

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

INTRINSICALLY-MOTIVATED EXPLORATION
AND TASK-AWARE DATA EXPLOITATION: A
PATH TO ACCELERATED REINFORCEMENT
LEARNING

MINGQI YUAN

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University

Department of Computing

Intrinsically-Motivated Exploration and Task-Aware Data
Exploitation: A Path to Accelerated Reinforcement Learning

Mingqi Yuan

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
January 2026

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Mingqi Yuan

Abstract

Learning through interaction with the environment is a lifelong behavior of human beings, which has significantly inspired the advancement of reinforcement learning (RL). As a general and powerful paradigm, RL has been widely adopted across the technology revolution, from large language models (LLMs) to cutting-edge semiconductor design. Despite its success, RL remains fundamentally constrained by severe sample inefficiency, in which training often requires millions of interactions and may still yield suboptimal policies or fail entirely. This limitation leads to substantial computational overhead and significantly restricts the applicability of RL in domains where data collection is costly or impractical.

As RL increasingly involves diverse domains, the need for more data-efficient algorithms becomes ever more critical. This thesis is dedicated to accelerating RL training by systematically enhancing exploration and exploitation, the two fundamental mechanisms that govern learning efficiency. In Chapter 4, we conduct a comprehensive study on the intrinsically-motivated exploration mechanism. More specifically, propose computation-efficient intrinsic reward formulations that provide sustained exploration incentives, alleviating the biased-objective issue arising from intrinsic reward shaping, and establish a unified benchmark to support reproducible research and principled comparison in this area. In Chapter 5, we focus on task-aware data exploitation and develop a series of acceleration strategies, including adaptive exploitation scheduling, ultra-lightweight hyperparameter optimization, and state rep-

resentation learning tailored for RL. These techniques collectively enhance training stability, sample efficiency, and generalization while maintaining low computational overhead.

Beyond algorithmic development, we demonstrate the practical impact of the proposed acceleration techniques on complex humanoid robot learning tasks. By applying the developed methods to whole-body control problems, we show remarkable improvements in learning speed, robustness, and real-world performance. In summary, this thesis makes significant contributions to the field of data-efficient RL, advancing the development of a more efficient and scalable RL paradigm to support future AI research across diverse domains.

Publications Arising from the Thesis

1. Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. “Rewarding Visitation Discrepancy for Exploration in Reinforcement Learning”, in *Deep Reinforcement Learning Workshop NeurIPS*, 2022.
2. Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. “Automatic Intrinsic Reward Shaping for Exploration in Deep Reinforcement Learning”, in *International Conference on Machine Learning*, 2023.
3. Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Wenjun Zeng, Glen Berseth. “RLeXplore: Accelerating Research in Intrinsically-Motivated Reinforcement Learning”, *Transactions on Machine Learning Research*, 2025
4. Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. “Adaptive Data Exploitation in Deep Reinforcement Learning”, in *International Conference on Machine Learning*, 2026, under review.
5. Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. ULTHO: Ultra-Lightweight yet Efficient Hyperparameter Optimization in Deep Reinforcement Learning. in *IEEE/CVF International Conference on Computer Vision*, 2025.
6. Mingqi Yuan, Tao Yu, Haolin Song, Bo Li, Xin Jin, Hua Chen, Wenjun Zeng. PvP: Data-Efficient Humanoid Robot Learning with Proprioceptive-Privileged

Contrastive Representations. in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisors, Prof. Wenjun Zeng, Prof. Bo Li, and my co-supervisor Prof. Qing Li, for their careful guidance, immense patience, and generous support throughout my doctoral journey. It is my great privilege to learn from and work under their supervision.

I am profoundly grateful to my close collaborator, Prof. Xin Jin, for his invaluable assistance with the research. His expertise and collaborative spirit were crucial to overcoming many challenges encountered during this project.

I extend my warm appreciation to Prof. Fumiya Iida, Dr. Kai-Fung Chu, and Dr. Huijiang Wang for their gracious hospitality and invaluable help during my visit to the University of Cambridge. Their support and stimulating discussions made my time there both productive and inspiring.

At last, I would like to express my sincere thanks to my parents and friends for their unwavering support and constant encouragement.

Table of Contents

Abstract	i
Publications Arising from the Thesis	iii
Acknowledgments	v
List of Figures	x
List of Tables	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	4
1.3 Contributions	7
1.4 Thesis Structure	9
2 Literature Review	10
2.1 Exploration Enhancement for RL Acceleration	10
2.1.1 Classic Exploration Strategies	10

2.1.2	Intrinsic Reward-Driven Exploration	12
2.1.3	LLM-Empowered Intrinsic Reward Generation	14
2.2	Exploitation Enhancement for RL Acceleration	15
2.2.1	Data Exploitation Scheduling	15
2.2.2	Data Augmentation Techniques	16
2.2.3	State Representation Learning	17
2.3	Engineering-Oriented RL Acceleration	19
2.3.1	Algorithm Implementation Acceleration	19
2.3.2	Environment Interaction Acceleration	19
3	Background	21
3.1	Reinforcement Learning	21
3.2	Multi-Armed Bandit	22
3.3	Representative MAB Algorithms	22
3.4	Intrinsic Reward Shaping	24
3.5	Representative RL Algorithms	24
3.6	Hyperparameter Optimization in RL	26
3.7	Humanoid Whole-Body Control	26
4	Intrinsically-Motivated Exploration for RL Acceleration	28
4.1	Rewarding Visitation Discrepancy for Efficient Exploration in RL . .	28
4.1.1	Introduction	28
4.1.2	Methodology	31

4.1.3	Experiments	35
4.1.4	Conclusion	39
4.2	Automatic Intrinsic Reward Shaping for Exploration in RL	40
4.2.1	Introduction	40
4.2.2	Methodology	43
4.2.3	Experiments	45
4.2.4	Conclusion	52
4.3	RLeXplore: Accelerating Research in Intrinsically-Motivated RL . . .	53
4.3.1	Introduction	53
4.3.2	Methodology	57
4.3.3	Architecture	58
4.3.4	Algorithmic Baselines	59
4.3.5	Experiments	62
4.3.6	Conclusion	74
5	Task-Aware Data Exploitation for RL Acceleration	77
5.1	ADEPT: Adaptive Data Exploitation in Deep RL	77
5.1.1	Introduction	77
5.1.2	Adaptive Data Exploitation	80
5.1.3	Experiments	85
5.1.4	Conclusion	92
5.2	ULTHO: Ultra-Lightweight yet Efficient Hyperparameter Optimization in Deep RL	93

5.2.1	Introduction	93
5.2.2	The ULTHO Framework	97
5.2.3	Experiments	101
5.2.4	Conclusion	109
5.3	PvP: Data-Efficient Humanoid Robot Learning with Proprioceptive-Privileged Contrastive Representations	111
5.3.1	Introduction	111
5.3.2	PvP Implementation	114
5.3.3	The SRL4Humanoid Framework	116
5.3.4	Experiments	119
5.3.5	Conclusion	131
6	Conclusion and Future Work	133
6.1	Conclusion	133
6.2	Future Work	135
	References	137

List of Figures

4.1	REVD evaluates the discrepancy of visitation distributions in different learning periods and transforms it into intrinsic rewards, providing powerful and sustainable exploration incentives.	29
4.2	Workflow of the REVD. The state is first encoded by a fixed, randomly initialized encoder before being compared with the historically and recently visited states. The intrinsic reward is computed as the summation of the distances to its k -NN in the two representation spaces (R. S.).	31
4.3	Comparison of different exploration statuses. The agent is expected to explore distinct and diverse states compared to the states it has previously visited.	33
4.4	Screenshots of the three selected MiniGrid environments.	36
4.5	The performance comparison of the A2C agent and its combinations with three intrinsic rewards. The solid line and shaded regions represent the mean and standard deviation, respectively.	38
4.6	The performance comparison of REVD with different k -values used for intrinsic reward computation. The solid line and shaded regions represent the mean and standard deviation, respectively.	39

4.7	The performance comparison of REVD with different buffer sizes used for state archival. The solid line and shaded regions represent the mean and standard deviation, respectively.	39
4.8	(a) AIRS employs two branches in the value network to disentangle the estimation of total return (evaluated by intrinsic and extrinsic reward function) and task return (only extrinsic reward function). (b) AIRS formulates the intrinsic reward selection as a multi-armed bandit problem and uses the upper confidence bound (UCB) to make decisions based on the estimated task return. Ext.: Extrinsic. Int.: Intrinsic. $\oplus$: Weighted summation operation.	41
4.9	The overview of AIRS. (a) The architecture of DAAC. Here, we use two branches in the value network to estimate $V_{\phi}^E(\mathbf{s})$ and $V_{\phi}^{E+I}(\mathbf{s})$. (b) AIRS selects the intrinsic reward function based on a queue (length= W) of estimated task returns.	43
4.10	Sample efficiency comparison between AIRS and baselines. The solid line and shaded regions represent the mean and standard deviation, which are computed over five random seeds.	49
4.11	PPO-Normalized test performance for the Procgen benchmark, which is aggregated over all sixteen tasks and ten random seeds.	50
4.12	Each row shows the probability of improvement, with 95% bootstrap confidence intervals, that the algorithm X on the left outperforms algorithm Y on the right, given that X was claimed to be better than Y . For all algorithms, results are based on ten random seeds per task.	51
4.13	Value loss of eight selected games during training. “E Return” indicates the branch for extrinsic value estimation in the value network. The solid line and shaded regions represent the mean and standard deviation over 10 random seeds, respectively.	51

4.14	The cumulative number of times AIRS selects each intrinsic reward function over the training progress, computed with ten random seeds.	52
4.15	The workflow of RLEXPLORE. (a) RLEXPLORE provides a decoupled module for intrinsic rewards that integrates seamlessly with the RL training loop. RLEXPLORE implements 8 SOTA intrinsic rewards and adapts to the unmodified RL training loop. (b) RLeXplore monitors the agent-environment interactions and gathers data samples using the <code>.watch()</code> function. After collecting experience rollouts, RLeXplore computes the corresponding intrinsic rewards using the <code>.compute()</code> function and updates the auxiliary models via the <code>.update()</code> function. (c) RLeXplore provides a <code>Fabric</code> class that allows developers to combine multiple intrinsic rewards in an elegant manner.	57
4.16	Screenshots of the selected exploration games. (a) <i>SuperMarioBros</i> . (b) <i>MiniGrid</i> . (c) <i>ALE-5</i> . (d) <i>Procgen-Maze</i> . (e) <i>Ant-UMaze</i>	63
4.17	Episode returns achieved by the intrinsic rewards in RLeXplore. (left) <i>SuperMarioBros</i> without access to the task rewards. (right) <i>MiniGrid-DoorKey-16×16</i> with sparse rewards.	63
4.18	Results for Q1, Q2, Q3, Q4, and Q5 in <i>SMB</i> (top) and <i>MGD</i> (bottom), which are normalized by the maximum score possibly achieved in the task. Here, ON is the observation normalization, RN is the reward normalization, UP is the update proportion, and WI is the weight initialization. The bar with a hatch mark is Combined , which refers to the results of using the best hyperparameters gathered in each question. Since RE3 only employs a fixed, randomly initialized neural network for encoding observations, there are no values in Q3. All the results are aggregated over 10 seeds, and each run uses 10M environment interactions.	66

4.19	Aggregated performance of the eight intrinsic rewards with different low-level hyperparameters over 10 random seeds. The vertical dashed line represents the performance of the extrinsic agent, which only has access to the task rewards. Here, ON is the observation normalization, RN is the reward normalization, UP is the update proportion, WI is the weight initialization, IQM is the interquartile mean, OG is the optimality gap (lower is better), and Combined refers to the results of using the best hyperparameters gathered in each question. All the computation is performed using the Rliable [2] library.	69
4.20	Performance of four selected intrinsic rewards in RLeXplore on the top eight most challenging tasks of the MGD suite. The solid line and shaded regions represent the mean and standard deviation computed with five random seeds, respectively.	71
4.21	(Left) During training, the extrinsic agent struggles to find the goal in the selected Maze, resulting in a reward of 0. While some intrinsic reward methods yield occasional non-zero rewards, the methods perform significantly better when intrinsic and extrinsic value estimation are decoupled using two distinct value heads in the agent’s network. (Right) In the Progen variant, where each maze represents a unique level, the baseline extrinsic agent achieves the goal 50% of the time, and intrinsic rewards don’t outperform the baseline significantly. We note that the presence of easier levels, where the goal may occasionally be near the agent’s starting point, results in generally less sparse rewards and an easier task to learn.	73

4.22	(Left) The performance ranking of single and mixed intrinsic rewards on the <i>SuperMarioBrosRandomLevels</i> . As expected, episodic bonuses (such as E3B and RIDE) demonstrate superior performance, attributed to the environment’s non-singleton MDP nature. (Right) Overall performance comparisons between the single and mixed intrinsic rewards. Here, G.E. denotes the six ”global+episodic” combinations, such as E3B+RND and E3B+ICM. G.G. denotes the three ”global+global” combinations, such as RND+ICM and RND+RIDE.	75
5.1	Aggregated training performance and computational overhead comparison on the Procgen benchmark. ADEPT serves as a plug-and-play module that enhances RL algorithms, significantly improving data efficiency and reducing computational costs.	78
5.2	Overview of the ADEPT framework. (a) A typical workflow of RL algorithms. (b) ADEPT optimizes data exploitation by adaptively adjusting IDU in the update phase, <i>e.g.</i> , the number of update epochs in the on-policy setting. (c) ADEPT formulates the IDU scheduling as a MAB problem, in which the estimated task return is leveraged as the utility function to evaluate the IDU values. Currently, we provide 4 scheduling strategies, including adaptive and rule-based approaches.	81
5.3	Aggregated training performance comparison on the Procgen benchmark with increasing NUE values.	82
5.4	Training performance and computational overhead comparison of the PPO and its combinations with ADEPT on eight Procgen environments. The solid line and shaded regions represent the mean and standard deviation, respectively, across five runs. Note that the dotted line and dashed line represent the highest score and the lowest overhead, respectively.	88

5.5	Training performance and computational overhead comparison of the DrAC and its combinations with ADEPT on eight Procgen environments. The solid line and shaded regions represent the mean and standard deviation, respectively, across five runs. Note that the dotted line and dashed line represent the highest score and the lowest overhead, respectively.	89
5.6	Aggregated performance of the PPO, DrAC, and their combinations with ADEPT on the test levels of the Procgen benchmark. All the scores are normalized by the corresponding PPO scores, and bars indicate 95% confidence intervals computed using stratified bootstrapping over five random seeds. Note that * represents the best scores gathered from all four ADEPT methods.	89
5.7	Ratio of dead units (RDU) of the policy networks trained by the vanilla PPO, DrAC, and their combinations with ADEPT. ADEPT significantly decreases the RDU and effectively preserves the richness of representation, thereby improving the agents' generalization ability. .	90
5.8	The aggregated decision processes of ADEPT-U and ADEPT-G for PPO on the eight selected Procgen environments. ADEPT effectively eliminates redundant updates, thereby improving data efficiency and reducing computational overhead.	90
5.9	Aggregated training performance comparison of ADEPT with different sets of NUE values. The mean and standard error are computed across all the environments.	91
5.10	Aggregated performance of three ADEPT methods on the MiniGrid benchmark. The mean and standard deviation are computed across all the environments.	91

5.11	Performance of the SAC and its combinations with four ADEPT methods on the DMC benchmark. The mean and standard deviation are computed using five random seeds.	92
5.12	Aggregated performance comparison on the Procgen benchmark with procedurally-generated environments. The choice of batch size has a significant impact on the agent’s performance. ULTHO can effectively perform HPO across different tasks and learning stages, thereby improving the overall performance.	94
5.13	Overview of ULTHO framework. (a) The key phases of RL algorithms. ULTHO serves as a plug-and-play module that feeds the optimized HPs to the RL algorithm. (b) ULTHO maintains inter-cluster and intra-cluster return distributions to perform HPO in a hierarchical manner. (c) A sliding window is used to store the estimated task returns for updating the return distributions.	97
5.14	Screenshots of the ALE (top) and Procgen (below) benchmarks. . . .	101
5.15	Performance comparison of the two ULTHO algorithms and HPO baselines on the ALE-5 benchmark, in which the mean and standard error are computed using five random seeds. All the scores are normalized using the min-max normalization with the human expert performance. The two ULTHO algorithms achieve similar or even higher performance with a few training budgets as compared to other HPO baselines. . .	103

5.16	Training performance of the PPO and its combinations with two ULTHO algorithms in eight Procgen environments, in which the mean and standard deviation are computed using five random seeds. ULTHO consistently outperforms the vanilla PPO agent in all the environments, significantly improving the sample efficiency. However, the Relay-ULTHO does not achieve any further performance gains, indicating that a more comprehensive policy, which continuously adapts multiple HPs, is crucial for highly dynamic environments.	104
5.17	Aggregated inter-cluster decision processes of ULTHO on the whole Procgen benchmark. Among the four HP clusters, the NUE cluster is selected the most, suggesting its high impact on overall performance.	106
5.18	Aggregated intra-cluster decision processes of ULTHO on the whole Procgen benchmark. Among the four HP clusters, the NUE cluster is selected the most, suggesting its high impact on overall performance.	107
5.19	The variation of aggregated confidence intervals on the whole Procgen environments. Here, the solid line represents the mean value, and the dashed line represents the final upper confidence bound. It is evident that the NUE cluster obtains the highest upper confidence bound at the end of training, which aligns with the selection proportion illustrated in Figure 5.18.	108
5.20	Aggregated performance of ULTHO on the Procgen benchmark with different configurations of c and W , and the mean and standard error are computed across all the environments. These results demonstrate that ULTHO is robust to the variation of the two internal HPs. . . .	109

5.21	Aggregated performance of ULTHO on the MiniGrid benchmark, and the mean and standard deviation are computed across all the environments. For the relatively stationary environments, Relay-ULTHO can further obtain performance gains through incremental optimization. .	110
5.22	Performance of the PPO baseline and two ULTHO algorithms on the PyBullet benchmark, and the mean and standard deviation are computed using five random seeds. Our method can effectively improve the sample efficiency through high-quality HPO in continuous control tasks.	110
5.23	(a) PvP employs contrastive learning between proprioceptive and privileged states to learn compact and task-relevant representations for policy learning. (b) The aggregated learning curves of PvP and algorithmic baselines across multiple evaluation tasks, where PvP significantly improves training efficiency. (c) The real-world evaluation of PvP on the LimX Oli humanoid robot.	112
5.24	An overview of the PvP approach. (a) The components of the privileged state and the proprioceptive state. (b) PvP conducts contrastive learning based on the intrinsic complementarity between the two state modalities.	115
5.25	The architecture of the SRL4Humanoid framework, in which the SRL and RL processes are fully decoupled.	117
5.26	The specifications of the LimX Oli humanoid robot used in the experiments, and the screenshots of the two designed tasks.	120
5.27	Training progress comparison between the vanilla PPO agent and its combination with four SRL methods on the two humanoid WBC tasks. The solid line and shaded region denote the mean and standard deviation, respectively.	121

5.28	The comparison of action smoothness optimization between the vanilla PPO agent and its combinations with the four SRL methods. The solid line and shaded region denote the mean and standard deviation, respectively.	122
5.29	The tracking performance comparison between the PPO agent and its combinations with the four SRL methods. Our PvP achieves the highest performance in terms of the three key tracking metrics. . . .	123
5.30	Training progress comparison of the four SRL methods with different training time proportions on the two humanoid WBC tasks. The solid line and shaded region denote the mean and standard deviation, respectively.	124
5.31	Training progress comparison of the four SRL methods with different training data proportions on the two humanoid WBC tasks. The solid line and shaded region denote the mean and standard deviation, respectively.	125
5.32	Learning curves of applying the SRL to the value encoder. The solid line and shaded region denote the mean and standard deviation, respectively.	126
5.33	Performance comparison of the PvP method with different weighting coefficients on the <i>LimX-Oli-31dof-Mimic</i> task. The solid line and shaded region denote the mean and standard deviation, respectively. .	128
5.34	Training progress comparison between the teacher-student distillation method and PvP in the <i>LimX-Oli-31dof-Velocity</i> task and <i>LimX-Oli-31dof-Mimic</i> task. The solid line and shaded region denote the mean and standard deviation, respectively.	129

5.35	(Left) Training progress comparison between the teacher-student distillation method and PvP in the <i>LimX-Oli-31dof-Velocity-Rough</i> task. (Right) Training progress comparison between the vanilla PPO agent and its combination with four SRL methods on the <i>Unitree-G1-29dof-Velocity</i> task. The solid line and shaded region denote the mean and standard deviation, respectively.	130
5.36	Sim2Sim evaluation on the MuJoCo simulator. The first two rows demonstrate motion imitation ability, and the last two rows show velocity tracking ability.	131

List of Tables

4.1	Hyperparameters used for A2C in the MiniGrid experiments.	37
4.2	Four intrinsic reward functions are employed in the Procgen experiments, where $\mathbf{e} = g(\mathbf{s})$ is the representation of $\mathbf{s}$, N_{ep} is the episodic state visitation frequency. For RE3 and RISE, g is a random and fixed encoder, while g is a learned encoder for RIDE. Finally, $\tilde{\mathbf{e}}$ is the k -nearest neighbor of $\mathbf{s}$ in the encoding space.	47
4.3	Summary of comparative results from our RLeXplore implementations and prior work. The right two columns report success rates (% of episodes solving the task), except for Procgen, where we report mean episode rewards after training.	55
4.4	Comparison between RLeXplore and other reported libraries of intrinsic rewards. Note that we focus on the intrinsic reward methods that are implemented. For instance, CleanRL has numerous implementations of various RL algorithms, but RND is the only intrinsic reward currently supported. Here, PnP is short of plug-and-play.	56
4.5	Details of the baseline settings.	65
4.6	PPO hyperparameters for <i>SuperMarioBros</i> , <i>MiniGrid</i> , and <i>Procgen</i> games. These remain fixed for all experiments.	67

5.1	The selected hyperparameter clusters for each benchmark.	103
5.2	Normalized test performance (%) comparison on the Progen benchmark. Here, IQM denotes the interquartile mean suggested by [2], and ‡ indicates the training is conducted by a population of agents. Similar to the training performance comparison, the normal ULTHO method achieves the highest score by providing a more comprehensive HPO policy.	105

Chapter 1

Introduction

1.1 Motivation

Learning is the process of gaining new understanding, acquiring knowledge, developing behaviors, building skills, and shaping values, attitudes, and preferences, which is a fundamental capability that contributes to the formation of intelligence [61]. Human learning behavior persists throughout life and is continuously shaped by interactions with the surrounding environment [185, 91]. Through repeated cycles of perception, action, and feedback, humans progressively refine their behavioral strategies, adapt to new situations, and develop increasingly sophisticated cognitive and motor capabilities [166, 21]. This interaction-driven and experience-based adaptation observed in human learning serves as a critical inspiration for computational learning paradigms, especially for reinforcement learning.

Reinforcement learning (RL) adheres to this interaction-driven learning philosophy, explicitly modeling learning through interactions, which is one of the three fundamental paradigms of machine learning. In RL, an agent observes the state of its environment, makes sequential actions, and receives reward signals, learning a policy that maximizes the long-term expected return [189]. Unlike supervised learning,

which relies on labeled data and explicitly specified targets [41, 68], RL agents need to discover desirable behaviors through trial-and-error interaction. Moreover, RL is fundamentally goal-driven, aiming to improve future performance and optimize cumulative utility, whereas unsupervised learning focuses on modeling data distributions or discovering latent structure [12, 69]. As a result, RL is uniquely suited to problems requiring continual adaptation, sequential decision-making, and learning under uncertainty.

RL has demonstrated remarkable success across a wide range of domains. In game playing, RL agents such as AlphaGo [179, 180] and AlphaStar [199] learn directly from raw observations and achieve superhuman performance. In the semiconductor manufacturing industry, RL-based graph placement methods [132] have been employed to automate chip floor planning, significantly reducing design time and producing layouts comparable to or exceeding those produced by human experts. In finance, frameworks such as FinRL enable deep RL applications in algorithmic trading, portfolio optimization, and other sequential decision-making problems under market uncertainty [117, 116]. Within the broader AI ecosystem, RL-based post-training [146] has become a cornerstone in shaping the behavior of large language models (LLMs), including ChatGPT [1] and DeepSeek [62], improving reasoning performance and aligning model behavior with human values and preferences [94]. Beyond these achievements, RL continues to play a critical role in areas like algorithm discovery [55] and embodied intelligence [194], establishing it as a central methodology in modern AI research.

Despite these advancements, RL is consistently constrained by several critical challenges that limit its broader applicability. RL agents often require millions of interactions with their environments to achieve satisfactory performance, yet still face the risk of learning failure, especially in sparse-reward, hard-exploration environments. This sample inefficiency issue is particularly significant in settings where interactions are expensive, unsafe, or slow, such as real-world robotics or LLMs training [22, 208, 223]. Furthermore, RL training is notoriously unstable, in which minor

changes in initialization, environment randomness, or algorithmic implementations can lead to drastically different outcomes [78, 80, 156]. For example, RL is highly sensitive to hyperparameters, which requires extensive effort (*e.g.*, by hyperparameter optimization or hand-crafted tuning) to identify suitable configurations [222, 52]. Additionally, generalization remains a persistent challenge, as policies trained on specific environments or task distributions frequently fail to transfer to unseen states, new conditions, or real-world variations [160, 159]. These challenges collectively motivate the pursuit of RL acceleration: developing methods that stabilize training, enhance sample efficiency and generalization, thereby enabling the reliable and scalable deployment of RL across a wide range of increasingly complex and dynamic decision-making scenarios.

The core of this acceleration objective lies in the exploration and exploitation capabilities of RL agents, which are two fundamental aspects of RL. In RL, exploration is the process by which an agent interacts with the environment to acquire diverse and informative experiences. In contrast, exploitation refers to how effectively the agent utilizes its accumulated experience to update its policy and value estimates, thereby improving task performance. RL agents often encounter issues of insufficient exploration and low exploitation efficiency, particularly in complex environments with high-dimensional observations and sparse reward spaces. Consequently, accelerating RL requires targeted improvements in both exploration and exploitation. In this thesis, we investigate the RL acceleration problem from both perspectives, aiming to improve exploration mechanisms and enhance exploitation efficiency. On the exploration side, we investigate intrinsically-motivated reward mechanisms that promote efficient and scalable exploration. On the exploitation side, we develop task-aware data-exploitation strategies that enhance training efficiency through principled scheduling, ultra-lightweight hyperparameter optimization, and effective state representation learning. We evaluate the proposed methods across diverse environments and provide benchmark implementations to support reproducible research. Through

these efforts, this thesis aims to advance the field of data-efficient RL and broaden its applicability in increasingly complex domains.

1.2 Research Objectives

The overall objective of this thesis is to accelerate RL by systematically enhancing both exploration and exploitation. From the exploration perspective, this thesis focuses on intrinsically motivated exploration through intrinsic reward shaping (IRS). Recent studies [149, 25, 10] have demonstrated that insufficient exploration remains a critical bottleneck in diverse RL scenarios, especially in visual control tasks such as video game playing. The agent must learn policies directly from high-dimensional pixel observations while facing sparse, delayed, or deceptive reward signals. As a result, RL agents are prone to prematurely converging to local optima after a finite number of iterations, and traditional exploration strategies (*e.g.*, ϵ -greedy or entropy bonuses) are often futile in such settings. IRS has emerged as a promising approach to alleviate these challenges by introducing additional learning signals that encourage agents to explore novel, informative, or uncertain states. By providing dense intrinsic feedback that complements sparse extrinsic rewards, IRS can significantly enhance learning efficiency, especially in hard-exploration tasks. On the exploitation side, this thesis focuses on three tightly coupled aspects that directly influence learning efficiency: data exploitation scheduling, hyperparameter optimization, and state representation learning. Finally, we summarize the detailed research objectives as follows:

- **Advanced intrinsically-motivated exploration.** Despite the success of IRS, many existing intrinsic reward methods suffer from the problem of vanishing intrinsic rewards and are unable to provide sustained exploration incentives. Moreover, these methods rely heavily on complex auxiliary models and repre-

sentation learning. It produces substantial computational overhead, and poor representation learning may result in severe performance loss, making it difficult to apply to arbitrary tasks. Additionally, most IRS approaches cannot guarantee the invariance of the optimal policy with respect to the extrinsic reward, and excessive or improperly applied exploration signals may even cause learning collapse. Since the joint objective composed of extrinsic and intrinsic rewards is inherently biased, optimizing it does not necessarily yield the optimal policy for the original task reward. Furthermore, intrinsic reward computation often requires auxiliary models, heavy engineering, and expensive computations, which significantly hampers reproducibility. Existing literature also reports highly variable performance for identical intrinsic rewards across different implementations, highlighting the need for a standardized framework and a deeper understanding of the optimization and implementation details. In response, this thesis aims to enhance this area by focusing on designing computation-efficient intrinsic rewards, completing their application mechanisms, and establishing a benchmark for intrinsic reward research.

- **Efficient and task-aware data exploitation.** While intrinsically-motivated exploration effectively addresses the data acquisition inefficiency, the exploitation inefficiency also becomes increasingly significant when scaling to complex environments. It directly determines how effectively collected experiences are transformed into policy improvement, thereby affecting overall training efficiency. This thesis investigates data exploitation from three critical perspectives. Firstly, we optimize the data exploitation scheduling, *i.e.*, the synchronization between data sampling and policy update. A representative concept is the update-to-data (UTD) ratio, which is defined as the number of update steps performed per environment step. Existing UTD research primarily applies to off-policy RL algorithms, while systematic investigation of on-policy algorithms is underexplored and will be deeply studied in this thesis. Secondly, we focus

on hyperparameter optimization (HPO) during training, a fundamental problem in ML that has been shown to significantly affect the learning efficiency and final performance of RL agents. Traditional HPO strategies (*e.g.*, random search, grid search, and Bayesian optimization) are consistently inefficient in RL due to its non-stationarity and high computational overhead. We aim to develop a more efficient HPO algorithm that operates on the fly, automatically and adaptively searching for optimal hyperparameters within a single run. Finally, we move to state representation learning (SRL), which aims to transform high-dimensional sensory inputs into compact and informative latent representations. By preserving task-relevant dynamics while filtering out noise and redundancy, effective SRL can substantially improve sample efficiency, generalization, and training stability. Accordingly, this thesis develops an RL-specific and task-aware SRL method to further maximize data exploitation efficiency.

- **Application to practical problem-solving.** Beyond algorithm innovation, this thesis also aims to demonstrate the impact of accelerated RL in practical problem-solving. In particular, we investigate learning acceleration for complex humanoid robot control, with an emphasis on whole-body control (WBC) tasks. Humanoid robots have emerged as a critical platform for embodied intelligence, whose human-like morphology offers inherent advantages in versatility, adaptability to human-centered environments, and intuitive interaction. However, deep RL applications to humanoid WBC remain severely constrained by sample inefficiency. The intricate dynamics and partial observability of humanoid robots, along with the necessity to optimize a composite reward structure to ensure task performance and reliability in real-world deployment, significantly increase the learning complexity. This thesis aims to leverage the proposed approaches to facilitate the solution of practical humanoid WBC tasks, thereby advancing data-efficient humanoid robot learning.

1.3 Contributions

In this section, we summarize the main contributions of each chapter in this thesis. Each point corresponds to the research objectives outlined in Section 1.2.

- In Section 4.1, we propose **REVD: REwarding Visitation Discrepancy**, which provides powerful exploration incentives with a simple, computation-efficient, and robust architecture. REVD maximizes the divergence-based visitation discrepancy to encourage exploration in RL, which is transformed into intrinsic rewards using an efficient estimator. REVD can be directly integrated with arbitrary RL algorithms, significantly promoting the sample efficiency with a simple and robust architecture. This work is **published in**:

Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. Rewarding Visitation Discrepancy for Exploration in Reinforcement Learning. *Deep Reinforcement Learning Workshop NeurIPS*, 2022 [213].

