a_{\text{rnvp}} \sim D_R$
$\tilde{g} \sim \mu_{\text{rnvp}}^{-1}(a_{z-\text{state}}, a_{\text{rnvp}})$
end for each

Algorithm 2 Find the goal using all parts of μ_{lo}

Inputs:
s, g, a_{rnvp}
Output:
$\tilde{g}$
Initialize:
Set the layer dimension of all policies (a forward part and an inverse part) inside Π to be the same except for μ_{hi}
Set the layer dimension of μ_{hi}
Set the action dimension of $\mu_{z-\text{state}}$
for each c **do**
Sample $s, g, a_{\text{rnvp}} \sim D_R$
$a_z^{'} \sim \mu_z(s, g)$
$a_{z-\text{state}}^{'} \sim \mu_{z-\text{state}}(s, a_z^{'})$
$\tilde{g} \sim \mu_{\text{rnvp}}^{-1}(a_{z-\text{state}}^{'}, a_{\text{rnvp}})$
end for each

Fig. 3.3 shows the full architecture of our model based on a two-level hierarchical structure. Each part of the lower-level policy, μ_{lo}, including the off-policy correction used in our model, is thoroughly explained in the following section.

3.3.1 Forward part μ_z

Similar to the lower-level policy of HRL, the action of the forward part interacts with an environment or another lower-level policy. We consider the forward part as the VAE framework of [1], which extracts the global representation of a goal g_t and a state s_t in

HRL as follows:

$$a_{t,z} \sim \mu_z(s_t, g_t) \text{ at time } t \tag{3.6}$$

where μ_z is the policy of the forward part, and $a_{t,z}$ is the action of μ_z. The action $a_{t,z}$ plays a role in the global representation, which helps in finding the goal $\tilde{g}_t$ in the inverse part for relabelling $g_t \sim D_R$.

The fixed dimension of action $a_{t,z}$, determined by a given environment and different from the dimension of g_t, can compensate for a potential weakness of the inverse part, which uses an FDGM and requires a dimension matching that of g_t. Since the outputs of both parts, $a_{t,z}$ and $a_{t,\text{rnvp}}$, are equivalent from the perspective of the higher-level policy, we can consider $a_{t,z}$, the output of the forward part, as a global representation that corresponds to the output of the inverse part, $a_{t,\text{rnvp}}$.

3.3.2 Inverse part μ_{lo}^{-1}

We aim to utilize the inverse operation of FDGM to compute a goal $\tilde{g}_t$ that corresponds to the updated lower-level policy, μ_{lo}. Our model architecture is inspired by the approach of [1], which employs two distinct representations to leverage the strengths and weaknesses of FDGM.

3.3.2.1 Conditional part $\mu_{z-\text{state}}$

While the conditional part serves a similar role to the observation embedding in LSP as a conditional input of FDGM, it differs in two key aspects. Firstly, unlike the observation embedding in LSP, which is not trained as an encoder using a fully connected neural network, it operates as an independent policy for improved performance. Secondly, $\mu_{z-\text{state}}$ takes in both state s_t and action $a_{t,z}$ to handle the global representation, whereas the observation embedding of LSP only takes in the state s_t to change the dimension of state s_t.

$$a_{t,z-\text{state}} \sim \mu_{z-\text{state}}(s_t, a_{t,z}) \text{ at time } t \tag{3.7}$$

where $\mu_{z-\text{state}}$ is the policy of conditional part and $a_{t,z}$ is the action of forward part.

There are three reasons why we consider the conditional part. The first reason is to provide a refined global representation $a_{t,z}$, while preserving the local representation $a_{t,\text{rnvp}}$, to the conditional input of FDGM. To address the chronic weakness in the FDGM part, which is a biased log-density estimation, we take into account the effect of the conditional part, including a global representation of a goal g_t and a state s_t.

The second reason is that controlling the dimension of the action $a_{t,z-\text{state}}$ enhances the role of $a_{t,z}$ effectively in the FDGM part. Although $a_{t,z}$ is fixed depending on the environment, we aim to indirectly control the dimension of $a_{t,z}$ through $a_{t,z-\text{state}}$ to optimize its effect on the output of the FDGM part.

The third reason is to modulate the dimension of the state s_t if its dimension is greater or less than that of the goal g_t, as both are original inputs in the FDGM part, to achieve a better $\tilde{g_t}$ through μ_{lo}^{-1}.

Finding the optimal dimension for $a_{t,z-\text{state}}$ involves a trade-off process that can be refined through experimentation.

3.3.2.2 FDGM part μ_{rnvp}

Its inputs are a goal g_t given from its higher-level policy as a main input and an action $a_{t,z-\text{state}}$ given from the conditional part as a conditional input of FDGM as follows:

$$a_{t,\text{rnvp}} \sim \mu_{\text{rnvp}}(g_t, a_{t,z-\text{state}}) \text{ at time } t \tag{3.8}$$

where μ_{rnvp} is the policy of FDGM part and $a_{t,\text{rnvp}}$ is the action of μ_{rnvp}. The principle of FDGM part can be explained using the forward transformation property of RealNVP [25] in a layer as follows:

$$\begin{aligned} a\_\text{rnvp}_{1:d} &= g_{1:d} \\ a\_\text{rnvp}_{d+1:D} &= g_{d+1:D} \odot \exp(s([g,a])) + t([g,a]) \end{aligned} \tag{3.9}$$

where the element-wise product is represented as the $\odot$ symbol. A goal g_t is split into a $g_{1:d}$ and a $g_{d+1:D}$. The concatenation of the $g_{1:d}$ and the action $a_{t,z-\text{state}}$ is a $[g,a]$. An

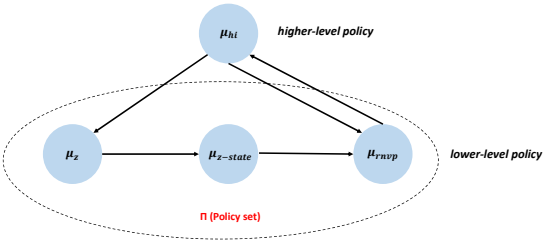


FIGURE 3.4: Communication graph representing the dependence of our model.

$a\_\text{rnvp}_{1:d}$ and an $a\_\text{rnvp}_{d+1:D}$ are the outputs of a coupling layer and will be concatenated for the other layer. Furthermore, $s(\cdot)$ and $t(\cdot)$ are scale and translation functions and map $R^d \mapsto R^{D-d}$. They are made by a deep neural network since the inverse function of each bijection and the Jacobian determinant does not require any computation regarding s and t. The inverse operation together with the forward operation can be calculated as follows:

$$\begin{cases} a\_\text{rnvp}_{1:d} = g_{1:d} \\ a\_\text{rnvp}_{d+1:D} = g_{d+1:D} \odot \exp(s([g,a])) + t([g,a]) \end{cases} \tag{3.10}$$

$$\iff \begin{cases} g_{1:d} = a\_\text{rnvp}_{1:d} \\ g_{d+1:D} = (a\_\text{rnvp}_{d+1:D} - t([a_\text{rnvp}, a_{z-\text{state}}])) \\ \odot \exp\left(-s([a_\text{rnvp}, a_{z-\text{state}}])\right) \end{cases} \tag{3.11}$$

where an action $a_{t,\text{rnvp}}$ is split into an $a\_\text{rnvp}_{1:d}$ and an $a\_\text{rnvp}_{d+1:D}$. The concatenation of $a_{t,\text{rnvp}}$ and $a_{t,z-\text{state}}$ is an $[a_\text{rnvp}, a_\text{z-state}]$. A $g_{1:d}$ and a $g_{d+1:D}$ are the outputs of a coupling layer in the inverse operation.

The dimension of $a_{t,\text{rnvp}}$ is determined by the dimension of g_t due to the bijective transformation property of FDGM. The output of the FDGM part, $a_{t,\text{rnvp}}$, is used not for giving an action to an environment but for the off-policy correction of its higher-level policy after it is stored in replay buffer D_R.

3.3.3 Off-policy correction, convergence and algorithm

A replay buffer collected by a higher-level policy for the off-policy RL is filled with state-action-reward transitions (s_t, g_t, $a_{t,z}$, $a_{t,z-\text{state}}$, $a_{t,\text{rnvp}}$, $\Sigma R_{t:t+c-1}$, s_{t+c}) based on a higher-level transition tuple ($s_{t:t+c-1}$, $g_{t:t+c-1}$, $a_{t:t+c-1,z}$, $a_{t:t+c-1,z-\text{state}}$, $a_{t:t+c-1,\text{rnvp}}$, $R_{t:t+c-1}$, s_{t+c}). Finally, in case of the off-policy training with the step t transition of replay buffer at time

$t + c$ steps,

$$J_{(..., g_t, ...) \sim D_R}(\theta)_{t+c} \neq J(\theta)^{OPT}_{t+c} \tag{3.12}$$

where $J_{(..., g_t, ...) \sim D_R}(\theta)_{t+c}$ is the objective function of $\mu_{hi,\theta}$ and $J(\theta)^{OPT}_{t+c}$ is the optimal objective function of $\mu_{hi,\theta}$ on condition that $a_t \neq a_{t+c}$ mentioned at time $t + c$ of Section 3.2.2. In this research, the inverse operation of FDGM of its lower-level policy is executed by

$$\tilde{g}_{t+c} \sim \mu^{-1}_{\text{rnvp}}(a_{t,\text{rnvp}}, a_{t,z-\text{state}}) \tag{3.13}$$

with $a_{t,\text{rnvp}}$ and $a_{t,z-\text{state}}$ taken from the replay buffer to get the $\tilde{g}_{t+c}$. To relabel g_t, $\tilde{g}_{t+c}$ is utilized for the level synchronization of the adjacent policies when the higher-level policy is trained after the lower-level policy is trained in a bottom-up layerwise fashion of HRL. As g_t is relabelled with $\tilde{g}_{t+c}$ by the off-policy correction of our model approaches to $g_{t+c,ideal}$ which ideally reflects the current internal parameter, $\mu_{rnvp,\theta}$, of μ_{lo}, the objective function of $\mu_{hi,t+c}$ closes to its optimal one if an agent experiences all states of an environment as follows:

$$\begin{gathered} \text{If } \lim_{s \to \infty} \tilde{g}_{t+c} = g_{t+c,ideal}, \\ \text{then } \lim_{s \to \infty} J_{(..., \tilde{g}_{t+c}, ...) \sim D_R}(\theta)_{t+c} = J(\theta)^{OPT}_{t+c}. \end{gathered} \tag{3.14}$$

Our model combines three policies of the same type along with the common loss $L_{\text{common actor}}$ and $L_{\text{common critic}}$ (See Section 3.4) into a lumped lower-level policy μ_{lo}. Meanwhile, the dependence of each policy in our model is represented in Fig. 3.4. Our research leverages TD3, which has been verified for its convergence [81], for all policies. Hence, the convergence of the policy of each level in our model also corresponds to the dependence of three sub-policies and the higher-level policy, relying on the convergence characteristic of the atomic policy of each level in the off-policy training.

The dynamics of the inverse operation using our inverse model for optimal performance varies for each task within the environment. Thus, we utilize Algorithm 1 and Algorithm 2 to find an optimal goal for the off-policy correction using our inverse model as follows.

- Firstly, set the layer dimension (dim.) of all policies of our model and the action dim. of $\mu_{z-\text{state}}$.

- Secondly, employ their own method to find a goal through μ_{lo}. Section 3.4 shows the layer dim. of all policies and the action dim. of $\mu_{z-\text{state}}$ according to an Algorithm used in each result of experiments.

3.4 Experiments

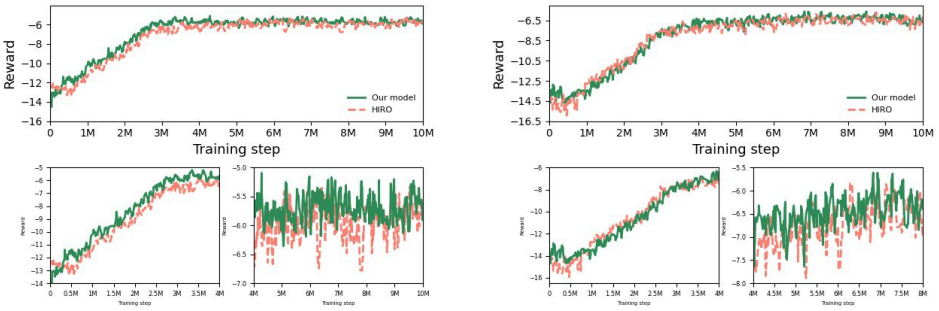
In this research, we investigate whether our inverse model in HRL, using the direct estimation with FDGM for off-policy correction, can achieve competitive performance against HIRO and LSP. Our implementation is based on the HIRO open code[1] and RealNVP open code[2] in LSP, using the same exact optimization algorithm for transparent benchmarking. Consequently, we integrate the two models, our model and LSP, into the HIRO code framework. We have released our code[3] with explanations.

The two-level hierarchical structure of HRL is used because the same HRL structure of HIRO is used. The intrinsic reward function of a lower-level policy defined in HIRO is used without modification. All other neural networks including the critic of the FDGM part except for the actor of the FDGM part, which is a RealNVP, consist of a fully connected neural network as utilized in HIRO.

HIRO utilizes the TD3 policy gradient algorithm, which is a variant of the DDPG for continuous control. Since our model is built based on the HIRO source, all policies of our model also utilize TD3. Therefore, all policies of the lower-level policy set, $\mathbf{\Pi}$, take advantage of the actor and critic loss of DDPG for their learning. Instead of using each loss for each policy of $\mathbf{\Pi}$, the losses used for training the actor and critic of the three policies are the common ones, $L_{\text{common actor}}$ and $L_{\text{common critic}}$, which are averaged over the sum of the losses of all three policies as follows:

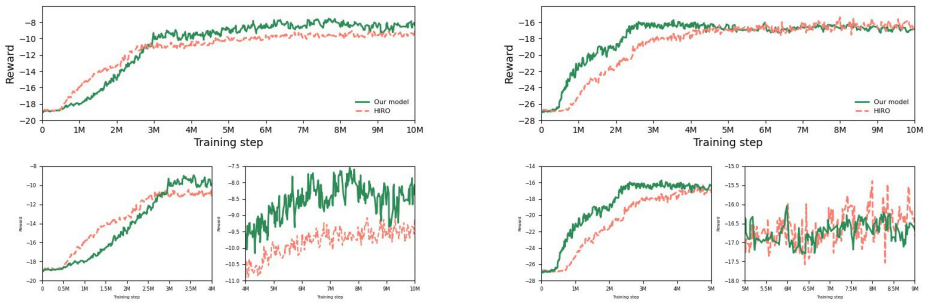
$$\begin{aligned} L_{\text{common actor}} &= (L_{z\,actor} + L_{z-\text{state actor}} + L_{\text{rnvp actor}})/3 \\ L_{\text{common critic}} &= (L_{z\,critic} + L_{z-\text{state critic}} + L_{\text{rnvp critic}})/3 \end{aligned} \tag{3.15}$$

[1] https://github.com/tensorflow/models/tree/master/research/efficient-hrl
[2] https://github.com/haarnoja/sac
[3] https://github.com/jangikim2/Hierarchical_Reinforcement_Learning/tree/master/distributed_inverse-efficeint-hrl-sac_210127_tfp_bijectors.bijector_dataefficient_three_reflected_888_0420



(A) Ant Push Multi

(B) Ant Fall Multi



(C) Ant Push Single

(D) Ant Fall Single

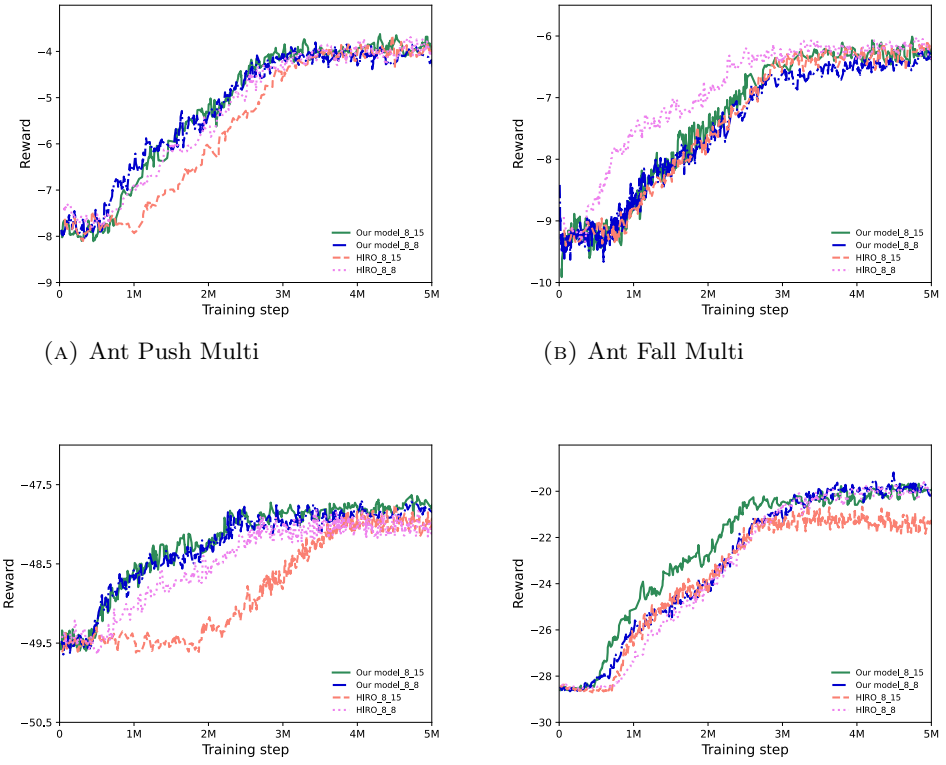
FIGURE 3.5: For the task of the higher-level policy, the average rewards of our model (green) are compared with that of HIRO (salmon) with the same NN size on condition of the optimal goal space.

where $L_{z\,actor}$ and $L_{z\,critic}$ are the actor loss and critic loss of μ_z, $L_{z-\text{state actor}}$ and $L_{z-\text{state critic}}$ are the actor loss and critic loss of $\mu_{z-state}$ and $L_{\text{rnvp actor}}$ and $L_{\text{rnvp critic}}$ is the actor loss and critic loss of μ_{rnvp}.

The implementation of the lower-level policy and off-policy correction of the HIRO open code has been modified. To achieve the main purpose of the experiment, we make two specific modifications to the open-source HIRO code as follows:

1. We replace the neural network of the lower-level policy of HIRO with the architecture of our model.

2. We replace the indirect probabilistic method of HIRO with the inverse operation of our model, as represented in Algorithm 1 or Algorithm 2, for off-policy correction. All other parts of the code remain unchanged.



(A) Ant Push Multi

(B) Ant Fall Multi

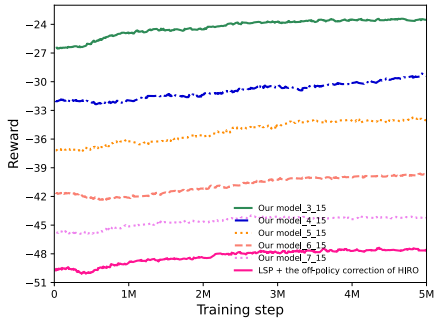
(C) Ant Push Single

(D) Ant Fall Single

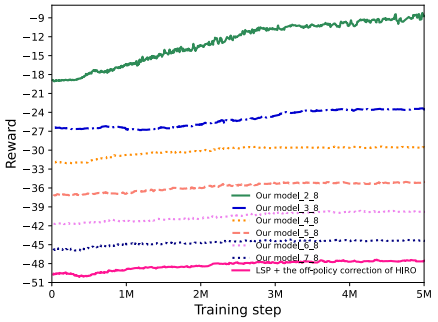
FIGURE 3.6: The average rewards of our model (green, blue) for the task of the higher-level policy are compared with that of HIRO (salmon, violet) on condition of a non-optimal goal space. The numbers of (our model or HIRO)_8_(15 or 8) are the goal dimension of higher-level policy and that of lower-level policy respectively.

For LSP, we replace all of LSP's policies with HIRO's model architecture on the goal space designed in HIRO, except for the actor of the FDGM part, as mentioned earlier. Despite LSP not originally using off-policy correction, we utilize off-policy correction using the inverse operation of FDGM of the lower-level policy for a fair comparison.

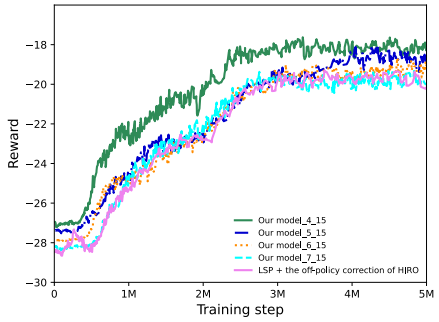
Since HIRO researches the Ant environment in Mujoco, which is part of the OpenAI Gym, the HIRO open-source code defines optimal maximum and minimum tuple values for the goals of the higher-level policy and lower-level policy. In this experiment, we use them as the standard goal spaces. For more details, refer to Appendix 3.6.1 and 3.6.2. Meanwhile, in our evaluation, we focus on the harder tasks of the Ant environment, such as Ant Push and Ant Fall, with different modes (Multi and Single) among the tasks which are available in HIRO.



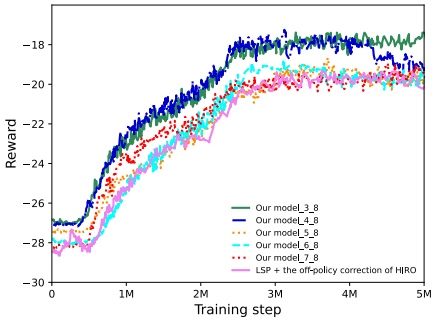
(A) Ant Push Single



(B) Ant Push Single



(C) Ant Fall Single



(D) Ant Fall Single

FIGURE 3.7: For the task of the higher-level policy, the average rewards of our model (green, blue, orange, salmon, violet and navy for Ant Push Single and green, blue, orange, cyan and red for Ant Fall Single) are compared with that of LSP (pink for Ant Push Single and violet for Ant Fall Single). They are based on the change of higher-level goal dimension of a) our model_8_15, (b) our model_8_8 (c) our model_8_15 (d) our model_8_8 used in the corresponding task of Fig. 3.6.

We conduct two comparisons between our model and HIRO regarding the methodology for off-policy correction and the biased log-density estimation related to FDGM.:

1. Optimal goal space comparison in Section 3.4.1 : by using the optimal maximum and minimum tuple values for the goals of the higher-level and lower-level as defined in HIRO as default parameters.

2. Non-optimal goal space comparison in Section 3.4.2 : by modifying some dimensions of optimal maximum and minimum tuple values of goal of higher-level and lower-level defined in HIRO.

Our model is also compared with LSP (see Section 3.4.3) in terms of the average reward result of the higher-level policy for the inherent restriction between input and output dimensions of FDGM. An ablation study is conducted to compare a variant of our model with our model (see Section 3.4.4). Both our model and HIRO are evaluated in tasks of the Mujoco ant environment from the OpenAI Gym, including Ant Push Multi, Ant Fall Multi, Ant Push Single, and Ant Fall Single, to evaluate the performance of our research. Our model is specifically evaluated in Ant Push Single and Ant Fall Single against LSP. Further details of the experiment are provided in Appendix 3.6.1 and 3.6.2.

The objectives of our experiments are as follows:

1. To verify that our model improves the performance of the agent more than HIRO regarding off-policy correction..
2. To determine the extent to which our model addresses the chronic issue of FDGM in HRL.
3. To evaluate the impact of the flexible goal dimension in our model on the agent's performance compared to LSP.
4. To compare our model with the variation of our model.

3.4.1 Comparative analysis with the optimal goal space

Algorithm 1 and 2 aim to find the optimal $\tilde{g}_t$ from the FDGM part reflecting the current knowledge of the lower-level policy in each task. In Fig. 3.5, the average results of our model for the tasks of the higher-level policy are compared with those of HIRO with default parameters including the optimal goal space. Our model shows obviously better performance in most tasks.

- Ant Push Multi (Actor of FDGM : 145 dim., All other NNs : 170 dim., $a_{t,z-\text{state}}$: 16 dim.) in Fig. 3.5a: Our model achieves better scores than HIRO with default parameters until 4.5M. Then our result marginally outperforms HIRO up to 10M. Algorithm 1 is used for our model.

- Ant Fall Multi (Actor of FDGM : 150 dim., All other NNs : 170 dim., $a_{t,z-\text{state}}$: 18 dim.) in Fig. 3.5b: Our model and HIRO with default parameters achieve similar results until 4.8M. Then, the performance of our model is slightly better up to 8M. Algorithm 1 is used for our model.

- Ant Push Single (Actor of FDGM : 130 dim., All other NNs : 150 dim., $a_{t,z-\text{state}}$: 19 dim.) in Fig. 3.5c: Our model achieves considerably better performance than HIRO with default parameters until 10M, although our model's result is not as good as HIRO's up to 3M. Algorithm 1 is used for our model.

- Ant Fall Single (All lower policy's NNs : 135 dim., All higher policy's NNs : 160 dim., $a_{t,z-\text{state}}$: 24 dim.) in Fig. 3.5d: Our model achieves a much higher speed of learning up to 5.25M than HIRO with default parameters. Our result is similar to HIRO's up to 8M, after which it is slightly lower until 10M. Algorithm 2 is used for our model.

3.4.2 Comparative analysis with a non-optimal goal space

A non-optimal goal space for two models, HIRO and our model, is required for fair comparison. We consider two cases for the non-optimal goal space. The first is 8 goal dimensions of the higher-level policy, whose default goal dimension is 2 or 3 depending on the task, replaced with the first 8 goals of the lower-level policy of HIRO, whose default goal is 15. It is represented as (Our model or HIRO)_8_15, which designates *Model name_ the goal dim. of the higher-level policy_ the goal dim. of the lower-level policy.* As a consequence, the first default 8 goals of the lower-level policy of HIRO are selected for the goal of the higher-level policy of (Our model or HIRO)_8_15. The goal space of the lower-level policy in (Our model or HIRO)_8_15 is the same as the default goal space of the lower-level policy of HIRO. The second is (Our model or HIRO)_8_8, which removes the last 7 goals of the lower-level policy of (Our model or HIRO)_8_15. Figure 3.9 in the Appendices 3.6 shows more details regarding the above two goal spaces.

For the comparison with HIRO, first, we tune the appropriate parameters, which are the layer dim. of all policies of each level and the $a_{t,z-\text{state}}$ dim. of the conditional part, for

our model to achieve the best performance in each task by using Algorithm 1 or 2. Then, the performance of HIRO with the found layer dim. of the best performance of our model is compared with our model for each task.

The conditions for achieving the best performance with our model are as follows:

- Ant Push Multi (Our model - Actor of FDGM : 135 dim., All other NNs : 160 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 12 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 20 dim.) - Our model_8_8 and Our model_8_15 show better performance than HIRO_8_8, which is the best among HIRO's two models. Algorithm 1 is used for our model.
- Ant Fall Multi (Our model - Actor of FDGM : 140 dim., All other NNs : 165 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 12 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 15 dim.) - HIRO_8_8 shows the best performance among all other models. Algorithm 1 is used for our model.
- Ant Push Single (Our model - Actor of FDGM : 130 dim., All other NNs : 150 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 9 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 22 dim.) - Both of our models show better performance than HIRO's two models. Algorithm 1 is used for our model.
- Ant Fall Single (Our model - Actor of FDGM : 135 dim., All other NNs : 160 dim., Our model_8_15 - $a_{t,z-\text{state}}$: 26 dim., Our model_8_8 - $a_{t,z-\text{state}}$: 22 dim.) - Our model_8_15 shows the best performance among all other models. The inverse operation of Algorithm 2 which is combined with the size of NN of Algorithm 1 is used for our model.

Our model shows the best performance in most tasks. In only Fig. 3.6b, our model's performance is behind HIRO's. Compared with tasks with a high reward shape, such as Ant Push Multi and Ant Fall Multi, our model demonstrates much superiority over HIRO in tasks with a low reward shape, such as Ant Push Single (Fig. 3.6c) and Ant Fall Single (Fig. 3.6d), where there is ample room for improvement. The performance of our model is impressive in the early stage, which does not have enough data in a replay buffer for off-policy correction. The main reason is that our model can take advantage of an exact goal, $\tilde{g}_t$, by using the direct inference from the inverse operation of FDGM.

3.4.3 Comparative analysis between our model and LSP

One of the purposes of our model is to make the goal space of all policies in HRL flexible without any restrictions. Our model_(2,3,4,5,6 or 7)_(8 or 15) exemplifies the goal flexibility of our model. These non-optimal goal spaces are constructed using the same method as (Our model or HIRO)_8_(8 or 15). The goal space of the higher-level policy is replaced with the first x goals, which is 2,3,4,5,6 or 7, of the lower-level policy of HIRO, whose default goal is 15. Fig. 3.9 in the Appendix shows examples of some cases of the above goal space.

In Fig. 3.7, our model_(2,3,4,5,6 or 7)_(8 or 15) with the same layer dim. as the corresponding task of our model_8_(8 or 15) in Fig. 3.6 is compared with LSP to demonstrate the superiority of the goal flexibility of our model over LSP. All output dim. of LSP for selecting a goal space for all level policies of HRL should match the dim. of the action space of the environment, which is 8 in the Mujoco ant environment, due to the characteristic of the invertible bijective transformation of FDGM. Thus, LSP should have LSP_8_8. The layer dim. of LSP is configured based on the configuration of our model in Ant Push Single (Fig. 3.6c) and Ant Fall Single (Fig. 3.6d). Additionally, the observation embedding dim. of LSP is tuned to achieve the best performance, as follows:

- Ant Push Single - Higher policy's NN : 150 dim., Lower policy's NN : 130 dim., The output dim. of each level's observation embedding : 23 dim. in Fig. 3.7a and 3.7b
- Ant Fall Single - Higher policy's NN : 160 dim., Lower policy's NN : 135 dim., The output dim. of each level's observation embedding : 14 dim. in Fig. 3.7c and 3.7d

Fig. 3.7 illustrates the effectiveness of the choice of goal dim. of the higher-level policy of our model. It shows that the smaller the goal dimension of the higher-level policy of our model, the better its performance. This trend indicates that the goal dim. of the higher-level policy of our model approaches the default goal dim. of HIRO's higher-level policy. LSP is significantly inferior to our model, highlighting the superiority of our model's flexibility in goal dimension.

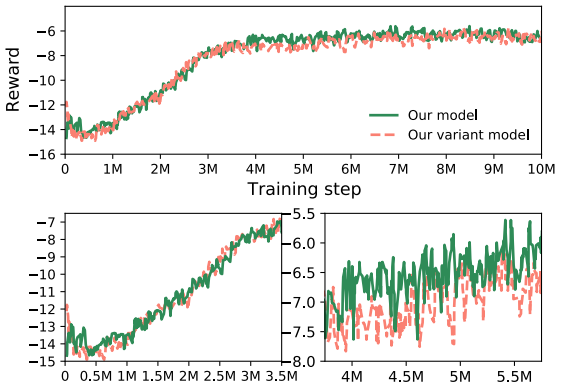


FIGURE 3.8: Average result of our method (green) compared with that of the variant model (salmon) in Ant Fall Multi.

3.4.4 Ablation study

The importance of $a_{t,z}$ is of interest due to its significant role in addressing the chronic issue of FDGM, specifically the biased log-density problem. Considering the potential impact on our model's performance, it would be intriguing to explore how replacing $a_{t,z}$ with goal g_t as the input of the conditional part affects our model's performance.

- Ant Fall Multi of variant model (Actor of FDGM : 150, All other NNs : 180, $a_{t,z-\text{state}} = 19$)

In this experiment, we do not set the parameters of each policy to be the same as that of our model in the comparative analysis. First, we find the parameters of the variant model with the optimal goal space for its best performance in the Ant Fall Multi task of Section 3.4.1. Then, we compare the performance of our model from the previous Ant Fall Multi experiment with the optimal goal space to the variant model with the newly found parameters. This comparison reveals that although both experiments result in the same performance until 3.5M, our original model marginally outperforms the variant model up to 7.5M. Subsequently, both models perform similarly up to 10M. The performance of our model against the variant model is shown in Fig. 3.8.

3.5 Discussion

3.5.1 The performance of the higher-level policy based on our model in all tasks

In most of the experiments, our model outperforms HIRO and LSP. Despite not using any indirect estimations such as the indirect probabilistic method of HIRO, our model achieves competitive performance against HIRO and LSP by utilizing the inverse operation of FDGM. Particularly, the lower-level policy, which comprises three policies, does not significantly affect the performance of the early learning stage of the higher-level policy. However, fine-tuning the dimension of $a_{t,z-\text{state}}$ and the layer dim. of both level policies is required, as they have an effect on the performance of the higher-level policy.

3.5.2 The inference from the performance of our model compared with LSP

Even though our model has a non-optimal goal space, it demonstrates competitive performance, highlighting the importance of freedom in choosing the goal dimension in HRL. In Fig. 3.7, LSP cannot show the competitive performance due to its inherent characteristic, a fixed goal space. In contrast, our model exhibits exclusive performance potential due to its flexibility in choosing the goal space.

3.5.3 The inverse operation and layer dim. in Algorithm 1 and Algorithm 2

The reason for considering Algorithm 1 and 2 is to address the level and speed of performance of the higher-level policy, respectively. Since our model utilizes TD3 in a policy, the FDGM part also includes a combination of actor and critic. Interestingly, when the layer dim. of the actor is the same as that of all other neural networks in Algorithm 1, the performance of the higher-level policy falls short of our expectations. However, when there is a difference in the layer dim. of the neural networks, particularly for the actor of FDGM compared to the other neural networks, the performance meets our expectations.

Algorithm 1 performs a simple inverse operation that reflects only the FDGM part of the lower-level policy, while Algorithm 2 considers all three sub-parts of the lower-level policy with different layer dimensions between the higher-level and lower-level policies. Through this research, the performance level of Algorithm 1 is superior to that of Algorithm 2. However, Algorithm 2 tends to achieve faster performance. In one of our experiments with a non-optimal goal space, namely Ant Fall Single, we combine the two algorithms to leverage their respective strengths.

3.5.4 The performance difference between the low reward shape and the high reward shape

Even though our model achieve better performance than HIRO for all tasks, there is a difference in performance between the high reward shape of Ant Push Multi and Ant Fall Multi and the low reward shape of Ant Push Single and Ant Fall Single. This difference may be attributed to two main reasons. Firstly, the chronic issue of FDGM, i.e., biased log-density estimation, still affects the inverse operation of the FDGM. In particular, RealNVP used in our implementation is the early version of FDGM. Secondly, the reconstruction loss of the Autoencoder also plays a role. The forward part and conditional part of our model function as an encoder and a decoder, respectively. In tasks with low or high reward shapes, our model exhibits different performance in environments where the size of the feature space is much larger than that of the action space due to the reconstruction loss of the Autoencoder.

3.5.5 The further research for an environment with a small action space and a big feature space

The inputs of conditional part are s_t and $a_{t,z}$. In environments where the action space is small and the feature space is large, the model architecture of [1] for unbiased log-density estimation in our model may not work properly because the dimension of the global representation, $a_{t,z}$, is much smaller than that of s_t. Therefore, to increase the dimension of the global representation, a decoder for $a_{t,z}$ is required.

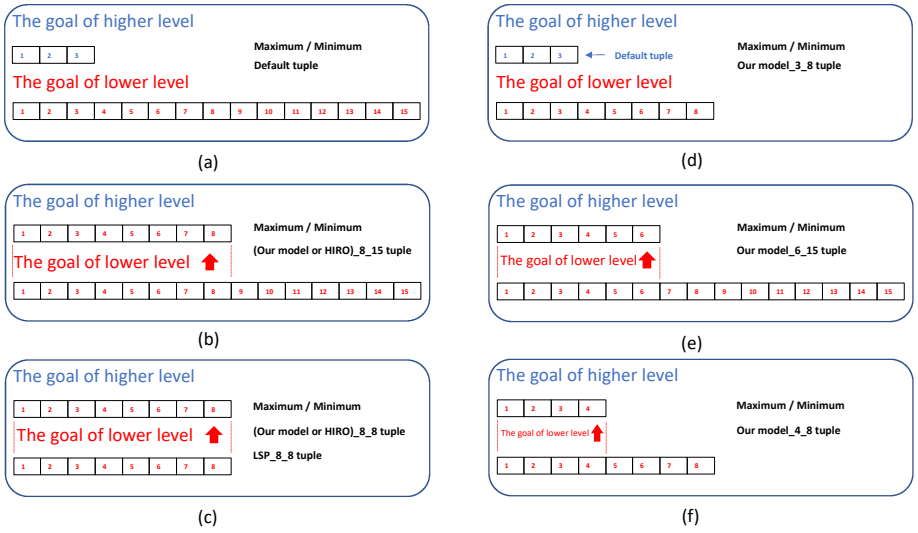


FIGURE 3.9: The example of how to create a non-optimal goal space of a higher-level policy and lower-level policy of our model from the default optimal goal space tuples of a higher-level policy and lower-level policy of HIRO. Each level's default tuple in (a) is represented with color, which is blue for higher-level policy or red for lower-level policy, and the number of tuple element's position. Only the tuple's elements of destination (higher-level's goal) are replaced by the corresponding tuple's elements of source (lower-level's goal). In the case of (Our model, HIRO, or LSP)_8_8 for the lower-level's goal, the last 7 tuple elements of the lower-level goal are removed. (a) The default tuple, which is optimal tuple, of a higher-level policy and lower-level policy of HIRO. Appendix 3.6.1 shows the definition of each level's tupe. (b) Our model_8_15 (c) Our model_8_8 (d) Our model_3_8. The goal space of the higher-level policy of 'Our model_2_8' and 'Our model_3_8' is constrained to be the same as the default goal space of Ant Push Single and Ant Fall Single of HIRO, respectively. (d) Our model_6_15 (e) Our model_4_8.