- In Section 4.2, we present **AIRS: Automatic Intrinsic Reward Shaping** that intelligently and adaptively provides high-quality intrinsic rewards to enhance exploration in RL. More specifically, AIRS selects the IRS function from a predefined set in real time based on the estimated task return, providing reliable exploration incentives and alleviating the biased objective problem. This work is **published in**:

Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. Automatic Intrinsic Reward Shaping for Exploration in Deep Reinforcement Learning. *International Conference on Machine Learning*, 2023 [214].

- In Section 4.3, we introduce **RLeXplore**, a unified, highly modularized, and plug-and-play framework offering reliable implementations of eight state-of-the-art intrinsic reward methods. Furthermore, we conduct an in-depth study that identifies critical implementation details and establishes well-justified standard practices for intrinsically-motivated RL. This work is **published in**:

Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Wenjun Zeng, Glen Berseth. RLeXplore: Accelerating Research in Intrinsically-Motivated Reinforcement Learning. *Transactions on Machine Learning Research*, 2025 [212]

- In Section 5.1, we propose **ADEPT: Adaptive Data ExPloiTation**, which adaptively controls the utilization of the sampled data across different tasks and learning stages. ADEPT formulates the scheduling process as a multi-armed bandit problem, where a set of extent values corresponds to the arms. ADEPT can automatically select the optimal extent value based on the estimated task return, thereby significantly maximizing data efficiency while reducing computational costs. This work is **submitted to**:

Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. Adaptive Data Exploitation in Deep Reinforcement Learning. *International Conference on Machine Learning*, 2026 [215]

- In Section 5.2, we propose **ULTHO: Ultra-Lightweight Hyperparameter Optimization**, a general, simple, yet powerful framework for achieving efficient and robust HPO in deep RL within single runs. Unlike previous methods that focus on short-term improvements or require complex learning processes, ULTHO performs HPO from a hierarchical bandit perspective, achieving long-term performance gains with minimal computational overhead. This work is **published in**:

Mingqi Yuan, Bo Li, Xin Jin, Wenjun Zeng. ULTHO: Ultra-Lightweight yet Efficient Hyperparameter Optimization in Deep Reinforcement Learning. *IEEE/CVF International Conference on Computer Vision*, 2025 [215]

- In Section 5.3, we propose **PvP**, a **P**roprioceptive-**P**rivileged contrastive learning framework that leverages the intrinsic complementarity between proprioceptive and privileged states. PvP learns compact and task-relevant latent representations without requiring hand-crafted data augmentations, enabling faster and more stable policy learning. To support systematic evaluation, we develop **SRL4Humanoid**, the first unified and modular framework that provides high-

quality implementations of representative SRL methods for humanoid robot learning. This work is **published in**:

Mingqi Yuan, Tao Yu, Haolin Song, Bo Li, Xin Jin, Hua Chen, Wenjun Zeng. PvP: Data-Efficient Humanoid Robot Learning with Proprioceptive-Privileged Contrastive Representations. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 presents a comprehensive review of the acceleration research in RL.
- Chapter 3 provides essential background information for this thesis, including the mathematical definition of RL, representative RL algorithms, and intrinsic reward shaping, etc.
- Chapter 4 presents our contributions to the intrinsically-motivated exploration in RL, including a computation-efficient intrinsic reward, an intrinsic reward scheduling strategy, and a benchmark for intrinsic reward research.
- Chapter 5 presents our contributions to the task-aware data exploitation in RL, including an adaptive data exploitation scheduling algorithm, an ultra-lightweight yet efficient HPO algorithm, and a novel SRL approach designed for humanoid robot learning.
- Chapter 6 concludes the entire thesis and identifies two future directions.

Chapter 2

Literature Review

In this chapter, we conduct a comprehensive review of prior research on RL acceleration. Specifically, we discuss existing studies from three perspectives: exploration enhancement for RL acceleration, which is detailed in Section 2.1; exploitation enhancement for RL acceleration, which is detailed in Section 2.2; and engineering-level enhancement for RL acceleration, which is detailed in Section 2.3.

2.1 Exploration Enhancement for RL Acceleration

2.1.1 Classic Exploration Strategies

To enhance exploration in RL, a wide range of strategies have been proposed from different perspectives, including uncertainty-based exploration, probability matching, and entropy-based methods. For instance, ϵ -greedy is a simple approach that selects the action with the highest estimated value with probability $1 - \epsilon$ and chooses a random action with probability ϵ , thereby providing a simple mechanism for balancing exploration and exploitation [189]. Similarly, Boltzmann exploration samples actions based on their estimated value and employs a temperature parameter that controls the

trade-off between exploration and exploitation [28]. In contrast, the upper confidence bound (UCB) strategy explicitly accounts for uncertainty in value estimates, favoring actions with higher uncertainty to ensure sufficient exploration in the early learning period [8]. Similarly, Thompson sampling builds upon the insight of probability matching, selecting actions by sampling from a posterior distribution over action values [29]. This approach encourages exploration in high-uncertainty areas by selecting each action with a probability proportional to the (posterior) belief that it is optimal, thereby asymptotically optimizing long-term performance. Additionally, in deep RL, entropy-based regularization has also been widely adopted to promote exploration by encouraging higher stochasticity in policy outputs [64, 65], thereby ensuring sufficient exploration before converging to a deterministic policy. These methods offer various approaches to improve exploration and are prone to being effective in tabular settings. However, they are consistently inefficient in environments with high-dimensional observations and sparse rewards [10], especially for hard-exploration tasks.

Noise-based exploration constitutes another representative and widely used approach in RL for promoting effective exploration of the environment. By injecting controlled noise into various components of the learning process, including the actions, observations, rewards, value functions, and model parameters, this approach enhances sample efficiency and generalization across diverse tasks. For instance, the DDPG [111] algorithm introduces the Ornstein-Uhlenbeck process to inject temporally correlated noise into the continuous action output, thereby improving exploration efficiency in physical control problems with inertia. Furthermore, [158] proposes a generalized state-dependent exploration (gSDE) mechanism, which shifts the noise injection from the action space to the policy parameter space. This method generates temporally consistent and state-dependent noise throughout an episode, resulting in smoother exploration trajectories that are crucial for safe and efficient learning on real-world robotic systems.

In addition to perturbing actions to enhance exploration, several approaches introduce

stochasticity into policy parameters or environment observations to further enhance exploration and robustness. For example, [57] proposes NoisyNet, which injects learnable parametric noise directly into the policy network weights, allowing the agent to adaptively control exploration intensity through gradient descent. This mechanism induces consistent, state-dependent policy variations and has demonstrated strong performance on the Atari benchmark. Similarly, [154] suggests perturbing policy parameters at the beginning of each episode and employs an adaptive scaling mechanism that dynamically adjusts the noise magnitude to maintain a meaningful level of perturbation in the action space. This approach ensures temporally consistent exploration throughout the rollout and can solve sparse-reward tasks where action noise fails. In contrast, [131] proposes a more structured approach entitled active domain randomization (ADR), which actively learns a sampling strategy over environment parameters. ADR employs a discriminator-based reward to prioritize environment variations that maximize the discrepancy between policy rollouts in reference and randomized instances, thereby concentrating training on the most challenging scenarios and resulting in more robust policies and superior sim-to-real transfer compared to uniform domain randomization. Collectively, these methods illustrate a progression from injecting simple noise to adaptively learning the noise distribution, significantly enhancing exploration efficiency and generalization in complex RL tasks.

2.1.2 Intrinsic Reward-Driven Exploration

Intrinsic reward-based approaches enhance exploration by introducing auxiliary reward signals that motivate the agent to seek out novel, informative, or learnable states, beyond the extrinsic rewards provided by the environment. This mechanism addresses the hard exploration problem in sparse-reward settings by ensuring the agent receives consistent feedback for exploratory behaviors. In the following, we provide an overview of existing formulations for intrinsic rewards of different natures and, incidentally, introduce the methods included in our **RLeXplore** benchmark.

Count-Based Exploration. Count-based exploration methods provide intrinsic rewards by measuring the novelty of states, defined to be inversely proportional to the state visitation counts [184, 190, 124, 87]. In finite state spaces, count-based methods perform near optimally [184]. For this reason, these methods have been established as appealing techniques for driving structured exploration in RL. However, they do not scale well to high-dimensional state spaces [15, 118]. Pseudo-counts provide a framework to generalize count-based methods to high-dimensional and partially observed environments [15, 143, 128]. [25] proposed random network distillation (RND), which uses the prediction error against a fixed network as a learning signal that is correlated to counts. Recently, [71] proposed E3B and showed that the intrinsic objective provides a generalization of counts to high-dimensional spaces. In **RLeXplore**, we include Pseudo-counts, RND, and E3B as representatives of the state-of-the-art count-based methods.

Curiosity-Driven Exploration. Curiosity-based objectives train agents to interact with the environment seeking to experience outcomes that are not aligned with the agents’ predictions [7]. Hence, curiosity-driven exploration usually involves training an agent to increase its knowledge about the environment (*e.g.*, environment dynamics) [182, 149, 211]. The intrinsic curiosity module (ICM) [149, 23] learns a joint embedding space with inverse and forward dynamics losses and was the first curiosity-based method successfully applied to deep RL settings. Disagreement [150] further extended ICM by using the variance over an ensemble of forward-dynamics models to compute curiosity. However, curiosity-driven methods are consistently found to be unsuccessful when the environment has irreducible noise [167]. To address the problem, [161] proposed RIDE, which uses the difference between two consecutive state embeddings as the intrinsic reward and encourages the agent to choose actions that result in significant state changes. In general, curiosity-based objectives remain amongst the most popular intrinsic rewards in deep RL applications to this day. In **RLeXplore**, we include ICM, Disagreement, and RIDE as representatives of the

state-of-the-art curiosity-driven methods.

Global and Episodic Exploration. Towards more general and adaptive agents, recent works have studied decision-making problems in contextual Markov decision processes (MDPs) (*e.g.*, procedurally-generated environments) [161, 71, 129]. Contextual MDPs require episodic-level exploration, where novelty estimates are reset at the beginning of each episode. [70] showed that both global and episodic exploration modalities have unique benefits and proposed combined objectives that achieve remarkable performance across many MDPs of different structures. NGU [9] and RIDE [161] also instantiate both global and episodic bonuses. Inspired by this recent line of work, in this paper, we study novel combinations of objectives for exploration that achieve impressive results in contextual MDPs.

2.1.3 LLM-Empowered Intrinsic Reward Generation

The technological revolution of large language models has significantly impacted all areas of AI research. Recent work has explored empowering LLMs to generate intrinsic rewards, progressively advancing from offline semantic guidance to scalable online learning systems. For instance, [20] introduces an LLM approach that leverages frozen LLMs to generate language-driven intrinsic rewards that encourage diverse yet human-meaningful behaviors, demonstrating improved sample efficiency over novelty-based curiosity methods in sparse-reward environments. Similarly, [48] proposes ELLM, which uses LLM-generated and context-sensitive goals to guide reward-free pretraining, thereby biasing exploration toward semantically meaningful behaviors and enhancing downstream task performance. Moving toward automated reward design, [165] introduces ProgressCounts, where LLMs generate task-specific progress functions that map high-dimensional states to coarse measures of task completion. These progress values are then discretized and used within a count-based intrinsic reward framework, where agents are rewarded for exploring novel states within this

simplified progress space. Meanwhile, [155] demonstrates that LLM-based guidance can complement classical intrinsic motivation by combining LLM-derived rewards with VAE-based state novelty, resulting in improved stability and performance in sparse-reward grid-world tasks. Finally, [227] proposes ONI, a distributed framework that learns intrinsic rewards from asynchronous LLM feedback in an online manner, distilling sparse LLM annotations into a learned reward model and enabling scalable, high-throughput RL without relying on pre-collected offline datasets [227]. Collectively, these recent advances demonstrate that intrinsic reward constitutes a sustainable mechanism for exploration, whose effectiveness can be further enhanced by integrating it with more powerful auxiliary models, such as LLMs.

2.2 Exploitation Enhancement for RL Acceleration

2.2.1 Data Exploitation Scheduling

In RL, data exploitation scheduling involves the systematic management of how collected experiences are utilized during training. This includes deciding which samples to replay, when to update the model, and balancing the importance of past data with new interactions. The overall objective is to maximize the data utility to accelerate convergence and improve sample efficiency.

A representative line of work in this area focuses on designing efficient experience replay mechanisms [168, 5, 121, 139, 82, 56, 75, 114]. For instance, prioritized experience replay [168] prioritizes transitions based on their temporal-difference error so that more informative or surprising experiences are sampled more frequently. Hindsight experience replay [5] augments trajectories with alternative goals, enabling learning even under sparse and binary rewards by reusing past experiences with different

hindsight goals. Distributed prioritized experience replay [75] scales this idea by decoupling acting and learning, where multiple actors collect data in parallel, while a central learner replays prioritized samples, thereby improving throughput and sample utilization. Finally, competitive experience replay [114] introduces a competitive dynamic between multiple agents, utilizing paired trajectories to drive exploration and better exploit sparse-reward environments.

In addition, recent research has focused on optimizing the update-to-data (UTD) ratio, which is defined as the number of gradient-based updates performed per environment step. For example, [32] introduced REDQ, a model-free algorithm that achieves state-of-the-art performance in continuous control tasks by combining high UTD ratios, Q-function ensembles, and in-target minimization. Meanwhile, [108] identified high validation TD error as a key bottleneck in sample-efficient deep RL and proposed AVTD, which dynamically adjusts regularization strategies to mitigate overfitting. Furthermore, [47] developed DUTD, a method that automatically tunes the UTD ratio in model-based RL by detecting underfitting and overfitting on a validation set, thereby eliminating the need for manual hyperparameter tuning while enhancing robustness. Furthermore, the recently proposed SPEQ [162] alternates between an online low-UTD phase and an offline stabilization phase. During the stabilization phase, Q-functions are fine-tuned on the replay buffer without additional interactions with the environment, preserving sample efficiency while reducing computational cost compared to standard high-UTD methods.

2.2.2 Data Augmentation Techniques

Data augmentation is a general yet effective technique across all areas of machine learning research. In RL, data augmentation is also leveraged to expand the training dataset, which can significantly improve sample efficiency and generalization of agents [105, 122]. For instance, [37] quantifies the impact of generalization in RL

and suggests that augmentations can reduce overfitting, thereby improving generalization capabilities. Building on this, [207] proposes a data-regularized Q (DrQ) algorithm, demonstrating that regularizing deep RL from pixel observations with image augmentations improves generalization and sample efficiency. Similarly, [101] also introduces augmented data as a regularization technique and develops a series of data augmentation operations for both pixel-based and state-based inputs. [206] further extends these methods and proposes DrQ-v2, which integrates data augmentation, exploration noise scheduling, and fast experience replay with a DDPG backbone algorithm. DrQ-v2 is the first model-free method that solves complex humanoid tasks directly from pixels.

In addition to visual augmentations, other methods for improving RL generalization include automatic data augmentation techniques that dynamically adjust augmentations during training to improve the model’s robustness across different data distributions. For instance, [160] proposes automatic data augmentation for RL, selecting augmentations based on the agent’s performance to reduce overfitting and improve generalization. Similarly, [67] introduces the concept of soft data augmentation, which regularizes RL models by applying stochastic perturbations to the state and action spaces, thereby facilitating better generalization across varied environments. [123] investigates the key attributes of data augmentation that account for its effectiveness, and proposes a novel cycling augmentation mechanism. This approach performs periodic cycles of different data augmentation operations to increase type diversity and maintain data distribution consistency, thereby achieving superior sample efficiency compared to prior methods.

2.2.3 State Representation Learning

State representation learning (SRL) has been widely applied in RL to enhance learning efficiency and generalization by extracting compact and informative latent represen-

tations from high-dimensional observations. Existing SRL approaches can be broadly categorized into three types: reconstruction-based methods [90, 73, 210, 50, 49], dynamics modeling-based methods [149, 83, 96, 174, 130], and contrastive learning-based methods [183, 209, 98, 99, 225]. For instance, dynamics modeling methods encourage representations that capture environment dynamics through predictive modeling. Forward models predict future states from current state–action pairs, while inverse models infer actions from transitions [149, 83, 96], thereby providing rich controllable features. In contrast, contrastive learning methods structure latent space by enforcing similarity between positive pairs (*e.g.*, temporally adjacent states) while separating negative pairs [142]. This yields invariant and temporally smooth representations that improve RL efficiency and generalization. For instance, CURL [98] leverages augmented image pairs to enforce invariances, while ATC [183] aligns embeddings of temporally close observations. Extensions such as CDPC [225] refine contrastive predictive coding with temporal-difference objectives to stabilize training in stochastic environments.

SRL holds great promise for enhancing humanoid robot learning by enabling efficient encoding and processing of complex sensory information, with pioneering work exploring its application in improving robot adaptability and performance in dynamic environments [119, 186, 187, 224, 226]. For example, [224] proposes an Any2Track framework that leverages SRL to enhance motion tracking by integrating a history-informed adaptation module. This module utilizes dynamics-aware world model prediction to extract informative dynamics features, enabling the robot to adapt to various disturbances, such as terrain, external forces, and changes in physical properties. Similarly, [187] proposed a world model reconstruction framework that uses sensor denoising and world state estimation to improve locomotion in unpredictable environments. These approaches highlight the potential of SRL to enhance the performance of humanoid robots, yet further exploration is needed to optimize and generalize these methods across diverse control tasks and environments.

2.3 Engineering-Oriented RL Acceleration

2.3.1 Algorithm Implementation Acceleration

In addition to advancements on algorithm-level acceleration, the practical engineering implementation of these algorithms is also critical. Even well-designed algorithms can fail to deliver their theoretical advantages if there are implementation inefficiencies, poor software abstractions, or underutilized hardware, which can increase training time. Consequently, there has been a growing focus on developing high-performance, reproducible, and developer-friendly RL frameworks that substantially reduce experimentation overhead and wall-clock training time.

Several open-source projects have made significant contributions to this area, providing optimized and user-friendly frameworks for RL research and deployment. For example, Stable-Baselines3 [156] offers a comprehensive library of state-of-the-art RL algorithms, designed for ease of use and efficient experimentation. Tianshou [202] focuses on high performance and scalability, supporting distributed training and multi-agent environments with a modular design. Elegant RL [115] and CleanRL [80] prioritize simplicity and efficiency, enabling researchers to quickly test RL algorithms without unnecessary complexity. PureJaxRL [120] takes advantage of JAX for GPU-accelerated RL, facilitating the rapid development of custom algorithms. Finally, RSL-RL [173] is specifically designed for robotics, featuring a compact and easily modifiable codebase optimized for pure GPU training to achieve high throughput in large-scale simulations.

2.3.2 Environment Interaction Acceleration

In addition to algorithm implementation, the speed of environment simulation constitutes a major bottleneck in RL, particularly for data-hungry methods such as large-

scale robotic training. To address this challenge, a new generation of environment engines and simulators has been developed with a focus on parallelism, hardware acceleration, and reduced CPU–GPU communication overhead.

Sample Factory [153] optimizes asynchronous policy gradients to achieve tens to hundreds of thousands of frames per second on a single GPU, enabling billion-transition training without a large cluster. For environments traditionally executed on CPUs, EnvPool [203] provides a parallelized C++ executor that achieves massive speedups on tasks such as Atari and MuJoCo. In contrast, JAX-based frameworks like Gymnax [97] utilize just-in-time compilation and vectorization to run environments directly on accelerators, thereby eliminating CPU-GPU communication overhead. This paradigm is further extended by Craftax [129], a complex, open-ended benchmark that runs orders of magnitude faster than its Python predecessor, allowing a billion interactions in under an hour. For robotics, GPU-accelerated simulators such as IsaacGym [125] and IsaacLab [140] provide high-fidelity physics for continuous control tasks. Collectively, these frameworks significantly reduce the wall-clock time for environment interaction, enabling faster iteration and large-scale training, which are essential for modern RL research in robotics and beyond.

Chapter 3

Background

In this chapter, we provide essential background information to support the subsequent methodological discussion in this thesis. Specifically, we begin by introducing the formal definition of RL, followed by specific research directions, such as intrinsic reward shaping and hyperparameter optimization in RL, etc.

3.1 Reinforcement Learning

In this thesis, we frame the RL problem as a partially observable Markov decision process (POMDP) [17, 88], which can be defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, P, \Omega, R, \gamma)$. Here, $\mathcal{S}$ is the state space, $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ is the transition function that defines a probability distribution over $\mathcal{S}$, $\Omega : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{O})$ is the observation function, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the extrinsic reward function, and $\gamma \in [0, 1]$ is a discount factor. The goal of RL is to learn a parameterized policy $\pi_{\theta}(\mathbf{a}|\mathbf{s})$ to maximize the expected discounted return:

$$J_{\pi}(\theta) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t) \right]. \quad (3.1)$$

3.2 Multi-Armed Bandit

The multi-armed bandit (MAB) problem is a classic example of the exploration–exploitation dilemma in RL and decision-making under uncertainty. Formally, it can be defined as follows: Let there be K arms (actions), each associated with an unknown reward distribution. At each time step $t = 1, 2, \dots, T$, the agent selects an arm from $\{1, 2, \dots, K\}$ and receives a stochastic reward R_t drawn from the distribution corresponding to the arm. Let μ_i denote the expected reward of i -arm, and let $\mu^* = \max_i \mu_i$ be the maximum expected reward. The goal of the agent is to maximize the cumulative reward over T steps, or equivalently, to minimize the cumulative regret, defined as:

$$\text{Regret}(T) = T\mu^* - \sum_{t=1}^T \mathbb{E}[R_t] \quad (3.2)$$

The regret measures the total loss incurred by not always selecting the optimal arm. In subsequent methods, we will integrate MAB as a subsystem into the RL framework to support exploration and exploitation scheduling.

3.3 Representative MAB Algorithms

To address the MAB problem, we introduce two critical algorithms, namely the upper confidence bound (UCB) [8] and Thompson sampling (TS) [193]. These two algorithms will also be employed in our subsequent methodologies.

The ϵ -greedy method selects non-greedy actions with equal probability, regardless of their estimated returns derived from the past attempts. However, it is more reasonable to choose actions based on their estimated returns by taking into account both their expected returns and the uncertainty associated with these estimates. Inspired

by this observation, the UCB algorithm selects actions by [8]

$$k_{\text{UCB}} = \operatorname{argmax}_i \left[Q(i) + c \sqrt{\frac{\ln t}{N_t(i)}} \right], \quad (3.3)$$

where $N_t(i)$ is the number of times that the i -th arm has been selected prior to time t , and $c > 0$ controls the degree of exploration. The second term measures the uncertainty in the return estimation of the i -th arm. The uncertainty term decreases when $N_t(i)$ increases. On the contrary, if the i -th arm is not selected but t increases, the uncertainty estimate increases. As a result, all arms will eventually be selected with arms of either low estimated return or high selection frequency being selected with decaying frequency.

In TS, the agent tracks a belief over the probability distribution of the optimal arms and samples from this distribution, which is shown to solve the MAB problem more effectively [43]. More specifically, TS assumes that $Q(i)$ follows a Beta distribution that is a family of continuous probability distributions defined on the interval $[0, 1]$ parameterized by two positive shape parameters, namely α and β . At each time step t , TS samples an expected reward $\tilde{Q}(i)$ from the prior distribution $\text{Beta}(\alpha_i, \beta_i)$ for every arm selection. After that, the action is chosen by

$$k_{\text{TS}} = \operatorname{argmax}_i \tilde{Q}(i). \quad (3.4)$$

After the true reward is observed, the Beta distribution is updated following

$$\begin{aligned} \alpha_i &\leftarrow \alpha_i + R(k) \mathbb{1}[k_{\text{TS}} = i], \\ \beta_i &\leftarrow \beta_i + (1 - R(k)) \mathbb{1}[k_{\text{TS}} = i], \end{aligned} \quad (3.5)$$

where $\mathbb{1}$ is the indicator function. Since TS samples reward estimates from prior distributions and the reward probability for each action is currently considered optimal, it effectively implements the idea of probability matching.

3.4 Intrinsic Reward Shaping

Reward shaping is a very powerful technique for scaling up RL methods to handle complex problems by supplying additional rewards to the agent to guide its learning process. The most well-known reward shaping method is potential-based reward shaping (PBRS) [138], which provides a theoretical guarantee of policy invariance.

Formally, let $\Phi : \mathcal{S} \rightarrow \mathbb{R}$ be a potential function defined over states. The shaping reward F induced by PBRS is defined as

$$F(\mathbf{s}_t, \mathbf{s}_{t+1}) = \gamma\Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t), \quad (3.6)$$

where $\gamma \in [0, 1]$ is the discount factor.

In contrast, intrinsic reward shaping does not strictly adhere to this definition. Following prior work in intrinsically-motivated RL [25], we consider an augmented reward function that combines both extrinsic and intrinsic components. The overall reward signal at time step t is defined as:

$$R_t^{\text{total}} = R_t + \beta_t \cdot I_t, \quad (3.7)$$

where $I : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the intrinsic reward, $\beta_t = \beta_0(1 - \kappa)^t$ is the exploration coefficient controlling the relative weight of intrinsic motivation over time, and κ is a decay rate. This formulation enables the agent to balance task-specific objectives with exploratory behavior, particularly in sparse or deceptive reward settings. Finally, the augmented optimization objective is:

$$J_\pi(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t (R_t + \beta_t \cdot I_t) \right]. \quad (3.8)$$

3.5 Representative RL Algorithms

We introduce two representative RL algorithms: proximal policy optimization (PPO) and soft actor-critic (SAC), which will be widely utilized in our experiments.

PPO [171] is an on-policy algorithm that is designed to improve the stability and sample efficiency of policy gradient methods, which uses a clipped surrogate objective function to avoid large policy updates.

The policy loss is defined as:

$$L_{\pi}(\boldsymbol{\theta}) = -\mathbb{E}_{\tau \sim \pi} [\min(\rho_t(\boldsymbol{\theta}) A_t, \text{clip}(\rho_t(\boldsymbol{\theta}), 1 - \epsilon, 1 + \epsilon) A_t)], \quad (3.9)$$

where

$$\rho_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)}, \quad (3.10)$$

A_t is the advantage function, and ϵ is a clipping range coefficient.

Meanwhile, the value network is trained to minimize the error between the predicted return and a target of discounted returns computed with generalized advantage estimation (GAE) [170]:

$$L_V(\phi) = \mathbb{E}_{\tau \sim \pi} \left[(V_{\phi}(\mathbf{s}) - V_t^{\text{target}})^2 \right]. \quad (3.11)$$

SAC [65] is an off-policy actor-critic algorithm that maximizes a maximum-entropy objective, encouraging efficient exploration while maintaining high sample efficiency. The objective function is defined as

$$J_{\pi}(\boldsymbol{\theta}) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t (R(\mathbf{s}_t, \mathbf{a}_t) + \alpha H(\pi(\cdot | \mathbf{s}_t))) \right], \quad (3.12)$$

where $H(\pi(\cdot | \mathbf{s}_t))$ denotes the policy entropy and α is a temperature parameter.

SAC employs twin Q-functions and a replay buffer to enable stable off-policy learning. The critic is trained using the soft Bellman backup, while the actor is updated by minimizing

$$L_{\pi}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_{\boldsymbol{\theta}}} [\alpha \log \pi_{\boldsymbol{\theta}}(\mathbf{a} | \mathbf{s}) - Q(\mathbf{s}, \mathbf{a})]. \quad (3.13)$$

3.6 Hyperparameter Optimization in RL

Hyperparameter optimization (HPO) in RL aims to find the optimal set of HPs that maximizes the expected return of the agent’s policy [59, 52]. In this context, HPs refer to parameters that are not directly learned by the RL algorithm but influence the learning process, such as learning rates, exploration parameters, and network architecture choices. The objective of HPO in RL can be formulated as solving the following black-box optimization problem:

$$\boldsymbol{\psi}^* = \operatorname{argmax}_{\boldsymbol{\psi} \in \boldsymbol{\Psi}} J_{\pi}(\boldsymbol{\theta}, \boldsymbol{\psi}, \mathbf{A}), \quad (3.14)$$

where $\boldsymbol{\psi}^*$ is the optimal configuration, $\boldsymbol{\Psi}$ is the HP space, and $\mathbf{A}$ is an RL algorithm.

In Section 5.2, we propose an ultra-lightweight yet efficient algorithm that performs HPO on the fly, which significantly improves the searching efficiency and reduces computational overhead.

3.7 Humanoid Whole-Body Control

In this thesis, we aim to examine the proposed methodology in practical problem-solving, and humanoid whole-body control (WBC) is selected as the test benchmark to align with the emerging trend of embodied intelligence.

WBC serves as the foundation for enabling humanoid robots to execute diverse and complex tasks. Formally, given a set of continuous commands $\mathcal{C}$ and observations $\mathcal{O}$, the objective is to design a control function that maps $(\mathcal{O}, \mathcal{C})$ to appropriate control signals. Equipped with learning-based methods, it is feasible to directly learn a parameterized policy $\pi_{\boldsymbol{\theta}} : \mathcal{O} \times \mathcal{C} \rightarrow \mathcal{A}$ that outputs joint actions [178]. In practice, actions are often defined as offsets to nominal joint positions for the upper body, lower body, and hands. The final joint reference is obtained by adding these offsets to nominal targets, which are then tracked using a proportional-derivative (PD) controller

with fixed gains.

RL offers a data-driven framework for optimizing humanoid WBC policies through interaction with the environment. Correspondingly, humanoid WBC provides a challenging yet realistic benchmark for evaluating the learning efficiency of RL, as effective whole-body coordination requires both rapid exploration and efficient exploitation of high-dimensional control data.

Chapter 4

Intrinsically-Motivated Exploration for RL Acceleration

4.1 Rewarding Visitation Discrepancy for Efficient Exploration in RL

4.1.1 Introduction

Efficient and complete exploration remains a significant challenge in deep RL. Inspired by human learning patterns, recent approaches propose leveraging intrinsic motivation to enhance exploration, particularly in real-world settings where the agent receives sparse or missing rewards [169, 145, 144]. Intrinsic rewards encourage the agent to visit novel states [15, 143, 190, 25] or to increase its knowledge and prediction accuracy of the environment dynamics [182, 149, 211, 76, 141]. For instance, [143, 15] leveraged a density model to evaluate state novelty, and the intrinsic rewards were defined to be inversely proportional to the visiting frequencies. As a result, the agent is motivated to revisit the infrequently seen states and achieve comprehensive exploration.

In contrast, [182, 149, 211] designed an auxiliary model to learn the environment’s dynamics, utilizing the prediction error as the intrinsic reward. Furthermore, [24] performed experiments on Atari games only using the prediction-based intrinsic rewards, demonstrating that the agent could also achieve remarkable performance.

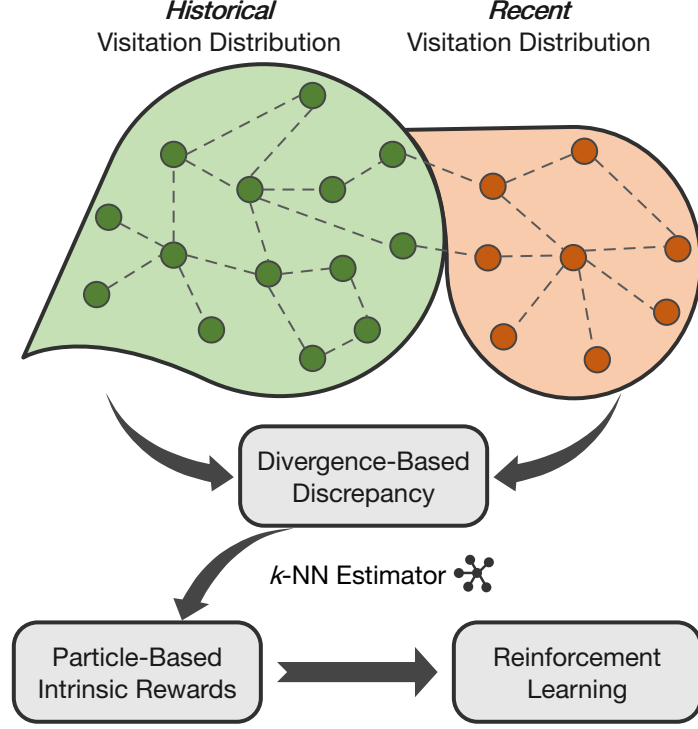


Figure 4.1: REVD evaluates the discrepancy of visitation distributions in different learning periods and transforms it into intrinsic rewards, providing powerful and sustainable exploration incentives.