3.6 Appendices

Fig. 3.9 shows the example of how to create a non-optimal goal space for a higher-level policy and lower-level policy of our model from the default optimal goal space tuples of a higher-level policy and lower-level policy of HIRO.

In addition, the training parameters for our model and the key points for experiments are summarized as follows.

3.6.1 The training parameters for our model

The training parameters defined in our model for all tasks are the same as those of HIRO, as follows:

- Discount $\gamma = 0.99$ for both controllers.
- Adam optimizer; actor learning rate 0.0001; critic learning rate 0.001.
- Soft update targets $\tau = 0.005$ for both controllers.
- Replay buffer of size 200,000 for both controllers.
- Lower-level train step and target update performed every 1 environment step.
- Higher-level train step and target update performed every 10 environment steps.
- No gradient clipping.
- Reward scaling of 1.0 for lower-level; 0.1 for higher-level.
- Lower-level exploration is Gaussian noise with $\sigma = 1.0$.
- Higher-level exploration is Gaussian noise with $\sigma = 1.0$.
- state s_t : 30 dim.
- action $a_{t,z}$: 8 dim.
- Training step : 10M steps
- Higher-level goal space for ant_fall_(multi or single)
  - meta_context_range = ((-4, -4, 0), (12, 28, 5))
  - goal dim. in higher-level policy : 3 dim.
- Higher-level goal space for ant_push_(multi or single)
  - meta_context_range = ((-16, -4), (16, 20))
  - goal dim. in higher-level policy : 2 dim.
- lower-level goal space :
  - CONTEXT_RANGE_MIN = (-10, -10, -0.5, -1, -1, -1, -1, -0.5, -0.3, -0.5, -0.3, -0.5, -0.3, -0.5, -0.3)
  - CONTEXT_RANGE_MAX = (10, 10, 0.5, 1, 1, 1, 1, 0.5, 0.3, 0.5, 0.3, 0.5, 0.3, 0.5, 0.3)
  - goal dim. in lower-level policy : 15 dim.

3.6.2 The key points for experiments

The key points to note during the experiment are reiterated below:

- Although the dim. of a_z and a_{rnvp} are fixed depending on the environment and the input of FDGM, the dim. of $a_{t,z-\text{state}}$ is determined through a trade-off process involving experimentation to find the best dimension.
- The layer dim. of all policies of 'Our model_8_8' in Fig. 3.6 is the same as that of 'Our model_8_15'.
- In Fig. 3.7, the goal space of the higher-level policy of 'Our model_2_8' and 'Our model_3_8' is constrained to be the same as the original goal space of Ant Push Single and Ant Fall Single of HIRO, respectively.
- The layer dim. of all policies of our model in Fig. 3.7 is the same as that of our model in Fig. 3.6.
- The best size of $a_{t,z-\text{state}}$ for each experiment in Fig. 3.7 is chosen.
- As with Ant Fall Single in Fig. 3.6, our model in Ant Fall Single in Fig. 3.7 takes advantage of the combination of the inverse operation of Algorithm 2 and the size of the neural network of Algorithm 1.
- Even though the framework of LSP does not include off-policy correction, LSP in Fig. 3.7 benefits from off-policy correction to support a fair comparison in the experiment.

Chapter 4

An Autonomous Non-monolithic Agent with Multi-mode Exploration based on Options Framework

4.1 Introduction

Exploration is the crucial part of RL algorithms because it gives an agent the choice to uncover unknown states. There have been many RL exploration research studies with various viewpoints, such as intrinsic reward [86], [87], [88], [89], skill discovery [90], [91], [92], Memory base [93], [94], [95], [96], and Q-value base [97]. Although exploration research has evolved, it has concentrated on 'how to explore', which is how an agent selects an exploratory action. However, the exploration research regarding 'when to explore' has not been researched in earnest.

There are two types of methodology regarding 'when to explore', which are monolithic exploration and non-monolithic exploration. The noise-based monolithic exploration, a representative monolithic exploration, is that a noise, which is usually sampled from a random distribution, is added to the original action of a behaviour policy before putting the final action to an environment. The original action of policy and the noise to be added act as an exploitation and exploration respectively. Hence, the behaviour policy using

monolithic exploration is affiliated to a time-homogeneous behaviour policy. However, in a non-monolithic exploration, the original action of a behaviour policy is not added to a noise. They act for their own purpose at a separate step. Therefore, the behaviour policy using a non-monolithic exploration belongs to a heterogeneous mode-switching one (Fig. 4.1).

We have investigated the initial research [2] of non-monolithic exploration. As the tentative work, there are still several limitations. Firstly, there is only one exploration policy (we call it one-mode exploration). An agent can require more choice of entropy of exploration mode which denotes more exploration modes greater than one-mode exploration. Secondly, the period of exploration to be controlled should be not fixed but variable. Thirdly, the research takes advantage of a simple threshold hyper-parameter function, which is named 'homeostasis', for the variable scale of trigger signals for switching exploration or exploitation. However, there should be a natural switching mechanism by using the policy itself. It also claims other informed triggers, which are action-mismatch-based triggers and variance-based triggers.

In this thesis, we propose an autonomous non-monolithic agent with multi-mode exploration based on an options framework to resolve the above-mentioned considerations. Specifically, we adopt a Hierarchical Reinforcement Learning (HRL) as an options framework chaining together a sequence of exploration modes and exploitation in order to achieve switching behaviours at intra-episodic time scales. Thus, we can achieve a multi-mode exploration with the different entropy. In order to enable autonomous switching between exploration policies and an exploitation policy where the switching is based on intrinsic signals, we adapt a guided exploration using a reward modification of each switching mode. A robust optimal policy is also researched to maintain the potential performance.

Meanwhile, for this research the following 5 questions should be answered. How can an options framework be adopted in order to take advantage of the context of a HRL for exploration modes and exploitation? How does an agent have the flexibility of the exploration period? How does an agent get more entropy choice of exploration mode? How can an agent determine the switching of non-monolithic multi-mode exploration by itself without any subsidiary function such as 'homeostasis'? How does an agent avoid the inherent

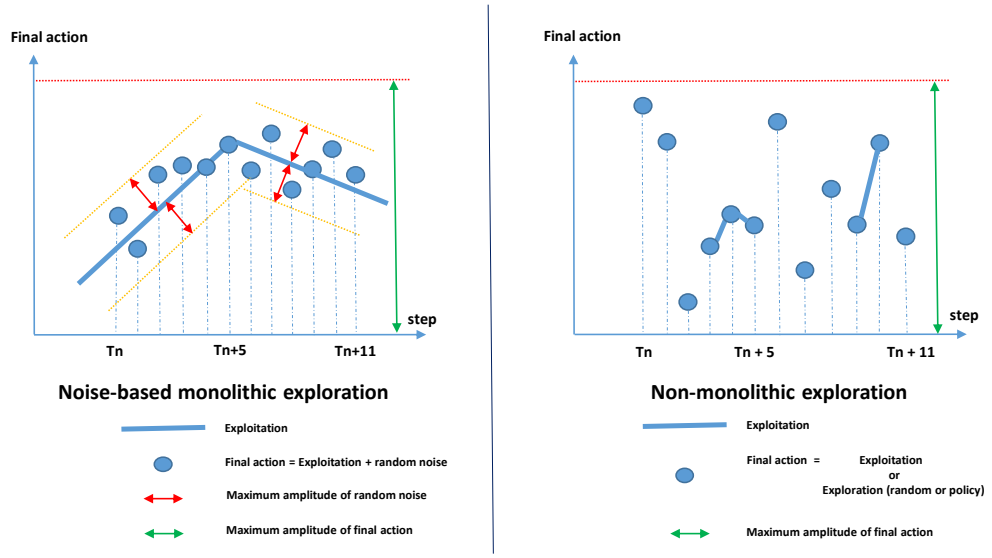


FIGURE 4.1: An example of noise-based monolithic exploration (left) and non-monolithic exploration (right). The final action, which is a scalar in this example for the well understanding explanation, denotes the action of an agent represented with a solid circle at each step. The solid line denotes the exploitation, an original action of a behaviour policy. The solid circle in the noise-based monolithic exploration is a final action which combines the original action of a behaviour policy and a sampled bounded noise at each step. However, the solid circle in the non-monolithic exploration is defined according to the mode of each step, i.e. exploitation which is an an original action of a behaviour policy or exploration which is a random noise or a policy.

disturbance of a policy but have a robust optimal policy?

It is worth mentioning that there are no similar works in the literature, so the reference methods are partially based on the method proposed in [2] even though their work is not based on an options framework. In the end, our exploration method shows a better performance.

The contributions are summarized as follows.

- *Development of an options framework model supporting an autonomous non-monolithic multi-mode exploration:* We introduce a novel HRL model architecture to support an autonomous non-monolithic multi-mode exploration for the first 3 research questions.
- *Development of a switching method for a non-monolithic exploration by using an inherent characteristic of a policy:* Our model use a guided exploration with a reward modification for the fourth research question.

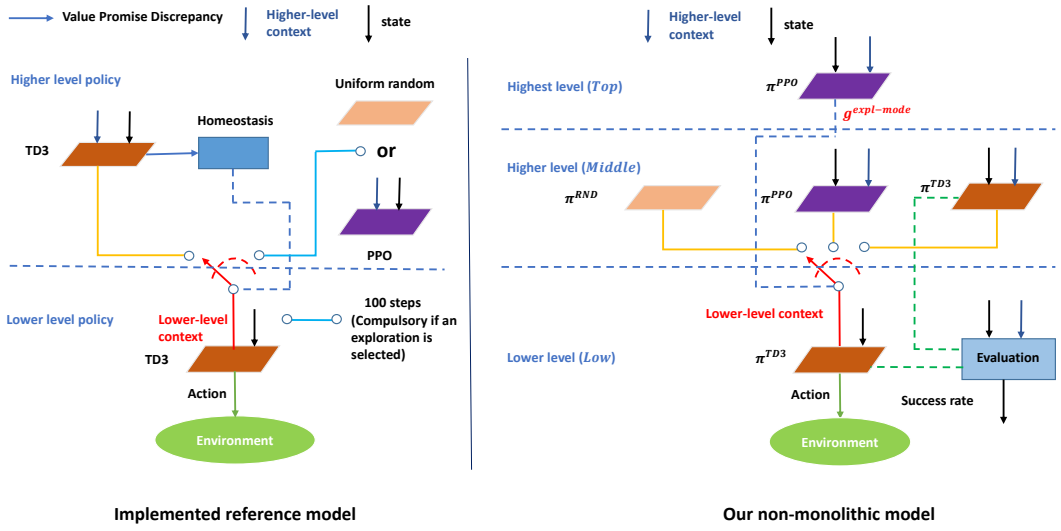


FIGURE 4.2: The architecture of our suggested model (right) compared with that of the reference paper using a homeostasis [2] (left).

- *Improved robustness of the policy:* A robust optimal policy can be ensured by taking advantage of an evaluation process for the last research question.

4.2 Our model

From the rising issue of value promise discrepancy used in [2], our research pays close attention to an autonomous multi-mode non-monolithic exploration model where an agent makes an action as to when an exploration mode starts and exits by itself. In addition, the expected model takes advantage of its inherent characteristic for the action. For the purpose, our research adopts an options framework.

An options framework especially in a goal-conditioned HRL is the appropriate consideration to control the multi-mode exploration through a fully state-dependent hierarchical policy. For the first research question proposed in Section 4.1, our model has three levels of HRL as shown in Fig. 4.2 together with the implemented model of [2]. Our model names each of the levels according to the height of the level: Top, $Middle$ and Low. The policies are in each level: π_T^{PPO} for Top, π_M^{TD3}, π_M^{PPO} and π_M^{RND} for $Middle$ and π_L^{TD3} for Low.

TABLE 4.1: Key Notations.

Symbol	Meaning
t	action step
$state$	current state
$next\_state$	next state
Top	The highest level
$Middle$	The higher level
Low	The lower level
$Action$	The action of *Low* level (The action of π_L^{TD3})
$target\_pos$	The context of *Top* and *Middle*
$goal$	The current lower-level context of three sub-policies of *Middle* level (The current goal for π_L^{TD3})
$next\_goal$	The next lower-level context of three sub-policies of *Middle* level (The next goal for π_L^{TD3})
R	The reward received from an environment (The sign of R is negative in Ant domain of OpenAI Gym)
π_T^{PPO}	The policy(on-policy) of *Top* level
π_M^{TD3}	The policy(off-policy) of *Middle* level
π_M^{PPO}	The policy(on-policy) of *Middle* level
π_M^{RND}	The policy (The uniform random for random policy) of *Middle* level
π_L^{TD3}	The policy(off-policy) of *Low* level
$g^{\text{expl-mode}}$	The action of π_T^{PPO} of *Top* level
$\alpha_{g^{\text{expl-mode}}}$	The preset value of α according to $g^{\text{expl-mode}}$
$S\_O_{g^{\text{expl-mode}}}$	The reference value of $Success\_ratio$ according to $g^{\text{expl-mode}}$
$loss$	The loss of π_T^{PPO} of *Top* level
S_E	The $Success\_rate$ of evaluation function of π^{TD3} of *Middle* level
$Done\_m$	The count of *Done* during the horizon of *Top* level
R_m	The sum of R during the horizon of *Top* level
$Count\_m$	The count during the horizon of *Top* level
S_O_m	The ratio of success count regarding $g^{\text{expl-mode}}$ of π_T^{PPO} during the horizon of *Top* level
ρ	The preset value of target rate, i.e. the average number of switches of the reference model

The hierarchical control process is easy to systematically construct a multi-mode exploration, $g^{\text{expl-mode}}$, as the option against a function control such as homeostasis. The exploration mode policy, π_T^{PPO}, can choose one of three policies of *Middle* level as follows

$$g^{\text{expl-mode}} \sim \pi_T^{PPO}. \tag{4.1}$$

Therefore, the value of $g^{\text{expl-mode}}$ denotes one of two exploration modes, which are uniform random and PPO, or one exploitation which is TD3. It also provides several control benefits for exploration as there are the inherent characteristics of the options framework.

In order to accomplish the purpose of our research, our options framework model comprises four elements: the inherent switching mode decision of the policy itself, empowering more entropy degrees for exploration, a guided exploration mode, and the use of an evaluation process for robustness.

4.2.1 The inherent switching mode decision of a policy itself

Since the inherent training method of π_T^{PPO} is used in our model, one policy of *Middle* level can be chosen according to an option, $g^{\text{expl-mode}}$, of π_T^{PPO}. π_T^{PPO} is synthesizing the reward-maximization of policy on all modes into its own policy without a subsidiary aid. This leads to the fact that the period of both exploration and exploitation is controlled by the inherent characteristic of an agent. In the end, all characteristic of the non-monolithic exploration mode policy can be integrated to the reward-maximization of policy for the second research question in Section 4.1. We can verify the choice of a switching mode on the count of each exploration mode as shown in Section 4.3.

4.2.2 Empowering more entropy choice for exploration

Our model pursues multi-mode exploration for the exploration mode policy according to the degree of entropy of exploration mode as the degree of optimism. Our model has two exploration modes, which are a π_M^{RND} and a π_M^{PPO}, and one exploitation policy, π_M^{TD3}, in *Middle* level for the third research question in Section 4.1. Thus, while an agent is being trained, we hypothesize that the degree of each entropy of three policies is as follows,

$$H\left(\pi_M^{RND}\right) > H\left(\pi_M^{PPO}\right) > H\left(\pi_M^{TD3}\right) \tag{4.2}$$

where $H\left(\pi_M^{\bullet}\right)$ denotes the overall entropy of a policy $\pi_M^{\bullet}$.

Our model just consumes PPO for an exploration mode so that it will be discarded at the end of training. Our model takes care of only off-policy, TD3, as a final target policy. Meanwhile, PPO and TD3 are trained together whenever a data occurs due to one of three sub-policies of *Middle* level. If PPO is trained to some degree, our model expects that the performance of PPO is higher than the performance of uniform random regarding the result of exploration.

4.2.3 Guided exploration

There are two phases of a potential reward progress during the training of our agent. Thus, our model takes a guided exploration into consideration for the agent in order to keep the first phase. Since our model pursues an options framework in a goal-conditioned HRL, the exploration mode policy can follow a reward-maximizing policy so that the modification of reward R from an environment is conducted with a preset parameter $\alpha_{g^{\text{expl-mode}}}$ as

$$R_{final} = R + \alpha_{g^{\text{expl-mode}}} * R \tag{4.3}$$

where R_{final} denotes a modified reward according to a preset parameter $\alpha_{g^{\text{expl-mode}}}$ and an environment reward R.

The value of $\alpha_{g^{\text{expl-mode}}}$ is differently or sometimes equally preset according to the type of $g^{\text{expl-mode}}$ as

$$\alpha_{\text{uniform random}} > \alpha_{\text{ppo}} > \textit{or equal to } \alpha_{\text{td3}} \tag{4.4}$$

where $\alpha_{\text{uniform random}}$, α_{ppo}, *and* α_{td3} denotes a preset parameter $\alpha_{g^{\text{expl-mode}}}$ of π_M^{RND}, π_M^{PPO} and π_M^{TD3} for *Middle* respectively.

Finally, since the value of R_{final} is utilized in the training of the exploration mode policy, a reward-maximized option for the fourth research question in Section 4.1 is preferred by the exploration mode policy depending on the value of $\alpha_{g^{\text{expl-mode}}}$. As the value of $\alpha_{g^{\text{expl-mode}}}$ gets bigger, the occurrence probability of its exploration mode gets smaller.