However, the aforementioned methods cannot provide sustainable exploration incentives, as the exploration bonus diminishes as the agent becomes familiar with the environment. As a result, the agent may miss the downstream learning opportunities and fail to learn in hard-exploration tasks. To address the problem, [9] proposed a NGU framework that produces intrinsic rewards composed of episodic and long-term state novelty. The episodic state novelty is evaluated using an episodic memory and pseudo-count method, encouraging the agent to visit diverse states in each episode.

Since the memory is erased at the beginning of each episode, the intrinsic reward will not vanish across episodes. In contrast, [161] proposed a more straightforward method, entitled RIDE, which calculates the difference between two consecutive encoded states as the intrinsic reward, encouraging the agent to choose actions that result in significant state changes. However, NGU and RIDE are expensive and unstable because they rely heavily on auxiliary models. Poor representation learning may incur severe performance loss, making it difficult to apply to arbitrary tasks.

In this section, we propose a novel intrinsic reward method entitled **REVD: REwarding Visitation Discrepancy**, which provides powerful exploration incentives with a simple, computation-efficient, and robust architecture. Our main contributions are summarized as follows:

- We propose to maximize the divergence-based visitation discrepancy to encourage exploration in RL, which is transformed into intrinsic rewards using an efficient estimator. Our method can be directly integrated with arbitrary RL algorithms, significantly promoting the sample efficiency with a simple and robust architecture.
- REVD is computation-efficient, as it requires neither representation learning nor additional models. This property also enables it to overcome the vanishing intrinsic rewards problem, offering sustainable exploration incentives throughout the learning process.
- Extensive experiments are performed to compare the performance of REVD against existing methods on the MiniGrid benchmark. Simulation results demonstrate that REVD achieves remarkable performance and outperforms the algorithmic baselines.

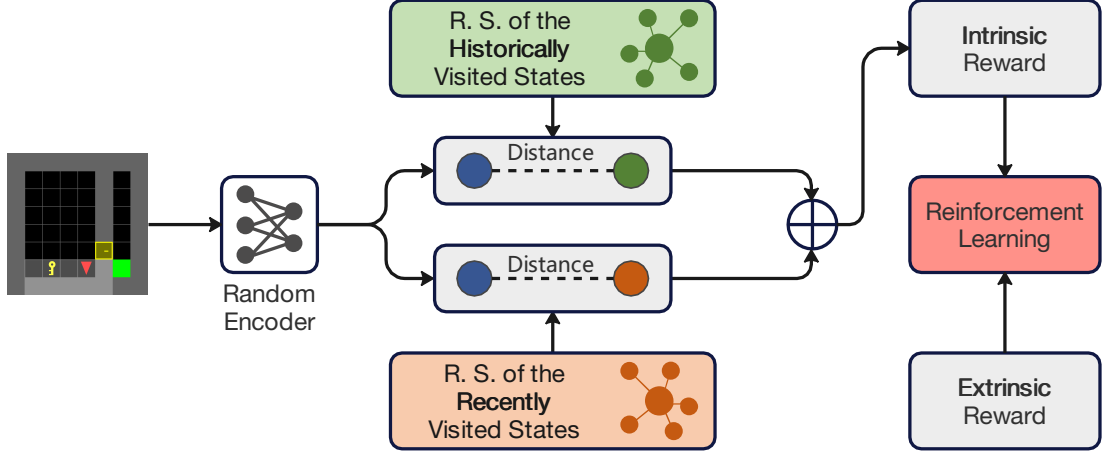


Figure 4.2: Workflow of the REVD. The state is first encoded by a fixed, randomly initialized encoder before being compared with the historically and recently visited states. The intrinsic reward is computed as the summation of the distances to its k -NN in the two representation spaces (R. S.).

4.1.2 Methodology

In this section, we present the **REVD: REwarding Visitation Discrepancy**, a computation-efficient framework for providing powerful and sustainable exploration incentives in RL. The key insight of REVD is to encourage exploration by providing intrinsic rewards based on the agent’s visitation discrepancy across different learning periods. To that end, we introduce the Kullback–Leibler (KL) divergence [40] to formally evaluate the discrepancy.

Let $X \in \mathbb{R}^d$ be a random vector that has two density functions $p(\mathbf{x})$, $q(\mathbf{x})$ with respect to Lebesgue measure on $\mathbb{R}^d$, and let $\{\mathbf{x} \in \mathbb{R}^d : p(\mathbf{x}), q(\mathbf{x}) > 0\}$ be the support set. The KL divergence is defined as

$$D_{\text{KL}}(p||q) = \int_{\mathcal{X}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \quad (4.1)$$

Denote by $\check{\rho}(\mathbf{s})$ and $\hat{\rho}(\mathbf{s})$ the historical and recent visitation distributions, the visita-

tion discrepancy can be evaluated as

$$D_{\text{KL}}(\hat{\rho}(\mathbf{s}) \parallel \check{\rho}(\mathbf{s})) = \int_{\mathcal{S}} \hat{\rho}(\mathbf{s}) \log \frac{\hat{\rho}(\mathbf{s})}{\check{\rho}(\mathbf{s})} d\mathbf{s}. \quad (4.2)$$

In the exploration phase of RL, the agent is expected to visit as many distinct states as possible within a limited number of iterations. It is feasible to encourage exploration by maximizing the divergence above. However, it is non-trivial to evaluate and optimize this objective directly when $\check{\rho}$ and $\hat{\rho}$ are unavailable. To circumvent this problem, we introduce the following k -nearest neighbor (k -NN) estimator and transform Eq. (4.2) into computable intrinsic rewards before jointly optimizing the policy with task rewards. Let $\mathcal{X} = \{X_i\}_{i=1}^N$ be i.i.d samples from the distribution with density p and $\mathcal{Y} = \{Y_i\}_{i=1}^M$ be i.i.d samples from the distribution with density q , we estimate the KL divergence as follows [201, 152]:

$$\hat{D}_{\text{KL}}^{k,N,M}(p \parallel q) = \frac{d}{N} \sum_{i=1}^N \log \frac{\nu(X_i, \mathcal{Y}, k)}{\nu(X_i, \mathcal{X}, k)} + \log \frac{M}{N-1}, \quad (4.3)$$

where $\nu(X_i, \mathcal{X}, k)$ is the Euclidean distance between X_i and its k -th nearest neighbor in $\mathcal{X}$, $\nu(X_i, \mathcal{Y}, k)$ is the Euclidean distance between X_i and its k -th nearest neighbor in $\mathcal{Y}$.

Denote by $\check{\mathcal{E}} = \{\check{\mathbf{s}}_i\}_{i=1}^M$ and $\hat{\mathcal{E}} = \{\hat{\mathbf{s}}_i\}_{i=1}^N$ the sets of the historically visited states and recently visited states, the Eq. (4.2) can be approximated using Eq. (4.3) as

$$\hat{D}_{\text{KL}}^{k,N,M}(\hat{\rho} \parallel \check{\rho}) = \frac{d}{N} \sum_{i=1}^N \log \frac{\nu(\mathbf{e}_i, \check{\mathcal{E}}, k)}{\nu(\mathbf{e}_i, \hat{\mathcal{E}}, k)} + \log \frac{M}{N-1}, \quad (4.4)$$

where $\mathbf{e}_i = f_{\phi}(\mathbf{s}_i)$ is an encoding vector produced by a fixed and randomly-initialized encoder, and $\check{\mathcal{E}}, \hat{\mathcal{E}}$ are the corresponding sets of encoding vectors.

It is natural to find that

$$\hat{D}_{\text{KL}}^{k,N,M}(\hat{\rho} \parallel \check{\rho}) \propto \sum_{i=1}^N \log \frac{\nu(\mathbf{e}_i, \check{\mathcal{E}}, k)}{\nu(\mathbf{e}_i, \hat{\mathcal{E}}, k)}. \quad (4.5)$$

Therefore, we define the intrinsic reward that takes each transition as a particle:

$$I(\mathbf{s}_i) = \log \frac{\nu(\mathbf{e}_i, \check{\mathcal{E}}, k)}{\nu(\mathbf{e}_i, \hat{\mathcal{E}}, k)}. \quad (4.6)$$

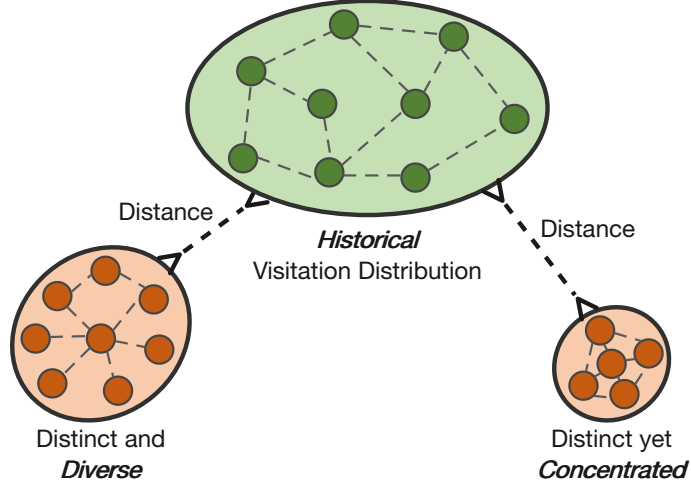


Figure 4.3: Comparison of different exploration statuses. The agent is expected to explore distinct and diverse states compared to the states it has previously visited.

However, it is risky to optimize this reward directly. As shown in Fig. 4.3, the agent is likely to wander in a distinct yet narrow area to get more rewards. The desirable exploration status is that the agent should explore more distinct and diverse states compared to the states it has historically visited. Mathematically, the agent is expected to maximize the divergence between the historical and recent visitation distribution, while maximizing the entropy of the recent visitation distribution.

Therefore, we consider reformulating the intrinsic reward as

$$I(\mathbf{s}_i) = \underbrace{\log (\nu(\mathbf{e}_i, \check{\mathcal{E}}, k) + 1)}_{\text{Outer Discrepancy}} + \alpha \cdot \underbrace{\log (\nu(\mathbf{e}_i, \hat{\mathcal{E}}, k) + 1)}_{\text{Intra Discrepancy}}, \quad (4.7)$$

where $0 < \alpha < 1$ is a weighting coefficient.

This formulation retains the divergence estimation while also accounting for the diversity of both long-term and short-term explorations.

Algorithm 1: REVD

```
1 Initialize the encoder  $f_\phi$  and policy network  $\pi_\theta$ ;
2 Initialize the buffer  $\mathcal{B}$ ;
3 for each episode do
4   Sample a rollout  $\{\mathbf{s}_0, \mathbf{a}_0, \dots, \mathbf{a}_{T-1}, \mathbf{s}_T\}$  using the policy;
5   Compute the encoding vectors of  $\mathbf{e}_t = f_\phi(\mathbf{s}_t)$ ;
6   for  $t = 0, \dots, T$  do
7     Compute the distance between  $\mathbf{e}_t$  and all elements in  $\mathcal{B}$  and find the
        $k$ -NN;
8     Compute the distance between  $\mathbf{e}_t$  and all elements in  $\{\mathbf{e}_t\}_{t=0}^T$  and find the
        $k$ -NN;
9     Compute the intrinsic reward  $I_t$  following Eq. (4.7);
10    Set  $R_t^{\text{total}} \leftarrow R_t + \lambda \cdot I_t$ ;
11  end
12  Update the policy network with transitions  $\{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, R_t^{\text{total}}\}_{t=0}^{T-1}$  using any
    RL algorithm
13  Add the rollout data the buffer  $\mathcal{B}$ ;
14 end
```

Equipped with the intrinsic reward, the total reward of each transition $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ is computed as

$$R_t^{\text{total}} = R(\mathbf{s}_t, \mathbf{a}_t) + \lambda_t \cdot I(\mathbf{s}_t), \quad (4.8)$$

where $\lambda_t = \lambda_0(1 - \kappa)^t \geq 0$ is a weighting coefficient that determines the exploration preference, where κ is a decay rate. In particular, this intrinsic reward can be utilized for unsupervised pre-training in the absence of extrinsic rewards. Then the pre-trained policy can be employed in the downstream task adaptation with extrinsic rewards. Letting π_θ denote the policy represented by a neural network with parameters θ , the training objective is to maximize the expected discounted return $\mathbb{E}_{\pi_\theta} [\sum_{k=0}^{\infty} \gamma^k R_k^{\text{total}}]$.

Finally, the detailed workflow of REVD is summarized in Algorithm 1.

4.1.3 Experiments

In this section, we design the experiments to answer the following questions:

- **Q1:** Can REVD improve the sample efficiency and policy performance of RL algorithms in environments with high-dimensional observations and sparse rewards?
- **Q2:** How does REVD compare to exploration approaches empowered by auxiliary models?
- **Q3:** How does REVD compare to other computation-efficient exploration approaches?
- **Q4:** How does the choice of k -value and buffer size affect the performance of REVD?

4.1.3.1 Experimental Setup

a) Benchmark Selection: As shown in Figure 4.4, we introduce the MiniGrid benchmark to test the effectiveness of REVD [35]. MiniGrid contains a collection of 2D grid-world environments with goal-oriented and sparse-reward tasks that are significantly challenging for exploration. Three environments are selected for our experiments, namely *DoorKey-6×6*, *DoorKey-8×8*, and *Empty-16×16*. In the *DoorKey* environment, the agent must collect a key and unlock a door before entering the room containing the goal. For the *Empty* environment, the agent must find the goal in the furthest corner of a large room. Notably, these environments are partially observable, as the agent only has access to a compact $7 \times 7 \times 3$ embedding of the 7×7 grid directly in front of it.

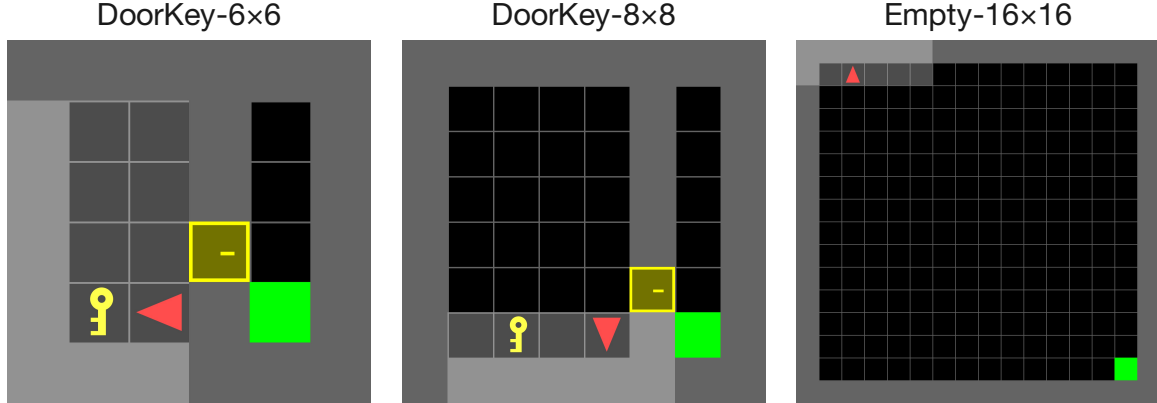


Figure 4.4: Screenshots of the three selected MiniGrid environments.

b) Algorithmic Baselines: For the backbone RL algorithm, we select the advantage actor-critic (A2C) [135], which operates across multiple parallel environments while synchronizing updates. For the baselines of intrinsic rewards, we select RE3 [177] and RND [25]. RE3 is also a computation-efficient approach that maximizes the entropy of the global state visitation distribution. RND is a prediction-based method that calculates the prediction error as the intrinsic rewards, which suffers from the issue of diminishing rewards. For RE3 and REVD, we maintain a buffer for them to store the historically visited states. Then we combine these intrinsic rewards with the A2C algorithm and compare their performance under the same number of environment steps. To make a fair comparison, the hyperparameters and network architectures of the A2C remain constant for all experiments, as shown in Table 4.1.

c) Evaluation Metrics: In the MiniGrid benchmark, the agent will continue to receive a reward of -1 unless the agent successfully reaches the goal. Therefore, the performance of the agent is evaluated by: (i) whether the agent can solve the environment, and (ii) the total environment steps consumed for problem solving. For each method, we run a hyperparameter search over the corresponding configurations and report the best results.

Table 4.1: Hyperparameters used for A2C in the MiniGrid experiments.

Hyperparameter	Value
Observation downsampling	(7, 7, 3)
Stacked frames	1
Environment steps	2400000 for DoorKey-8×8, otherwise 400000
Number of workers	16
Episode steps	5
Optimizer	RMSprop
Learning rate	0.001
GAE coefficient	0.95
Action entropy coefficient	0.01
Value loss coefficient	0.5
Max gradient norm	0.5
Value clipping coefficient	0.2
Batch size	256
Gamma γ	0.99

4.1.3.2 Result Analysis

The following analysis of results is conducted based on the predefined research questions.

a) Effectiveness of REVD: We first examine the effectiveness of REVD, *i.e.*, whether it can accelerate the problem-solving of the agent. As illustrated in Figure 4.5, REVD successfully reaches the goal with the fewest environment steps, while the vanilla A2C agent fails to solve the *DoorKey-8×8* environment. This indicates that REVD provides accurate exploration guidance for the agent, thereby enhancing overall learning efficiency.

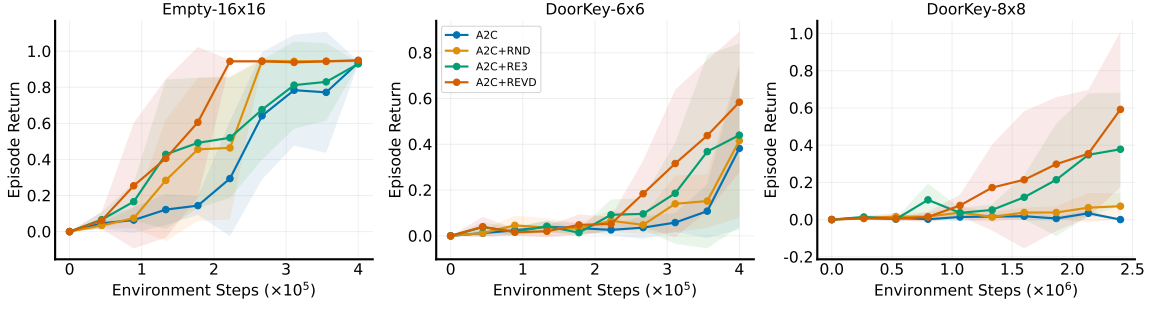


Figure 4.5: The performance comparison of the A2C agent and its combinations with three intrinsic rewards. The solid line and shaded regions represent the mean and standard deviation, respectively.

b) Comparison with Other Intrinsic Rewards: In contrast, RND slightly accelerates the learning process in *Empty-16 $\times$ 16* and *DoorKey-6 $\times$ 6*, yet also fails in the *DoorKey-8 $\times$ 8* environment. The problem of diminishing rewards becomes more significant as environmental complexity increases, while REVD can provide sustainable exploration incentives. As a result, REVD consistently outperforms the RND in all environments.

c) Impact of the k -Value Selection: The selection of the k -value directly affects the estimation accuracy of the divergence. A small k value leads to large variance, while a large k value leads to large bias. Therefore, we conduct an ablation study to compare the performance of REVD with different k values. As illustrated in Figure 4.6, $k = 3$ is the generally best option for all the environments, which effectively balances the estimation variance and bias.

d) Impact of the Buffer Size: Likewise, the buffer size affects the estimation quality of the historical visitation distribution. We also conduct an ablation study to compare the performance of REVD with different buffer sizes used for state archival. As illustrated in Figure 4.7, the buffer size of 10000 achieves the best performance in the *DoorKey-8 $\times$ 8* environment. This suggests that a larger buffer is necessary for

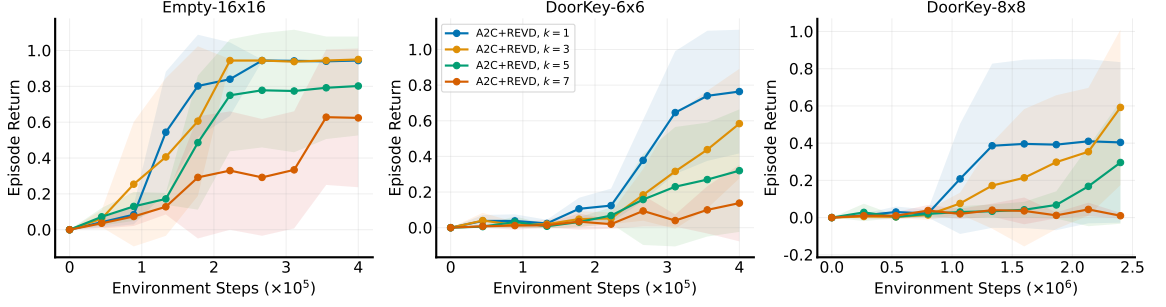


Figure 4.6: The performance comparison of REVD with different k -values used for intrinsic reward computation. The solid line and shaded regions represent the mean and standard deviation, respectively.

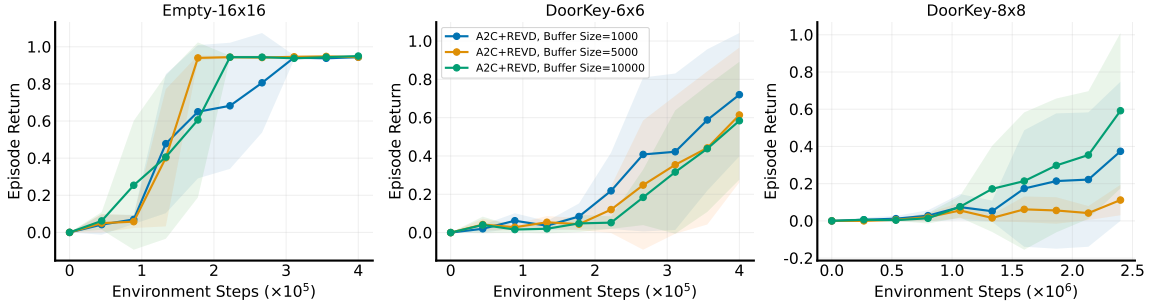


Figure 4.7: The performance comparison of REVD with different buffer sizes used for state archival. The solid line and shaded regions represent the mean and standard deviation, respectively.

complex environments, whereas a smaller buffer is sufficient for relatively simple ones.

4.1.4 Conclusion

In this section, we propose a novel and computation-efficient exploration method entitled REVD, which provides high-quality intrinsic rewards without relying on auxiliary models or representation learning. Since REVD computes intrinsic rewards based on

visitation discrepancy in different periods, it can overcome the problem of vanishing rewards and provide sustainable exploration incentives. We test REVD on the MiniGrid benchmark, and extensive experimental results demonstrate that REVD significantly improves sample efficiency and outperforms algorithmic baselines. It can be envisioned that computation-efficient exploration will be widely applied to various tasks due to its high efficiency and convenience. This work is expected to facilitate and inspire more subsequent research on data-efficient RL.

4.2 Automatic Intrinsic Reward Shaping for Exploration in RL

4.2.1 Introduction

Striking an appropriate balance between exploration and exploitation remains a long-standing problem in reinforcement learning (RL) [188]. Sufficient exploration can prevent the RL agent from prematurely falling into local optima after a finite number of iterations, thereby contributing to the learning of better policies [58]. To address this problem, classical exploration strategies, such as ϵ -greedy or Boltzmann exploration, will randomly choose all possible actions with a non-zero probability [137]. But these approaches are inefficient when handling complex environments with high-dimensional observations. Moreover, extrinsic rewards are consistently found to be sparse or even absent in many real-world scenarios, and they may completely fail to learn.

Recent approaches have proposed to leverage intrinsic rewards to encourage exploration [141, 76, 15, 149, 66, 143]. For instance, [15] leverages a density model to approximate the state visitation frequency and define the intrinsic reward as inversely proportional to the pseudo-count. As a result, the agent is encouraged to visit the

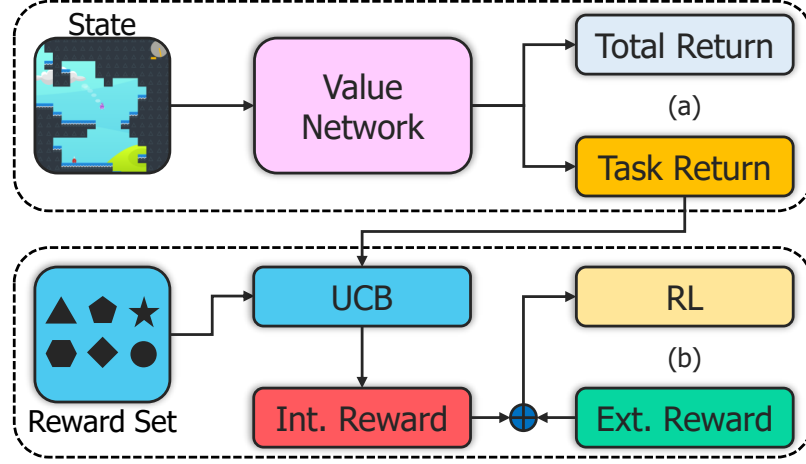


Figure 4.8: (a) AIRS employs two branches in the value network to disentangle the estimation of total return (evaluated by intrinsic and extrinsic reward function) and task return (only extrinsic reward function). (b) AIRS formulates the intrinsic reward selection as a multi-armed bandit problem and uses the upper confidence bound (UCB) to make decisions based on the estimated task return. **Ext.**: Extrinsic. **Int.**: Intrinsic. $\oplus$: Weighted summation operation.

infrequently-seen states, increasing the probability of encountering states with higher task rewards. In contrast, curiosity-driven exploration aims to learn the dynamics of the environment and utilizes the prediction error as the intrinsic reward [182, 211, 25]. For example, [149] uses an inverse-forward dynamics model to learn the representation of state space, which only encodes the part that affects the decision-making and ignores environment noise and other irrelevant interference. After that, the intrinsic reward is defined as the prediction error of the encoded next-state based on the current state-action pair.

Despite the excellent performance of intrinsic rewards, they cannot guarantee the invariance of the optimal policy, and excessive exploration may lead to learning collapse [138]. Since the joint objective composed of extrinsic and intrinsic rewards

is biased, maximizing it only sometimes yields the optimal policy for the extrinsic reward alone. To alleviate this problem, [30] proposes an extrinsic-intrinsic policy optimization (EIPO) approach, which automatically tunes the importance of the intrinsic reward via a principled constrained policy optimization procedure. As a result, EIPO can restrain the intrinsic reward when exploration is unnecessary and increase it when exploration is required. However, EIPO is too sophisticated to implement and cannot provide increments for arbitrary RL algorithms. Moreover, many experiments reported in existing work [24, 24, 161, 9, 30] demonstrate that the performance of the same intrinsic rewards varies significantly across different tasks and learning stages. Selecting the best intrinsic reward for a specific task is always challenging. With this in mind, [228] proposed a stochastic gradient-based method for learning parametric intrinsic rewards, thereby improving the performance of policy gradient-based algorithms.

In this section, we solve the aforementioned problems using a novel framework entitled **AIRS: Automatic Intrinsic Reward Shaping**. Our main contributions can be summarized as follows:

- Firstly, we suggest adapting the intrinsic reward shaping function across different tasks and learning stages, thereby enhancing exploration efficiency and alleviating the problem of the biased objective. This means using different types of intrinsic rewards or choosing not to use them when exploration is sufficient. To the best of our knowledge, this is the first work that proposes the concept of adaptive intrinsic reward shaping.
- Secondly, we formulate the intrinsic reward selection as a multi-armed bandit problem, in which different intrinsic reward functions are regarded as arms. AIRS can automatically select the best shaping function at different learning stages based on estimated task returns, enabling dynamic adaptation and efficient allocation of exploration incentives without prior knowledge or manual

intervention.

- Finally, we test AIRS on the MiniGrid and Procgen (*sixteen games with procedurally-generated environments*) benchmark. Extensive simulation results demonstrate that AIRS achieves superior performance and generalization, outperforming the benchmarking schemes.

4.2.2 Methodology

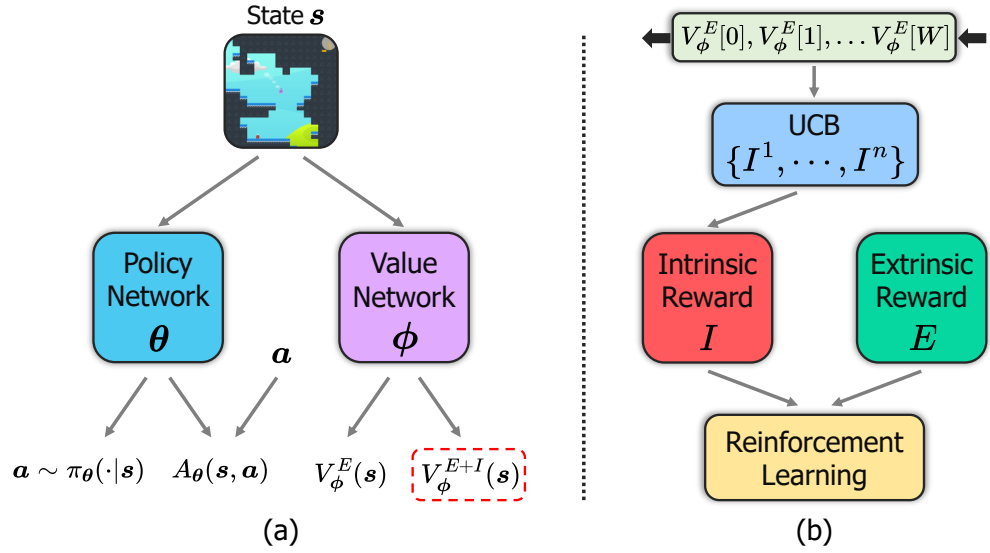


Figure 4.9: The overview of AIRS. (a) The architecture of DAAC. Here, we use two branches in the value network to estimate $V_\phi^E(s)$ and $V_\phi^{E+I}(s)$. (b) AIRS selects the intrinsic reward function based on a queue (length= W) of estimated task returns.

In this section, we propose the AIRS framework, which enhances the exploration and generalization capabilities of the RL agent by intelligently providing high-quality intrinsic rewards. Our critical insight is that different tasks and learning stages may benefit from distinct intrinsic reward functions. For instance, in the early stages of learning, aggressive exploration can enable agents to gain a large number of discrim-

inative experiences in a short time. To that end, the simple idea is to design an intrinsic reward function that rewards significant state changes. However, in the later stages of learning, excessive exploration is unnecessary and may disrupt the learned policy; it suffices to maintain appropriate exploration. The discussions above can be summarized as the following problems:

- Use or not use intrinsic reward function?
- Use which intrinsic reward function?

Denoting by $\mathcal{I} = \{I^1, \dots, I^n\}$ the set of intrinsic reward functions, the problem of intrinsic reward selection can be formulated as a multi-armed bandit (Figure 4.9b) [103]. Each reward function is considered an arm, and the objective is to maximize the long-term return evaluated by the extrinsic reward function.

To solve the MAB problem above, we introduce the UCB algorithm, which selects actions by the following policy:

$$I_t = \operatorname{argmax}_{I \in \mathcal{I}} \left[U_t(I) + c \sqrt{\frac{\log t}{N_t(I)}} \right], \quad (4.9)$$

where $U_t(I)$ is the estimated utility of action I , $N_t(I)$ is the number of times that function I has been selected prior to time t , and $c > 0$ is a constant that controls the degree of exploration. To update U_t , we employ two branches (Figure 4.9a) in the value network to estimate the state-value function evaluated by the mixed reward function $E + I$ and extrinsic reward function E , respectively. For each intrinsic reward function I , a queue of length W is leveraged to store the average estimated task return, and $U_t(I)$ is computed as

$$U_t(I) = \frac{1}{W} \sum_{i=1}^W \bar{V}_\phi^E[i], \quad (4.10)$$

where $\bar{V}_\phi^E$ is the average estimated return of a rollout. Meanwhile, the loss function

of the value network is redefined as

$$L_V(\phi) = \left[V_\phi^{E+I}(\mathbf{s}_t) - \hat{V}_t^{E+I} \right]^2 + \left[V_\phi^E(\mathbf{s}_t) - \hat{V}_t^E \right]^2. \quad (4.11)$$

Finally, the detailed workflow of AIRS based on DAAC is summarized in Algorithm 2. We want to highlight two critical facts about AIRS. Here, we use DAAC as the benchmark due to its improved generalization performance. But AIRS can be combined with many RL algorithms, and it suffices to use two branches in the value network to estimate task and total return, respectively. In addition, any appropriate bandit algorithms can replace the UCB algorithm. For example, a normal Thompson sampling [164] can be deployed when the intrinsic reward pool has only two reward functions. Therefore, AIRS is a very open architecture, and various attempts are possible.