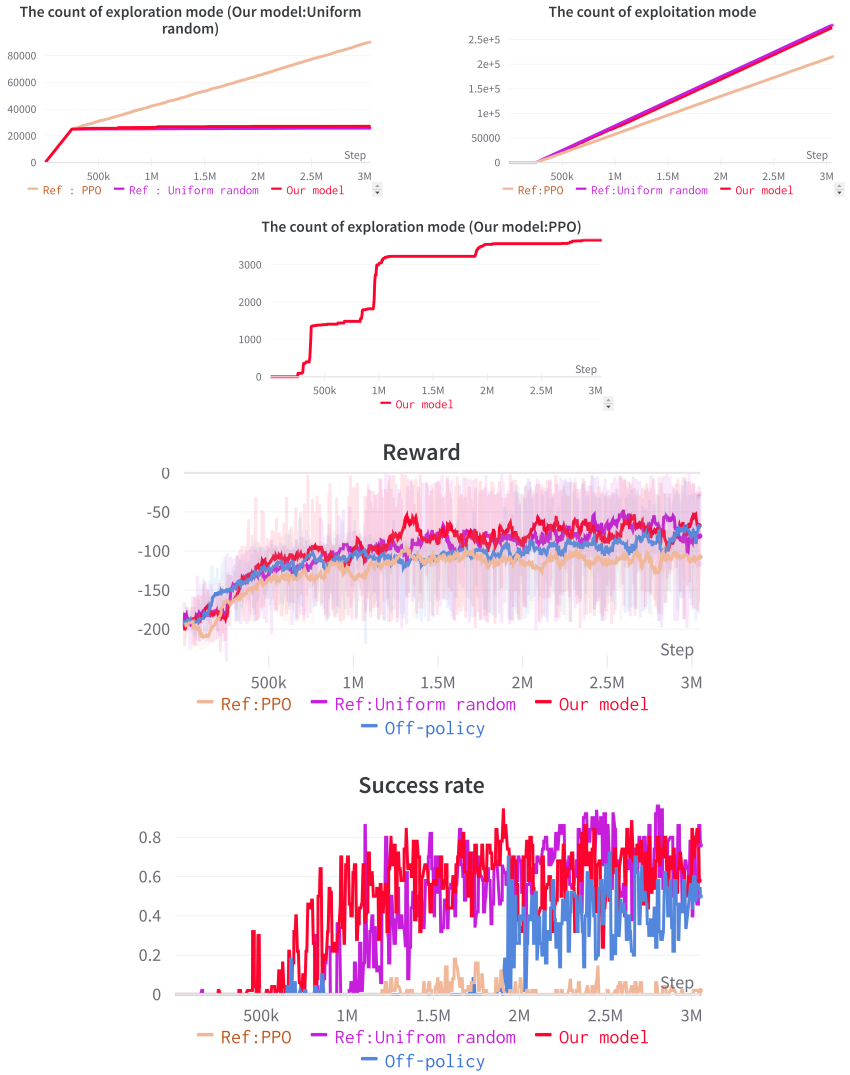


FIGURE 4.3: The count of exploration modes and exploitation and the reward and success rate of higher level policy for our model, Ref:Uniform random, Ref:PPO and HIRO in Ant Push.

4.2.4 Evaluation for robustness

For the second phase of the potential reward progress of our agent, our model adopts the online evaluation process to keep a robust optimal policy. The occurrence of success rate in the online evaluation process shows that the performance of an agent enters the second stage of reward progress in this research. From the second stage, our agent is required to have robust optimal policy by using online evaluation process. The online process evaluating the off-policy, π_M^{TD3}, operates every preset step. Then, it outputs the success

FIGURE 4.4: The count of exploration modes and exploitation and the reward and success rate of higher level policy for our model, Ref:Uniform random, Ref:PPO and HIRO in Ant Fall.

rate, S_E, according to the type of $g^{\text{expl-mode}}$. Thus, $loss_{final}$ of the exploration mode policy, π_T^{PPO}, for the fifth research question in Section 4.1 is calculated in this research as

$$loss_{final} = loss + S\_E * loss \tag{4.5}$$

where $loss_{final}$ is a modified loss according to S_E.

In our model, as the online value of S_E increases, the loss of the exploration mode policy in the mode of uniform random and online policy becomes bigger than its online original

loss.

4.3 Experiments

The control of multi-mode exploration of our model as an autonomous non-monolithic agent is shown by the count of exploration modes and exploitation, since their counts are critical for the analysis of our model. Each count describes the current situation of reward-maximization of policy on all modes. Through the analysis, we aim to answer the following crucial question. *Can our model show better performance than that of the representative model of the reference paper, [2], and a noise-based monolithic exploration policy?* We evaluate our model and them in two tasks, Ant Push and Ant Fall, of Ant domain of OpenAI Gym. The reference models for the comparison are two models of [2], XU-intra(100, informed, p^*, X) and XI-intra(100, informed, p^*, X), which are called 'Ref:Uniform random' and 'Ref:PPO' respectively in our reference model[1]. PPO is utilized for the intrinsic explore mode of our reference model. A noise-based monolithic exploration policy is HIRO, which is composed of TD3 at each level. Our model and two reference models are also implemented based on HIRO. In order to evaluate the best performance among three models, we have the four analysis items as follows:

1. How many counts are assigned to each policy through whole training steps?
2. How does the transition of our model between exploration mode and exploitation occur compared with the forced exploration transition of reference model?
3. How much is the difference between uniform random and on-policy as the exploration policy of our model based on a guided exploration strategy?
4. How much does the evaluation process influence the performance of the second reward phase?

[1] Please read the section 3.1 of [2] for the experimental details

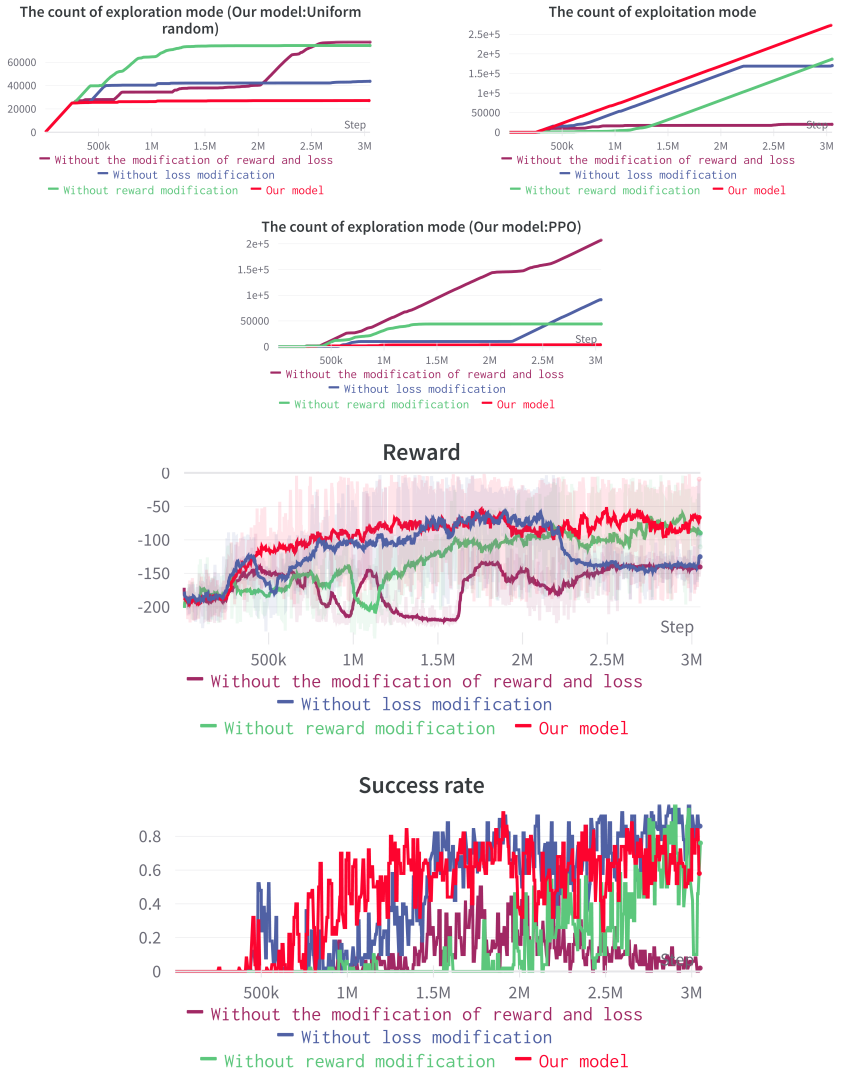


FIGURE 4.5: Three types of ablation study against our normal model in Ant Push.

The results of Ant Push and Ant Fall are represented in Fig. 4.3 and Fig. 4.4 respectively. Moreover, our source and implementation details are available online[2]. Algorithm 3 shows the main part of algorithm which is implemented on the reference code.

[2] https://github.com/jangikim2/An-Autonomous-Non-monolithic-Agent-with-Multi-mode-Exploration-based-on-Options-Framework

4.3.1 Comparison with the reference paper and pure off-policy

4.3.1.1 Ant Push

Our model outperforms all other models through almost all training steps. The exploitation of our model and two reference models occurs during the most of the training steps. The exploration mode of our model and two reference models takes place less than the exploitation of them. HIRO shows the best performance in the early period but quickly loses the potential through whole training steps as the other models takes advantage of the diverse exploration modes. The performance of 'Ref:Uniform random' is better than that of 'Ref:PPO'.

The exploration mode of Uniform random of our model and 'Ref:Uniform random' does not take place for a long time, but for a short time and gradually. Meanwhile, more exploration of 'Ref:PPO' occurs than that of 'Ref:Uniform random' according to a preset target rate ρ where the incessant exploration occurs after the starting mode.

After the starting mode in Algorithm 3, the PPO exploration mode of our model has about 3600 steps, which is more than the Uniform random exploration mode of our model, which is about 2100 steps. Most of the PPO exploration mode of our model occurs before 1M steps. The comparison of the total steps of the two exploration modes and exploitation of our model is

$$Total\_Step\left(\pi_M^{TD3}\right) >> Total\_Step\left(\pi_M^{PPO}\right) > Total\_Step\left(\pi_M^{RND}\right) \tag{4.6}$$

where $Total\_Step\left(\pi_M^{\bullet}\right)$ denotes total conducted steps of a policy of $\pi_M^{\bullet}$.

The guided exploration strategy produces the exploration mode of our model based on the modification of reward, Equation (4.3), and the ratio of success count S_O_m.

The second phase in Ant PUSH task starts from the steps when the reward occurs above -100 since the success rate, S_E, of the evaluation process occurs from about 500K and passes 0.6 at 1M steps. The situation of collapsed reward does not take place for a long time because of the loss modification, Equation (4.5), relying on the evaluation process.

4.3.1.2 Ant Fall

Our model shows a competitive performance against all other models after 3M steps. The preset of reward modification of Ant Fall is different from that of Ant Push, which means that $\alpha_{\text{on-policy}}$ is equal to $\alpha_{\text{off-policy}}$. The exploitation of our model occurs over 1M steps less than that of the two reference models, which is different from the situation of Ant Push. The performance of HIRO is stationary and decrease in the latter part. Unlike Ant Push, the performance of 'Ref:PPO' is better than that of 'Ref:Uniform random'.

The exploration mode of 'Ref:Uniform random' and our model takes place for longer than that of 'Ref:Uniform random' and our model in Ant Push. Meanwhile, the exploration mode of 'Ref:PPO' and that of 'Ref:Uniform random' are almost the same since a preset target rate ρ for 'Ref:Uniform random' and 'Ref:Uniform random' is the same.

Although $\alpha_{\text{on-policy}}$ is equal to $\alpha_{\text{off-policy}}$, since the ratio of success count regarding each action of the second level is also modified, after the starting mode, the total step comparison of two exploration mode and exploitation of our model is through whole training steps as

$$Total\_Step\left(\pi_M^{TD3}\right) > Total\_Step\left(\pi_M^{PPO}\right) >> Total\_Step\left(\pi_M^{RND}\right). \tag{4.7}$$

Unlike the Ant PUSH task, the second phase in the Ant Fall task suffers a drop of reward between 4M and 4.5 M steps. The success rate of the evaluation process stays between 0.5 and 0.6 during the period. The recovery of reward quickly takes place due to the success rate compared with 4.3.2.2.

4.3.2 Ablation study

We investigate our model without the reward modification, the loss modification and both modifications. Fig. 4.5 shows the results of the experiment compared with our normal model in the Ant Push task. Therefore, the part related to our normal model in Ant Push is removed for the purpose of experimenting each case.

4.3.2.1 Without the reward modification

While the exploitation has less steps than our normal model, the exploration of Uniform random and PPO has more steps than our normal model. The performance of reward and success rate slowly increase.

4.3.2.2 Without the loss modification

Again, the two exploration modes have more steps than our normal model and the exploitation has less steps than our normal model. It shows a drop of reward between 2.2M steps and 3M steps due to the increase in PPO exploration. Although the success rate is better than that of our normal model during the period, its performance is worse than that of our normal model.

4.3.2.3 Without both the reward modification and the loss modification

Too many explorations and less exploitation cause the worst performance.

4.4 Discussion

4.4.1 The effect of on-policy for exploration

When the on-policy operates in the beginning of exploration, the performance of on-policy, π_M^{PPO}, is not competitive. However, after it is trained by itself or other policies to some extent, the performance of on-policy shows better performance than the random policy. Meanwhile, In practice π_T^{PPO} is likely not to suffer a local minima due to the three policies of *Middle* level.

4.4.2 The effect of reward modification

In Ant Fall task, the performance of our model in the early steps up to about 2M steps lags behind all other models. The reason is that the on-policy operates for long time up to then

since $\alpha_{\text{on-policy}}$ is equal to $\alpha_{\text{off-policy}}$. The reward modification for the guided exploration takes advantage of the fixed value of $\alpha_{g^{\text{expl-mode}}}$, which is not an adaptive strategy.

4.4.3 The effect of loss modification

The occurrence of on-policy and random policy in the Ant Fall task between 4M and 4.5M steps gives rise to a drop in performance of the agent. In particular, the modeling of uncertainty reflecting the success rate, S_E, can be considered. The higher S_E is, the lower the uncertainty is. Thus, S_E is related to the uncertainty.

Algorithm 3 Multi-exploration mode based on options framework

Initialize:
  Set the value of $\alpha_{g^{expl\text{-}mode}}$ *according to* $g^{expl\text{-}mode}$ *of* π_T^{PPO}
  Set the value of $S\_O_{g^{expl\text{-}mode}}$ *according to* $g^{expl\text{-}mode}$ *of* π_T^{PPO}
procedure $EVALUATE\_\pi_M^{TD3}(..., \pi_M^{TD3}, \pi_L^{TD3}, ...)$
  According to π_M^{TD3} *and* π_L^{TD3}, compute S_E
end procedure
procedure $TRAIN_T(..., S\_E, g^{\text{expl-mode}}, ...)$
  if $g^{\text{expl-mode}}$ *is Random uniform or PPO* **then**
    $loss_{final} = loss + S\_E * loss$
  else
    $loss_{final} = loss - S\_E * loss$
  end if
end procedure
for $t = 0, \ldots, T-1$ **do**
  $action \leftarrow Clamp\_Max(\pi_L^{TD3}(state, goal) + Noise)$
  Execute action, observe R *and* $next\_state$
  $Done \leftarrow Judge\_success(state, target\_pos)$
  Increase $Done\_m$ *by 1 if Done is True*
  Increase R_m *by* R
  Increase $Count\_m$ *by 1*
  if *the horizon of Top level* **then**
    $S\_O\_m \leftarrow Done\_m \,/\, Count\_m$;
    $R_{final} \leftarrow R\_m + \alpha_{g^{\text{expl-mode}}} * R\_m$
    if $S\_O\_m >= S\_O_{g^{\text{expl-mode}}}$ **then**
      $Done\_m \leftarrow True$
    end if
    if *the training time of Top level* **then**
      $Train_T(..., S\_E, g^{\text{expl-mode}}, ...)$
    end if
    if *the starting mode* **then**
      $g^{\text{expl-mode}} \leftarrow$ *Random policy*
    else
      $g^{\text{expl-mode}} \sim \pi_T^{PPO}$
    end if
    $R\_m, Count\_m, Done\_m \leftarrow 0$
  end if
  if *the horizon of Middle level* **then**
    By $g^{expl\text{-}mode}$, $next\_goal \sim \pi_M^{RND}, \pi_M^{PPO}$ *or* π_M^{TD3}
    if *the training step of* π_M^{PPO} **then**
      $Train\_\pi_M^{PPO}(...)$
    end if
  end if
  $state \leftarrow next\_state$; $goal \leftarrow next\_goal$
  if *the training step of* π_M^{TD3} **then**
    $Train\_\pi_M^{TD3}(...)$
  end if
  if *the evaluation step of Middle level* **then**
    $Evaluate\_\pi_M^{TD3}(..., \pi_M^{TD3}, \pi_M^{TD3}, ...)$
  end if
end for

Chapter 5

Decoupling Exploration and Exploitation for Unsupervised Pre-training with Successor Features

5.1 Introduction

Unsupervised pre-training leverages previously acquired behaviour without requiring an extrinsic reward, enabling an agent to rapidly adapt to new downstream tasks during fine-tuning. It has contributed significantly to Computer Vision (CV) [98] and Natural Language Processing (NLP) [99], overcoming the weak point of supervised reinforcement learning which is limited to a single task.

As skill-based unsupervised reinforcement learning (RL) has been researched, one of current mainstream unsupervised pre-training RLs involves an agent maximizing the mutual information between an encoded state and a policy skill to learn an explicit skill vector. This approach takes advantage of algorithms such as VIC [100], DIAYN [53], and VALOR [101]. However, during fine-tuning, these algorithms may suffer from issues related to inferior generalization and sluggish task inference due to the inefficient interpolation relying on latent behaviour spaces learnt during unsupervised pre-training.

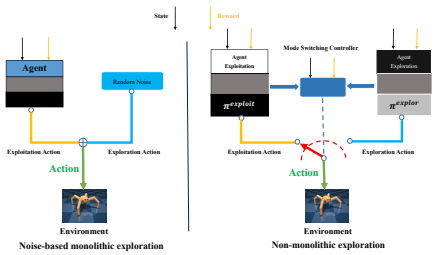


FIGURE 5.1: An example of noise-based monolithic exploration (left) and non-monolithic exploration (right). In the noise-based monolithic exploration, noise and the agent's action play roles in exploration and exploitation, respectively. The final action, 'Action', taken by the agent in the environment is the result of adding the action with a noise. In the non-monolithic exploration, the exploitation agent and the exploration agent work for their own purposes with the help of a mode-switching controller. The mode-switching controller considers the state of one or both agents.

To address these issues, research on the value function using Successor Features (SFs) has been conducted to decouple the dynamics of environment from the reward [52, 102]. Using SFs for value function approximation makes it possible to relearn only the applicable component in case of changes to dynamics or reward. VISR [55] takes advantage of SFs focusing on 'task inference' for fine-tuning after it is pre-trained. Although it has strengths against VIC and DIAYN in terms of certain issues, it also has the weakness of inefficient exploration.