4.2.3 Experiments

In this section, we designed experiments to answer the following questions:

- **Q1:** Can AIRS alleviate the aforementioned biased objective problem?
- **Q2:** How does AIRS compare to single intrinsic reward-driven approaches?
- **Q3:** Can the redesigned value network make effective value estimation?
- **Q4:** What is the detailed decision process of AIRS during training?

4.2.3.1 Experimental Setup

a) Benchmark Selection: Similar to the REVD, we first test AIRS on the Mini-Grid benchmark to highlight the effectiveness of automatic intrinsic reward shaping [35]. MiniGrid contains a collection of 2D grid-world environments with goal-oriented and sparse-reward tasks that are extremely challenging for exploration. Additionally,

Algorithm 2: Automatic Intrinsic Reward Shaping (AIRS)

```

1 Initialize policy network  $\pi_{\theta}$  and value network  $V_{\phi}$  with parameters  $\theta$  and  $\phi$ 
2 Initialize the set of intrinsic reward functions  $\mathcal{I} = \{I^1, \dots, I^n\}$ , exploration
   coefficient  $c$ , window length  $W$  for estimating the utility functions, and total
   number of updates  $K$ 
3  $N(I) = 1, \forall I \in \mathcal{I}$  // Initialize the number of times each  $I$  was
   selected.
4  $U(I) = 0, \forall I \in \mathcal{I}$  // Initialize the utility functions for all  $I$ .
5  $R(I) = \text{FIFO}(W), \forall I \in \mathcal{I}$  // Initialize the lists of returns for all  $I$ .
6 for  $k = 1, \dots, K$  do
7    $I_k = \underset{I \in \mathcal{I}}{\operatorname{argmax}} \left[ U(I) + c \sqrt{\frac{\log k}{N(I)}} \right]$  // Use UCB to select an  $I$ .
8   Collect a rollout  $\{(\mathbf{s}_t, \mathbf{a}_t, E_t, \mathbf{s}_{t+1})\}_{t=1}^T$  using  $\pi_{\theta}$ 
9   Use  $I_k$  to compute intrinsic rewards
10  Compute the value and advantage targets  $\hat{V}_t^E, \hat{V}_t^{E+I}$  and  $\hat{A}_t$  for all states  $\mathbf{s}_t$ 
11   $\theta \leftarrow \underset{\theta}{\operatorname{argmax}} J_{\text{DAAC}}$  // Update the policy network.
12   $\phi \leftarrow \underset{\phi}{\operatorname{argmax}} L_V$  // Update the value network.
13  Compute the mean return  $\bar{V}_{\phi}$  obtained by the new policy
14  Add  $\bar{V}_{\phi}$  to the  $R(I_k)$  list using the first-in-first-out rule
15   $U(I_k) \leftarrow \frac{1}{|R(I_k)|} \sum_{\bar{V}_{\phi} \in R(I_k)} \bar{V}_{\phi}$ 
16   $N(I_k) \leftarrow N(I_k) + 1$ 
17 end

```

we introduce the full Procgen benchmark, which contains sixteen games with procedurally generated environments [36]. Procgen is developed similarly to the ALE benchmark, in which agents must learn motor control directly from images [16]. However, Procgen has higher requirements for generalization, and the agent must conduct sufficient exploration and learn transferable skills rather than memorize specific tra-

jectories. These attributes make it a good benchmark for our research. All Progen games utilize a discrete action space with 15 possible actions and produce $64 \times 64 \times 3$ RGB observations. The agents are trained on the easy level and tested on the entire distribution of levels.

b) Algorithmic Baselines: For the backbone RL algorithm, we select A2C [135] for the MiniGrid experiments and consider the combination of A2C and RE3. For AIRS, we build it on top of A2C+RE3, in which the intrinsic reward function pool induces only identity (ID) shaping and RE3. In particular, we added the intrinsic rewards directly into the estimated advantages, rather than using an additional branch in the value network. For each benchmark, we report only the best results obtained after a hyperparameter search.

Table 4.2: Four intrinsic reward functions are employed in the Progen experiments, where $\mathbf{e} = g(\mathbf{s})$ is the representation of $\mathbf{s}$, N_{ep} is the episodic state visitation frequency. For RE3 and RISE, g is a random and fixed encoder, while g is a learned encoder for RIDE. Finally, $\tilde{\mathbf{e}}$ is the k -nearest neighbor of $\mathbf{s}$ in the encoding space.

Reward	Formulation	Remark
RE3 [177]	$I_t = \frac{1}{k} \sum_{i=1}^k \log(\ \mathbf{e}_t - \tilde{\mathbf{e}}_t^i\ _2 + 1)$	Shannon entropy maximization
RISE [216]	$I_t = \frac{1}{k} \sum_{i=1}^k (\ \mathbf{e}_t - \tilde{\mathbf{e}}_t^i\ _2)^{1-\alpha}$	Rényi entropy maximization
RIDE [161]	$I_t = \ \mathbf{e}_{t+1} - \mathbf{e}_t\ _2 / \sqrt{N_{\text{ep}}(\mathbf{s}_{t+1})}$	Significant state changes
ID	$I_t = 0$	No shaping

For the Progen experiments, four intrinsic reward functions are selected to construct the reward pool, namely RE3 [177], RISE [216], RIDE [161], and ID, respectively. The detailed information of these functions is illustrated in Table 4.2. The reasons for selection are as follows: (i) they can provide sustainable exploration incentives, *i.e.*, the intrinsic rewards will not vanish with visits; (ii) they are computation-efficient

and easy to implement, and (iii) they can leverage fixed and random encoder to encode the state space, which provides relatively stable intrinsic reward space and guarantees the algorithm convergence. Furthermore, we select DrAC [160], IDAAC [159], DAAC [159], and PPO [171] as benchmarks. DrAC theoretically combines data augmentation with actor-critic algorithms, achieving remarkable performance on the Procgen benchmark. DAAC is a state-of-the-art on-policy algorithm that decouples policy and value optimization to improve generalization in RL. IDAAC is a variant of DAAC that adds a regularizer to the DAAC policy encoder, ensuring it contains only episode-specific information. For each benchmark method, we report only its best performance after a hyperparameter search.

c) Evaluation Metrics: For the MiniGrid experiments, we simply compare the problem-solving speed between AIRS and other algorithmic baselines. For the Procgen experiments, we report results using a reliable and robust toolbox entitled *rl* to guarantee reliability and reduce uncertainty [2]. Four metrics are introduced in the following sections, **Median:** Aggregate median performance, higher is better. **IQM:** Aggregate interquartile mean performance, higher is better. **Mean:** Aggregate mean performance, higher is better. **Optimality Gap:** The amount by which the algorithm fails to meet a minimum score; the lower, the better.

4.2.3.2 Results Analysis

The following results analysis is performed based on the pre-defined research questions.

a) Performance Comparison on the MiniGrid Benchmark: Figure 4.10 illustrates the sample efficiency comparison between AIRS and benchmarking schemes. A2C+RE3 successfully outperformed the vanilla A2C agent in all three games, demonstrating that RE3 can effectively promote the sample efficiency using the same environment steps. In contrast, A2C+AIRS achieves the highest performance and con-

vergence rate in all three games. This indicates that AIRS successfully alleviated the biased objective problem and intelligently controlled the exploration degree during the training process.

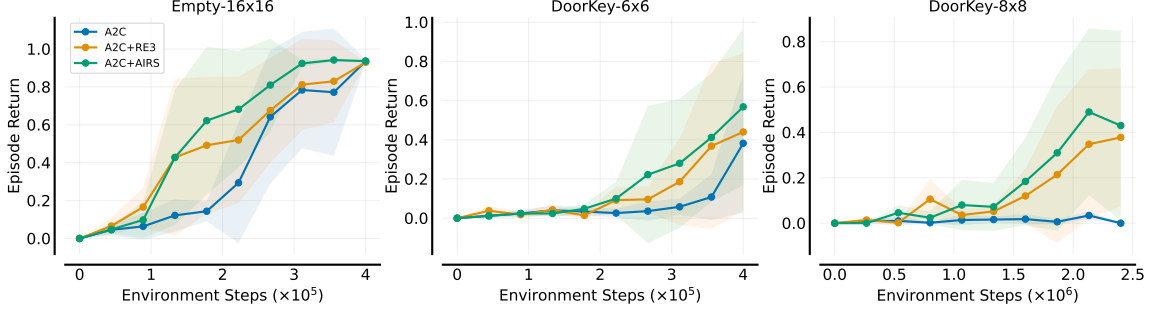


Figure 4.10: Sample efficiency comparison between AIRS and baselines. The solid line and shaded regions represent the mean and standard deviation, which are computed over five random seeds.

b) Performance Comparison on the Procgen Benchmark: The game complexity of test levels is much higher than that of the train levels, providing a direct measure to evaluate the generalization ability of the agents. Figure 4.11 illustrates the test performance comparison evaluated by four metrics. AIRS achieves the highest performance across all four metrics, especially in **Median** and **IQM**. In contrast, IDAAC achieves the second highest performance in all four metrics. The performance of IDAAC and DAAC is very close in most games. This indicated that the additional regularization term did not significantly improve the performance of DAAC, which is consistent with the results reported in [159]. Moreover, the combination of DAAC and single intrinsic rewards achieves lower performance than the vanilla DAAC agent. Furthermore, we compute the probability of improvement of AIRS over the benchmarks. Figure 4.12 indicates that the probability of AIRS being better than DAAC is 67%, and the probability of AIRS being better than IDAAC is 61%.

Similar to the MiniGrid experiments, we conduct an ablation study by setting the

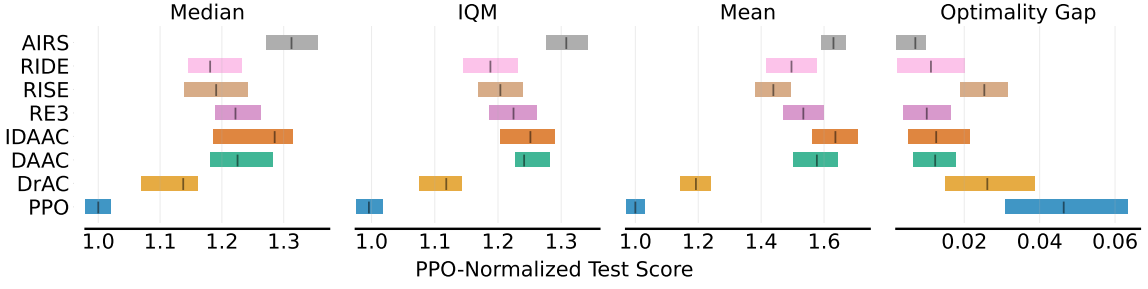


Figure 4.11: PPO-Normalized test performance for the Procgen benchmark, which is aggregated over all sixteen tasks and ten random seeds.

intrinsic reward pool to contain only two modules: RE3 and ID. As a result, the test **IQM** and **OG** are 1.27 and 0.015, which are better than DAAC+RE3. The ablation results further demonstrated that AIRS effectively alleviates the biased objective problem, providing appropriate exploration incentives when necessary. It also demonstrated that AIRS could assemble advantages by automatically selecting from multiple intrinsic rewards.

c) Accuracy Analysis of Value Estimation: Figure 4.13 illustrates the value loss of eight games during training. It is evident that the loss of the two branches stabilizes after 1 million environment steps. But AIRS still achieved remarkable performance gains. This indicates that AIRS can effectively improve the agent’s exploration and maintain tracking of the fundamental objective, *i.e.*, the task reward.

d) Detailed Decision Process of AIRS: Furthermore, we document the specific decision-making process of AIRS. Figure 4.14 illustrates the cumulative number of times AIRS selects each intrinsic reward function over the training progress of eight games. In the *BigFish* game, the RIDE is the most selected function while AIRS chose no shaping in 33.7% of the training time. In the *Miner* game, the RISE is the most selected function while AIRS chose no shaping in 59.3% of the training time. Similarly, the RISE is also the most selected function in the *Ninja* game, and AIRS

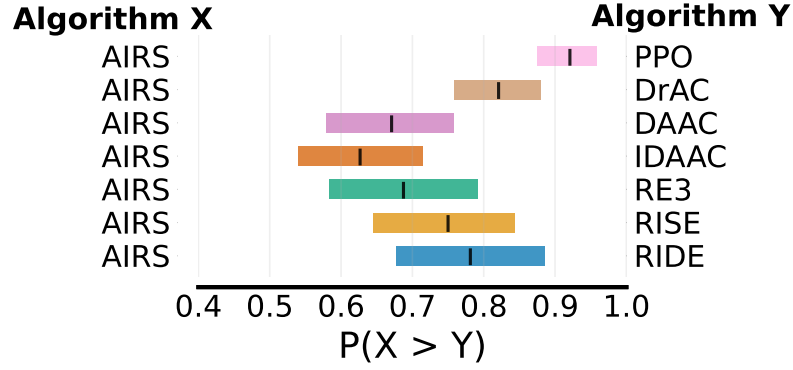


Figure 4.12: Each row shows the probability of improvement, with 95% bootstrap confidence intervals, that the algorithm X on the left outperforms algorithm Y on the right, given that X was claimed to be better than Y . For all algorithms, results are based on ten random seeds per task.

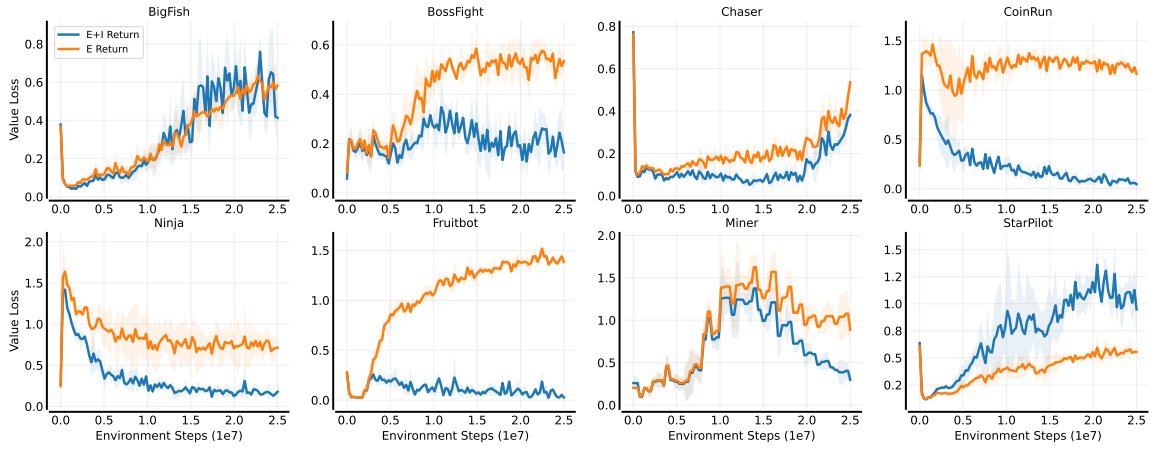


Figure 4.13: Value loss of eight selected games during training. “E Return” indicates the branch for extrinsic value estimation in the value network. The solid line and shaded regions represent the mean and standard deviation over 10 random seeds, respectively.

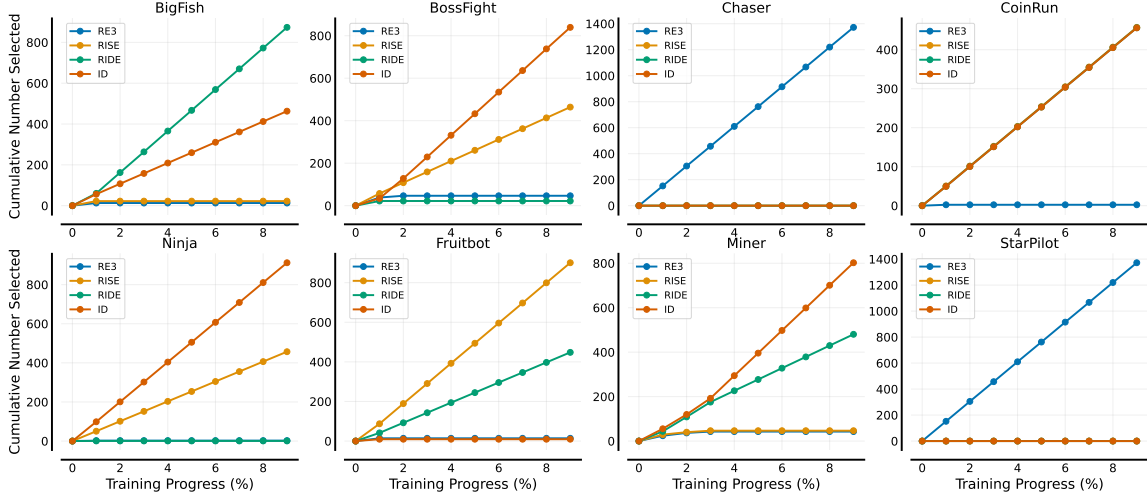


Figure 4.14: The cumulative number of times AIRS selects each intrinsic reward function over the training progress, computed with ten random seeds.

chooses no shaping in 66.5% of the training time. Meanwhile, AIRS divides the choice opportunity into three functions in *CoinRun* game and *BossFight* game. It is evident that no shaping played an essential role throughout the training process, and appropriate exploration can improve sample efficiency and achieve higher performance. In contrast, AIRS selects RISE and RIDE at most in the *Fruitbot* game. In the *Chaser* and *StarPilot* game, AIRS only selects the RE3 function during the training, and the final performance of AIRS and DAAC+RE3 is almost the same.

4.2.4 Conclusion

In this section, we investigate how to enhance exploration in RL and propose an intrinsic reward-driven method entitled AIRS. AIRS can intelligently select the best intrinsic reward function from a pre-defined set in real time, providing reliable exploration incentives and guaranteeing policy invariance. In particular, we develop a toolkit to provide high-quality implementations of diverse intrinsic reward approaches,

which is expected to inspire more subsequent research. We test AIRS on multiple tasks from MiniGrid, Progen, and DeepMind Control Suite. Extensive simulation results demonstrate that our method can achieve superior performance.

However, there are some remaining limitations of AIRS. Firstly, AIRS performs selection based on a pre-defined reward set. The quality of selected reward modules will inevitably affect the final policy performance. It's critical to find an effective method to build the reward set. Meanwhile, different intrinsic reward spaces may interfere with one another, leading to unexpected learning collapse. Secondly, AIRS makes decisions based on estimated task returns, which require an independent module to make those estimates. The estimation accuracy also determines the reliability of the actions of AIRS, and additional modules will increase the complexity of the deployment of AIRS. For algorithms like A2C and PPO, a simple approach is to add intrinsic rewards to the estimated advantages directly, rather than using an additional value network branch, as in our MiniGrid experiments. Finally, AIRS formulates the reward selection as a bandit problem and uses UCB to solve it, requiring sufficient attempts to estimate the bound. This will significantly increase the computational overhead and decrease the convergence rate. We will address these issues in future work.

4.3 RLeXplore: Accelerating Research in Intrinsically-Motivated RL

4.3.1 Introduction

RL provides a framework for training agents to solve tasks by learning from interactions with an environment. At the core of RL is the optimization of a reward function, where agents aim to maximize cumulative rewards over time [188]. How-

ever, in complex environments, defining extrinsic rewards that effectively guide an agent’s learning process can be impractical, often requiring domain-specific expertise. In practice, poorly defined extrinsic rewards can lead to sparse-reward settings, where RL agents struggle due to the lack of a meaningful learning signal [23].

As the RL community tackles increasingly complex problems, such as training generally capable RL agents, there is a need for more autonomous agents capable of learning valuable behaviors without relying on dense supervision [86]. To address this challenge, the concept of intrinsic rewards has emerged as a promising approach in the RL community [25, 149, 161, 9, 71, 150]. Intrinsic rewards provide agents with additional learning signals, enabling them to explore and acquire skills across diverse environments beyond what extrinsic rewards alone can offer. However, computing intrinsic rewards often requires learning auxiliary models, heavy engineering, and performing expensive computations, making reproducibility challenging.

While several formulations of intrinsic rewards have been proposed [149, 9, 102], each with its potential benefits for improving agent learning, the field lacks a comprehensive understanding of the comparative advantages and challenges posed by these methods. Importantly, existing literature reports varying performance when using the same intrinsic rewards, reinforcing the need for a standardized framework and a deeper understanding of the optimization and implementation details.

In this paper, we introduce **RLeXplore**¹, an open-source library containing high-quality implementations of state-of-the-art (SOTA) intrinsic rewards. RLeXplore offers a plug-and-play framework for researchers working on intrinsically-motivated RL, enabling them to seamlessly integrate intrinsic rewards into their projects. Specifically, RLeXplore (1) facilitates fair comparisons across multiple intrinsic reward methods, (2) can be easily integrated with various RL frameworks, and (3) streamlines the development of new intrinsic reward methods. In Table 4.3, we compare the performance of the implementations in RLeXplore with the original results reported in

¹<https://github.com/RLE-Foundation/RLeXplore>

previous works.

Table 4.3: Summary of comparative results from our RLeXplore implementations and prior work. The right two columns report success rates (% of episodes solving the task), except for Procgen, where we report mean episode rewards after training.

Environment	Reward	Original	RLeXplore
SuperMarioBros (10M Steps)	RIDE	23%	50%
SuperMarioBros (10M Steps)	ICM	30%	30%
MiniGrid-DoorKey-16×16 (10M Steps)	ICM	0%	60%
MiniGrid-DoorKey-16×16 (10M Steps)	RND	0%	60%
MiniGrid-DoorKey-16×16 (10M Steps)	RIDE	25%	12%
MiniGrid-DoorKey-8×8 (1M Steps)	RE3	50%	95%
MiniGrid-DoorKey-8×8 (1M Steps)	RND	0%	82%
MiniGrid-DoorKey-8×8 (1M Steps)	ICM	20%	83%
Procgen - 200 Mazes (25M Steps)	E3B	3.00	4.10
Procgen - 200 Mazes (25M Steps)	ICM	2.50	5.90
Procgen - 200 Mazes (25M Steps)	RND	1.70	5.00

To support these capabilities, we have provided extensive documentation that includes detailed guides on using RLeXPLORE, along with comprehensive code tutorials. These resources are designed to make it straightforward for users to get started with RLeXPLORE, regardless of their prior experience with intrinsic rewards in RL. In Table 4.4, we provide an overview of the main differences and advantages of RLeXPLORE compared to existing RL libraries.

We aim for the community to adopt RLeXPLORE as a standard tool for evaluating intrinsic reward methods, reducing implementation efforts, and mitigating inconsistencies in results and conclusions.

Table 4.4: Comparison between RLeXplore and other reported libraries of intrinsic rewards. Note that we focus on the intrinsic reward methods that are implemented. For instance, CleanRL has numerous implementations of various RL algorithms, but RND is the only intrinsic reward currently supported. Here, PnP is short of plug-and-play.

Framework	ML Framework	Number of Algorithms	PnP	Doc.	Benchmark Results
CleanRL	PyTorch	1	✗	✓	1 task, 1 experiment
DI-Engine	PyTorch	3	✗	✓	5 tasks, 19 experiments
rllib	TensorFlow	2	✗	✓	N/A
RLeXplore	PyTorch	8	✓	✓	17 tasks, over 2000 experiments

Our work presents a systematic study aimed at addressing gaps in understanding the critical implementation and optimization details of intrinsic rewards. We investigate the design of different intrinsic reward methods and (1) highlight challenges in the reproducibility of prior work, and (2) share highly performant reimplementations of many popular methods. To guide our investigation, we formulate numerous questions, aiming to uncover the intricacies of intrinsic rewards and their impact on RL agent performance. Our results highlight the importance of thoughtful implementation design for intrinsic rewards, showing that naive implementations can lead to suboptimal performance. Through carefully studied design decisions, we demonstrate significant performance gains.

Our results show that with RLeXplore, RL agents can learn emergent behaviours autonomously, solving multiple levels of SuperMarioBros without task rewards. Ad-

ditionally, we show that intrinsic rewards enable RL agents to obtain great performance on complex sparse-reward tasks like Procgen-Maze, MiniGrid, the ALE-5 hard-exploration tasks and Ant-UMaze.

4.3.2 Methodology

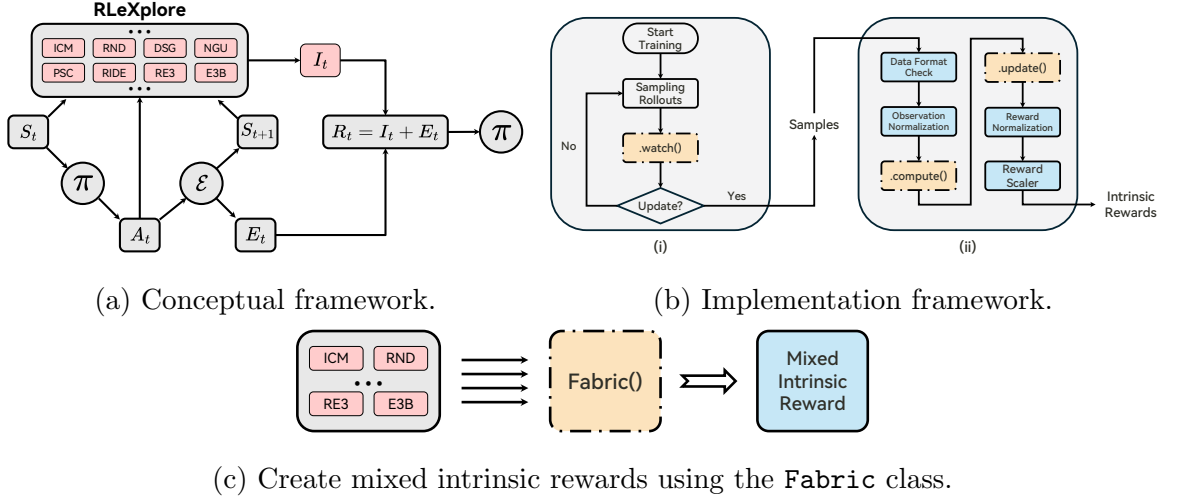


Figure 4.15: The workflow of RLeXplore. (a) RLeXplore provides a decoupled module for intrinsic rewards that integrates seamlessly with the RL training loop. RLeXplore implements 8 SOTA intrinsic rewards and adapts to the unmodified RL training loop. (b) RLeXplore monitors the agent-environment interactions and gathers data samples using the `.watch()` function. After collecting experience rollouts, RLeXplore computes the corresponding intrinsic rewards using the `.compute()` function and updates the auxiliary models via the `.update()` function. (c) RLeXplore provides a `Fabric` class that allows developers to combine multiple intrinsic rewards in an elegant manner.

In this section, we present **RLeXplore**, a unified, highly-modularized, and plug-and-play framework that currently provides high-quality and reliable implementations of eight SOTA intrinsic reward methods². Comparing multiple intrinsic reward methods

²RLeXplore complies with the MIT License.

under fair conditions is challenging due to various confounding factors, such as using distinct RL frameworks (*e.g.*, PPO [171], DQN [136], IMPALA [54]), optimization (*e.g.*, reward and observation normalization, network architecture) and evaluation details (*e.g.*, environment configuration, algorithm hyperparameters). RLeXplore is designed to provide a unified framework with standardized procedures for implementing, computing, and optimizing intrinsic rewards.

4.3.3 Architecture

The core design decision of RLeXplore involves decoupling the intrinsic reward modules from the RL optimization algorithms, which enables our intrinsic reward implementations to be integrated with any desired RL algorithm or existing libraries, such as CleanRL [80], Stable-Baselines3 [157], and RLLTE [219]. Figure 4.15 illustrates the basic workflow of RLeXplore, which consists of two parts: data collection (*i.e.*, policy rollout) and reward computation.

Commonly, at each time step, the agent receives observations from the environment and predicts actions. The environment then executes the actions and returns feedback to the agent, which consists of a next observation, a reward, and a terminal signal. During the data collection process, the `.watch()` function is used to monitor the agent-environment interactions. For instance, E3B [71] updates an estimate of an ellipsoid in an embedding space after observing every state. At the end of the data collection rollouts, `.compute()` computes the corresponding intrinsic rewards. Note that `.compute()` is only called once per rollout using batched operations, which makes RLeXplore a highly efficient framework. Additionally, RLeXplore provides several utilities for reward and observation normalization. Finally, the `.update()` function is called immediately after `.compute()` to train the reward module if necessary (*e.g.*, train the forward dynamics models in Disagreement [150] or the predictor network in RND [25]). All operations are subject to the standard workflow of the Gymnasium

API [197].

In particular, recent research [70] has highlighted that mixed intrinsic rewards can significantly promote the agent’s exploration capability by providing comprehensive exploration incentives. In RLeXplore, we provide a **Fabric** class that allows developers to combine multiple intrinsic rewards in an elegant manner, as illustrated in Figure 4.15.

RLeXplore offers several benefits to the research community:

- For researchers seeking reliable tools for benchmarking and general applications: RLeXplore provides high-quality implementations of popular intrinsic reward methods, useful in both research and practical applications. It can be seamlessly integrated with existing RL libraries, such as Stable-Baselines3 [157], CleanRL [80], and RLLTE [218].
- For developers experimenting with new intrinsic rewards: RLeXplore offers modular components, such as various embedding networks and a standardized workflow. This setup facilitates the creation, modification, and testing of new ideas. Detailed examples are available in the code repository and documentation.
- For promoting collaboration and accelerating progress: We have published a space using Weights & Biases (W&B) to store reusable experiment results on recognized benchmarks. This initiative aims to enhance collaboration within the research community and speed up progress by providing easy access to established benchmark results.

4.3.4 Algorithmic Baselines

In RLeXplore, we implement eight widely-recognized intrinsic reward methods spanning the different categories in this area, namely ICM [149], RND [25], Disagreement

[150], NGU [9], PseudoCounts [9], RIDE [161], RE3 [177], and E3B [71], respectively. We selected them based on the following tenet:

- The algorithm represents a unique design philosophy;
- The algorithm achieved superior performance on well-recognized benchmarks;
- The algorithm can adapt to arbitrary tasks and can be combined with arbitrary RL algorithms.

The detailed descriptions of each method are as follows:

ICM [149]. ICM leverages an inverse-forward model to learn the dynamics of the environment and uses the prediction error as the curiosity reward. Specifically, the inverse model infers the current action $\mathbf{a}_t$ based on the encoded states $\mathbf{e}_t$ and $\mathbf{e}_{t+1}$, where $\mathbf{e} = \psi(\mathbf{s})$ and $\psi(\cdot)$ is an embedding network. Meanwhile, the forward model f predicts the encoded next-state $\mathbf{e}_t$ based on $(\mathbf{e}_t, \mathbf{a}_t)$. Finally, the intrinsic reward is defined as

$$I_t = \|f(\mathbf{e}_t, \mathbf{a}_t) - \mathbf{e}_{t+1}\|_2^2. \quad (4.12)$$

RND [25]. RND produces intrinsic rewards via a self-supervised manner, in which a predictor network $\hat{f}$ is trained to approximate a fixed and randomly-initialized target network f . As a result, the agent is motivated to explore unseen parts of the state space. The intrinsic reward is defined as

$$I_t = \|\hat{f}(\mathbf{s}_{t+1}) - f(\mathbf{s}_{t+1})\|_2^2. \quad (4.13)$$

Disagreement [150]. Disagreement is a variant of ICM that leverages an ensemble of forward models and calculates the intrinsic reward as the variance among these models. Accordingly, the intrinsic reward is defined as

$$I_t = \text{Var}\{f_i(\mathbf{e}_t, \mathbf{a}_t)\}, i = 0, \dots, N. \quad (4.14)$$

NGU [9]. NGU is a mixed intrinsic reward approach that combines global and episodic exploration and is the first method to achieve non-zero rewards in the game of *Pitfall!* without using demonstrations or hand-crafted features. The intrinsic reward is defined as

$$I_t = \min\{\max\{\alpha_t, C\}/\sqrt{N_{\text{ep}}(\mathbf{s}_t)}, \quad (4.15)$$

where α_t is a life-long curiosity factor computed following the RND method, C is a chosen maximum reward scaling, and N_{ep} is the episodic state visitation frequency computed by pseudo-counts.