APS [4], one of the competence-based algorithms mentioned in [103], utilizes combined intrinsic reward with rewards of VISR and Active Pre-Training (APT) [104], allowing 'task inference' and 'exploration', respectively. Even though the mutual information exploration method of APS is considered a recent leading method, its performance is not sufficient since its combined intrinsic reward interferes with the original intention of the respective intrinsic reward from VISR or APT. In addtion, the competence-based approaches such as APS have a chronic issue supporting small skill spaces due to the requirement of a precise discriminator [103]. The linear reward-regression problem mentioned in [55] gets worse since these interference can ultimately cause damage to the role of classifier or discriminator for fine-tuning the pre-trained RL agent.

In our research, we propose a novel unsupervised pre-training model for the agent based on SFs ensuring the decomposition of 'task inference' and 'exploration' to improve performance. In the methodology, the agent that conducts 'task inference' should primarily focus on the 'exploitation' without 'exploration'. To avoid a single agent that conducts

both 'exploitation' and 'exploration', there should be discrete agents responsible for each task.

Since a RL agent built on noise-based monolithic exploration typically is not suitable for our research, we focus on separating 'exploitation' and 'exploration' using a non-monolithic exploration methodology, as shown in Fig.5.1. There should be several answers to build a decoupled exploration methodology that allows an agent using SFs to switch between two modes, exploitation and exploration, during unsupervised pre-training. The questions that arise are: How can such a methodology be conducted for unsupervised pre-training of SFs to switch an exploration mode? How can separate agents be trained during the pre-training of the decoupled exploration methodology? Is it possible for an agent using SFs to overcome a chronic issue supporting small skill spaces resulting from the requirement of a precise discriminator? The contributions of our research are three-folded:

- The development of a decoupled exploration methodology based on non-monolithic exploration methodology to improve the performance of a monolithic agent using successor features during unsupervised pre-training.
- An optimized training method of separate agents during the pre-training phase of decoupled exploration methodology for the agent using successor features.
- The ability taking advantage of a competence-based algorithm as a separate exploration agent for greater flexibility and generalization of discriminator of exploitation agent.

TABLE 5.1: Key Notations.

Symbol	Meaning
t	action step
s	current state
s'	next state
a	The final action to an environment
$a_t^{exploit}$	The action of $\pi^{exploit}$
a_t^{explor}	The action of π^{explor}
$\mathbf{z}$	Encoded observation (or state)
$\mathbf{w}$	Task vector
$r^{exploitation}$ *or* $r^{exploit}$	Intrinsic reward of APS exploitation
$r^{exploration}$ *or* r^{explor}	Intrinsic reward of APS exploration
$Exploit$	The agent of exploitation
$Explor$	The agent of exploration
$\pi^{exploit}$	The policy of Exploit
π^{explor}	The policy of Explor
$\phi^{exploit}$	The feature of Exploit
ϕ^{explor}	The feature of Explor
Q^{exploit}	The state-action value function of Exploit
$Action$	The action, chosen by Homeo, to an environment
$D_{promise}(t-k,t)$	The value promise discrepancy
$D_{promise}^{explot}$	The value promise discrepancy of $\pi^{exploit}$
f_{homeo}	The function of homeostasis of the policy of Exploit, $\pi^{exploit}$
ρ	The preset value of target rate, i.e. the average number of switches of the reference model, which is selected among (0.1; 0.01; 0.001; 0.0001) for pretraining
$D_R^{\#}$	The data of a common replay buffer and its encoded state through the state encoder of # which is Exploit or Explor

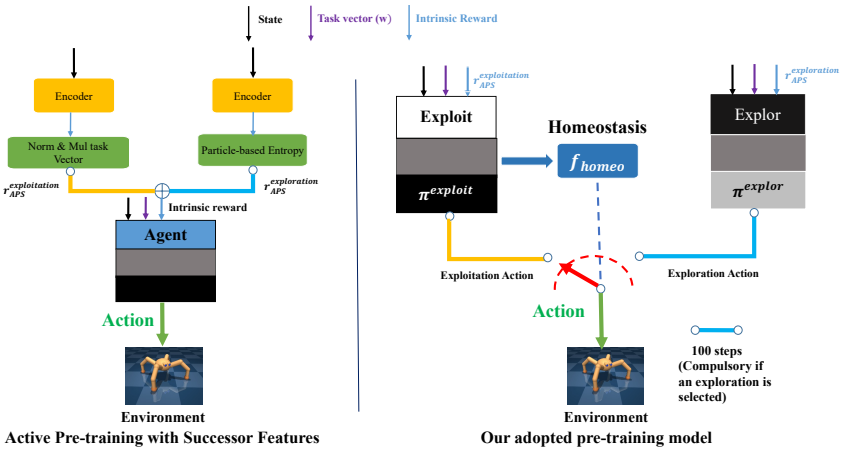


FIGURE 5.2: The architecture of our proposed pre-training model, NMPS, (right) compared with Active Pre-training with Successor Features (left).

5.2 PRELIMINARY KNOWLEDGE

5.2.1 Successor features

[105] further extends the successor representation of [106] to a continuous space as

$$\begin{aligned} r(s, a_t, s') &= \phi(s, a, s')^T \mathbf{w} \\ Q^{\pi}(s, a) &= E^{\pi}\left[\sum_{i=t}^{\infty} \gamma^{i-t} \phi_{i+1} \mid S_t = s,\ A_t = a\right]^T \mathbf{w} \\ &= \psi^{\pi}(s, a)^T \mathbf{w} \end{aligned} \tag{5.1}$$

where $\mathbf{w}$ is a task vector, $r(s, a_t, s')$ is a reward function, s is a state and $\phi_t = \phi(S_t, A_t, S_t')$ and $\psi^{\pi}(s, a)$ are a successor feature of policy π. Generalized Policy Improvement (GPI) based on SFs has also been claimed to have a transfer reinforcement learning framework [107, 108].

5.2.2 Non-monolithic exploration

As described in [109, 110], [111] claims the importance of 'when' to explore, which is a non-monolithic exploration approach, as opposed to 'how' to explore which is typically associated with monolithic exploration. As its switching method, it adopts 'Homeostasis' [112], which is abbreviated as 'Homeo', using the difference of value function between k

TABLE 5.2: Comparing all variants of NMPS based on 4 factors that are 'Reward', '$D_R^{\#}$', 'The train of feature or skill ('T. f or s' marked in the table)' and 'Action' on the model notation of $NMPS\_\{X\ or\ D\}\_\{sep,\ exploit\ or\ explor\}\_\{e,\ x\ or\ *\}$.

	Reward		$D_R^{\#}$		T. f or s		
The variants of NMPS	Exploit	Explor	Exploit	Explor	Et	Er	Action
$NMPS\_X\_sep^{ex}$		r^{explor}	$D_R^{exploit}$	D_R^{explor}	✓	✓	Homeo
$NMPS\_X\_sep^{e*}$		r^{explor}	$D_R^{exploit}$	D_R^{explor}	✓	x	Homeo
$NMPS\_X\_exploit^{ex}$		r^{explor}	$D_R^{exploit}$	common	✓	✓	Homeo
$NMPS\_X\_exploit^{e*}$		r^{explor}	$D_R^{exploit}$	common	✓	x	Homeo
$NMPS\_X\_explor^{ex}$		r^{explor}	common	D_R^{explor}	✓	✓	Homeo
$NMPS\_X\_explor^{e*}$	$r^{exploit}$	r^{explor}	common	D_R^{explor}	✓	x	Homeo
$NMPS\_D\_sep^{ex}$		DIAYN	$D_R^{exploit}$	D_R^{DIAYN}	✓	✓	Homeo
$NMPS\_D\_sep^{e*}$		DIAYN	$D_R^{exploit}$	D_R^{DIAYN}	✓	x	Homeo
$NMPS\_D\_sep^{ex\_D}$		DIAYN	$D_R^{exploit}$	D_R^{DIAYN}	✓	✓	DIAYN
$NMPS\_D\_sep^{e*\_D}$		DIAYN	$D_R^{exploit}$	D_R^{DIAYN}	✓	x	DIAYN
$NMPS\_D\_sep^{ex\_D\_A10}$		DIAYN	$D_R^{exploit}$	D_R^{DIAYN}	✓	✓	DIAYN
$NMPS\_D\_sep^{e*\_D\_A10}$		DIAYN	$D_R^{exploit}$	D_R^{DIAYN}	✓	x	DIAYN

steps which is called the 'value promise discrepancy' as

$$D_{promise}(t-k,t) := \left|V(s_{t-k}) - \sum_{i=0}^{k-1} \gamma^i R_{t-i} - \gamma^k V(s_t)\right| \tag{5.2}$$

where $V(s)$ denotes the agent's value estimate at state s, R denotes the reward and γ denotes a discount factor.

5.3 Our methodology

An agent using SFs computes $\psi^{\pi}(s,a)$ during unsupervised pre-training. Then, during supervised fine-tuning, the agent can find a $\mathbf{w} \in R^d$ by solving a regression problem through $r(s,a_t,s')$ of Eq. (5.1). Afterward, it computes based on $Q^{\pi}(s,a)$ of Eq. (5.1). However, APS cannot exactly solve a regression problem during supervised fine-tuning with $\psi^{\pi}(s,a)$ computed based on $r_{APS}(s,a,s')$ of Eq. (2.12) during unsupervised pre-training.

The discrete agents using SFs, which are Exploit and Explor, are required to divide the united intrinsic reward of APS, which is Eq. (2.12), to overcome an implicit issue that

violates the linear reward-regression problem of Eq. (5.1) during task-specific fine-tuning. After pre-training, Exploit is used for fine-tuning and Explor is discarded.

Since our model utilizes discrete agents, we explore various model variants to enhance its performance across different domains. Table 5.2 shows all evaluated models, which are composed of 4 independent factors, to find the optimal model in each domain. There are 4 groups: $NMPS\_X\_sep$, $NMPS\_X\_exploit$, $NMPS\_X\_explor$, $NMPS\_D\_sep$ in NMPS model. Among them, the standard type is $NMPS\_X\_sep^{ex}$.

- In the 'Exploit' of 'Reward' column, the intrinsic rewards of exploitation agent which is abbreviated as 'Exploit' are $r_{APS}^{explitation}$ of APS. In the 'Explor' of 'Reward' column, the intrinsic rewards of exploration agent which is abbreviated as 'Explor' are $r_{APS}^{exploration}$ of APS or the reward of DIAYN with X and D of model notation, respectively. DIAYN in 'Reward' also means that Explor is DIAYN.

- 'common' in '$D_R^{\#}$' means that the active $D_R^{\#}$ is used for both agents in common. Furthermore, 'sep(erate)', 'exploit' and 'explor' of model notation means the use of $D_R^{\#}$ of both agents, only $D_R^{exploit}$ of exploitation agent and only D_R^{explor} of exploration, respectively.

- '✓' and 'x' in 'T. f or s' means that the feature or skill of Explor is trained or not. 'e', 'x' or '*' of model notation means that the feature of Exploit or the feature or skill of Explor is trained or not.

- 'Homeo' in 'Action' means that the final action is determined in Exploit or Explor depending on the decision of Homeo. 'DIAYN' in 'Action' with 'D' of notation means that the only exploration agent of $NMPS\_D\_sep^{ex}$, DIAYN, decides Action. $NMPS\_D\_sep^{\#\#\_D}$ denotes that the default dimensions of feature of Exploit and skill of Explor are 10 and 16, respectively. 'A10' in $NMPS\_D\_sep^{\#\#\_D\_A10}$ denotes that all dimensions of feature of Exploit and skill of Explor are 10.

The details of four independent factors are explained in the following section.

5.3.1 The model with a non-monolithic exploration

We adopt the non-monolithic model used in [111] as a reference method inspiring our model, as shown in Fig.5.2 for the decoupled exploration methodology that our research seeks. In our adopted pre-training model, the decision of Action is decided by Homeo using the value promise discrepancy of Exploit, based on Q^{exploit}, as an input:

$$
\begin{aligned}
D^{\text{exploit}}_{\text{promise}} &= D^{\text{exploit}}_{\text{promise}}(t-k,t) \\
Action &= \begin{cases} a_t^{exploit} \sim \pi^{exploit}, & \text{for } f_{homeo}(D^{exploit}_{promise}) = 0 \\ a_t^{explor} \sim \pi^{explor}, & \text{for } f_{homeo}(D^{exploit}_{promise}) \neq 0 \end{cases}
\end{aligned} \tag{5.3}
$$

where $f_{homeo}(D_{promise})$ with the input of $D_{promise}(t-k,t)^{exploit}$ is Homeo for the decision of exploration mode switching in which its result value are based on Bernoulli distribution and $D_{promise}(t-k,t)^{exploit}$ is the value promise discrepancy of $\pi^{exploit}$.

Then, we expect that the entropy of an Explor agent using π^{explor} is greater than that of an Exploit agent using $\pi^{exploit}$ as follows:

$$
H\Big(\pi^{explor}\Big) > H\Big(\pi^{exploit}\Big) \tag{5.4}
$$

where $H\Big(\cdot\Big)$ denotes the overall entropy of a policy. Since Exploit using SFs is used during fine-tuning and Explor takes a supportive role for the exploration of Exploit, π^{explor} with $r_{APS}^{exploration}(s,a,s')$ in Eq. 2.12 is discarded when a pre-training is finished. Only $\pi^{exploit}$ with $r_{APS}^{exploitation}(s,a,s')$ in Eq. 2.12 is used for fine-tuning.

5.3.2 The optimized training method of separate agents

In this research, our implementation utilizes an off-policy, DDPG, used as a base algorithm of [103] for all agents. A data from $D_R^{\#}$ of either a single agent or both agents can be used to train both agents and determine whether they will synchronize as follows:

$$
(..., \mathbf{w}, state, ...) \sim D_R^{exploit} \text{ or/and } D_R^{explor} \tag{5.5}
$$

where (..., $\mathbf{w}$, state, ...) is the data loaded from $D_R^{\#}$ of both or one of two agents according to the case of 'and' or 'or' to be initialized for pre-training and $\mathbf{w}$ is a task vector. In the case of 'and' of Eq. (5.5), each agent is separately trained with its own $D_R^{\#}$. Both agents regarding 'or' of Eq. (5.5) are trained with one of $D_R^{\#}$ in common (see Table 5.2 for more details).

In addition, to maximize the effect of exploration for the model, the other variance of Explor using a feature or a skill can be considered (see 'T. f or s' of Table 5.2 for more details). If a feature or skill of Explor is not trained, the feature or skill of Explor with its initial value is used for the agent during pre-training. In the case of feature, the following result of action-value function, Q, can be expected:

$$Q^{Explor}_{\phi^{Explor}_{Trained}} \neq Q^{Explor}_{\phi^{Explor}_{NonTrained}} \tag{5.6}$$

where $\phi^{Explor}_{Trained}$ and $\phi^{Explor}_{NonTrained}$ are the trained and the nontrained feature of Explor, respectively. This relies on the fact that Explor will be discarded when the pre-training is finished.

5.3.3 The greater flexibility and generalization of discriminator

We consider that since Eq. (5.1) shows that a successor feature and a task vector are a pair to produce a reward, the combination of two factors is crucial to synchronize the two agents. In this research, the decomposition of 'task inference' and 'exploration' extends to the combination of $\pi^{exploit}$ using SFs and π^{explor} of the competence-based algorithms for the purpose of improving exploration performance. The competence-based algorithm maximizes the mutual information $I(\mathbf{z}; \mathbf{w})$ between the encoded observation $\mathbf{z}$ and skill $\mathbf{w}$ to learn an explicit skill vector as follows:

$$I(\mathbf{z}; \mathbf{w}) = H(\mathbf{z}) - H(\mathbf{z}; \mathbf{w}) = H(\mathbf{w}) - H(\mathbf{w}; \mathbf{z}). \tag{5.7}$$

Each competence-based algorithm has a different performance in an environment since a competence-based algorithm has a different algorithm based on a different decomposition of the mutual information mentioned above. As a variant of our model, we leverage a

Algorithm 4 The main algorithm of NMPS

Initialize:
  Set the value of the duration step of exploration mode
  Set the value of ρ in (0.1; 0.01; 0.001; 0.0001) for Homeo
  Set whether the feature or skill of Explor is or not for *Train_Both_Agents*
procedure EVALUATE_$\pi^{exploit}$($D_{promise}$ of $\pi^{exploit}$)
  Compute the output of Homeo with $D_{promise}$ of $\pi^{exploit}$
end procedure
procedure TRAIN_BOTH_AGENTS(..., **w**, *state*, ...)
  Train the feature of $\pi^{exploit}$
  Train $\pi^{exploit}$ with $r_{APS}^{exploitation}(s, a, s')$
  Train the feature or skill of π^{explor} according to the initialization procedure
  Train π^{explor} with $r_{APS}^{exploration}(s, a, s')$
end procedure
for $t = 0, \ldots, T-1$ **do**
  if *starting mode* **then**
    The output of Homeo $\leftarrow$ exploration mode
  else
    Compute $D_{promise}$ of $\pi^{exploit}$
    $Evaluate\_\pi^{exploit}(D_{promise}$ of $\pi^{exploit})$
  end if
  w $\sim$ exploitation agent
  if *output of Homeo == exploitation mode* **then**
    action $\leftarrow \pi^{exploit}$
  else
    action $\leftarrow \pi^{explor}$
  end if
  if *training step of* $\pi^{exploit}$ *and* π^{explor} **then**
    $Train\_Both\_Agents((..., \mathbf{w}, state, ...)$
    $\sim D_R^{exploit}$ or/and $D_R^{explor})$
  end if
  Execute action, observe reward (for evaluation) and next_state
end for

competence-based algorithm for exploration. This approach enhances the flexibility and generalization of the Exploit's discriminator, allowing it to overcome the limitations associated with supporting small skill spaces (see $NMPS\_D$ of Table 5.2). In the model, the reward of Explor is the reward of DIAYN, which is $NMPS\_D\_sep^{\#\#}$. Action can also be the

action of DIAYN instead of the decision of Homeo, which is $NMPS\_D\_sep^{\#\#\_D\_\#\#\#}$. We also expect the following entropy comparison for each agent of NMPS:

$$H\Big(\pi^{explor}_{competence-based\ algorithm}\Big) > H\Big(\pi^{exploit}\Big). \tag{5.8}$$

5.4 Experiments

The open source code[1] for pre-training and fine-tuning of a competence-based approach evaluated in this research is used for the comparison, which follows a state-based agent. Our code is available on the following github[2]. The hyper-parameters and Homeo used in this experiment are explained in 'Table I' and 'Algorithm I' of Appendix in 'supplementary document.pdf' on the above github. The evaluation of this research has been conducted in three domains: Walker, Quadruped and Jaco Arm. Three domains are based on the DeepMind Control Suite (DMC), which is a collection of continuous control tasks.

The discrete agents of $NMPS\_X$ of our model are based on APS implemented in open source code. Both Exploit and Explor are implemented with $\log q(w|s)$ and $r_t^{APT}(s)$ of Eq. 2.12, respectively. That is to say, in $NMPS\_X$, each agent has a separate APS intrinsic reward. Meanwhile, in $NMPS\_D$, DIAYN in competence-based algorithm is used as an Explor instead of Explor of $NMPS\_X$.

NMPS is implemented on the source code with XI-intra(100, informed, p^*, X)[3] where 'XI' means an intrinsic reward, '100' is an explore duration, 'informed' is a trigger type, p^* is the exploit duration parameterized by a target rate ρ and 'X' is the starting mode of explore. Instead of RND used in [111], this research takes advantage of DDPG since all reference algorithms are based on DDPG in the reference code[1]. Therefore, the design of NMPS in this research hinges on off-policy.

[1] https://github.com/rll-research/url_benchmark
[2] https://github.com/jangikim2/NMPS
[3] The section 3.1 of [111]

Since NMPS is applied to only pre-training, the original code of fine-tuning in the reference code is utilized for the fine-tuning without a modification. After fine-tuning, Exploit of NMPS is compared with APS, DIAYN and SMM [113]. On the other hand, Explor of NMPS is discarded after pre-training.

Algorithm 4 shows the explanation of main algorithm of our non-monolithic exploration model during the pre-training phase. During the pre-training phase of $NMPS\_X\_exploit$ or $NMPS\_X\_explor$, the two agents, Exploit and Explor, use a shared replay buffer that is generated by an active agent among the discrete agents. Meanwhile, for pre-training of $NMPS\_D$, separate replay buffers are used for each agent. $NMPS\_D\_sep^{\#\#\_D\_\#\#\#}$ does not need Homeo since only Explor is involved in Action.

The most important tuned parameter in our source code is the target rate, ρ, for Homeo. The best result of each NMPS among 4 values of target rate ρ, which are (0.1; 0.01; 0.001; 0.0001), has been selected. The total number of pre-training steps in the reference code is 2M steps. We make a starting mode during 100K steps for exploration by using only the Explor. The snapshot of pre-training made in 1M steps is used for fine-tuning. A task vector used in both agents is generated by Exploit.

The best results of NMPS and other models (APS, SMM and DIAYN) are compared in Section 5.4.1, while all NMPSs and APS are evaluated in Section 5.4.2. In Section 5.4.3, four factors of NMPS are analysed.

5.4.1 Using the intrinsic reward of APS and DIAYN

Fig. 5.3 presents a comparison of the best performance achieved by NMPS, APS, DIAYN and SMM in all three domains. According to the results, DIAYN and SMM perform worse than or similarly to APS in Walker and Jaco Arm domains. However, in only Quadruped domain, DIAYN shows remarkable performance in the early stage. Hence, in the experiment on Quadruped domain, NMPS employs DIAYN as Explor for taking advantage of the performance of DIAYN. Overall, due to the expected effect of aforementioned four factors of NMPS, NMPS shows the best performance among all competence-based algorithms.

- Walker, (a) of Fig. 5.3: $NMPS\_X\_exploit^{e*}$, which makes the training of feature different from $NMPS\_X\_exploit^{ex}$, outperforms all other models across all steps. The performance of APS is simialr to that of SMM. DIAYN, a skill discovery algorithm, has the worst performance in Walker domain.

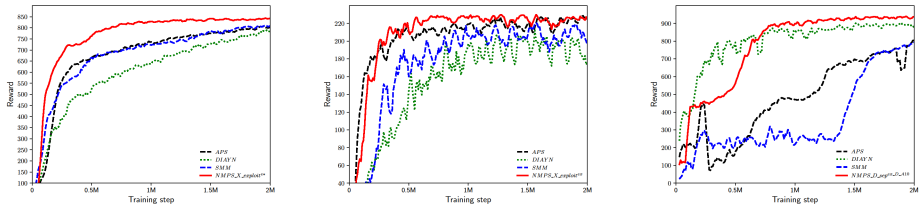
- Jaco Arm, (b) of Fig. 5.3: The performance of $NMPS\_X\_exploit^{ex}$ is better than that of APS after around 0.25M steps and is better than those of SMM and DIAYN (the worst) over all steps. SMM and DIAYN exhibit poor performance even when compared to APS. The skill discovery algorithm DIAYN is also not competitive in Jaco Arm.

- Quadruped, (c) of Fig. 5.3: Although DIAYN exhibits remarkable performance in the early stage, $NMPS\_D\_sep^{ex\_D\_A10}$ shows the best performance after around 0.68M steps. APS has the third best performance among 4 models. SMM has the worst performance after around 0.6M steps in Quadruped.

In Walker, $NMPS\_X\_exploit^{e*}$ shows overwhelming performance across all steps. It dramatically improves the performance of APS, which is similar to that of DIAYN, since it has three components which are decoupling the combined intrinsic reward of APS, not training feature of Explor and using $D_R^{exploit}$ in common for two agents. The above first two components boosts its exploration by empowering much entropy to Explor. On the other hand, Exploit can hold the privilege of 'task inference' for fine-tuning. Using $D_R^{exploit}$ in common can help two agents to synchronize them by using the common data.

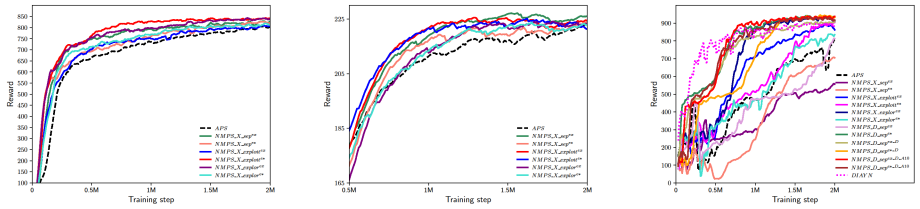
In Jaco Arm, the performance of $NMPS\_X\_exploit^{ex}$ is behind in the very early stage. The decoupled agents causes lagging behind APS during the stage. However, after entering the steady state after 0.25M steps, it outperforms APS. Ensuring training feature of Explor makes the stable operation of Explore in the domain.

In Quadruped, $NMPS\_D\_sep^{ex\_D\_A10}$, a variant type, exhibits the best performance in the Quadruped domain, even though π_{DIAYN}^{explor} feeds all data into $\pi^{exploit}$ during its pre-training. In the early stage of fine-tuning, its Exploit is behind against DIAYN since it has an exploitation agent, $\pi^{exploit}$, with $r^{exploit}$ during its pre-training. However, during the early stage, the performance of Exploit is better than APS and SMM since the data



(A) Walker (B) Jaco Arm (C) Quadruped

FIGURE 5.3: The comparison result of fine-tuning of NMPS (the best one), APS, DIAYN and SMM on Walker (a), Jaco Arm (b) and Quadruped (c) by using the intrinsic reward of APS or DIAYN for Explor of NMPS.



(A) Walker (B) Jaco Arm (C) Quadruped

FIGURE 5.4: The comparison result of fine-tuning of all $NMPS_X$s of NMPS and APS on Walker (a) and Jaco Arm (b, smoothed line) and most NMPS variants and APS on Quadruped (c) by using the intrinsic reward of APS or DIAYN for Explor of NMPS.

of π^{explor}_{DIAYN} is also in charge of the performance of $\pi^{exploit}$. Although its performance is not enough compared with DIAYN in the early stage, it eventually improves and becomes better than DIAYN, whose performance starts to stabilize after around 0.68M steps. The reason is that SFs of Exploit gives rise to the exact 'task inference' during the steady state of DIAYN.

5.4.2 The performance of all NMPSs against that of APS

$NMPS\_X$ in Waker and Jaco Arm are compared with APS. All NMPSs in Quadruped are compared with APS.

- Walker, (a) of Fig. 5.4: $NMPS\_X$ is superior to APS across all steps. The overall performance of each variant type is as follows.

$$NMPS\_X\_explor > NMPS\_X\_exploit >$$
$$NMPS\_X\_sep > APS$$

- Jaco Arm, (b) of Fig. 5.4: Since the performance difference between $NMPS\_X$ and APS is small, the graph shows a smoothed line. However, all variants of $NMPS\_X$ exhibit better performance than APS after at least 0.7M steps. The overall performance of each variant type is as follows.

$$NMPS\_X\_exploit > NMPS\_X\_sep >$$
$$NMPS\_X\_explor > APS$$

- Quadruped, (c) of Fig. 5.4: Most NMPS variants are superior to APS. $NMPS\_D\_sep^{\#\#\_D\_A10}$ shows the best performance. The overall performance order of each variant is as follows.

$$NMPS\_D\_sep^{\#\#\_D\_A10} > NMPS\_D\_sep^{\#\#\_D} >$$
$$NMPS\_X\_explor > NMPS\_D\_sep >$$
$$NMPS\_X\_exploit > APS > NMPS\_X\_sep$$

In Walker, $NMPS\_X$ has the strength against APS. Especially, in this domain, the performance of $NMPS\_X\_sep$ is behind against that of other types, $NMPS\_X\_explor$ and $NMPS\_X\_exploit$.

In Jaco Arm, even though the difference of performance between $NMPS\_X$ and APS is small, all $NMPS_X$s are superior to APS. The performance of $NMPS\_X\_explor$ is behind that of other two types, $NMPS\_X\_exploit$ and $NMPS\_X\_sep$.

In Quadruped, although most $NMPS\_X$ variants are superior to APS, $NMPS\_D$ improves the performance of $NMPS\_X$ further. The difference between the performance of $NMPS\_D$ and that of $NMPS\_X$ is big. In the performance difference, the potential of $NMPS\_D$ is remarkable for how to use an agent with SFs. $NMPS\_D\_sep^{\#\#\_D\_\#\#\#}$ shows high performance among all NMPSs.

Among them, the performance of $NMPS\_D\_sep^{\#\#\_D\_A10}$ is highest. We pay attention to the dimension of feature and skill of $NMPS\_D\_sep^{\#\#\_D\_\#\#\#}$. In addition, it is noticeable that the performance of $NMPS\_D\_sep^{e*}$ is better than that of $NMPS\_D\_sep^{\#\#\_D}$. If the dimension of feature and skill of NMPS is not the same as that of practically optimized feature and skill such as $NMPS\_D\_sep^{\#\#\_D\_A10}$, the performance of $NMPS\_D\_sep^{\#\#\_D}$ is not enough to overcome even that of $NMPS\_D\_sep^{e*}$.

5.4.3 The analysis of four factors of NMPS

Although NMPS using decoupled intrinsic rewards gives rise to the performance improvement of APS, $NMPS\_X\_sep$ is not enough to overcome APS as shown in Quadruped of Fig. 5.4.

In Walker and Jaco Arm, $NMPS\_X\_exploit$ verifies a potential competitiveness across two domains. Furthermore, it is noteworthy that synchronizing both Exploit and Explor through $NMPS\_X\_exploit$ and $NMPS\_X\_explor$ is significant to boost the performance of normal type, $NMPS\_X\_sep$.

The effect of the factor regarding "The train of feature or skill" depends on the domain, as the purpose of this factor introduces a significant variance into NMPS.

Based on the performance results in Quadruped of Fig. 5.4, an agent with SFs can overcome the limitation of supporting only small skill spaces of a discriminator by leveraging the superior performance of another competence-based algorithm in the environment.

5.5 Discussion

5.5.1 Decoupling a monolithic exploration agent with SFs

The variants of NMPS demonstrate the contribution of NMPS. Most NMPS variants outperform APS in all environments. Decoupling exploitation and exploration in the original monolithic algorithm with SFs improves its performance. The mode-switching controller (Homeo in this research) for both agents impacts the rate of communication with the environment. As a result, agents with SFs based on NMPS show better performance than APS during fine-tuning.

5.5.2 The factors to maximize the performance of NMPS

In this research, several factors are found to influence the performance of decoupled unsupervised pre-training. These factors include the factor of mode switching controller,

i.e. the target rate ρ, and four factors related to the variants of NMPS. The Homeo controller's target rate, ρ, is found to be a crucial factor. Furthermore, 5.4.2 shows that the performance difference among the variants of $NMPS\_X$ and $NMPS\_D$ are due to the four factors related to their design. These factors result in performance differences among the four groups of NMPS across different environments. Notably, even though there are the agents of same type, taking advantage of $D_R^{\#}$ and $'T.\ f\ or\ s'$ makes a different performance. They are other breakthroughs for $NMPS\_X$ and $NMPS\_D$ to overcome their own limitation.

5.5.3 The vulnerable point of competence-based approach

We believe that the performance of DIAYN in Quadruped domain is better than that of APS due to the feature dimension of APS to some extent since the feature dimension of APS, 10, is smaller than the skill dimension of DIAYN which is 16.

Interestingly, the performance of $NMPS\_D\_sep^{\#\#\_^D\_^{A10}}$ based on the feature dimension of APS is better than that of $NMPS\_D\_sep^{\#\#\_^D}$ and DIAYN. The extreme non-monolithic operation of Explor of $NMPS\_D\_sep^{\#\#\_^D\_\#\#\#}$ has the full charge of exploration as well as exploitation. Finally, $NMPS\_D\_sep^{\#\#\_^D\_^{A10}}$ suggests a new methodology in order to overcome the linear reward-regression issue of agent using SFs.

Chapter 6

A Non-Monolithic Policy Approach of Offline-to-Online Reinforcement Learning

6.1 Introduction

Offline RL [114] has been researched to address the disadvantages of online RL, such as the non-trivial cost associated with collecting data from a real environment and the risks of physical interactions with the environment, such as with real robots [115, 116]. However, it still suffers from limited performance due to sub-optimal datasets in downstream tasks.

Offline-to-online RL addresses the disadvantages of both offline RL and online RL [117, 118]. It involves pre-training a policy on an offline dataset, followed by fine-tuning the online policy using the pre-trained offline policy in a real environment.

However, two main issues arise in offline-to-online RL [76]. The first issue stems from offline RL and involves out-of-distribution (OOD) challenges due to distribution mismatch, which impacts the exploitation phase during the fine-tuning process. The second issue pertains to the exploration aspect of offline-to-online RL being hindered by the constrained policy of offline RL.

To address these persistent issues, numerous studies have been conducted within the RL community [75, 119, 120]. Among these approaches, Policy Expansion (PEX) [5], the current state-of-the-art offline-to-online RL algorithm that utilizes an unmodified offline policy, leverages a policy set comprising both policies without altering the offline policy in the pre-training-to-fine-tuning scheme for exploration and learning. However, the excessive involvement of both policies in exploration leads to deficiencies in both exploitation and exploration. Its step operation unit cannot ensure the autonomous training of the online agent, ultimately deteriorating the performance of PEX.

In a pre-training-to-fine-tuning scheme, the offline policy is pre-trained using an offline dataset, proving beneficial for exploiting the distribution of a downstream task. The unmodified offline policy in offline-to-online RL can also assist its counterpart, the online policy, in guiding exploration to some extent. Thus, the unmodified offline policy becomes crucial for effectively exploiting the downstream task. Simultaneously, the online policy has the potential to explore a downstream task due to its immature policy, addressing the distribution shift between the offline dataset and the downstream task. Achieving optimal agent performance requires ensuring proper execution timing and duration for the online policy in relation to the offline policy. Therefore, our research focuses on how to reconcile the advantages of the offline policy (exploitation) and the online policy (exploration) in offline-to-online RL, without compromising the integrity of the offline policy, to enhance overall agent performance.

In this research, we propose a novel model for offline-to-online RL that utilizes a non-monolithic exploration methodology without modifying the offline policy. In our model, an offline policy and an online policy are specialized in exploitation and exploration, respectively. Thus, our offline-to-online model features a policy set similar to PEX, complemented by a mode-switching controller to select an active policy. An agent utilizing this model without modifying the original offline policy should address the following questions: How does the agent leverage the offline policy for exploitation based on the nature of the downstream task? Additionally, how does the agent effectively explore with the online policy?

The main contributions of our research are outlined below:

- The development of our offline-to-online RL model leverages a non-monolithic exploration methodology to strategically focus on both exploitation and exploration.
- The ability to leverage flexibility and generalization in a downstream task.

6.2 Our methodology

The ultimate goal of offline-to-online RL is to leverage the pre-learned knowledge of the offline policy and the adaptability of the online policy. Unlike the unified framework, which compromises the integrity of a pre-trained offline policy, it becomes challenging to timely extract the original knowledge from the offline policy. Conversely, an offline-to-online RL scheme that retains an unmodified offline policy can achieve its objectives if equipped with a suitable mode-switching controller. Although PEX employs such a scheme, it struggles to ensure sufficient online training for exploration and learning, as indicated in Eq. (2.15) and Eq. (2.16).

6.2.1 Offline-to-online RL with a mode-switching controller

Our model incorporates a non-monolithic exploration methodology into offline-to-online RL, featuring a policy set Π comprised of π^{off} and π^{on}, as illustrated in Fig. 6.1.

$$\Pi = [\pi^{\text{off}}, \pi^{\text{on}}] \tag{6.1}$$

The composite policy $\tilde{\pi}$, based on Π, consists of both π^{off} and π^{on}, with their roles separated into exploitation and exploration, respectively. The activation of π^{off} and π^{on} is determined by a mode-switching controller. We use Homeo, whose algorithm is defined in [58], as the mode-switching controller in our model, as detailed in 'Algorithm 1' of 'supplementary document.pdf'. The D_{promise} function[2.13] of the Homeo, denoted as f_{homeo}, utilizes $Q^{\text{off}}(s, \mathit{Action})$, the state-action value function of π^{off}, to monitor changes in the pre-trained Q^{off} during a predefined timeframe. The decision of Action of $\tilde{\pi}$ as determined

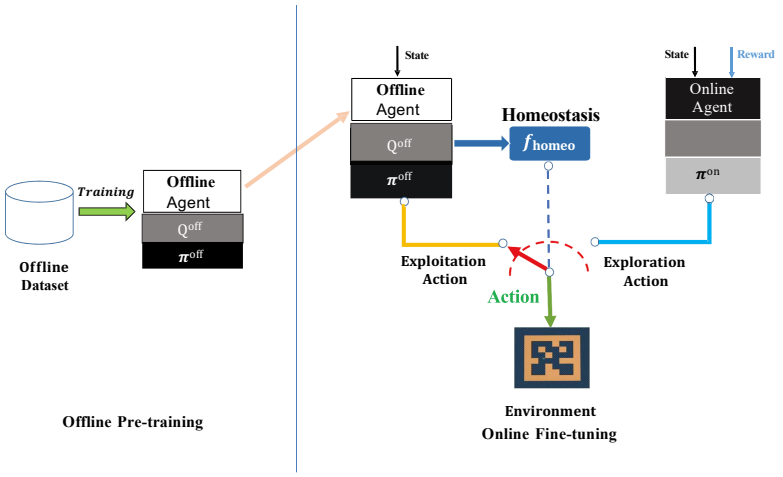


FIGURE 6.1: Illustration of offline-to-online RL training schemes employed in our model with an unmodified offline policy.

by Homeo is as follows:

$$
\begin{aligned}
D^{\text{off}}_{\text{promise}} &= D^{\text{off}}_{\text{promise}}(t-k,t) \\
\tilde{\pi}(Action|s) &= \begin{cases} a^{\text{off}}_t \sim \pi^{\text{off}}, & \text{for } f_{\text{homeo}}(D^{\text{off}}_{\text{promise}}) = 0 \\ a^{on}_t \sim \pi^{on}, & \text{for } f_{\text{homeo}}(D^{\text{off}}_{\text{promise}}) \neq 0 \end{cases}
\end{aligned} \tag{6.2}
$$

where $f_{\text{homeo}}(D_{\text{promise}})$ with the input of $D^{\text{off}}_{\text{promise}}(t-k,t)$ using $Q^{\text{off}}(s,a)$ denotes Homeo for the decision of mode switching controller in which $D^{\text{off}}_{\text{promise}}(t-k,t)$ denotes the value promise discrepancy of π^{off} and the result value of $f_{\text{homeo}}(D_{\text{promise}})$ is based on Bernoulli distribution.