PseudoCounts [9]. Pseudo-counts has been widely used in count-based exploration approaches [15, 143] with diverse implementations like neural density models. In this paper, we follow NGU [9] that computes pseudo-counts via k -nearest neighbor estimation, which is highly efficient and can be applied to arbitrary tasks. Given the encoded observations $\{\mathbf{e}_0, \dots, \mathbf{e}_{T-1}\}$ visited in the an episode, we have

$$\sqrt{N_{\text{ep}}(\mathbf{s}_t)} \approx \sqrt{\sum_{\tilde{\mathbf{e}}_i} K(\tilde{\mathbf{e}}_i, \mathbf{e}_t) + c}, \quad (4.16)$$

where $\tilde{\mathbf{e}}_i$ is the first k nearest neighbors of $\mathbf{e}$, K is a Dirac delta function, and c guarantees a minimum amount of pseudo-counts. Finally, the intrinsic reward is defined as

$$I_t = 1/\sqrt{N_{\text{ep}}(\mathbf{s}_t)} \quad (4.17)$$

RIDE [161]. RIDE is designed based on ICM that learns the dynamics of the environment and rewards significant state changes. Accordingly, the intrinsic reward is defined as

$$I_t = \|\mathbf{e}_{t+1} - \mathbf{e}_t\|_2 / \sqrt{N_{\text{ep}}(\mathbf{s}_{t+1})}, \quad (4.18)$$

where $N_{\text{ep}}(\mathbf{s}_{t+1})$ is used to discount the intrinsic reward and prevent the agent from lingering in a sequence of states with a large difference in their embeddings.

RE3 [177]. RE3 is an information theory-based and computation-efficient exploration approach that aims to maximize the Shannon entropy of the state visiting distribu-

tion. In particular, RE3 leverages a random and fixed neural network to encode the state space and employs a k -nearest neighbor estimator to estimate the entropy efficiently. Then, the estimated entropy is transformed into particle-based intrinsic rewards. Specifically, the intrinsic reward is defined as

$$I_t = \frac{1}{k} \sum_{i=1}^k \log(\|\mathbf{e}_t - \tilde{\mathbf{e}}_t^i\|_2 + 1). \quad (4.19)$$

E3B [71]. E3B provides a generalization of count-based rewards to continuous spaces. E3B learns a representation mapping from observations to a latent space (*e.g.*, using inverse dynamics). At each episode, the sequence of latent observations parameterizes an ellipsoid [106, 8, 42], which is used to measure the novelty of the subsequent observations. In tabular settings, the E3B ellipsoid reduces to the table of inverse state-visitation frequencies [71]. Given a feature encoding f , at each time step t of the episode the elliptical bonus I_t is defined as follows:

$$I_t = f(\mathbf{s}_t)^T C_{t-1} f(\mathbf{s}_t), \quad (4.20)$$

$$C_{t-1} = \sum_{i=1}^{t-1} f(\mathbf{s}_i) f(\mathbf{s}_i)^T + \lambda \mathbf{I}, \quad (4.21)$$

where f is the learned representation mapping, C_{t-1} is the episodic ellipsoid [71], λ is a scalar coefficient, and $\mathbf{I}$ is the identity matrix.

4.3.5 Experiments

Our experiments aim to achieve two main objectives: (i) highlight how intrinsic reward methods are sensitive to implementation details, and (ii) identify the best algorithmic and design choices to ensure high performance across various sparse-reward environments to demonstrate the generality and robustness of our framework.

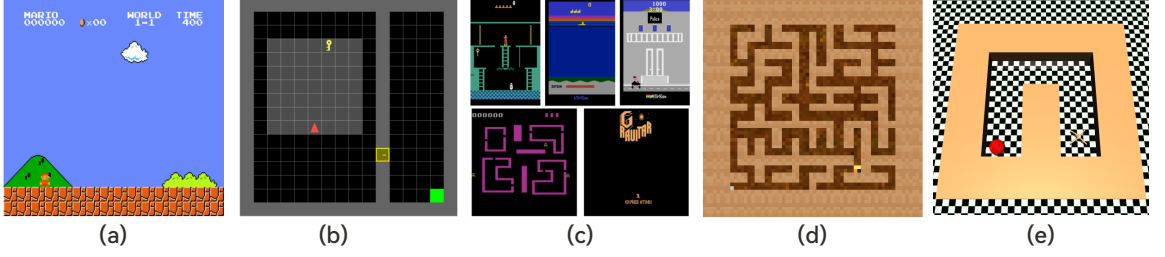


Figure 4.16: Screenshots of the selected exploration games. (a) *SuperMarioBros.* (b) *MiniGrid.* (c) *ALE-5.* (d) *Procgen-Maze.* (e) *Ant-UMaze.*

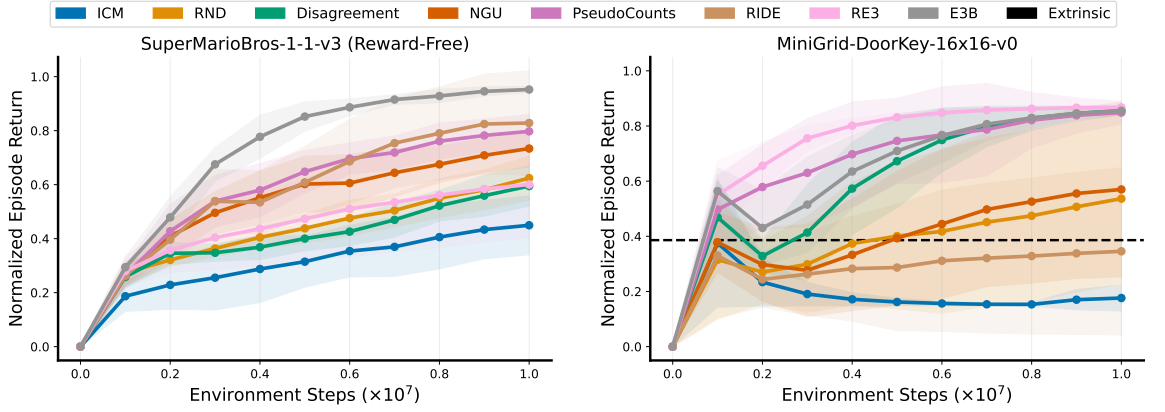


Figure 4.17: Episode returns achieved by the intrinsic rewards in RLeXplore. (left) *SuperMarioBros* without access to the task rewards. (right) *MiniGrid-DoorKey-16 $\times$ 16* with sparse rewards.

First, we use *SuperMarioBros* (*SMB*) without access to the environment’s rewards to study the low-level implementation details of intrinsic reward methods that drive robust exploration. We selected *SMB* because effective exploration within this environment strongly correlates with task performance, making it an excellent benchmark for measuring the efficacy of exploration techniques. This environment has been widely used in previous studies on exploration in RL [150, 161, 23]. To further generalize our findings, we also use the *MiniGrid-DoorKey-16 $\times$ 16* (*MGD*) environment, which is challenging due to the sparse rewards, making it difficult to solve with classical RL

algorithms³. The effectiveness of intrinsic rewards in *MiniGrid* environments has also been highlighted in prior works [161, 71, 70]. With these two environments, we aim to study the implementation details in both reward-free and sparse-reward tasks.

Secondly, to showcase the generalizability of RLeXplore, we evaluate our implementations in additional sparse-reward environments, including *Procgen*, *MiniGrid*, *Ant-UMaze*, and the set of five hard-exploration games in the *arcade learning environment* (*ALE*) suite. These experiments are designed to test how well our methods balance the use of dense intrinsic rewards with sparse extrinsic rewards across a variety of tasks.

Lastly, we explore recent advancements in using combined intrinsic rewards [70] to enhance exploration in contextual MDPs. Specifically, we use the full set of levels in *SMB* to evaluate how well both single and combined intrinsic rewards can explore various game versions and generalize their exploration across different levels. In the following sections, we present results from *SMB* and *MiniGrid* for objective (i) and from *Procgen-Maze* for objective (ii).

The design of these experiments is driven by our primary goal: to provide a general and reliable set of intrinsic reward implementations within a user-friendly framework. Instead of attempting to benchmark all methods across every possible domain, we focus on verifying the generality of each method within a carefully selected subset of popular exploration tasks.

4.3.5.1 Low-level Implementation Details of Intrinsic Rewards

The performance of intrinsic rewards is affected by various factors that tend to vary with the complexity of the task. For instance, the RL algorithm used for optimization, the architecture of the networks, algorithm-specific hyperparameters, and the joint optimization of intrinsic and extrinsic rewards. As a result, implementing and

³<https://minigrid.farama.org/environments/minigrid/DoorKeyEnv>

Table 4.5: Details of the baseline settings.

Hyperparameter	Value
Observation normalization	RMS
Reward normalization	RMS
Weight initialization	Orthogonal
Update proportion	100%
with LSTM	False

reproducing intrinsic reward methods is challenging. To tackle this problem, we first formulate five questions to investigate how various low-level implementation details impact the training of intrinsically motivated agents. We first define an initial baseline configuration for optimizing the intrinsic rewards, shown in Table 4.5. These baseline settings are selected based on the most common configurations reported in the literature. Next, we address each question by modifying only one hyperparameter in the baseline configuration at a time. Finally, we evaluate the performance of these intrinsic rewards with the best parameters gathered in each question. All the experiments are conducted using *SMB* and *MGD* to investigate the effects in sparse-rewards and reward-free (*i.e.*, without access to extrinsic rewards) scenarios, respectively.

Importantly, as shown in Figure 4.15, we keep the PPO hyperparameters fixed and the overall RL training loop unmodified throughout all the experiments in the paper in order to isolate the effect of the questions on the intrinsic reward components. Previous work has shown that PPO has many implementation details that are key to achieving great performance [79, 53]. In the following, we study implementation details for the intrinsic reward components. The fixed PPO hyperparameters are shown in Table 4.6.

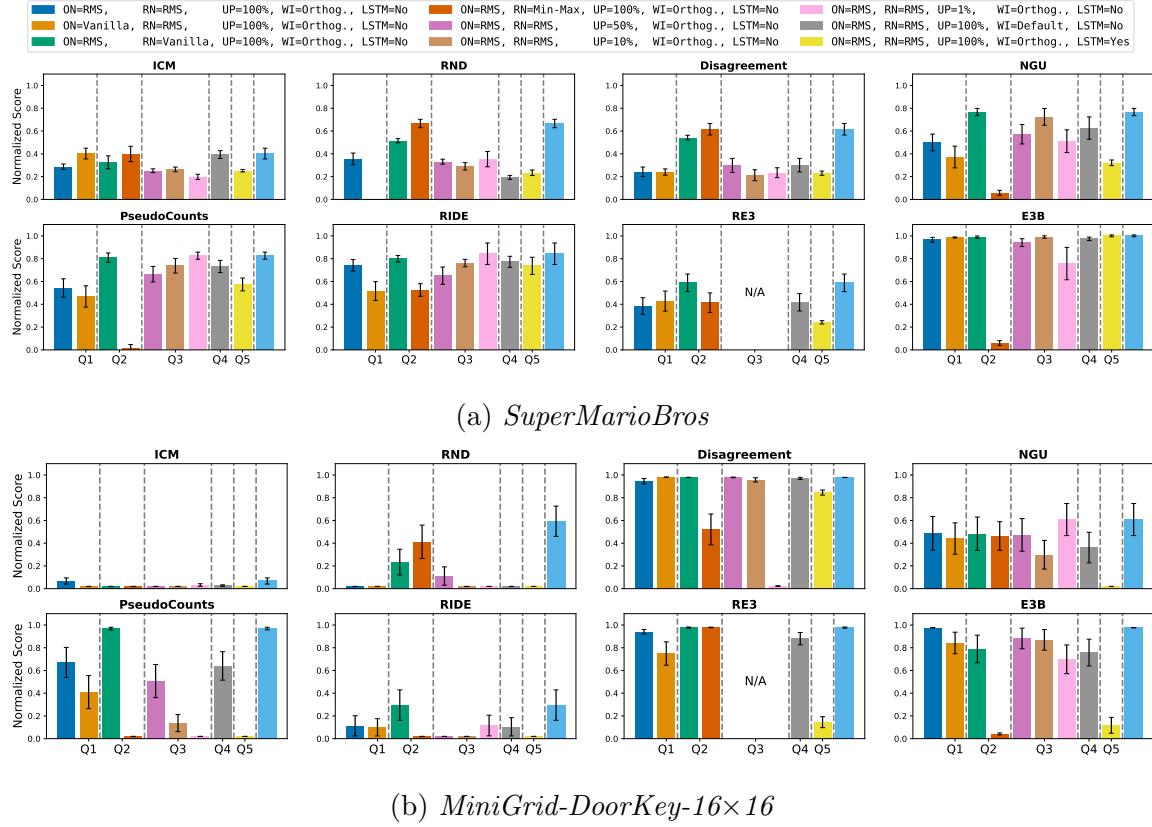


Figure 4.18: Results for Q1, Q2, Q3, Q4, and Q5 in *SMB* (top) and *MGD* (bottom), which are normalized by the maximum score possibly achieved in the task. Here, **ON** is the observation normalization, **RN** is the reward normalization, **UP** is the update proportion, and **WI** is the weight initialization. The bar with a hatch mark is **Combined**, which refers to the results of using the best hyperparameters gathered in each question. Since RE3 only employs a fixed, randomly initialized neural network for encoding observations, there are no values in Q3. All the results are aggregated over 10 seeds, and each run uses 10M environment interactions.

Q1: The impact of observation normalization.

Observation normalization is crucial in deep learning to avoid numerical instabilities during optimization. Image observations, where each pixel value typically ranges from 0 to 255 per colour channel, are commonly normalized to a range of 0 to 1 using Min-

Table 4.6: PPO hyperparameters for *SuperMarioBros*, *MiniGrid*, and *Procgen* games. These remain fixed for all experiments.

Hyperparameter	SuperMarioBros	MiniGrid	Procgen
Observation downsampling	(84, 84)	(7, 7)	(64, 64)
Observation normalization	/ 255.	No	/ 255.
Reward normalization	No	No	No
Weight initialization	Orthogonal	Orthogonal	Orthogonal
LSTM	No	No	No
Stacked frames	No	No	No
Environment steps	10000000	10000000	25000000
Episode steps	128	32	256
Number of workers	1	1	1
Environments per worker	8	256	64
Optimizer	Adam	Adam	Adam
Learning rate	2.5e-4	2.5e-4	5e-4
GAE coefficient	0.95	0.95	0.95
Action entropy coefficient	0.01	0.01	0.01
Value loss coefficient	0.5	0.5	0.5
Value clip range	0.1	0.1	0.2
Max gradient norm	0.5	0.5	0.5
Epochs per rollout	4	4	3
Batch size	256	1024	2048
Discount factor	0.99	0.99	0.999

Max normalization by dividing each pixel value by 255. However, previous studies suggest that Min-Max normalization may not be ideal for all representation learning algorithms [25].

In Q1, we compare Min-Max normalization with using an exponential moving average (EMA) of the mean and standard deviation for observation normalization (RMS) for the inputs to the intrinsic reward modules. RMS normalizes observations by subtracting the running mean and dividing it by the running standard deviation of all observations collected by the agent thus far. Our results shown in Figures 4.18 and 4.19 indicate that using RMS for observation normalization generally reduces the variance and achieves better asymptotic performance across all the environments of study. Importantly, some intrinsic rewards, such as RND, NGU, PseudoCounts, and RIDE, benefit significantly from RMS normalization. Critically, RND achieves zero rewards in SMB if observations are not normalized with RMS. These results indicate that RMS normalization is important for intrinsic reward methods that use random networks, since the lack of normalization can result in the embeddings produced by the random networks carrying very little information about the inputs [25].

Q2: The impact of reward normalization.

Similarly to Q1, reward normalization can have a large impact when using deep neural networks to compute the intrinsic rewards, since the scale of these rewards can be arbitrary and vary significantly over time. To mitigate the non-stationarity of intrinsic rewards, in Q2, we compare three normalization approaches for the reward outputs of the intrinsic reward modules: (1) Min-Max normalization, (2) using an RMS of the standard deviation, and (3) no reward normalization.

Reward normalization smooths the optimization process, which can be beneficial for stability but can lead to slower convergence [25]. Our findings show that almost all intrinsic rewards critically require some form of reward normalization, as agents fail to explore without normalized rewards. Importantly, the latter applies to all the environments that we experiment with. Additionally, while RMS is generally the default strategy for reward normalization, our results in Figure 4.19 show that Min-Max normalization is a more robust option in *SMB*, improving the performance and

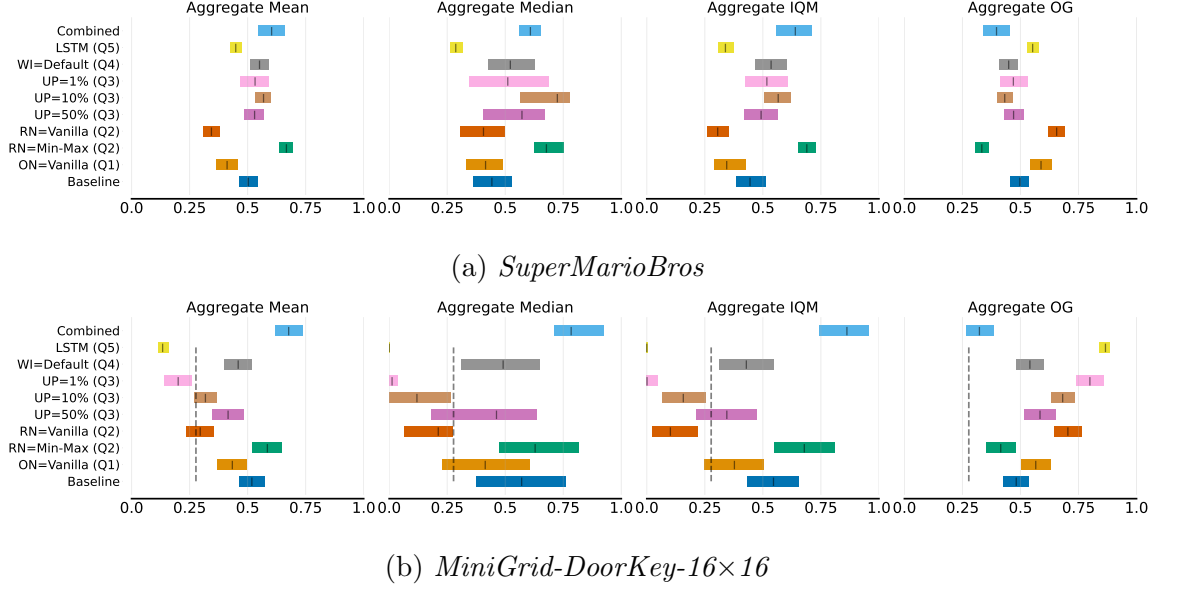


Figure 4.19: Aggregated performance of the eight intrinsic rewards with different low-level hyperparameters over 10 random seeds. The vertical dashed line represents the performance of the extrinsic agent, which only has access to the task rewards. Here, **ON** is the observation normalization, **RN** is the reward normalization, **UP** is the update proportion, **WI** is the weight initialization, **IQM** is the interquartile mean, **OG** is the optimality gap (lower is better), and **Combined** refers to the results of using the best hyperparameters gathered in each question. All the computation is performed using the Rliable [2] library.

reducing the variance of the majority of the methods.

Q3: The co-learning dynamics of policies and auxiliary tasks for intrinsic rewards.

Optimizing intrinsic rewards in deep RL often involves training additional networks for auxiliary tasks (*e.g.*, the predictor network in RND, the inverse dynamics encoder in ICM, and the forward dynamics encoders in Disagreement). However, managing the co-learning dynamics of the auxiliary networks and policies is challenging. In Q3, we explore three update strategies for the auxiliary networks in the intrinsic reward

modules: (1) updating them at the same frequency as the policy, (2) updating them 50% of the time, (3) updating them 10% of the time, and (4) updating them 1% of the time. This comparison sheds light on the trade-off between the number of gradient updates in the auxiliary networks and the performance of the policy. Additionally, lower update frequencies have the benefit of reducing computational overhead and training time by limiting the number of gradient updates required.

Our findings indicate that the auxiliary networks generally perform robustly across the range of update frequencies studied. Additionally, there is no clear configuration that seems generally better for all intrinsic rewards across environments, rendering this implementation detail worth tuning for specific environments.

Q4: The impact of weight initialization.

Weight initialization plays a crucial role in optimizing deep neural networks, enabling faster convergence. In Q4, we compare two approaches for weight initialization in the auxiliary networks of the intrinsic reward modules: (1) orthogonal weight initialization and (2) uniform weight initialization (PyTorch’s default). Note that again, the policy and value networks remain unchanged.

Our results highlight the importance of weight initialization in intrinsically-motivated RL. Specifically, we found that orthogonal weight initialization is beneficial for most intrinsic rewards, regardless of their specific optimization tasks (*e.g.*, inverse dynamics, forward dynamics), and even in random networks (*e.g.*, RND and RE3). This benefit is evidenced by reduced variance in episode returns and generally higher mean returns. This observation aligns with previous research indicating that orthogonal weight initialization can improve performance stability in deep RL agents [79, 53]. Importantly, RND is the intrinsic reward method that shows the highest variability for this implementation detail, where orthogonal weight initialization works better in *SMB* but worse than uniform initialization in *MGD*.

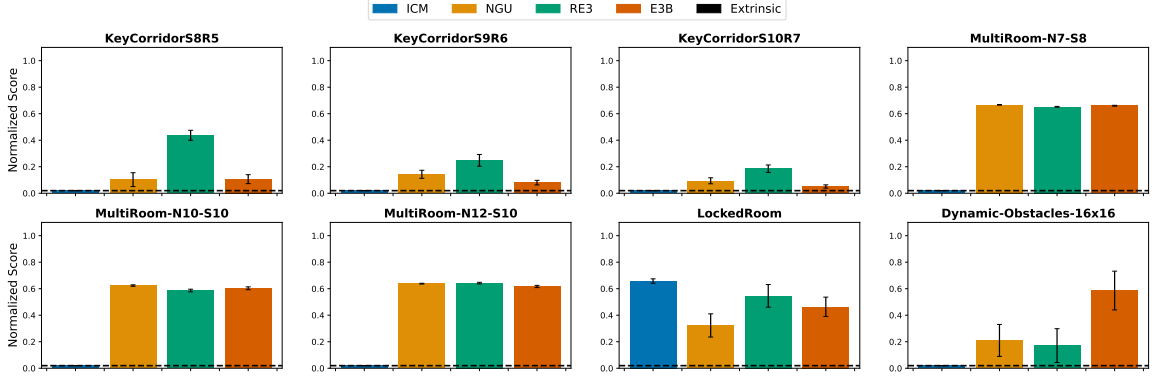


Figure 4.20: Performance of four selected intrinsic rewards in RLeXplore on the top eight most challenging tasks of the MGD suite. The solid line and shaded regions represent the mean and standard deviation computed with five random seeds, respectively.

Q5: Is memory required to optimize intrinsic rewards?

In Q5, we investigate whether the intrinsic rewards included in RLeXplore benefit from memory-enabled architectures. We compare the optimization of intrinsic rewards using a vanilla policy network and one equipped with a long short-term memory (LSTM) [74] module, while keeping PPO as the RL backbone algorithm.

Some intrinsic reward methods exhibit significantly lower performance when using LSTM policies. This observation aligns with the fact that LSTMs provide episodic context to policies, whereas most intrinsic reward methods define exploration as a global problem.

Finally, we utilize the best-performing implementation details observed in Q1-5 to experiment with the set of most challenging exploration tasks from *MiniGrid*. Our results in Figure 4.20 show that with our implementations of intrinsic rewards in RLeXplore, researchers can make progress in training RL agents in challenging tasks where vanilla RL agents are unable to learn due to the sparsity of the task rewards. In summary, by systematically addressing the implementation details, our work sig-

nificantly enhances the reproducibility of intrinsic reward methods. These thoughtful design choices not only improve performance but also ensure that our implementations can be reliably reproduced and generalized across various environments.

4.3.5.2 Combination of Intrinsic and Extrinsic Rewards

Q6: Joint optimization of intrinsic and extrinsic rewards.

In sparse-reward environments, the objective is for agents to explore the state space by optimizing intrinsic rewards until they discover the task rewards, at which point they should focus solely on optimizing the task rewards. However, many intrinsically motivated RL applications naively optimize the sum of intrinsic and extrinsic rewards, potentially leading to learning fuzzy value functions and suboptimal policies [27]. In this section, we compare this common approach with learning two separate value functions, one for each reward function [25]. The advantages of the latter include the ability to disentangle the effects of intrinsic and extrinsic rewards on the agent’s behaviour, leading to cleaner learning dynamics and potentially more efficient exploration. In these settings, both value functions are used during the advantage estimation phase of PPO. Specifically, we compute separate GAE values - one using the intrinsic value function and one using the extrinsic value function. The resulting advantages are then summed to compute the policy loss term for PPO. This separation facilitates more accurate advantage estimates for each reward type, leading to improved learning dynamics.

For this analysis, we used the *Procggen-Maze* task [36] as a sparse-reward benchmark. RL agents often struggle to learn meaningful behaviours from the extrinsic reward alone in this task. We evaluate different variants of the task (*e.g.*, 1 maze vs. 200 mazes) to examine singleton versus contextual MDPs. We note that in our framework, we do not provide different context information to the agents for singleton versus contextual MDPs (*e.g.*, the context ID). We refer to these frameworks to formalize

the agent-environment interaction when the environment remains static throughout training (*i.e.*, singleton - 1 maze) versus when it varies at each episode (*i.e.*, contextual - a different maze at each episode).

Figure 4.21 demonstrates that learning two separate value functions [80], which we refer to as the *TwoHead* architecture, outperforms the naive approach of simply adding the two rewards in the complex sparse-reward environment of *Procgen-Maze*, both in singleton and contextual settings. Importantly, all methods outperform the extrinsic agent, especially in the *1 Maze* environment.

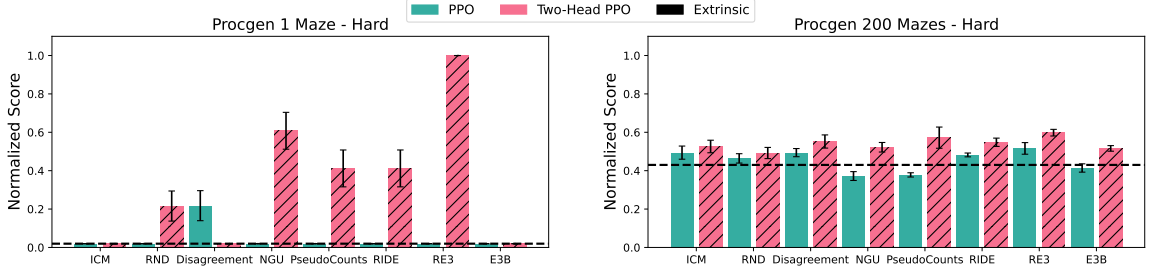


Figure 4.21: (Left) During training, the extrinsic agent struggles to find the goal in the selected Maze, resulting in a reward of 0. While some intrinsic reward methods yield occasional non-zero rewards, the methods perform significantly better when intrinsic and extrinsic value estimation are decoupled using two distinct value heads in the agent’s network. (Right) In the Procgen variant, where each maze represents a unique level, the baseline extrinsic agent achieves the goal 50% of the time, and intrinsic rewards don’t outperform the baseline significantly. We note that the presence of easier levels, where the goal may occasionally be near the agent’s starting point, results in generally less sparse rewards and an easier task to learn.

4.3.5.3 Unlocking the Potential of Intrinsic Rewards

Q1-6 extensively discuss the tuning of intrinsic rewards under both normal and reward-free scenarios, revealing significant insights into the optimization processes. However, we aim to delve deeper into the capabilities of intrinsic rewards to address

the evolving challenges in the RL community. Specifically, in Q7, we investigate recent developments in the exploration literature in RL, such as combined intrinsic rewards and exploration in contextual MDPs. For our experiments, we use the *SMB-RandomStages* environment variant, where agents play a different level in the game at each episode. Our results indicate that the recent developments in combined intrinsic rewards merit further research, as we demonstrate that such methods can enable agents to learn exploratory behaviours of exceptional quality in both singleton and contextual MDPs.

Q7: The performance of mixed intrinsic rewards.

We run experiments using all the levels in the game of *SMB*, and we sample them uniformly during training. As in Q1-5, we do not use the extrinsic reward for training the agents but use it as an evaluation metric to show how much agents actively explore the environment.

Our results show that combined objectives enable emergent behaviours of much better quality than single objectives. Interestingly, E3B and RIDE are the best-performing single objectives, and E3B+RIDE also achieves the highest performance among all the combinations. Similarly, RND and ICM, combined with other intrinsic rewards, outperform their original performance. This indicates that different intrinsic rewards can provide orthogonal gains that can be leveraged together.

4.3.6 Conclusion

In this section, we introduce RLeXplore, a comprehensive open-source repository that not only implements state-of-the-art intrinsic rewards but also provides a systematic evaluation framework for understanding their impact on agent performance. Our results show that with RLeXplore, RL agents can learn emergent behaviors

4.3. RLeXplore: Accelerating Research in Intrinsically-Motivated RL

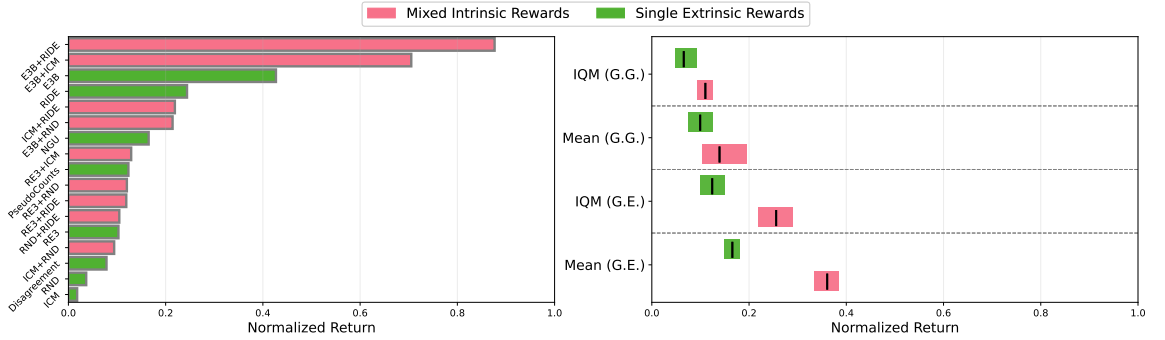


Figure 4.22: (Left) The performance ranking of single and mixed intrinsic rewards on the *SuperMarioBrosRandomLevels*. As expected, episodic bonuses (such as E3B and RIDE) demonstrate superior performance, attributed to the environment’s non-singleton MDP nature. (Right) Overall performance comparisons between the single and mixed intrinsic rewards. Here, **G.E.** denotes the six ”global+episodic” combinations, such as E3B+RND and E3B+ICM. **G.G.** denotes the three ”global+global” combinations, such as RND+ICM and RND+RIDE.

autonomously, solving multiple levels of *SuperMarioBros* without task rewards. Additionally, we show that intrinsic rewards enable RL agents to obtain great performance on complex sparse-reward tasks like *Procgen-Maze*, *MiniGrid*, the *ALE-5 hard-exploration tasks*, and *Ant-UMaze*. Finally, RLeXplore facilitates further research in mixed intrinsic rewards [70], uncovering the potential of such methods.

Through our study, we emphasize the importance of thoughtful implementation design, demonstrating that well-considered approaches lead to significant performance gains over naive implementations. Our contributions extend to establishing standardized practices for implementing and optimizing intrinsic rewards, laying the groundwork for future advancements in intrinsically motivated RL.

RLeXplore is designed to benchmark end-to-end intrinsic reward methods. These end-to-end methods are more commonly used and under-evaluated by the community. Skill-based algorithms, which typically involve separate phases for skill discovery

and skill learning, are more complex and left for future work. For an alternative perspective that includes skill-based approaches, we refer readers to the unsupervised RL benchmark by [102]. Additionally, RLeXplore was designed with accessibility in mind, ensuring that the implemented methods can be run on standard computational resources by any researcher. To maintain this accessibility, we have not included more complex and potentially powerful methods like BYOL-Explore [63], or RECODE [89]. These methods are not open-source and have been optimized exclusively with non-open-source RL algorithms, which further limits their integration into RLeXplore.

Chapter 5

Task-Aware Data Exploitation for RL Acceleration

5.1 ADEPT: Adaptive Data Exploitation in Deep RL

5.1.1 Introduction

Deep reinforcement learning (RL) has achieved remarkable success across diverse domains such as complex games [137, 179, 199], algorithmic innovation [55, 127], large language models (LLMs) [146], and chip design [60]. However, promoting **data efficiency** and **generalization** remains a long-standing challenge in deep RL, especially when learning from complex environments with high-dimensional observations (*e.g.*, images). The agents often require millions of interactions with their environment, resulting in substantial computational overhead and restricting applicability in scenarios where data collection is costly or impractical. Moreover, the learned policy frequently struggles to adapt to dynamic environments, where minor variations in

tasks or conditions can significantly degrade its performance.

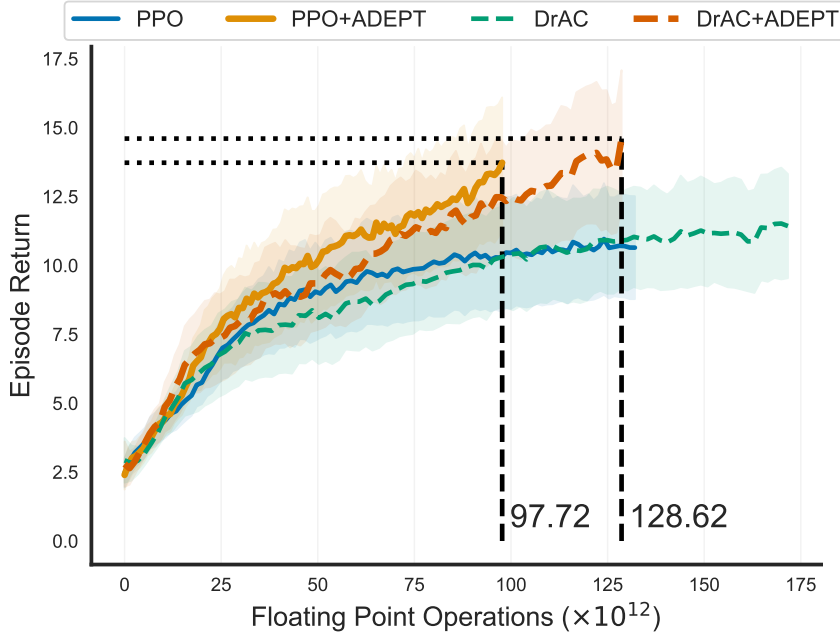


Figure 5.1: Aggregated training performance and computational overhead comparison on the Procgen benchmark. ADEPT serves as a plug-and-play module that enhances RL algorithms, significantly improving data efficiency and reducing computational costs.

To tackle the data efficiency problem, research has primarily focused on two key aspects: **data acquisition** and **data exploitation**. Data acquisition aims to maximize the quality and diversity of the data collected during interactions with the environment, while data exploitation seeks to optimize the utility of this data to enhance learning efficiency. Prominent techniques include data augmentation [149, 213, 71, 214, 101, 100], efficient experience replay [168, 75, 5], distributed training [54, 14], and environment acceleration [153, 125, 203]. For example, [25] proposed RND that utilizes the prediction error against a fixed network as intrinsic rewards, enabling structured and efficient exploration. In contrast, RAD [101] applies simple transfor-

mations to input observations during training, improving both data efficiency and generalization. On the exploitation side, [168] developed prioritized experience replay (PER), which enhances learning efficiency by sampling high-priority experiences more frequently during training. However, these methods struggle to balance data efficiency and computational efficiency. Techniques like data augmentation often introduce auxiliary models, storage, and optimization objectives, thereby significantly increasing computational overhead [9, 207]. Moreover, they often lack guarantees for optimizing long-term returns. For instance, intrinsic reward approaches suffer from the policy-invariance problem, in which exploration driven by intrinsic rewards may diverge from the optimal policy.

Beyond data efficiency, generalization represents another critical challenge in deep RL, as agents often overfit to their training environments [95, 4, 159]. To address this issue, [38] proposed a phasic policy gradient (PPG), which explores decoupling the representation learning of policy and value networks. PPG also incorporates an auxiliary learning phase to distill the value function and constrain the policy, thereby enhancing generalization. Similarly, [159] proposed decoupled advantage actor-critic (DAAC), which leverages decoupled policy and value networks while eliminating the need for auxiliary learning. Both PPG and DAAC can achieve superior generalization performance on the Procgen benchmark [36]. However, they rely on sophisticated hyperparameter tuning (*e.g.*, the number of update epochs of the two separated networks), which limits their adaptability across diverse scenarios. Moreover, they lack mechanisms to dynamically adjust to different learning stages, further constraining their applicability.

Inspired by the discussions above, we aim to enhance the data efficiency and generalization of RL agents while minimizing computational overhead. To that end, we propose a novel framework entitled **ADEPT: Adaptive Data ExPloiTation**, which incorporates three scheduling algorithms to assist RL algorithms. Our main contributions are summarized as follows:

- We propose to adaptively control the intensity of data utilization (IDU) across different tasks and learning stages. This scheduling process is formulated as a multi-armed bandit (MAB) problem, in which a set of IDU values represents the arms. ADEPT can automatically select the appropriate IDU value using adaptive and rule-based scheduling strategies, significantly improving the data efficiency.
- Through the IDU scheduling, ADEPT effectively eliminates redundant updates and is found to preserve the richness of policy representation, thereby improving the generalization of RL agents while reducing the training overhead. Notably, ADEPT requires no auxiliary learning processes, making it lightweight and broadly compatible with existing RL algorithms.
- Finally, we evaluate ADEPT on diverse benchmarks, including Procgen (sixteen procedurally-generated environments), MiniGrid (environments with sparse rewards), and DeepMind Control Suite (robotics environments with continuous action space). Extensive experiments demonstrate that ADEPT consistently improves agents' performance and achieves substantial reductions in computational cost.

5.1.2 Adaptive Data Exploitation

In this section, we propose the ADEPT framework to enhance data efficiency and generalization while reducing computational overhead during training. For on-policy algorithms like PPO [171], IDU can be explicitly interpreted as the **number of update epochs (NUE, denoted by K)** in the update phase, while the **frequency of training (FoT denoted by F)** ratio applies for off-policy algorithms. For the sake of convenience, we conduct the analysis considering the on-policy setting in the following contents. However, the proposed methods can also be applied to off-policy settings.

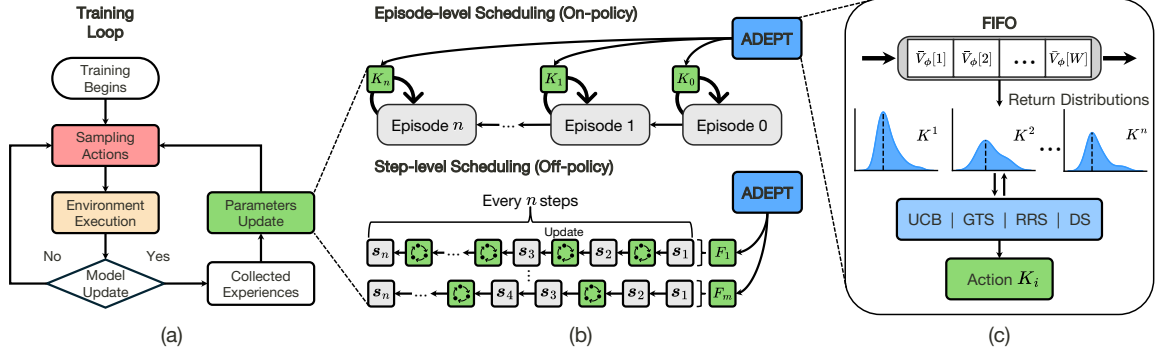


Figure 5.2: Overview of the ADEPT framework. (a) A typical workflow of RL algorithms. (b) ADEPT optimizes data exploitation by adaptively adjusting IDU in the update phase, *e.g.*, the number of update epochs in the on-policy setting. (c) ADEPT formulates the IDU scheduling as a MAB problem, in which the estimated task return is leveraged as the utility function to evaluate the IDU values. Currently, we provide 4 scheduling strategies, including adaptive and rule-based approaches.

IDU versus Data Efficiency. Figure 5.3 illustrates an intuitive experiment that trains PPO [171] agents on the Procgen benchmark using increasing NUE values. Procgen contains 16 procedurally-generated environments with high dynamics, which pose great challenges to data efficiency and generalization. As the figure shows, increasing the NUE in the learning process initially results in better performance, as the agent can exploit the available data more effectively. However, as the NUE continues to increase, the improvement in performance diminishes and ultimately degrades the performance. Excessive updates introduce redundancy and lead to inefficiencies in data utilization. Thus, optimizing IDU by selecting an appropriate NUE value helps to improve data efficiency, ensuring that each data point contributes effectively to the learning process without unnecessary repetition.

IDU versus Generalization. While increasing IDU ensures the sampled data is fully utilized, excessive updates can lead to overfitting and significantly degrade the

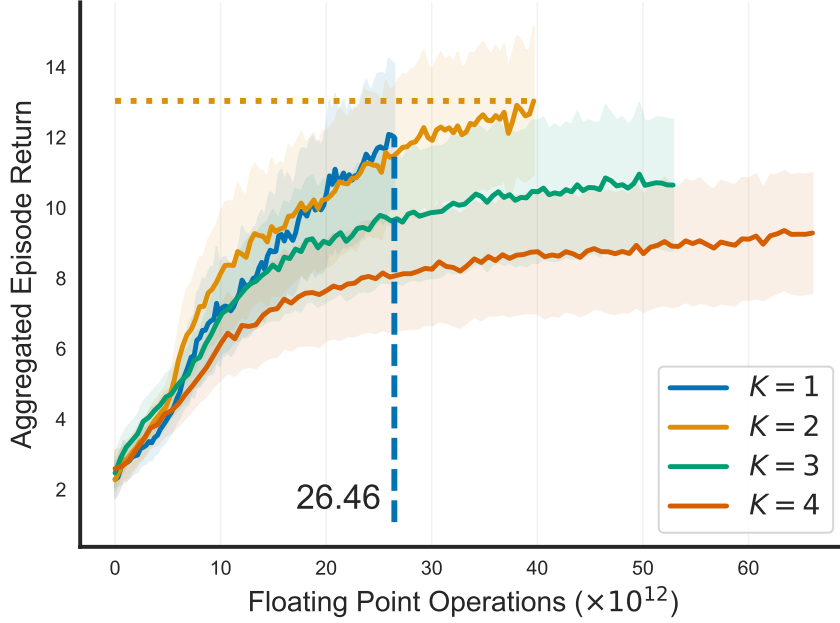


Figure 5.3: Aggregated training performance comparison on the Procgen benchmark with increasing NUE values.

agent’s generalization ability. This is evident in Figure 5.3, where $K = 4$ causes the agents’ learning to stagnate. This issue is particularly notable in non-stationary environments like Procgen, where agents must adapt to constantly changing conditions and reduce reliance on specific data. In the latter experiments section, we will show that controlling the IDU can effectively preserve the plasticity and representation richness of the policy network, thereby improving the agents’ generalization ability.

IDU versus Overhead. Figure 5.3 clearly illustrates that as the NUE increases, the computational overhead grows significantly. Excessive updates lead to substantial computational redundancy without proportional improvements in performance. By reducing unnecessary updates, data efficiency and generalization can be enhanced while simultaneously lowering computational overhead.

Therefore, the core objective of ADEPT is to adaptively control the IDU across different tasks and learning stages, striking an appropriate balance between data

Algorithm 3: ADEPT for On-policy

 Setting

- 1 Initialize the policy network π_θ and value network V_ϕ ;
 - 2 Initialize a set $\mathcal{K}$ of NUE values and corresponding hyperparameters of ADEPT;
 - 3 **for** *each episode* e **do**
 - 4 Sample rollouts using the policy π_θ ;
 - 5 Select K_e using ADEPT;
 - 6 Update the policy network and value network following K_e ;
 - 7 Update ADEPT if necessary;
-

Algorithm 4: ADEPT for Off-policy

 Setting

- 1 Initialize the policy network π_θ and value network Q_ϕ ;
 - 2 Initialize a set $\mathcal{F}$ of FoT values and corresponding hyperparameters of ADEPT;
 - 3 **repeat**
 - 4 **for** *every* T **steps do**
 - 5 Select a F using ADEPT;
 - 6 Update ADEPT if necessary;
 - 7 Perform update following F in the next T steps;
 - 8 **until** *training is over*;
-

efficiency, generalization, and overhead. This scheduling process can be considered a MAB problem that adaptively adjusts the IDU, and the objective is to maximize the long-term return evaluated by the task reward function. In the following, we introduce four specific scheduling methods to solve the defined MAB problem.

5.1.2.1 Adaptive Scheduling

ADEPT-U. We first leverage the UCB [8] to solve the defined MAB problem. Given a set of NUE values $\mathcal{K} = \{K^1, K^2, \dots, K^n\}$, for the i -th episode, we select a K_i from $\mathcal{K}$ using the following policy:

$$\underbrace{K_i = \operatorname{argmax}_{K \in \mathcal{K}} \left[U_i(K) + c \sqrt{\frac{\log i}{N_i(K)}} \right]}_{\text{Scheduling Policy}}, \quad \underbrace{U_i(K) = \frac{1}{W} \sum_{j=1}^W \bar{V}_\phi[j]}_{\text{Utility Evaluation}}, \quad (5.1)$$

where $N_i(K)$ is the number of times that K has been chosen before the i -th episode, and c is the exploration coefficient. Before the i -th update, we select a K using Eq. (5.1) and update the model for K epochs. Then, the counter is updated by $N_i(K) = N_i(K) + 1$. Next, we collect rollouts with the new policy and update the utility function $U_i(K)$, which is computed as a sliding window average of the past mean returns obtained by the agent after using K . Here, W is the window length and $\bar{V}_\phi[j]$ is the j -th past mean return stored in the queue.

The key insight of ADEPT-U is to leverage the value function as a performance indicator to evaluate different K values, ensuring the selection of NUE always improves the performance. Note that ADEPT-U does not directly participate in the optimization of the value function.

ADEPT-G. Similar to the UCB-based scheduling, we further introduce **Gaussian Thompson sampling (GTS)** [193] to solve the same MAB problem by modeling the return distribution of each K as a Gaussian distribution. GTS allows for a more flexible exploration pattern that adapts dynamically to new information compared to the fixed confidence bound strategy in UCB. For each episode, GTS samples from the distributions corresponding to all NUE candidates and selects the one with the highest sampled value:

$$\underbrace{K_i = \operatorname{argmax}_{K \in \mathcal{K}} \mathcal{N}(\mu_i(K), \sigma_i^2(K))}_{\text{Scheduling Policy}}, \quad (5.2)$$

where $\mu_i(K)$ and $\sigma_i^2(K)$ are the mean and variance of the reward distribution for K . Then the reward distributions are updated as follows

$$\underbrace{\mu_{i+1}(K) = \mu_i(K) + \eta \frac{\Delta_i(K)}{\hat{N}_i(K)}, \sigma_{i+1}^2(K) = \frac{N_i(K)\sigma_i^2(K) + \Delta_i^2(K)}{\hat{N}_i(K)}}_{\text{Incremental Update}}, \quad (5.3)$$

where $\Delta_i(K) = U_i(K) - \mu_i(K)$, $\hat{N}_i(K) = N_i(K) + 1$, and η is a step size.

5.1.2.2 Rule-based Scheduling

The two adaptive scheduling methods rely on the quality of the value function learning, as their effectiveness is closely tied to the accuracy of the value predictions. We further introduce two rule-based methods to manage the IDU without direct dependence on the value function.

ADEPT-R. We first employ Round-Robin scheduling (RRS) [6] to ensure that each candidate in $\mathcal{K}$ is selected in a cyclical order, giving equal opportunity to all options without bias. This strategy is widely used in various domains, such as network packet scheduling, load balancing in distributed systems, and time-sharing in resource management. The selection at the i -th episode follows: $K_i = \mathcal{K}[(i \bmod |\mathcal{K}|) + 1]$, where $|\mathcal{K}|$ is the cardinality of the set $\mathcal{K}$.

ADEPT-D. Another straightforward way is the decay scheduling, in which the K value is gradually reduced over time. This approach allows the system to initially make frequent updates for faster learning and then decrease the frequency as the model approaches convergence, thereby improving data efficiency while reducing computational overhead.

Finally, we summarize the general workflow of combining ADEPT with both on-policy and off-policy RL algorithms in Algorithm 3 and Algorithm 4.

5.1.3 Experiments

- **Q1:** Can ADEPT improve data efficiency versus using a fixed IDU?
- **Q2:** Can ADEPT achieve higher generalization performance in procedurally-generated environments?
- **Q3:** Can ADEPT reduce the overall computational overhead?
- **Q4:** How does ADEPT fundamentally affect the learned policies?

- **Q5:** What are the detailed decision processes of ADEPT?
- **Q6:** How does ADEPT behave in sparse-rewards environments and continuous control tasks?

5.1.3.1 Experimental Setup

a) Benchmark Selection: We first evaluate the ADEPT using the Procgen benchmark [36], which contains sixteen procedurally-generated environments. We select Procgen for two reasons. First, Procgen is similar to the arcade learning environment (ALE) benchmark [16] that requires the agent to learn motor control directly from images and presents a clear challenge to the agent’s data efficiency. Moreover, Procgen provides procedurally generated levels to evaluate the agent’s generalization ability with a well-designed protocol. All the environments use a discrete fifteen-dimensional action space and generate $(64, 64, 3)$ RGB observations. We use the *easy* mode and train the agents on 200 levels before testing them on the full distribution of levels. Furthermore, we introduce the MiniGrid [34], and DeepMind Control Suite (DMC) [191] to test ADEPT in sparse-rewards environments and continuous control tasks.

b) Algorithmic Baselines: For the on-policy RL algorithm, we select the proximal policy optimization (PPO) [171] and data regularized actor-critic (DrAC) [160] as the baselines. PPO is a representative algorithm that produces considerable performance on most existing RL benchmarks, while DrAC integrates data augmentation techniques into AC algorithms and significantly improves the agent’s generalization ability in procedurally-generated environments. For the off-policy RL algorithm, we select the soft actor-critic (SAC) [65] as the baseline, a powerful algorithm for continuous controls, which maximizes a weighted objective of the reward and the policy entropy.

c) IDU Candidates: [36] and [160] reported the overall best hyperparameters for the two algorithms on the Procgen benchmark. Both PPO and DrAC update their pa-

rameters for 3 epochs after each episode, so we conduct experiments with $\mathcal{K} = \{3, 2, 1\}$ and $\mathcal{K} = \{5, 3, 2, 1\}$. For the first set, we aim to assess whether ADEPT can enhance RL algorithms while reducing computational overhead. In contrast, the second set allows us to explore whether a broader range of NUE values further improves performance. For MiniGrid experiments, following the implementation of [34], we evaluate PPO+ADEPT-R and PPO+ADEPT-U using three NUE sets: $\{4, 2, 1\}$, $\{6, 4\}$, and $\{8, 4\}$. Finally, for the DMC experiments, we test the combinations of SAC and four DGO methods using two FoT sets: $\mathcal{F} = \{1, 2, 3\}$ and $\mathcal{F} = \{1, 2, 3, 5\}$.

d) Evaluation Metrics: We evaluate the data efficiency and generalization of each method using three metrics: (i) the average floating point operations (FLOPS) over 16 environments and all the runs, and the calculation process is depicted in [44], (ii) the aggregated mean scores on 200 levels, and (iii) the aggregated mean, median, interquartile mean (IQM), and optimality gap (OG) [2] on the full distribution of levels. Note that the score for each method in each environment is computed as the average episode return over 100 episodes and 5 random seeds.

5.1.3.2 Results Analysis

The following results analysis is performed based on the predefined research questions.

a) Data Efficiency Comparison: Figure 5.4 and Figure 5.5 compare the data efficiency and computational overhead of vanilla PPO, DrAC, and their combinations with four ADEPT methods on the Procgen benchmark. By alternating the NUE values from $\mathcal{K}$, both ADEPT-R and ADEPT-D achieve significant performance improvements over vanilla PPO, particularly in the *BigFish*, *CaveFlyer*, and *Chaser* environments. The average computational overhead of PPO+ADEPT-R and PPO+ADEPT-D is reduced to 70% and 80% of the vanilla PPO’s overhead, respectively. In contrast, ADEPT-U and ADEPT-G also outperform the baseline RL algorithms and achieve more extreme overhead compression, especially in the *Plunder* environment. Specifi-

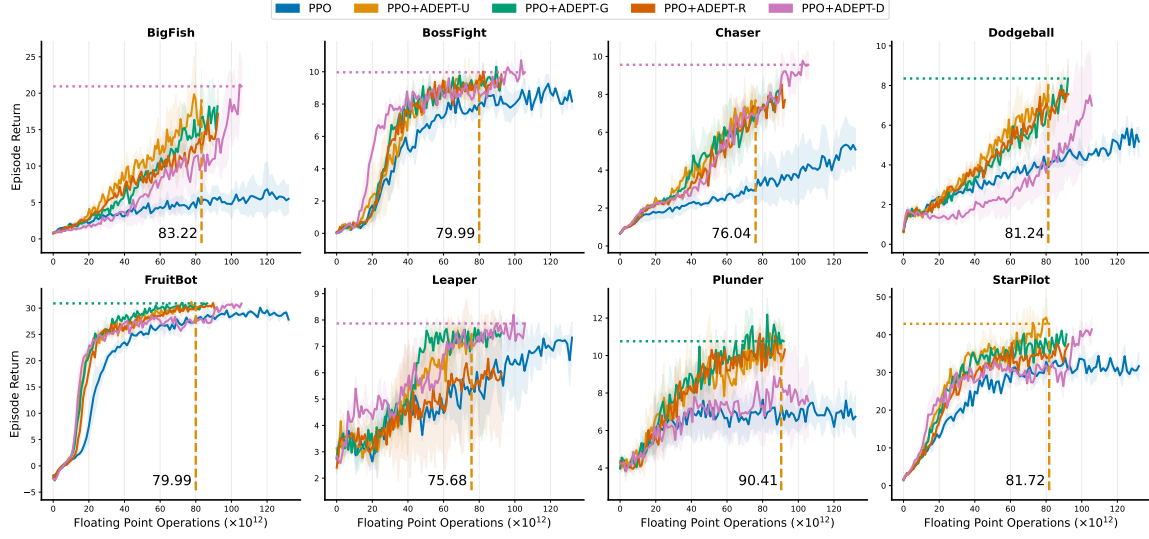


Figure 5.4: Training performance and computational overhead comparison of the PPO and its combinations with ADEPT on eight Procgen environments. The solid line and shaded regions represent the mean and standard deviation, respectively, across five runs. Note that the dotted line and dashed line represent the highest score and the lowest overhead, respectively.

cally, the average computational overhead of DrAC+ADEPT-U and DrAC+ADEPT-G is reduced to 69.1% and 68.7% of the overhead of the vanilla DrAC agent. The experimental results for PPO and DrAC demonstrate that ADEPT can significantly improve the data efficiency of RL algorithms and reduce computational overhead.

b) Generalization Performance on Procgen: We further evaluate the generalization performance of PPO, DrAC, and their combinations with ADEPT on the Procgen benchmark, as illustrated in Figure 5.6. All four ADEPT methods significantly promote the generalization performance of the two baseline RL algorithms. Surprisingly, PPO+ADEPT-D achieves the highest score among all the combinations with a simple decay scheduling strategy. To study the fundamental effect of ADEPT, we follow [46] and compute the ratio of dead units (RDU) of the feature layer in the learned policy network. As shown in Figure 5.7, ADEPT can effectively restrain

5.1. ADEPT: Adaptive Data Exploitation in Deep RL

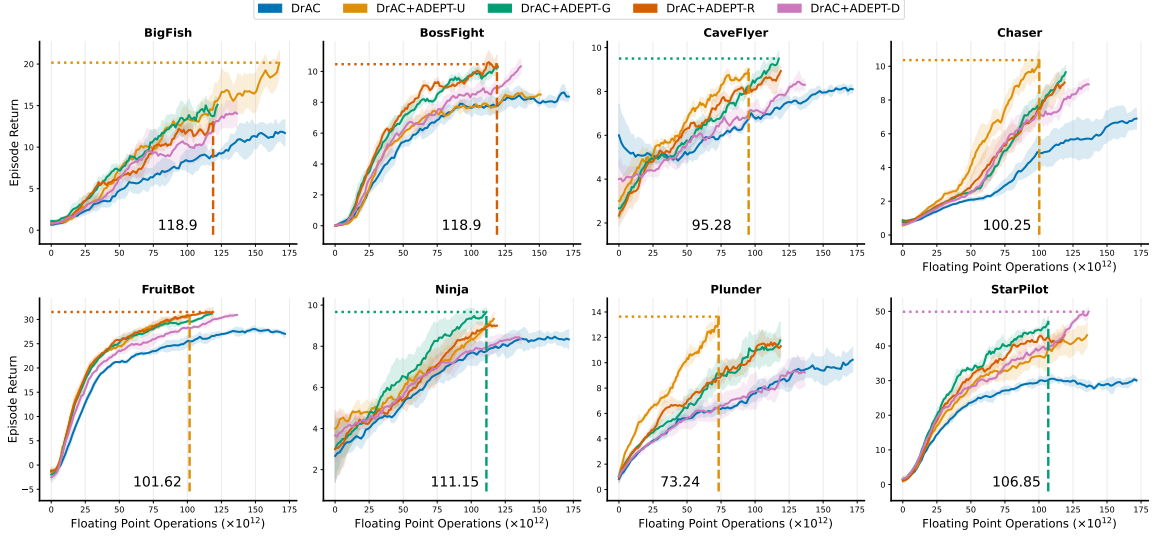


Figure 5.5: Training performance and computational overhead comparison of the DrAC and its combinations with ADEPT on eight Progen environments. The solid line and shaded regions represent the mean and standard deviation, respectively, across five runs. Note that the dotted line and dashed line represent the highest score and the lowest overhead, respectively.

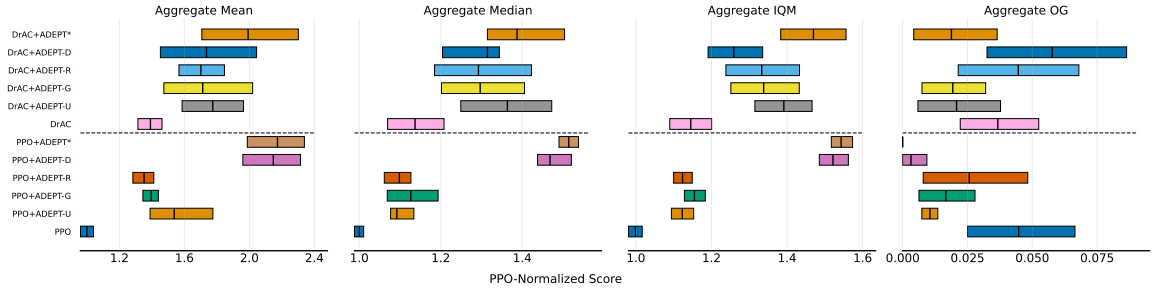


Figure 5.6: Aggregated performance of the PPO, DrAC, and their combinations with ADEPT on the test levels of the Progen benchmark. All the scores are normalized by the corresponding PPO scores, and bars indicate 95% confidence intervals computed using stratified bootstrapping over five random seeds. Note that * represents the best scores gathered from all four ADEPT methods.

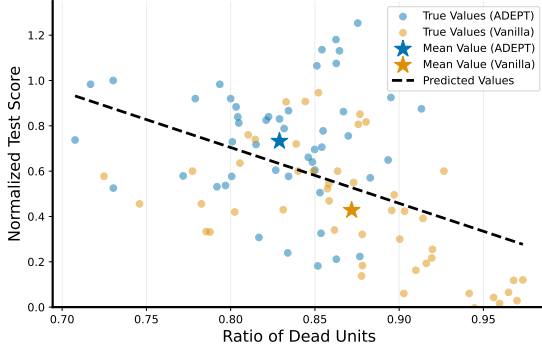


Figure 5.7: Ratio of dead units (RDU) of the policy networks trained by the vanilla PPO, DrAC, and their combinations with ADEPT. ADEPT significantly decreases the RDU and effectively preserves the richness of representation, thereby improving the agents’ generalization ability.

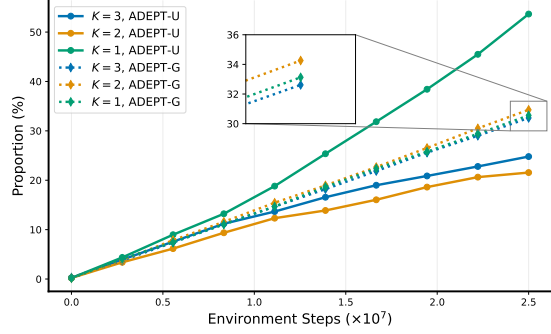


Figure 5.8: The aggregated decision processes of ADEPT-U and ADEPT-G for PPO on the eight selected Procgen environments. ADEPT effectively eliminates redundant updates, thereby improving data efficiency and reducing computational overhead.

the RDU and preserve the richness of policy representation, thereby enhancing RL agents’ generalization ability.

c) Analysis of the Decision Process: Next, we analyze the detailed decision processes of ADEPT. Figure 5.8 illustrates the cumulative proportion of each NUE value selected during the whole training of the PPO experiments. It is clear that ADEPT-U primarily selects $K = 1$, while $K = 2$ and $K = 3$ each account for approximately 20%. In contrast, ADEPT-G distributes its selections more evenly across all NUE values, reflecting a dynamic scheduling strategy enabled by its incremental update mechanism. These findings demonstrate that varying tasks and learning stages benefit from adaptive data utilization, and that ADEPT consistently selects the most appropriate NUE values to maximize data efficiency. Additionally, ADEPT exhibits a similar pattern to DrAC as PPO.

d) Ablation Studies: We also conducted a number of ablation experiments to study the importance of hyperparameters used in ADEPT, such as the exploration coefficient c and the window length W of ADEPT-U. We found that $c = 5.0$ and $W = 10$ are the overall best for ADEPT-U in both PPO and DrAC experiments. For PPO+ADEPT-G, the best options are $\eta = 1.0$ and $w = 50$, while for DrAC+ADEPT-G, they are $\eta = 0.1$ and $w = 50$. Additionally, we examine ADEPT with different sets of NUE values, as shown in Figure 5.9. The results demonstrate that a larger $\mathcal{K}$ can only enhance ADEPT in a few environments but degrade overall performance on the Procgen benchmark.

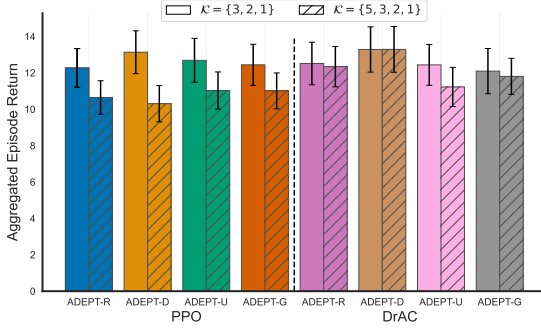


Figure 5.9: Aggregated training performance comparison of ADEPT with different sets of NUE values. The mean and standard error are computed across all the environments.