The policy $\tilde{\pi}$ operates with a union replay buffer $\mathrm{D_{off}} \cup \mathrm{D_{on}}$. Consequently, π^{on} also utilizes the same union replay buffer for its own training, whether π^{off} or π^{on} is active. This union replay buffer facilitates the rapid improvement of π^{on}'s performance. The framework with discrete agents, π^{off} and π^{on}, ensures the primary objective of our research, where π^{off} focuses on exploitation and π^{on} on exploration. However, the role of π^{on} will change during the late online fine-training stage, a topic that will be discussed later.

Our model adopts a heterogeneous temporal structure for mode-switching exploration. Two agents, each with a distinct purpose, monitor Q^{off} over a specific period. If the state of Q^{off} deteriorates within a predefined timeframe, the mode of policy $\tilde{\pi}$ transitions from

Algorithm 5 The main algorithm of online stage

Initialize:
  Set the value of *explore_fixed_steps*
  Set the value of *update_timestep*
  Set the value of ρ in (0.0001 ~ 0.9) for Homeo
  Set π^{off} and Q^{off} through a pre-trained offline policy and then freeze them
procedure EVALUATE $\pi^{\text{OFF}}(D_{\text{promise}}$ of $\pi^{\text{off}})$
  Compute the output of Homeo with D_{promise} of
    π^{off} based on *explore_fixed_steps*
end procedure
for $t = 0, \ldots, T-1$ **do**
  if *update_timestep* **then**
    Compute D_{promise} of π^{off} through Q^{off}
    Evaluate $\pi^{\text{off}}(D_{\text{promise}}$ of $\pi^{\text{off}})$
  end if
  if *output of Homeo == exploitation mode* **then**
    Action $\leftarrow \pi^{\text{off}}$
  else
    Action $\leftarrow \pi^{\text{on}}$
  end if
  Execute *Action*, observe reward and next_state
end for

exploitation to exploration. What triggers this switch? The mode-switching controller recognizes when the current exploitation by the offline policy is no longer reliable and consequently terminates it. This initiates an exploration phase for the downstream task, leading to a shift in $\tilde{\pi}$'s mode to exploration. Control of action is then transferred from π^{off} to π^{on}.

Meanwhile, during the online fine-training, we hypothesize that the role of π^{on} on Π changes as follows. During the beginning and middle of online fine-training,

$$H\Big(\pi^{off}\Big) < H\Big(\pi^{on}\Big), \quad \text{therefore, } \pi^{\text{on}} : \text{exploration policy} \tag{6.3}$$

where $H\Big(\cdot\Big)$ denotes the overall entropy of a policy. In the late online fine-training stage,

$$\begin{gathered} \mathrm{D_{off}} \cup \mathrm{D_{on}} \approx \mathrm{D_{Down,OPT}}, \\ \text{so, } H\Big(\pi^{off}\Big) \geq H\Big(\pi^{on}\Big), \\ \text{therefore, } \pi^{\mathrm{on}} : \text{exploitation-oriented policy.} \end{gathered} \tag{6.4}$$

During the beginning and middle of online fine-training, π^{on} primarily focuses on exploration. However, as training progresses, π^{on} becomes increasingly specialized in the downstream task. Consequently, in the late stages of online training, the Homeo controller selects between the exploitation policy π^{off} and the now exploitation-oriented policy π^{on} to enhance exploitation.

PEX also employs a heterogeneous temporal structure in mode-switching exploration but utilizes a different switching mode controller. This controller assesses the value function Q_ϕ of both agents not just during a specific period but at every training step, as illustrated in Equations (2.15) and (2.16). Consequently, PEX considers both agents for the combined roles of exploitation and exploration, making these its primary focus at each training step. To align with PEX's main objectives, π_θ should undergo sufficient training during a designated period, with an emphasis on exploring through π_θ.

In PEX, frequent interventions of π_β within $\tilde{\pi}$ can negatively affect the exploration capabilities of π_θ. Our model addresses this concern by focusing exclusively on Q^{off} during specific periods, which allows π^{on} to improve independently. In this model, π^{off} is solely responsible for exploitation, while π^{on} is dedicated to exploration until the midpoint of the online fine-training. During the late stages of online fine-training, π^{on}, now more exploitation-oriented, shifts its focus towards enhancing exploitation. This non-monolithic exploration approach efficiently meets these specific requirements.

6.2.2 The ability to leverage flexibility and generalization on a downstream task

Equations (2.15) and (2.16) illustrate the operation of $\tilde{\pi}$ per training step during the online fine-tuning phase of PEX. The premise for ensuring PEX performance is the sufficient

training of the online policy π_θ, emphasizing exploration. However, the operational mechanism of PEX does not guarantee this due to the step operation. Conversely, in our model, if π^{off} does not perform well during a predefined period (referred to as *update_timestep*), π^{on} can undergo sufficient sequential training over a designated period (referred to as *explore_fixed_steps*) based on its exploration capabilities. This setup ensures the autonomous training of π^{on} during the specified timeframe.

Therefore, our model can modulate the execution times of both π^{off} and π^{on} based on the dissimilarity between the distribution of the offline dataset and that of the downstream task. This allows our model to support a flexible and generalized approach, adapting effectively to various situations in both the offline dataset and the downstream task.

Given that π^{off} is extractable and π^{on} is trainable, controlling the performance of $\tilde{\pi}$ flexibly involves addressing two main concerns. Consequently, the role of the modulator, which comprises *update_timestep* and *explore_fixed_steps*, is crucial for the performance of $\tilde{\pi}$. The characteristics of the modulator enable our model to exhibit either an exploitative or exploratory bias, adapting dynamically to the needs of the task.

An optimal policy can be achieved through the careful negotiation of the modulator, striking a balance between exploitative and exploratory biases. Once the aforementioned parameters are well-tuned, this balance becomes achievable for any downstream task.

The modulating and mode-switching characteristics of our model provide a robust adaptive capacity for various downstream tasks, a feature absent in PEX. Consequently, our model excels in maximizing both pre-experienced and newly acquired knowledge, effectively adapting to different downstream tasks.

The table 6.1 shows the comparison between our model and PEX.

6.3 Experiments

The open-source code of PEX[1], which employs implicit Q-learning (IQL) [65] as the backbone algorithm, serves as a reference for our experiments on the standard D4RL benchmark

[1] https://github.com/Haichao-Zhang/PEX

TABLE 6.1: The comparison between our model and PEX.

Item	**Our model**	**PEX**
Operation unit of policy	Period	Step
Switching condition on $\mathbf{\Pi}$	Only Q^{off}	Q_{ϕ}
Switching-mode controller	Homeostasis	$P_{\mathbf{w}}$
Are the roles of agents on $\mathbf{\Pi}$ separated?	Yes	No
The role of online agent changes?	Yes	No
Is the online agent trained autonomously?	Yes	No
A modulator exists?	Yes	No

[121]. All tasks tested are summarized in 'Table 1' of 'supplementary document.pdf'. Training is conducted over five different random seeds. The common hyper-parameters used for all baselines are detailed in 'Table 2' of 'supplementary document.pdf', aligning with those in the open-source code of PEX to ensure a fair comparison.

Several baselines are employed for comparison with our model: **(i) PEX**: offline-to-online RL using Policy Expansion, serving as the primary benchmark; **(ii) Offline**: uses only an offline policy with IQL, without online policy training during the online fine-tuning phase; **(iii) Buffer**: involves only online policy training using IQL with a replay buffer of $\mathrm{D_{off}} \cup \mathrm{D_{on}}$ without an offline policy [122]. The contrasting characteristics of the Offline and Buffer models make them suitable for our evaluation. All baseline models are implemented using the open-source code of PEX, with IQL as the foundational algorithm. Online fine-tuning is performed across all models over 1M environmental steps.

To implement our model, we adhere to the format described in Section 3.1 of [58], labeled as XI-intra(100, informed, p, G) (See 'Appendix' of 'supplementary document.pdf'), where 'XI' denotes an intrinsic reward, '100' specifies an explore duration, 'informed' identifies a trigger type, p represents the exploitation duration parameterized by a target rate ρ (See 'Table 3' in 'supplementary document.pdf'), and 'G' indicates the greedy mode without a starting mode for exploration. In our implementation, we have made several modifications: the value of the target rate ρ is adjusted to [0.0001, 0.9], differing from the values (0.1, 0.01, 0.001, 0.0001) used in [58]. The intrinsic reward is replaced with an 'external reward,' specifically, an environmental reward used during online fine-tuning. The 'explore duration'

is replaced with '$explore\_fixed\_steps$'. For triggering, 'Homeo' is employed, focusing on the 'value promise' of Q^{off}.

The comparison process of our implementation unfolds as follows: First, an offline policy is pre-trained on a benchmark task from D4RL. Second, π^{off} and Q^{off}, which are derived from this offline pre-training, are frozen and then utilized during online fine-tuning. The online policy is fine-tuned using three main parameters: ρ, '$explore\_fixed\_steps$' and '$update\_timestep$', tailored to a benchmark task from D4RL. To ensure a fair comparison, PEX, Offline, and Buffer are tested using the same pre-trained offline policy. The main algorithm of our non-monolithic exploration RL model during the fine-tuning phase is detailed in Algorithm 5.

There are two comparative aspects in our research. Firstly, our model is primarily compared with PEX. Secondly, all models experimented with in this study are compared to further research interests. In these experiments, our model demonstrates superior performance over PEX, Offline, and Buffer.

6.3.1 The comparison between our model and PEX

Our research focuses on two key metrics: normalized return and execution count for both models. Initially, we analyze the normalized return across each task to compare the performance of the two models. This analysis will highlight the differences between our model, which employs non-monolithic exploration, and PEX, which utilizes policy expansion. Despite both models leveraging the same offline-to-online RL approach, our comparison aims to demonstrate their distinct performances without compromising the integrity of the offline RL policies.

Additionally, we analyze the number of times each policy—both offline and online—executes during the online fine-tuning phase for both models. By monitoring execution at every training step, this analysis helps illustrate the development of $\tilde{\pi}$ and estimate the contribution of each policy throughout the training process.

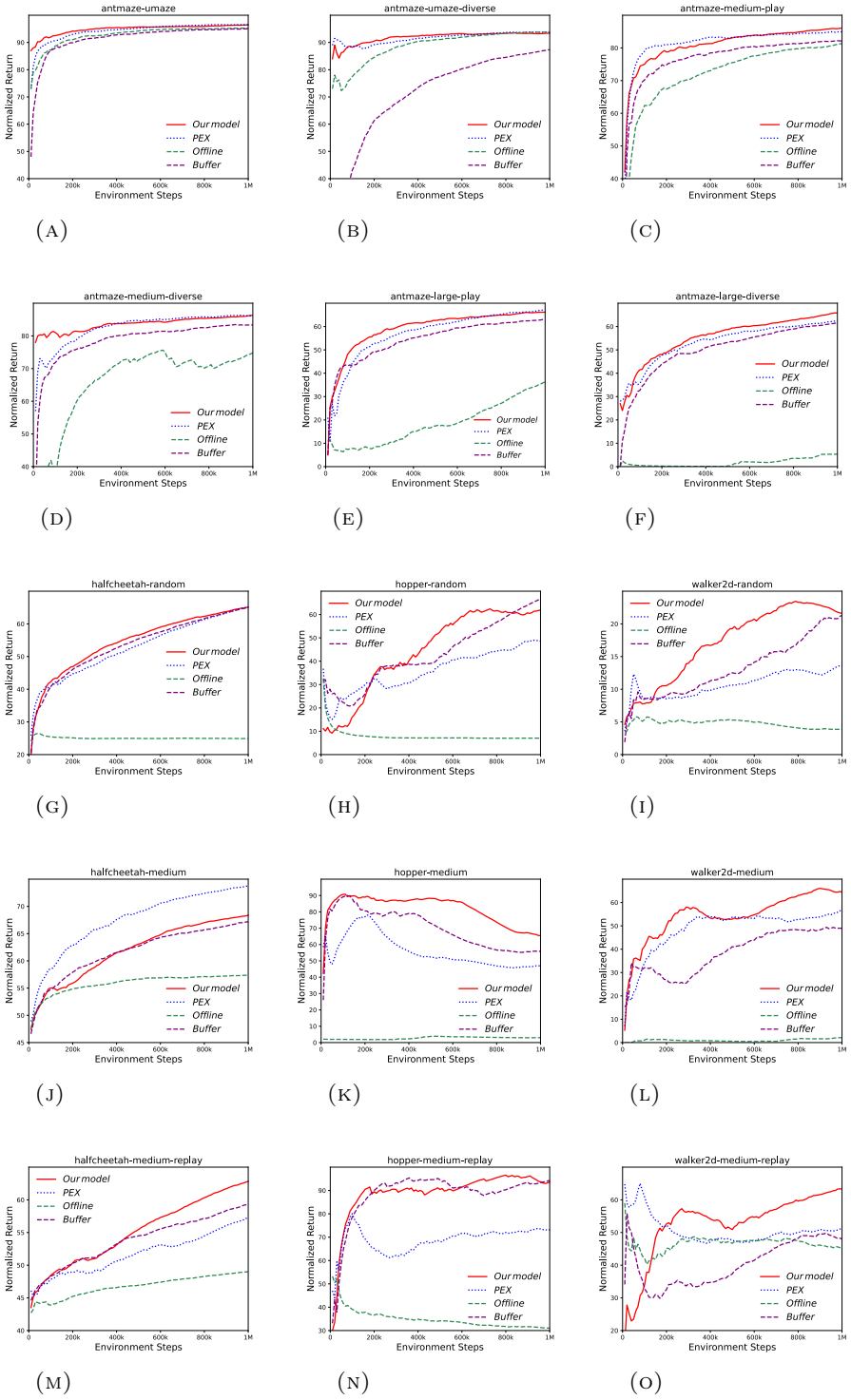


FIGURE 6.2: Normalized Return Curves of different methods, which are our model, PEX, Offline and Buffer, on benchmark tasks from D4RL. IQL is used for all methods as the backbone.

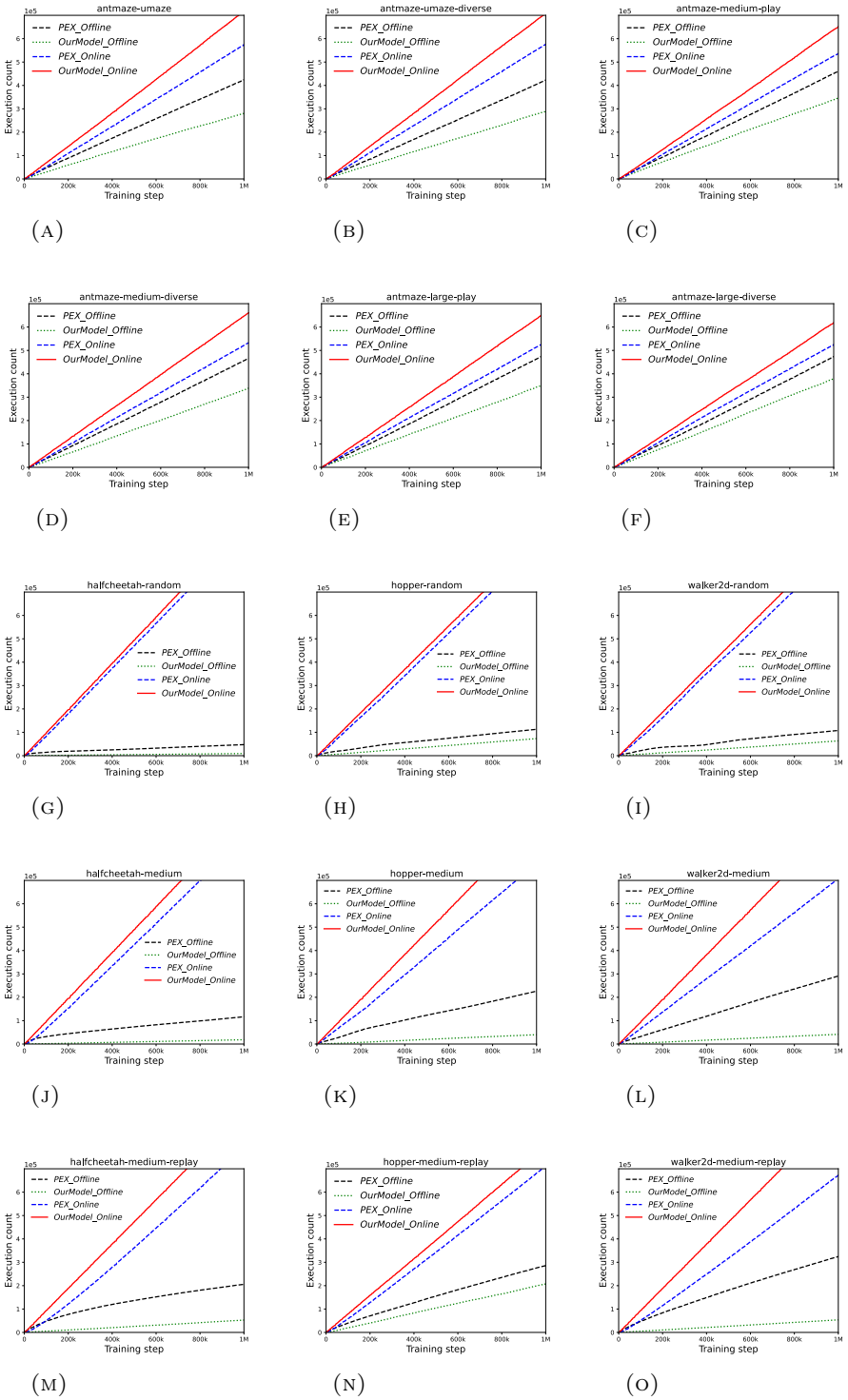


FIGURE 6.3: Execution count of our model and PEX on benchmark tasks from D4RL. The execution counts of offline policy and online policy of PEX or our model are referred to as $PEX\_Offline$ and PEX_Online or $OurModel\_Offline$ and $OurModel\_Online$, respectively.