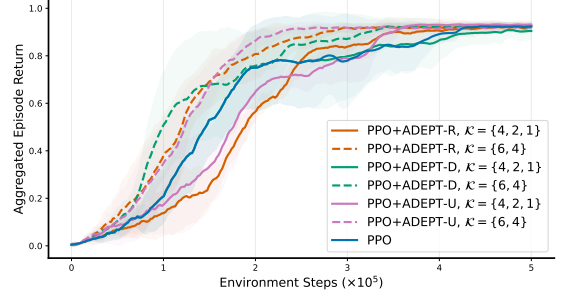


Figure 5.10: Aggregated performance of three ADEPT methods on the MiniGrid benchmark. The mean and standard deviation are computed across all the environments.

e) Performance in Sparse-rewards Environments: Additionally, we evaluate ADEPT on the MiniGrid benchmark with sparse-rewards and goal-oriented environments. Specifically, we conduct experiments using *DoorKey-6 $\times$ 6*, *LavaGapS7*, and *Empty-16 $\times$ 16*. Figure 5.10 illustrates the aggregated learning curves of the vanilla PPO agent and its combinations with 3 ADEPT methods using various sets of NUE candidates. It is clear that ADEPT takes fewer environment steps to solve tasks, highlighting its ability to accelerate RL algorithms in both dense- and sparse-reward

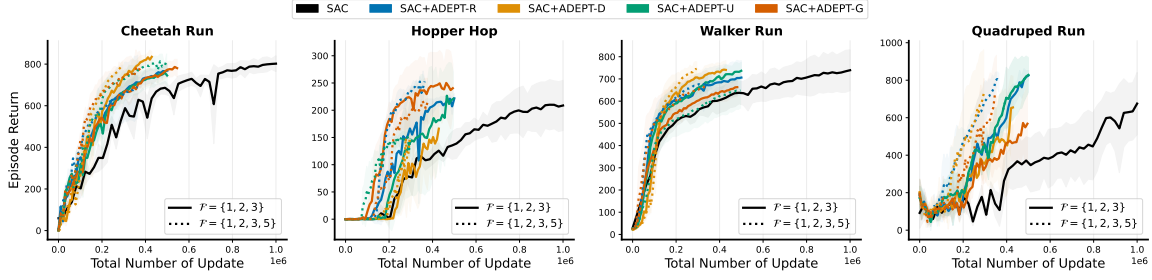


Figure 5.11: Performance of the SAC and its combinations with four ADEPT methods on the DMC benchmark. The mean and standard deviation are computed using five random seeds.

settings.

f) Performance on Continuous Control Tasks: Finally, we evaluate ADEPT on the DMC benchmark with continuous control tasks. Four environments are utilized, namely *CheetahRun*, *HopperHop*, *WalkerRun*, and *QuadrupedRun*. Figure 5.11 illustrates the learning curves of the vanilla SAC agent and its combinations with four ADEPT methods. ADEPT outperforms the SAC agent with fixed FoT values, especially in *HopperHop* and *QuadrupedRun* environments. These results underscore the effectiveness of ADEPT in enhancing both on-policy and off-policy RL algorithms across both discrete and continuous control tasks.

5.1.4 Conclusion

In this section, we propose ADEPT, a lightweight and effective framework for improving data efficiency and generalization, and reducing computational overhead in deep RL. By adaptively managing the intensity of data utilization during training, ADEPT optimizes data exploitation, eliminates redundant updates, and preserves representation diversity. Extensive experiments across diverse benchmarks demonstrate that ADEPT consistently improves learning performance for both on-policy

and off-policy algorithms, providing a practical solution for data-efficient RL.

Still, there are currently remaining limitations to this work. Specifically, the decision-making processes of ADEPT-U and ADEPT-G rely on the task return predicted by the value network, and inaccurate predictions directly affect scheduling quality. Furthermore, oscillatory scheduling may lead to underfitting in the value network, potentially degrading overall performance. Additionally, for ADEPT-G, we assume the arm reward follows a normal distribution, which may not generalize well across all scenarios. Future work will focus on mitigating these issues by improving the robustness of value network predictions and exploring more generalized reward modeling techniques, further solidifying the applicability and reliability of ADEPT.

5.2 ULTHO: Ultra-Lightweight yet Efficient Hyperparameter Optimization in Deep RL

5.2.1 Introduction

Deep reinforcement learning (RL) has propelled significant advancements across various fields, such as game playing [137, 179, 199], chip design [60], algorithm innovation [55, 127], and large language model (LLM) [146, 11, 19, 113]. However, training deep RL agents involves a number of design decisions and hyperparameter configurations, in which minor variations can result in substantial effects on learning efficiency and final performance [52]. While many efforts have been devoted to improving the design choices [72, 53, 77, 79] and implementation details [157, 4, 80, 219], RL-specific hyperparameter optimization (HPO) does not receive sufficient attention in the community. This gap is particularly notable given the high sensitivity of deep RL algorithms to HPs, which restricts their broader applications and the development of advanced and automated RL systems [59].

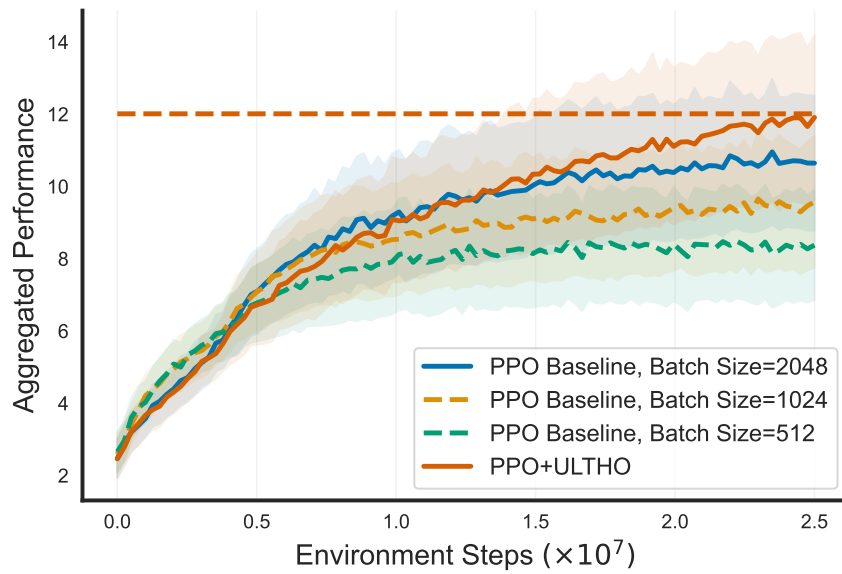


Figure 5.12: Aggregated performance comparison on the Progen benchmark with procedurally-generated environments. The choice of batch size has a significant impact on the agent’s performance. ULTHO can effectively perform HPO across different tasks and learning stages, thereby improving the overall performance.

HPO is a fundamental problem in machine learning, with traditional methods, such as grid search [205] and random search [18], being widely used to automate the process. More advanced techniques, such as Bayesian optimization [181] and multi-fidelity approaches like Hyperband [107], have been developed to improve efficiency by leveraging surrogate models and early stopping mechanisms. However, these methods are primarily designed for static optimization problems, where the optimal HPs remain fixed throughout the training process. In contrast, deep RL operates in non-stationary environments where the optimal HPs may vary across different training stages, especially in dynamic environments such as Procgen [36] and Craftax [129]. This non-stationarity, along with the high computational cost of deep RL training, makes traditional HPO methods less effective for these applications.

To tackle the HPO challenge in deep RL, several approaches have been proposed to dynamically adjust HPs during training. For instance, [85] introduced population-based training (PBT) that evolves a population of agents to explore and exploit different HP schedules, enabling adaptation to non-stationary settings. Notably, PBT is utilized to solve the complex 3D multiplayer game, Quake III Arena, achieving human-level performance by optimizing the agent policies along with internal reward signals in a hierarchical manner [84]. [59] further extended PBT and proposed SEARL, which optimizes the HPs and also the neural network architecture while simultaneously training the agent, significantly improving sample efficiency by sharing experience across the population. However, PBT-like methods remain sample-inefficient and computationally expensive, limiting their practicality to a wider range of tasks. To address these issues, single-run HPO methods such as HOOF [151] have been developed. HOOF adapts HPs using off-policy importance sampling, optimizing a one-step improvement objective with sampled trajectories. As a gradient-free algorithm, HOOF can significantly reduce overhead and promote sample efficiency. However, HOOF greedily focuses on HP configurations that maximize the value of the updated policy without introducing an exploration mechanism. Moreover, its reliance on importance

sampling also makes it prone to high variance when policy distributions shift, hurting its robustness in highly dynamic environments.

Inspired by the discussions above, we propose **ULTHO** : **U**ltra-**L**ightweigh**T** **H**yperparameter **O**ptimization, a general, simple, yet powerful framework for achieving efficient and robust HPO in deep RL within single runs. Unlike previous methods that focus on short-term improvements or require complex learning processes, ULTHO performs HPO from the perspective of the hierarchical bandits, ensuring long-term performance gains with minimal computational overhead. Our main contributions are summarized as follows:

- We formulate the HPO process as a multi-armed bandit with clustered arms (MABC), which optimizes HPs adaptively across different tasks and learning stages in a hierarchical fashion and effectively reduces the sample complexity. ULTHO selects the appropriate HPs based on the estimated task return, ensuring the maximization of long-term returns while balancing exploration;
- Our framework currently provides two algorithms, *i.e.*, a normal version and an extended version for continual optimization, solving HPO within single runs and providing a quantified and statistical perspective to analyze the potential of distinct HPs. In particular, ULTHO has a simple architecture and requires no additional learning processes, which can be compatible with a broad range of RL algorithms;
- Finally, we evaluate ULTHO on ALE (arcade game environments), Procgen (sixteen procedurally-generated environments), MiniGrid (environments with sparse rewards), and PyBullet (robotics environments with continuous action space). Extensive experiments demonstrate that ULTHO can achieve superior performance with remarkable computational efficiency.

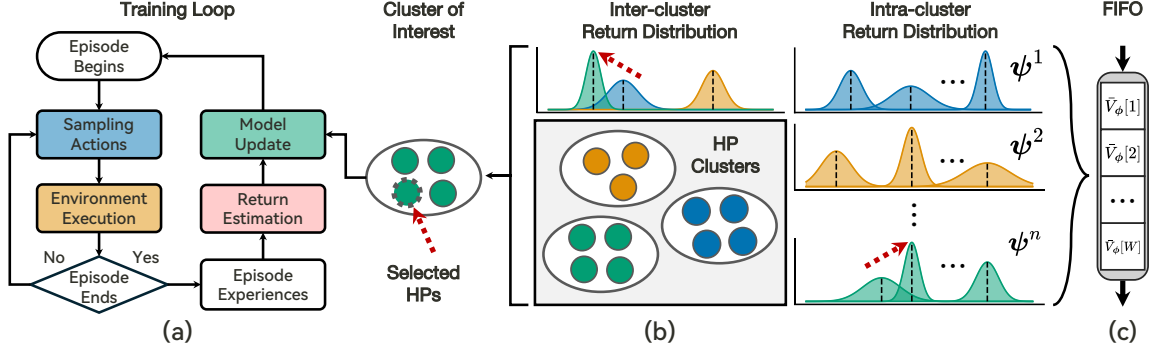


Figure 5.13: Overview of ULTHO framework. (a) The key phases of RL algorithms. ULTHO serves as a plug-and-play module that feeds the optimized HPs to the RL algorithm. (b) ULTHO maintains inter-cluster and intra-cluster return distributions to perform HPO in a hierarchical manner. (c) A sliding window is used to store the estimated task returns for updating the return distributions.

5.2.2 The ULTHO Framework

In this section, we introduce ULTHO, an ultra-lightweight, unified, yet powerful framework designed for efficient and robust HPO in RL, whose overview is presented in Figure 5.13. Our key insights are as follows: (i) Similar to [151] and [220], ULTHO also dynamically tunes HPs within a single training run, enabling adaptation to different learning stages and significantly reducing the trial-and-error cost; (ii) ULTHO adopts a hierarchical approach [84, 13] to organize and select the HP candidates. Specifically, we perform a two-tier HPO to traverse the search space more efficiently; (iii) Unlike prior methods [85, 151, 220] often prioritizes short-term improvements, ULTHO focuses on optimizing long-term returns, providing a systematic approach that ensures more sustainable performance gains; (iv) To facilitate a wide range of RL algorithms, ULTHO requires no additional learning processes or access to internal data, such as gradients, making it universally applicable without requiring complex modifications to the underlying algorithm.

5.2.2.1 MABC for HPO

To simplify the notations, we reuse Ψ to denote the set of HP clusters:

$$\Psi = \{\psi^1, \psi^2, \dots, \psi^n\}, \quad (5.4)$$

where

$$\psi^i = \{\psi_1^i, \psi_2^i, \dots, \psi_m^i\} \quad (5.5)$$

is an individual cluster of HPs.

Then, the HPO at different learning stages can be formulated as a multi-armed bandit with clustered arms (MABC) [26], and the optimization objective is to maximize the long-term return evaluated by the task reward function.

We solve the defined MABC problem by hierarchically applying the UCB [8] algorithm, which is an effective and widely-used method to balance exploration-exploitation in bandit problems. Specifically, at the i -th episode, we first select a cluster from Ψ by the following policy:

$$\underbrace{\psi_i = \operatorname{argmax}_{\psi \in \Psi} \left[U_i(\psi) + c \sqrt{\frac{\log i}{N_i(\psi)}} \right]}_{\text{Inter-cluster Scheduling Policy}}, \quad (5.6)$$

where $N_i(\psi)$ is the number of times that ψ has been chosen before the i -th episode, and c is the exploration coefficient.

Equipped with the selected cluster, we then select a specific HP value following a similar policy:

$$\underbrace{\psi_i = \operatorname{argmax}_{\psi \in \psi_i} \left[U_i(\psi) + c \sqrt{\frac{\log i}{N_i(\psi)}} \right]}_{\text{Intra-cluster Scheduling Policy}}. \quad (5.7)$$

The selected HP value will be used for policy updates. Then the utility functions of the cluster and the corresponding value are updated using a sliding window average

Algorithm 5: The ULTHO with UCB

- 1 Initialize the policy network π_{θ} and value network V_{ϕ} ;
- 2 Initialize a set Ψ of HPs, an exploration coefficient c , a window length W for estimating the utility functions;
- 3 $\forall \boldsymbol{\psi} \in \Psi, \forall \psi \in \boldsymbol{\psi}$, let

$$N(\boldsymbol{\psi}) = 1, U(\boldsymbol{\psi}) = 0, R(\boldsymbol{\psi}) = \text{FIFO}(W)$$

$$N(\psi) = 1, U(\psi) = 0, R(\psi) = \text{FIFO}(W)$$

for *each episode* e **do**

- 4 Sample rollouts using the policy network π_{θ} ;
 - 5 Perform the generalized advantage estimation (GAE) to get the estimated task returns;
 - 6 Select a cluster $\boldsymbol{\psi}_e$ using Eq. (5.6);
 - 7 Select a HP ψ_e from the $\boldsymbol{\psi}_e$ using Eq. (5.7);
 - 8 Update policy network and value network;
 - 9 Compute $\bar{V}_{\phi}$ obtained by the new policy;
 - 10 Add $\bar{V}_{\phi}$ to the queue $R(\boldsymbol{\psi}_e)$ and $R(\psi_e)$;
 - 11 Update $U(\boldsymbol{\psi}_e)$ and $U(\psi_e)$ using Eq. (5.8);
 - 12 $N(\boldsymbol{\psi}_e) \leftarrow N(\boldsymbol{\psi}_e) + 1, N(\psi_e) \leftarrow N(\psi_e) + 1$.
-

of the past mean episode returns obtained by the agent:

$$\underbrace{U_i(\boldsymbol{\psi}) = \frac{1}{W} \sum_{j=1}^W \bar{V}_{\phi}[j], \bar{V}_{\phi} = \frac{1}{T} \sum_{t=1}^T V_{\phi}(\mathbf{s}_t)}_{\text{Utility Evaluation}}, \quad (5.8)$$

where $V_{\phi}(\mathbf{s}_t)$ is the estimated return predicted by the value network, T is the episode length, and W is the window length.

5.2.2.2 Relay-ULTHO

Algorithm 5 summarizes the workflow of ULTHO with UCB. It is evident that the ULTHO changes only one HP at a time, which helps stabilize the learning process due to the high sensitivity of RL to HPs. However, optimizing multiple HPs simultaneously can lead to unstable learning behavior, slower convergence, and potentially sub-optimal policies in certain environments. Moreover, it is unclear whether all HPs consistently affect performance across different environments and learning stages. With this in mind, we propose an extended algorithm entitled Relay-ULTHO, as depicted in Algorithm 6.

Algorithm 6: The Relay-ULTHO

- 1 Execute Algorithm 5;
 - 2 Get ψ^{COI} and ψ^{NOI} using Eq. (5.9);
 - 3 Execute Algorithm 5 solely with ψ^{COI} or ψ^{NOI} ;
 - 4 Output the best-performing cluster.
-

The core idea behind Relay-ULTHO is to identify two HP clusters: the cluster of interest (COI) and the neglected cluster of interest (NOI). COI is the cluster that is selected most frequently, indicating it has the greatest impact on the agent’s performance. In contrast, NOI is the cluster with the least number of selections. After executing Algorithm 5 with an initial set of clusters Ψ , we compute the total count $N_{\text{end}}(\psi)$ for each cluster. Then we define the COI and NOI as

$$\begin{aligned}\psi^{\text{COI}} &= \underset{\psi \in \Psi}{\operatorname{argmax}} N_{\text{end}}(\psi), \\ \psi^{\text{NOI}} &= \underset{\psi \in \Psi}{\operatorname{argmin}} N_{\text{end}}(\psi).\end{aligned}\tag{5.9}$$

Once the COI and NOI are identified, the optimization process is focused on these clusters, allowing the algorithm to refine the most influential cluster and further explore the neglected cluster. This approach can improve the completeness of exploration, thereby squeezing out additional performance gains. However, we also need

to highlight that this algorithm may result in sub-optimal performance due to its restricted landscape.

5.2.3 Experiments

In this section, we design the experiments to investigate the following questions:

- **Q1:** Can ULTHO improve performance as compared to using fixed HP values?
- **Q2:** Can the relay optimization further enhance the ULTHO algorithm?
- **Q3:** What are the detailed decision processes of the ULTHO algorithm?
- **Q4:** How does ULTHO behave in sparse-rewards environments and continuous control tasks?
- **Q5:** How robust is the ULTHO algorithm?



Figure 5.14: Screenshots of the ALE (top) and Procgen (below) benchmarks.

5.2.3.1 Setup

a) Benchmark Selection: We first evaluate the ULTHO using the arcade learning environment (ALE) benchmark [16], a collection of arcade game environments that

requires the agent to learn motor control directly from images. Specifically, we focus on a subset of ALE known as ALE-5, which typically produces median score estimates for all games that are within 10% of their true values [3]. For all environments, we stack 4 consecutive frames to form an input state with the data shape of (84, 84, 4). Additionally, we introduce the Procgen benchmark with 16 procedurally-generated environments. Procgen is similar to the ALE benchmark yet involves much higher dynamicity and presents a more difficult challenge for HPO. All the environments use a discrete fifteen-dimensional action space and generate (64, 64, 3) RGB observations, and we use the *easy* mode and train the agents on 200 levels before testing them on the full distribution of levels. Furthermore, we introduce the MiniGrid [34] and PyBullet [39] to test ULTHO in sparse-rewards environments and continuous control tasks.

b) Algorithmic Baselines: For the RL algorithm, we select the proximal policy optimization (PPO) [171] as the baseline, which is a representative algorithm that produces considerable performance on most existing RL benchmarks. For the HPO algorithms, we select random search (RS) [18], PBT [85], PB2 [148], the Bayesian optimization tool SMAC [112], the combination of SMAC and the Hyperband scheduler (SMC+HB) [107], and TMIHF [147].

c) HP Clusters: We select HP clusters based on the prior practice reported in [85, 151, 59, 79, 215]. Specifically, our HP clusters include the learning rate (LR), batch size (BS), value loss coefficient (VLC), entropy loss coefficient (ELC), and the number of update epochs (NUE). To ensure a balanced exploration, we let each cluster have the same number of values. For the detailed configuration of each cluster, please refer to Table 5.1.

d) Evaluation Metrics: For the baselines that require multiple training runs (*e.g.*, PBT), we use the maximum observed past scores as their final performance. For our ULTHO algorithm, we simply use the observed episode return at the end of training as

Table 5.1: The selected hyperparameter clusters for each benchmark.

HP	ALE	Procgen	MiniGrid	PyBullet
LR	N/A	{2.5e-4, 5e-4, 1e-3}	{1e-3, 2.5e-3, 5e-3}	{2e-4, 5e-4, 7e-4}
BS	{128, 256, 512}	{512, 1024, 2048}	{128, 256, 512}	{64, 128, 256}
VLC	{0.25, 0.5, 1.0}	{0.25, 0.5, 1.0}	{0.25, 0.5, 1.0}	{0.25, 0.5, 1.0}
ELC	{0.01, 0.05, 0.1}	N/A	N/A	N/A
NUE	N/A	{3, 2, 1}	N/A	N/A

its performance. Note that the score of each method on each environment is computed as the average episode returns over 100 episodes and 5 random seeds. Additionally, we compare computational efficiency by evaluating the required training budgets for all the methods.

5.2.3.2 Results Analysis

The following results analysis is performed based on the pre-defined research questions.

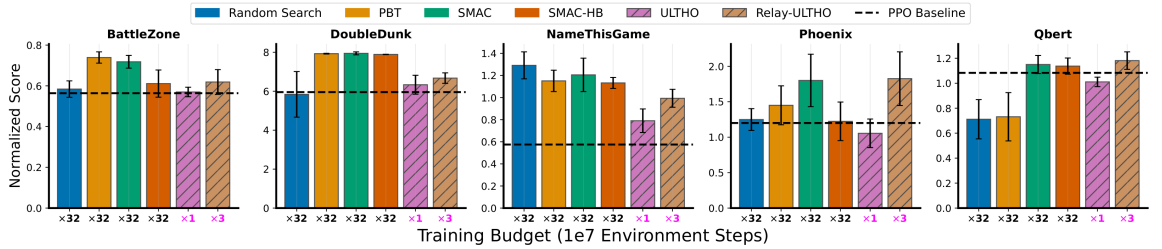


Figure 5.15: Performance comparison of the two ULTHO algorithms and HPO baselines on the ALE-5 benchmark, in which the mean and standard error are computed using five random seeds. All the scores are normalized using the min-max normalization with the human expert performance. The two ULTHO algorithms achieve similar or even higher performance with a few training budgets as compared to other HPO baselines.

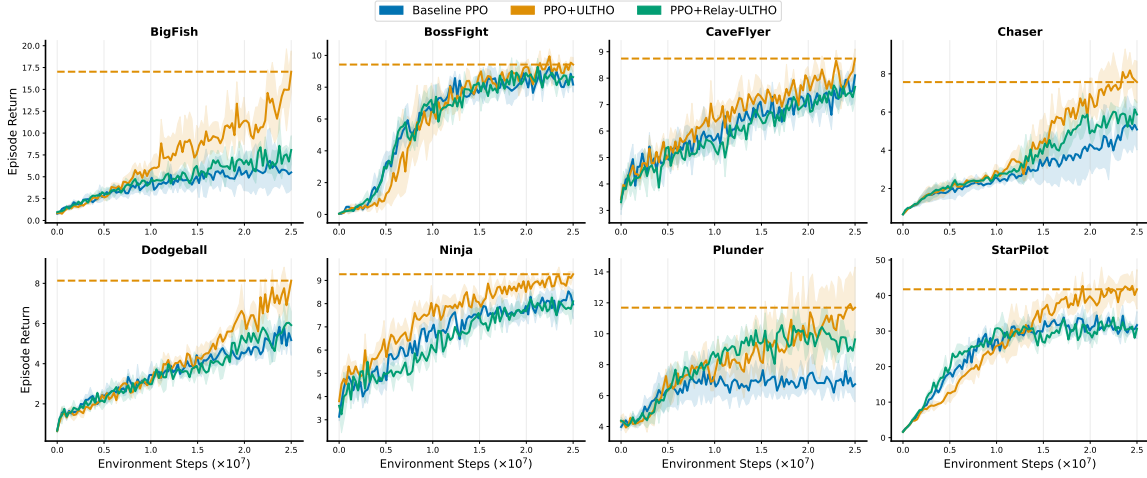


Figure 5.16: Training performance of the PPO and its combinations with two ULTHO algorithms in eight Procgen environments, in which the mean and standard deviation are computed using five random seeds. ULTHO consistently outperforms the vanilla PPO agent in all the environments, significantly improving the sample efficiency. However, the Relay-ULTHO does not achieve any further performance gains, indicating that a more comprehensive policy, which continuously adapts multiple HPs, is crucial for highly dynamic environments.

a) Comparison with HPO Baselines: We first compare the performance of ULTHO and four HPO baselines on the ALE benchmark, and their normalized scores and training budgets are illustrated in Figure 5.15. Here, the baseline PPO is trained using the common HPs reported in its original paper [171]. ULTHO successfully outperforms the PPO baseline in all five environments and achieves the highest performance in two environments, especially in Q^*Bert environment. In contrast, the SMAC and SMAC+HB rank second and third in terms of average performance across all the environments. The PBT method excels in the *BattleZone* environment, but it fails to outperform the baseline PPO in the Q^*bert environment. By adaptively tuning HPs within the training run, ULTHO can achieve remarkable efficiency with much lower computational cost as compared to the baselines.

Table 5.2: Normalized test performance (%) comparison on the Procgen benchmark. Here, **IQM** denotes the interquartile mean suggested by [2], and ‡ indicates the training is conducted by a population of agents. Similar to the training performance comparison, the normal ULTHO method achieves the highest score by providing a more comprehensive HPO policy.

Method	Aggregated Mean	Aggregated IQM
PPO	40.42±25.18	38.41±6.59
PBT‡	33.55±23.49	29.27±13.68
PB2‡	44.63±23.56	40.5±11.41
TMIHF‡	51.16±20.16	48.7±4.67
ULTHO	56.0±21.87	56.41±6.95
Relay-ULTHO	48.04±26.88	49.01±5.0

Next, we report the performance of the ULTHO on the Procgen benchmark. Similarly, the baseline PPO is trained using the reported HPs in [36]. Figure 5.16 illustrates the training performance comparison in eight environments. ULTHO outperforms the baseline PPO in most environments, achieving significant performance gains in environments like *BigFish*, *Chaser*, and *StarPilot*. Additionally, we evaluate ULTHO on the full distribution of levels, and Table 5.2 illustrates the performance comparison with three HPO baselines. ULTHO outperforms these baselines regarding both the aggregated mean and IQM. These results demonstrate that ULTHO can effectively select the appropriate HPs at different learning stages and significantly improve learning efficiency even in highly dynamic environments.

b) Capability of Relay Optimization: The high sensitivity to HPs of deep RL makes it risky to optimize multiple HPs simultaneously, leading to the development of the Relay-ULTHO algorithm to enhance exploration and stability. To evaluate its effectiveness, we test it on both the ALE and Procgen benchmarks. Relay-ULTHO

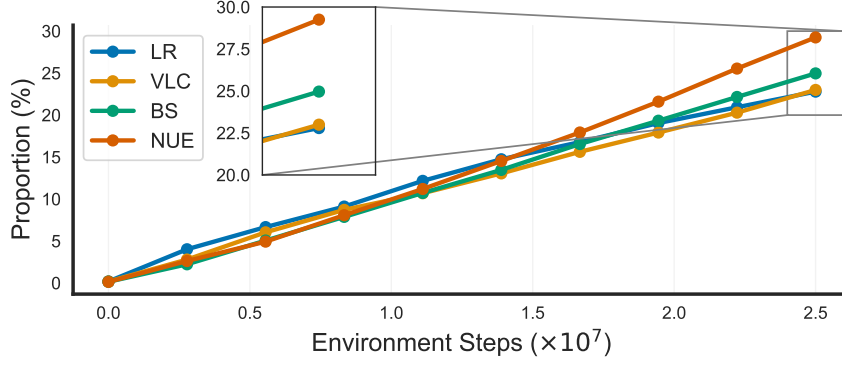


Figure 5.17: Aggregated inter-cluster decision processes of ULTHO on the whole Procgen benchmark. Among the four HP clusters, the NUE cluster is selected the most, suggesting its high impact on overall performance.

produces a much higher performance than the normal version, showing the incremental potential by identifying the COI and NOI from the HP space. However, Relay-ULTHO fails to outperform the normal version on the Procgen benchmark. While Relay-ULTHO is effective in environments with relatively stable dynamics, it struggles in highly non-stationary settings like Procgen, where optimal HP configurations may shift continually across training stages. Therefore, it is feasible to combine the two algorithms to realize a more robust HPO process.

c) The Detailed Decision Process: Furthermore, we analyze the detailed decision processes of ULTHO. Figure 5.17 illustrates the cumulative proportion of each cluster selected during the whole training. It is clear that ULTHO primarily selects the NUE cluster, while the other three clusters account for approximately 20%. For intra-cluster decisions, we find that the $LR=5e-4$, $VLC=1.0$, $BS=2048$, and $NUE=2$ are the most popular choices. Additionally, Figure 5.19 illustrates the variation of aggregated confidence intervals of these clusters during the training. ULTHO shows the explicitly distinct preferences of HP clusters, which aligns with the cumulative proportion illustrated in Figure 5.17. These confidence intervals provide a statistical and quantified perspective to identify the importance of each HP cluster, contributing

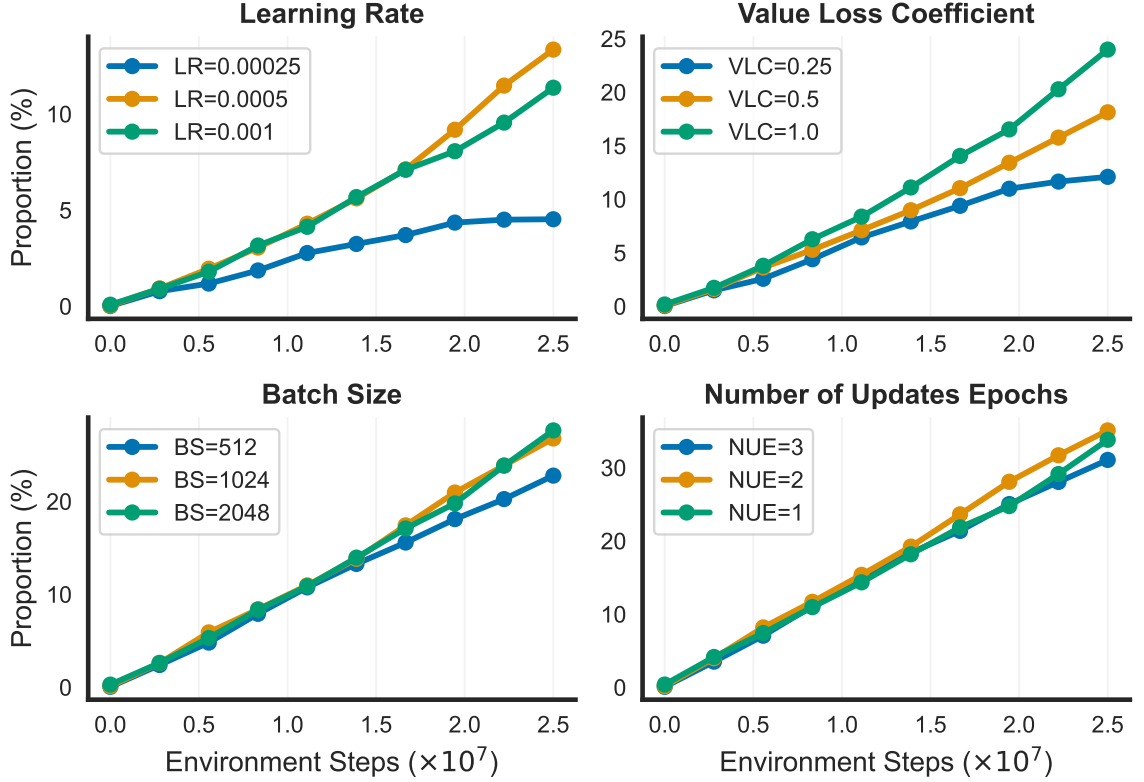


Figure 5.18: Aggregated intra-cluster decision processes of ULTHO on the whole Procgen benchmark. Among the four HP clusters, the NUE cluster is selected the most, suggesting its high impact on overall performance.

to a more efficient filtering approach.

d) Ablation Studies: While ULTHO achieves efficient HPO, the utilized UCB method relies on two HPs, the exploration coefficient c and the length of the sliding window W . To evaluate the robustness of ULTHO, we perform experiments using different configurations of the two HPs on the Procgen benchmark, and the aggregated training performance is illustrated in Figure 5.20. It is evident that the ULTHO is relatively insensitive to the choice of c and W , yet a bigger W helps achieve a more reliable return estimation. Therefore, ULTHO maintains stable performance across different HPs and can be easily adapted to various training scenarios.