6.3.1.1 Normalized return

In the Antmaze environment, which features non-Markovian policies, sparse rewards, undirected, and multitask data, our model either matches or outperforms PEX in most cases, with the exception of 'antmaze-medium-play'. Initially, our model faces challenges in 'antmaze-umaze-diverse' and 'antmaze-large-diverse' due to the immature online policy that undertakes exploration in datasets characterized by random goals and start locations. Paradoxically, the performance of our model surpasses that of PEX from an early stage, attributed to the sufficient exploration by the online policy. Notably, our model excels in 'antmaze-large-diverse', which presents the most challenging conditions.

In environments such as HalfCheetah, Hopper, and Walker, which involve suboptimal agents and narrow data distributions, our model significantly outperforms PEX, with the exception of 'halfcheetah-medium'. Initially, our model faces challenges due to exploration across most datasets. However, performance improves markedly from the early stages and is substantially better than that of PEX. Notably, our model demonstrates greater robustness in the latter stages, unlike PEX, whose performance tends to decline in 'hopper-medium', 'hopper-medium-replay', and 'walker2d-medium-replay'.

6.3.1.2 Execution count

'Execution count' refers to the number of times each policy is executed at every training step. Our implementation code meticulously tracks the execution count of each policy at every training step. In the Antmaze environment, the discrepancy in execution count between the offline and online policies in PEX is smaller compared to our model. Our model significantly increases this difference, which correlates with enhanced performance. This indicates that PEX does not adequately reflect the current performance of $\tilde{\pi}$ during online fine-tuning, as evidenced by its overall performance metrics.

In the HalfCheetah, Hopper, and Walker environments, although there is a significant difference in the execution count between the offline and online policies in PEX, our model further increases this disparity. This observation suggests that the offline dataset does not substantially contribute to the performance of $\tilde{\pi}$ in PEX. In contrast, our model efficiently

utilizes the offline policy to enhance the performance of $\tilde{\pi}$, even though the offline policy is executed less frequently. Meanwhile, the online policy is given more opportunities to execute within these domains, allowing it to further refine and solidify its own policy.

Across both models, the execution count of the online policy exceeds that of the offline policy, underscoring the reliance of $\tilde{\pi}$'s performance on the online policy. However, despite the lower execution count, the offline policy's contribution to $\tilde{\pi}$'s performance remains critical. The disparity between the execution counts of the offline and online policies varies with the dataset type, with a particularly pronounced difference observed in the 'random' dataset. Our model exhibits a larger discrepancy than PEX, even where the difference in PEX is substantial. This suggests that more extensive training of the online policy enhances performance on the dataset, while less frequent execution of the offline policy may be sufficient. Furthermore, in our model, the execution count for the offline policy is consistently lower than in PEX, and conversely, the count for the online policy is consistently higher (for further details, see 'C Ablation Study' in 'supplementary document.pdf').

6.3.2 The comparison between our model and others

In Antmaze, while the performance of both Offline and Buffer trails behind our model and PEX, an interesting observation is that Buffer outperforms Offline in tasks such as 'antmaze-medium-play', 'antmaze-medium-diverse', 'antmaze-large-play', and 'antmaze-large-diverse'. This indicates that training the online policy is more crucial than relying solely on the pre-trained offline policy for these tasks. Notably, in 'antmaze-large-diverse', the performance of the Offline model is particularly low, suggesting that the offline policy alone cannot significantly enhance the performance of $\tilde{\pi}$.

In HalfCheetah, Hopper, and Walker environments, Buffer outperforms PEX in all cases except for 'halfcheetah-medium', 'walker2d-medium', and 'walker2d-medium-replay'. This underlines the significance of online fine-tuning, as demonstrated by Buffer's superior performance compared to PEX. Our model, which shows even better performance than Buffer, effectively harnesses the synergy between the offline policy and the online policy, optimizing their combined impact on performance.

In three Antmaze tasks—'antmaze-umaze', 'antmaze-umaze-diverse', and 'antmaze-medium-diverse'—our model's performance is comparable to PEX. However, in 'antmaze-medium-play' and 'halfcheetah-medium', our model underperforms relative to PEX. Despite these outcomes, our model outperforms PEX in 10 out of all 15 tasks. Against Buffer, our model's performance is similar in 'halfcheetah-random', 'halfcheetah-medium', and 'hopper-medium-replay'. Yet, it excels in 12 out of the 15 tasks evaluated, demonstrating a higher overall performance than Buffer.

6.4 Discussion

6.4.1 Will π^{on} be also suboptimal and even unsafe?

Thanks to the adaptive role of π^{on} throughout the stages of online fine-training (refer to Eq. 6.3 and Eq. 6.4), π^{on} is reliable. Except for 'hopper-random' and 'halfcheetah-medium', our model maintains competitive performance even in the late stages of training, demonstrating its effectiveness across various experiments.

6.4.2 The effect of non-monolithic exploration methodology for offline-to-online RL

As depicted in Fig. 6.3.1, PEX faces challenges in ensuring sufficient training for the online policy. The execution counts of the offline and online policies in PEX are higher and lower, respectively, compared to those in our model. This suggests that PEX's design does not effectively manage the training of the online policy during the online fine-tuning phase. In contrast, our model provides adequate training for the online policy, effectively compensating for the relatively limited knowledge from the offline policy, which was pre-trained during the offline phase. This more robust approach in handling online policy training is further supported by the data in table 6.1.

6.4.3 The efficient usage of offline policy during online fine-tuning

Fig. 6.3.2 demonstrates that our model utilizes the offline policy more effectively than PEX, despite having a smaller execution count for the offline policy. This efficient use of the offline policy significantly enhances the performance of $\tilde{\pi}$, optimizing the balance between policy use and overall effectiveness.

6.4.4 How can the method be translated to a wider real-world problems?

The application of our methodology to the distributed model of federated learning can be envisioned. In this setup, some local models might operate using an unmodified off-policy, while others adopt an on-policy approach, varying key parameters accordingly. The global model could then leverage a suitable mode-switching controller to select the optimal policy from among all the local models, effectively integrating diverse learning strategies to enhance overall performance.

6.4.5 The considerations of further research

Currently, the online fine-tuning process requires manual and fixed adjustment of three key parameters: ρ, '$explore\_fixed\_steps$', and '$update\_timestep$'for the online policy. However, we hypothesize that the performance of offline-to-online RL could be significantly improved if these parameters were dynamically adjusted by the agent itself. In this proposed scenario, a parameter controller such as 'Meta RL' would modulate these key parameters. A crucial aspect for effective offline-to-online RL, especially when the offline policy remains unmodified, involves how the execution times for each policy—offline or online—are adjusted.

This research primarily focuses on offline-to-online reinforcement learning (RL) using an unmodified offline policy. Consequently, PEX, which is the current state-of-the-art offline-to-online RL algorithm that leverages an unmodified offline policy, is chosen as the main reference for comparison. Additionally, we plan to extend our research to include comparisons with offline-to-online RL models based on the unified framework, where the online policy operates based on the parameters of the offline policy.

Chapter 7

Conclusion

We introduced various policy compositions in hierarchical reinforcement learning (HRL) and transfer reinforcement learning (TRL) to identify an optimal design. The conclusions of each chapter, from Chapter 3 to Chapter 6, are summarized as follows.

In chapter 3, we have introduced a novel hierarchical reinforcement learning (HRL) inverse model that supports level synchronization using FDGM for off-policy correction in HRL. Our model outperforms HIRO and LSP, as demonstrated in four test tasks within the Ant environment, which includes two reward shapes (high and low) in a two-level hierarchical structure. Our model presents a generalizable approach for the inverse model in model-free HRL.

Our research demonstrates that off-policy correction in HRL can be effectively implemented using data acquired from a policy based on FDGM. This approach, known as the direct method, is more practical and generally applicable, as it relies on natural data computation without requiring peripheral information used in indirect estimations.

To date, there has been little research on FDGM in the RL area. Typically, most RL research utilizes general neural networks like fully connected neural networks. As RL might require additional functionality such as the inverse operation, more RL research in the future may benefit from using a generative model.

Further research is needed to enhance our model. Future studies should focus on RL using a general FDGM algorithm with feature-based data, rather than image-based data, which is large in size. Additionally, our generalized inverse model could be explored in atomic RL, even though it has been studied in HRL. We also consider extending our work to non-Markovian environments in the future.

In chapter 4, in order to overcome the issues of a non-monolithic exploration, this paper introduces an autonomous non-monolithic agent with multi-mode exploration based on options framework. We reveal the potential of our model to follow a behaviour thought of humans and animals. Our model takes advantage of the difference in the degree of entropy of each exploration policy with a guidance-exploration framework. A robust optimal policy can be expected due to the evaluation process. The research on a guided exploration of the adaptive strategy for the multi-mode exploration of an autonomous non-monolithic agent is required. The further research on the modeling of S_E in the agent is also required for the robust optimal policy.

In chapter 5, the research proposes a novel unsupervised pre-training method with SFs, NMPS. To improve performance, it incorporates a non-monolithic exploration methodology with greater flexibility, generalization of the discriminator, and optimized training methods for each agent's feature or skill.

Further research is needed to enhance our model for an agent which is not based on features or skills for Explor.

In chapter 6, the research introduces a novel non-monolithic exploration methodology for offline-to-online reinforcement learning (RL), which effectively utilizes an unmodified offline policy. This approach significantly enhances the agent's ability to leverage flexibility and generalization in downstream tasks, thereby improving performance across various scenarios.

Our research identifies the potential of offline-to-online RL with a non-monolithic exploration. To enhance the effectiveness of offline-to-online RL, various further researches of offline-to-online RL with a non-monolithic exploration framework are required.

Policy composition in deep reinforcement learning enables the development of diverse algorithmic approaches to enhance agent performance. Our research specifically focuses on advancing artificial general intelligence (AGI) cognition, with particular emphasis on improving agents' cognitive capabilities through various policy architectures.

References

[1] Xuezhe Ma, Xiang Kong, Shanghang Zhang, and Eduard Hovy. Decoupling global and local representations from/for image generation. *arXiv preprint arXiv:2004.11820*, 2020.

[2] Miruna Pislar, David Szepesvari, Georg Ostrovski, Diana Borsa, and Tom Schaul. When should agents explore? *arXiv preprint arXiv:2108.11811*, 2021.

[3] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018.

[4] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021. ISBN 2640-3498.

[5] Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. *arXiv preprint arXiv:2302.00935*, 2023.

[6] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[7] Volodymyr Mnih, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, Koray Kavukcuoglu, et al. Strategic attentive writer for learning macro-actions. *arXiv preprint arXiv:1606.04695*, 2016.

[8] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[9] Jacob Rafati and David C Noelle. Learning representations in model-free hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10009–10010, 2019.

[10] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[11] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 12, 2017.

[12] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

[13] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

[14] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[17] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[18] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[19] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[20] Jianlin Su and Guang Wu. f-vaes: Improve vaes with conditional flows. *arXiv preprint arXiv:1809.05861*, 2018.

[21] Soheil Kolouri, Phillip E Pope, Charles E Martin, and Gustavo K Rohde. Sliced-wasserstein autoencoder: An embarrassingly simple generative model. *arXiv preprint arXiv:1804.01947*, 2018.

[22] Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning*, pages 4277–4285. PMLR, 2018.

[23] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv: 1711.01558*, 2017.

[24] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[25] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[26] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[27] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[28] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[29] Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.

[30] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

[31] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[32] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[33] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

[34] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

[35] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. ISSN 0004-3702.

[36] Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008. ISSN 1083-4419.

[37] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.

[38] Sebastian Flennerhag, Jane X Wang, Pablo Sprechmann, Francesco Visin, Alexandre Galashov, Steven Kapturowski, Diana L Borsa, Nicolas Heess, Andre Barreto, and Razvan Pascanu. Temporal difference uncertainties as a signal for exploration. *arXiv preprint arXiv:2010.02255*, 2020.

[39] Andrea L Thomaz and Cynthia Breazeal. Experiments in socially guided exploration: Lessons learned in building robots that learn with and without human teachers. *Connection Science*, 20(2-3):91–110, 2008. ISSN 0954-0091.

[40] Esther Derman, Daniel Mankowitz, Timothy Mann, and Shie Mannor. A bayesian approach to robust reinforcement learning. In *Uncertainty in Artificial Intelligence*, pages 648–658. PMLR, 2020. ISBN 2640-3498.

[41] Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021.

[42] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. ISBN 2374-3468.

[43] Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. Adaptive skills adaptive partitions (asap). *Advances in neural information processing systems*, 29, 2016.

[44] Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *International Conference on Machine Learning*, pages 3130–3139. PMLR, 2019. ISBN 2640-3498.

[45] Nicolas Bougie and Ryutaro Ichise. Fast and slow curiosity for high-level exploration in reinforcement learning. *Applied Intelligence*, 51(2):1086–1107, 2021. ISSN 1573-7497.

[46] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[47] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[48] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378. IEEE, 2016. ISBN 1538626829.

[49] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision*, pages 483–492, 2017.

[50] Lucas Lehnert, Stefanie Tellex, and Michael L Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

[51] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[52] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

[53] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[54] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[55] Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

[56] Samuel J Gershman and Bastian Greshake Tzovaras. Dopaminergic genes are associated with both directed and random exploration. *Neuropsychologia*, 120: 97–104, 2018. ISSN 0028-3932.

[57] R Becket Ebitz, Brianna J Sleezer, Hank P Jedema, Charles W Bradberry, and Benjamin Y Hayden. Tonic exploration governs both flexibility and lapses. *PLoS computational biology*, 15(11):e1007475, 2019. ISSN 1553-734X.

[58] Miruna Pislar, David Szepesvari, Georg Ostrovski, Diana Borsa, and Tom Schaul. When should agents explore? *arXiv preprint arXiv:2108.11811*, 2021.

[59] Gina G Turrigiano and Sacha B Nelson. Homeostatic plasticity in the developing nervous system. *Nature reviews neuroscience*, 5(2):97–107, 2004. ISSN 1471-0048.

[60] Zhihui Xie, Zichuan Lin, Junyou Li, Shuai Li, and Deheng Ye. Pretraining in deep reinforcement learning: A survey. *arXiv preprint arXiv:2211.03959*, 2022.

[61] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

[62] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

[63] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. In *European Conference on Computer Vision*, pages 111–128. Springer, 2022.

[64] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[65] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[66] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[67] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[68] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[69] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. ISSN 2162-237X.

[70] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019. ISBN 2640-3498.

[71] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[72] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

[73] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

[74] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021. ISBN 2640-3498.

[75] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022. ISBN 2640-3498.

[76] Siyuan Guo, Yanchao Sun, Jifeng Hu, Sili Huang, Hechang Chen, Haiyin Piao, Lichao Sun, and Yi Chang. A simple unified uncertainty-guided framework for offline-to-online reinforcement learning. *arXiv preprint arXiv:2306.07541*, 2023.

[77] Deyao Zhu, Yuhui Wang, Jürgen Schmidhuber, and Mohamed Elhoseiny. Guiding online reinforcement learning with action-free offline pretraining. *arXiv preprint arXiv:2301.12876*, 2023.

[78] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[79] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[80] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[81] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[82] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[83] Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. Hierarchical reinforcement learning via advantage-weighted information maximization. *arXiv preprint arXiv:1901.01365*, 2019.

[84] John Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, pages 1009–1018. PMLR, 2018.

[85] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[86] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[87] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[88] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[89] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[90] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[91] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[92] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[93] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, and Andew Bolt. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

[94] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

[95] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[96] Yijie Guo, Jongwook Choi, Marcin Moczulski, Shengyu Feng, Samy Bengio, Mohammad Norouzi, and Honglak Lee. Memory based trajectory-conditioned policies for learning from sparse rewards. *Advances in Neural Information Processing Systems*, 33:4333–4345, 2020.

[97] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[98] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, and Mohammad Gheshlaghi Azar. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[99] Suzana Ilić, Edison Marrese-Taylor, Jorge A Balazs, and Yutaka Matsuo. Deep contextualized word representations for detecting sarcasm and irony. *arXiv preprint arXiv:1809.09795*, 2018.

[100] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[101] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[102] Xu Han and Feng Wu. Meta reinforcement learning with successor feature based context. *arXiv preprint arXiv:2207.14723*, 2022.

[103] Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

[104] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pretraining. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

[105] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312, 2016*, 2016.

[106] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993. ISSN 0899-7667.

[107] André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Žídek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. *arXiv preprint arXiv:1901.10964*, 2019.

[108] Chen Ma, Dylan R Ashley, Junfeng Wen, and Yoshua Bengio. Universal successor features for transfer reinforcement learning. *arXiv preprint arXiv:2001.04025*, 2020.

[109] Samuel J Gershman and Bastian Greshake Tzovaras. Dopaminergic genes are associated with both directed and random exploration. *Neuropsychologia*, 120: 97–104, 2018. ISSN 0028-3932.

[110] R Becket Ebitz, Brianna J Sleezer, Hank P Jedema, Charles W Bradberry, and Benjamin Y Hayden. Tonic exploration governs both flexibility and lapses. *PLoS computational biology*, 15(11):e1007475, 2019. ISSN 1553-734X.

[111] Miruna Pislar, David Szepesvari, Georg Ostrovski, Diana Borsa, and Tom Schaul. When should agents explore? *arXiv preprint arXiv:2108.11811*, 2021.

[112] Gina G Turrigiano and Sacha B Nelson. Homeostatic plasticity in the developing nervous system. *Nature reviews neuroscience*, 5(2):97–107, 2004. ISSN 1471-0048.

[113] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[114] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[115] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[116] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020.

[117] Yihuan Mao, Chao Wang, Bin Wang, and Chongjie Zhang. Moore: Model-based offline-to-online reinforcement learning. *arXiv preprint arXiv:2201.10070*, 2022.

[118] Nair Ashvin, Dalal Murtaza, Gupta Abhishek, and L Sergey. Accelerating online reinforcement learning with offline datasets. *CoRR, vol. abs/2006.09359*, 2020.

[119] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022. ISBN 2640-3498.

[120] Han Zheng, Xufang Luo, Pengfei Wei, Xuan Song, Dongsheng Li, and Jing Jiang. Adaptive policy learning for offline-to-online reinforcement learning. *arXiv preprint arXiv:2303.07693*, 2023.

[121] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[122] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.