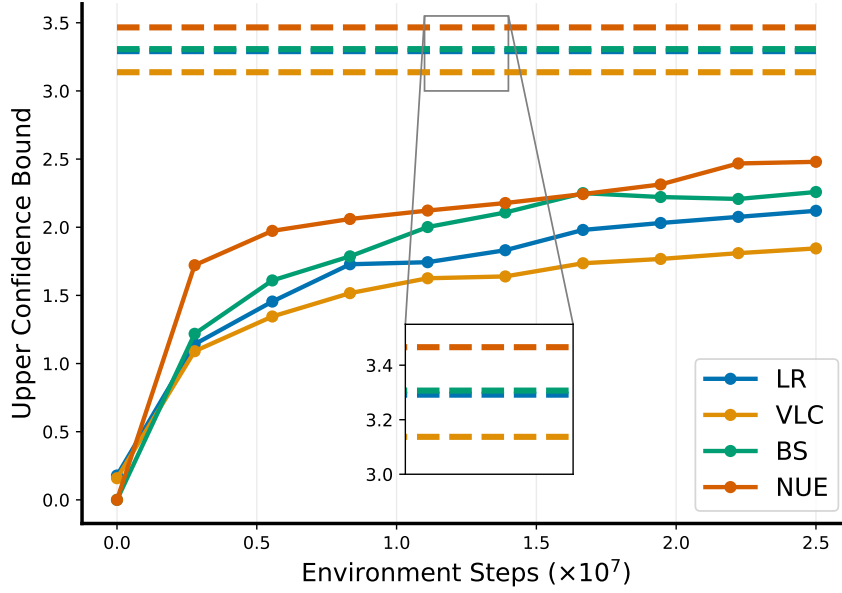


Figure 5.19: The variation of aggregated confidence intervals on the whole Procgen environments. Here, the solid line represents the mean value, and the dashed line represents the final upper confidence bound. It is evident that the NUE cluster obtains the highest upper confidence bound at the end of training, which aligns with the selection proportion illustrated in Figure 5.18.

e) Performance in Sparse-rewards Environments: Additionally, we evaluate ULTHO on the MiniGrid benchmark with sparse-rewards and goal-oriented environments. Specifically, we conduct experiments using *DoorKey-6×6*, *LavaGapS7*, and *Empty-16×16*. Figure 5.21 illustrates the aggregated learning curves of the PPO baseline and two ULTHO algorithms. While ULTHO fails to outperform the PPO baseline, the relay optimization takes fewer environment steps to solve the tasks, highlighting its capability to achieve efficient HPO in both dense and sparse-reward settings.

f) Performance on Continuous Control Tasks: Finally, we evaluate ULTHO on the PyBullet benchmark with continuous control tasks. Four environments are utilized, namely *Ant*, *HalfCheetah*, *Hopper*, and *Walker2D*. Figure 5.22 illustrates the

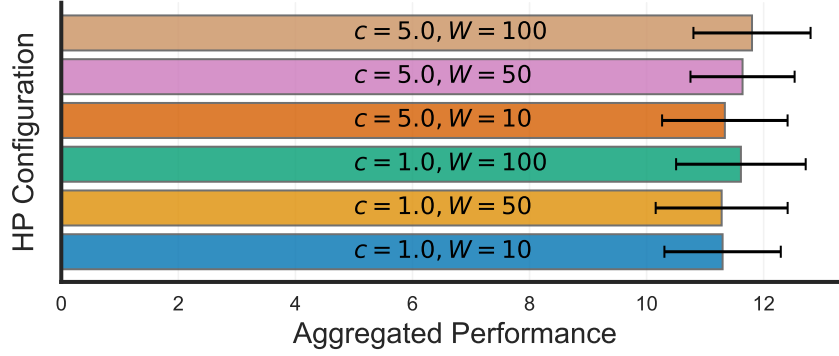


Figure 5.20: Aggregated performance of ULTHO on the Progen benchmark with different configurations of c and W , and the mean and standard error are computed across all the environments. These results demonstrate that ULTHO is robust to the variation of the two internal HPs.

aggregated learning curves of the vanilla PPO agent and two ULTHO methods, which produce significant performance gain on the whole benchmark. Similarly, as analyzed before, relay optimization brings advantages in relatively stationary environments. These results underscore the effectiveness of ULTHO in enhancing RL algorithms across both discrete and continuous control tasks.

5.2.4 Conclusion

In this section, we investigated the problem of HPO in deep RL and proposed an ultra-lightweight yet powerful framework entitled ULTHO, which performs HPO on the fly and requires no complex learning process. ULTHO formulates the HPO process as a multi-armed bandit with clustered arms, enabling efficient and adaptive HP selection across different learning stages. We evaluate ULTHO on ALE, Progen, MiniGrid, and PyBullet benchmarks. Extensive experiments demonstrate that ULTHO can effectively enhance RL algorithms with a simple architecture, contributing to the

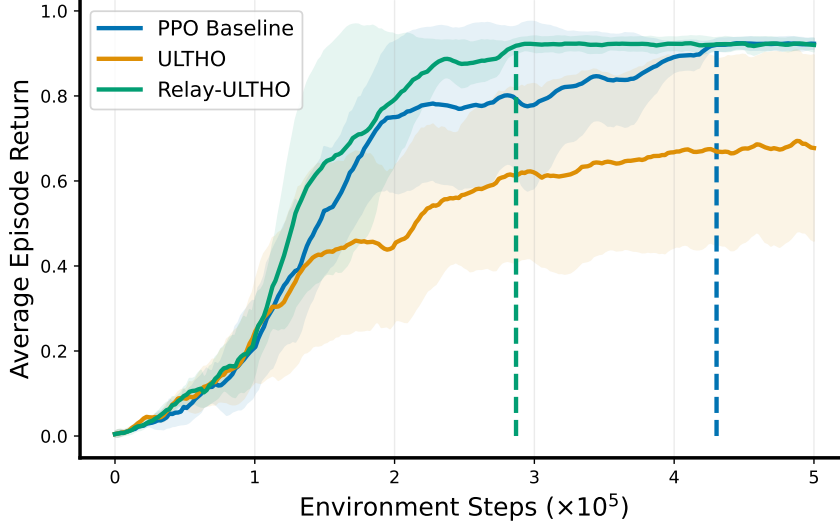


Figure 5.21: Aggregated performance of ULTHO on the MiniGrid benchmark, and the mean and standard deviation are computed across all the environments. For the relatively stationary environments, Relay-ULTHO can further obtain performance gains through incremental optimization.

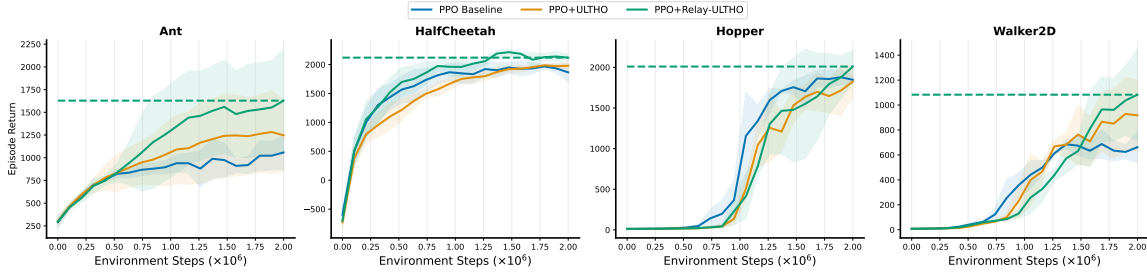


Figure 5.22: Performance of the PPO baseline and two ULTHO algorithms on the PyBullet benchmark, and the mean and standard deviation are computed using five random seeds. Our method can effectively improve the sample efficiency through high-quality HPO in continuous control tasks.

development of advanced and automated RL systems.

Still, there are currently remaining limitations to this work. As a lightweight framework, ULTHO currently operates on categorical HP values, which discretizes HP search into predefined clusters. While this ensures computational efficiency and stable optimization, it may limit the granularity of HP tuning in some cases. Extending ULTHO to handle continuous HP spaces for more precise tuning is a promising direction for future work. Additionally, while ULTHO uses a UCB-based strategy for exploration-exploitation balance, we have yet to explore alternative bandit algorithms, such as Exp3 or Thompson sampling. Evaluating these strategies within ULTHO could provide valuable insights into their impact on RL-specific HPO and potentially further improve performance in dynamic environments. Future work will focus on mitigating these issues and further enhancing ULTHO, making it more robust, adaptive, and efficient for HPO in deep RL.

5.3 PvP: Data-Efficient Humanoid Robot Learning with Proprioceptive-Privileged Contrastive Representations

5.3.1 Introduction

Humanoid robots have emerged as a critical platform for embodied intelligence, whose human-like morphology provides inherent advantages such as versatility, adaptability to human-centered environments, and intuitive interaction [45, 196, 198]. To enable these capabilities, whole-body control (WBC) is essential for coordinating numerous joints and actuators to achieve balanced, agile, and safe behaviors in real-world settings [176, 33]. However, designing effective WBC policies is particularly challenging due to the complex dynamics of humanoids, underactuation, and strong coupling

between locomotion, manipulation, and balance [92, 217]. As a result, traditional model-based methods often fail to ensure flexible real-time control and robust performance under non-stationary conditions [175, 163].

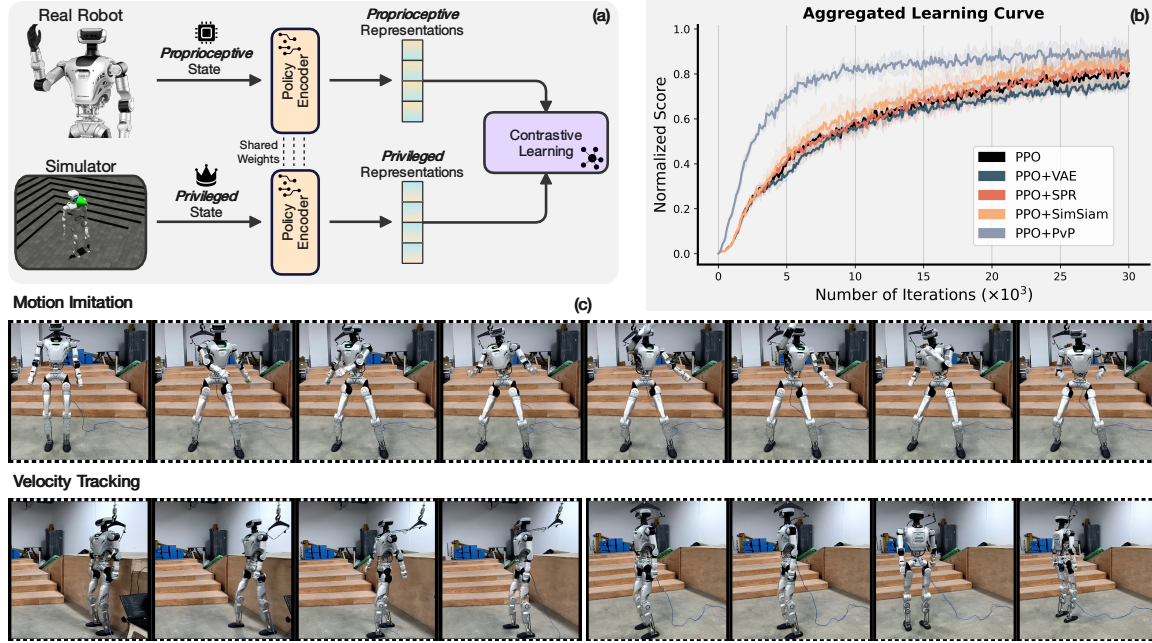


Figure 5.23: (a) PVP employs contrastive learning between proprioceptive and privileged states to learn compact and task-relevant representations for policy learning. (b) The aggregated learning curves of PVP and algorithmic baselines across multiple evaluation tasks, where PVP significantly improves training efficiency. (c) The real-world evaluation of PVP on the LimX Oli humanoid robot.

To overcome these limitations, recent research has increasingly turned to data-driven approaches, with reinforcement learning (RL) emerging as a dominant paradigm for humanoid WBC [110, 217, 204, 221, 81, 109]. For instance, BeyondMimic [110] leverages large-scale RL-based motion tracking to achieve strong generalization across diverse human-like whole-body motions, while HugWBC [204] employs RL to train a single policy for multi-gait locomotion (*e.g.*, standing, walking, and jumping) within a unified command space. Despite these promising results, RL applications for humanoid WBC remain constrained by sample inefficiency. The intricate dynamics

and partial observability of humanoid robots, along with the necessity to optimize a composite reward structure to ensure task performance (*e.g.*, tracking accuracy) and reliability in real-world deployment (*e.g.*, energy efficiency), significantly increase the sample complexity.

In response to this challenge, recent work has focused on simulation acceleration [126, 134, 192], data augmentation [172, 133, 173], and state representation learning (SRL) [119, 186, 224]. Among these strategies, SRL offers a promising solution by transforming high-dimensional sensory inputs into compact and informative latent representations. These representations preserve task-relevant dynamics while filtering out noise and redundancy, which can significantly improve sample efficiency, generalization, and overall performance in RL [100, 174, 98, 51] tasks. Pioneering work like [119, 186] demonstrates the potential of SRL in humanoid robot learning, where reconstruction-based representations (*e.g.*, elevation-based internal maps [186]) and contrastive learning-based abstractions (*e.g.*, perceptive internal models from height maps [119]) both serve to refine state embeddings and improve RL performance in complex environments. However, integrating SRL techniques into humanoid WBC remains underexplored, especially the seamless combination of SRL with RL in a unified, end-to-end framework that enhances both learning efficiency and real-world deployment reliability.

In this section, we investigate SRL-empowered humanoid robot learning and propose a simple yet powerful approach entitled **PvP: Proprioceptive-Privileged** contrastive learning. Our contributions are threefold:

- We propose to conduct contrastive learning between proprioceptive and privileged states of the humanoid robot to enhance the proprioceptive representations for policy learning. PvP leverages the intrinsic complementarity between the two state modalities without relying on handcrafted data augmentations, producing stable improvements across a broad range of tasks.

- We introduce **SRL4Humanoid**, to the best of our knowledge, the first unified and modular open-source framework that provides high-quality implementations of representative SRL methods for humanoid robot learning, enabling reproducible research and facilitating future progress in the community. Equipped with SRL4Humanoid, we conduct a systematic study of how different SRL methods and configurations affect the efficiency and performance of humanoid WBC learning.
- We validate our approach on the **LimX Oli** humanoid robot through two representative tasks: **velocity tracking** and **motion imitation**. Extensive experiment results demonstrate that PvP significantly enhances sample efficiency and policy performance in complex WBC scenarios, outperforming existing SRL baselines.

5.3.2 PvP Implementation

Recent approaches in humanoid robot learning have introduced SRL as an auxiliary learning objective to enhance policy learning. However, most of them follow a reconstruction-based approach [200, 187] (*e.g.*, predicting privileged information such as root linear velocity from proprioceptive states), yet they often struggle with suboptimal representation quality and poor generalization. This is because these methods focus on preserving the complete state, including irrelevant details, rather than learning task-relevant features. While methods like PIM [119] employ contrastive learning to obtain more robust representations, they only rely on a single state modality without incorporating privileged state information. This limitation restricts their ability to capture the full spectrum of task-relevant dynamics, leading to less informed representations.

With this in mind, PvP utilizes both proprioceptive and privileged states to perform contrastive learning, combining complementary information from different sensory

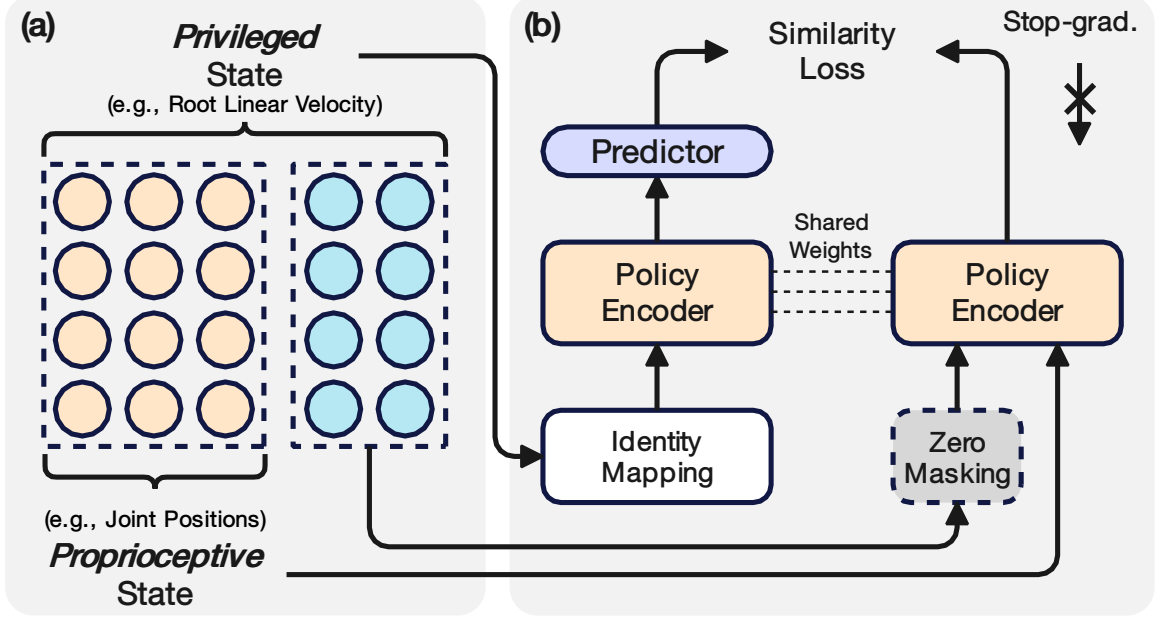


Figure 5.24: An overview of the PvP approach. (a) The components of the privileged state and the proprioceptive state. (b) PvP conducts contrastive learning based on the intrinsic complementarity between the two state modalities.

modalities. As shown in Figure 5.24, the privileged state $\mathbf{s}$ contains both proprioceptive observations (e.g., joint positions and angular velocities) and privileged information (e.g., root linear velocity), which can be considered as the pseudo augmentation of the proprioceptive state $\mathbf{o}$. Meanwhile, we apply the zero-masking to the privileged information part in $\mathbf{s}$ and keep the proprioceptive observations solely:

$$\tilde{\mathbf{s}}_t = \text{ZeroMasking}(\mathbf{s}_t). \quad (5.10)$$

Then, the derived data pair $(\mathbf{s}, \tilde{\mathbf{s}})$ is used to train the policy encoder, following the SimSiam algorithm [31]. Formally, denote by f_{θ} the policy encoder, h_{ψ} the predictor, we have

$$\begin{aligned} \mathbf{z} &= f_{\theta}(\mathbf{s}), & \tilde{\mathbf{z}} &= f_{\theta}(\tilde{\mathbf{s}}) \\ \mathbf{p} &= h_{\psi}(\mathbf{z}), & \tilde{\mathbf{p}} &= h_{\psi}(\tilde{\mathbf{z}}) \end{aligned} \quad (5.11)$$

Finally, the PvP loss is defined as

$$L_{\text{PvP}} = D_{\text{ncs}}(\mathbf{p}, \text{sg}(\tilde{\mathbf{z}})) + D_{\text{ncs}}(\tilde{\mathbf{p}}, \text{sg}(\mathbf{z})), \quad (5.12)$$

where $D_{\text{ncs}}(\mathbf{p}, \mathbf{z}) = -\frac{\mathbf{p}}{\|\mathbf{p}\|_2} \cdot \frac{\mathbf{z}}{\|\mathbf{z}\|_2}$ is the negative cosine similarity loss between the pair and $\text{sg}(\cdot)$ is the stop-gradient operation.

We would like to highlight the advantages of PvP. First, PvP leverages both proprioceptive and privileged states for contrastive learning, effectively reducing the complexity of SRL while enhancing the learned representations by incorporating richer and more comprehensive information. This approach also offers an alternative method for the policy to access privileged information, thereby enabling the agent to gain a better understanding of its environment. Additionally, PvP leverages the intrinsic complementarity between the two state modalities without relying on hand-crafted data augmentations. As a result, PvP is highly versatile and can be applied to a wide range of tasks.

5.3.3 The SRL4Humanoid Framework

To facilitate our experiments and subsequent research, we develop SRL4Humanoid, a unified, highly modularized framework that provides high-quality and reliable implementations of representative SRL techniques for humanoid robot learning. Figure 5.25 illustrates the architecture of the SRL4Humanoid framework. Following the practice of a series of humanoid robot research [33, 119, 186, 204], we select the proximal policy optimization (PPO) [171] as the backbone RL algorithm. Notably, the policy network accepts the proprioceptive state of the robot to generate actions, while the value network accepts the privileged state of the environment to perform value estimation. The SRL and RL processes are designed to be fully decoupled, and depending on the training configuration, the SRL objective can be applied to either the policy encoder or the value encoder.

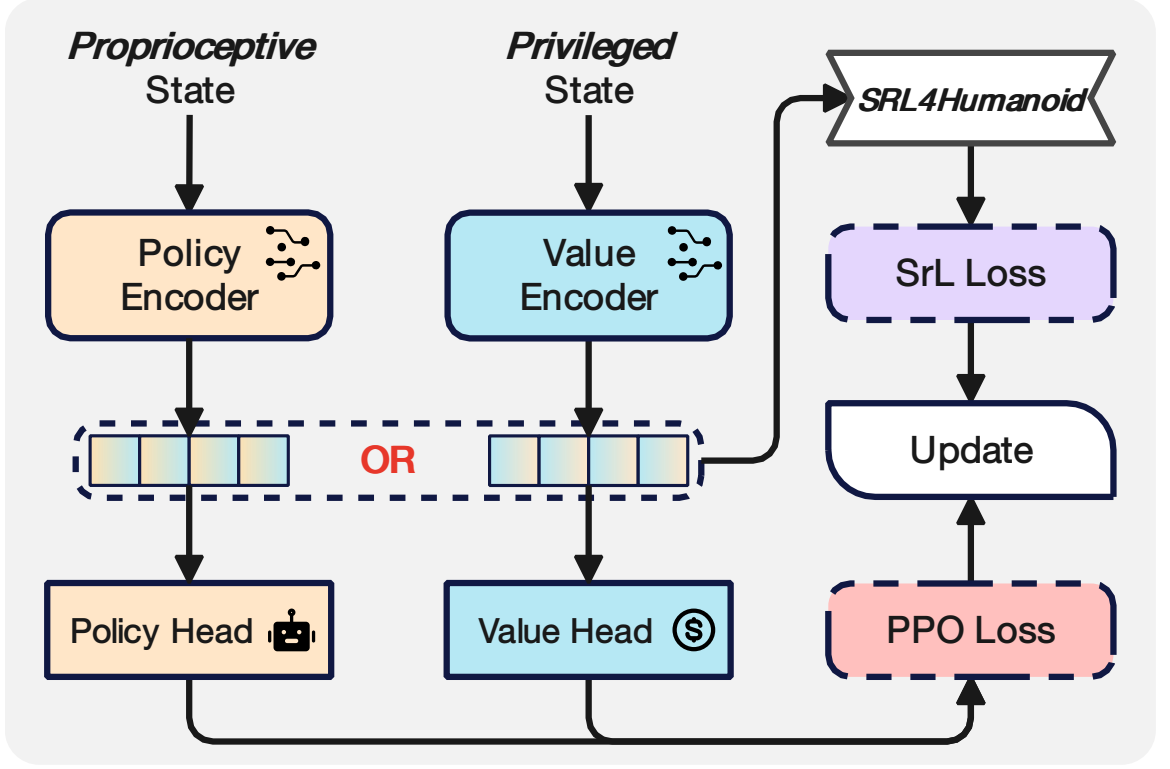


Figure 5.25: The architecture of the SRL4Humanoid framework, in which the SRL and RL processes are fully decoupled.

To ensure both diversity and representativeness, we currently implement three widely studied SRL algorithms, each exemplifying a distinct methodological paradigm introduced in Section 2.2.3. These categories capture the main approaches explored in prior SRL research, allowing us to analyze their relative strengths when applied to humanoid WBC tasks. Finally, the joint optimization objective of the RL and the SRL is defined as

$$L_{\text{Total}} = L_{\text{RL}} + \lambda \cdot L_{\text{SRL}}, \quad (5.13)$$

where λ is a weighting coefficient.

By default, the updates of the two loss items are synchronized. This means that the training of the SRL module shares the data batches with RL and follows the update frequency of RL to isolate the effects of SRL. However, in practical experiments, we

Algorithm 7: Workflow of the SRL4Humanoid

```
1 Initialize the policy  $\pi_{\theta}$  and value network  $V_{\phi}$ ;
2 Initialize the SRL module  $S_{\psi}$ ;
3 Set all the hyperparameters, such as the maximum number of episodes  $E$ , and
  the number of update epochs  $K$ , etc.
4 for  $episode = 1$  to  $E$  do
5   Sample rollouts using the policy network  $\pi_{\theta}$ ;
6   Perform the generalized advantage estimation (GAE) to get the estimated
     task returns;
7   for  $epoch = 1$  to  $K$  do
8     Sample a mini-batch  $\mathcal{B}$  from the rollouts data;
9     Use  $\mathcal{B}$  to compute the policy and value loss;
10    Use  $\mathcal{B}$  to compute the SRL loss;
11    Compute the total loss following Eq. (5.14);
12    Update the policy network, value network, and the SRL module;
13  end
14 end
15 Output the optimized policy  $\pi_{\theta}$ 
```

found that consistently applying the SRL loss does not always have a positive influence on policy learning and can sometimes degrade learning efficiency. In massively parallel RL, a large amount of repetitive and low-quality data is produced during the early stages of training. This is likely to cause the SRL to prematurely fall into local optima and fail to continuously influence policy learning. To address this issue, we employ an interval update mechanism:

$$L_{\text{Total}} = L_{\text{RL}} + \mathbb{1}(T) \cdot \lambda \cdot L_{\text{SRL}}, \quad (5.14)$$

where $\mathbb{1}(T)$ is an indicator function that equals 1 every T time steps; otherwise, it

equals 0. This phenomenon will be detailed in the experiment section. Finally, we summarize the workflow of SRL4Humanoid in Algorithm 7.

5.3.4 Experiments

In this section, we design the experiments to achieve the two main objectives: (i) evaluate the performance of the proposed PvP algorithm and the SRL4Humanoid framework on challenging humanoid robot tasks, and (ii) conduct a systematic analysis of the application of SRL for humanoid WBC. Specifically, we formulate the following research questions:

- **Q1:** Can the proposed PvP algorithm outperform the baseline methods?
- **Q2:** How does the proportion of training time affect SRL’s performance?
- **Q3:** How does the proportion of training data affect SRL’s performance?
- **Q4:** Which encoder (policy or value) benefits more from applying SRL loss?
- **Q5:** How computation-efficient are these SRL methods?
- **Q6:** How do the SRL-enhanced methods behave in real-world deployment?

5.3.4.1 Experimental Setup

a) Benchmark Design: As shown in Figure 5.26, we utilize the LimX Oli as the test platform, which is a full-size humanoid robot with 31 degrees of freedom (DoF). Based on this platform, we design two tasks: *LimX-Oli-31dof-Velocity* and *LimX-Oli-31dof-Mimic*. Specifically, *LimX-Oli-31dof-Velocity* requires the robot to track a velocity command on flat terrain, which is resampled every 10 seconds. The range of linear velocity on the x -axis is $(-0.5, 1.0)$ m/s, $(-0.3, 0.3)$ m/s on the y -axis, and

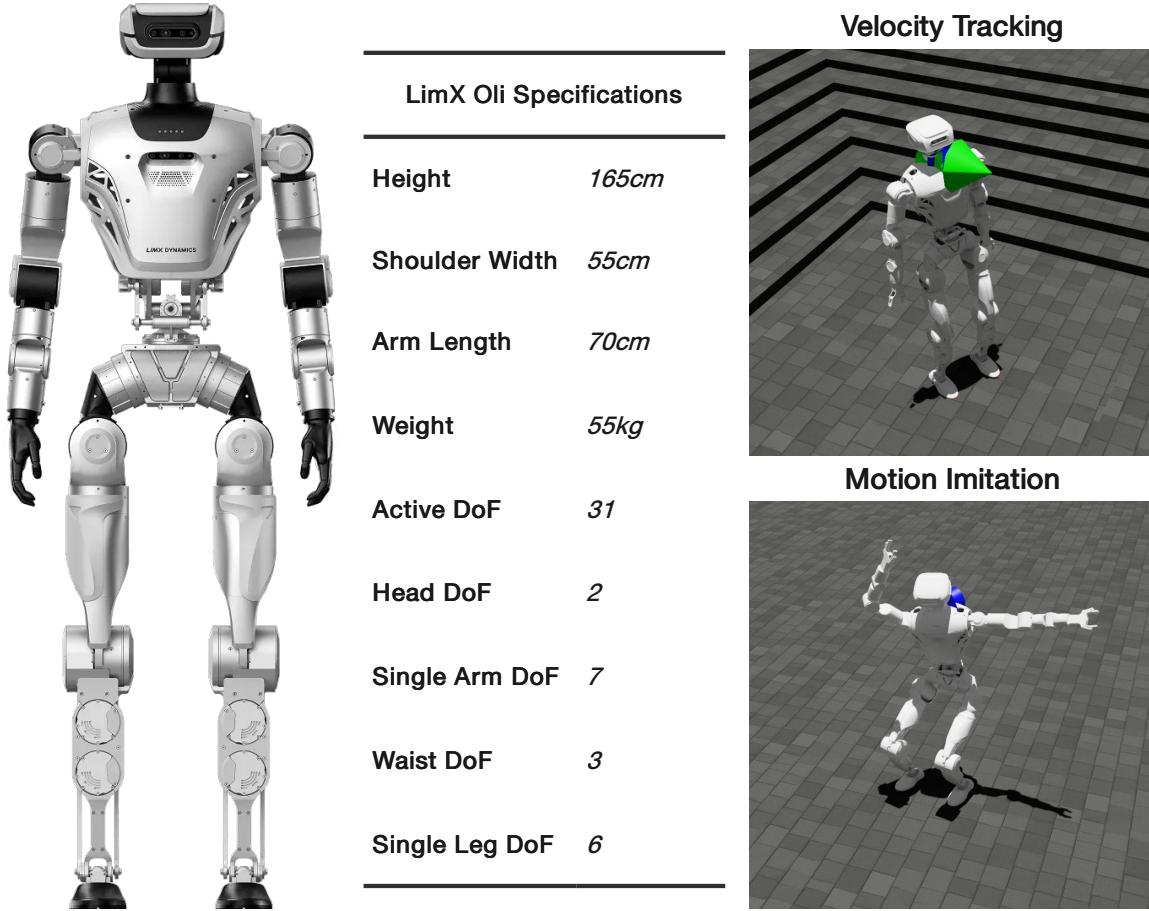


Figure 5.26: The specifications of the LimX Oli humanoid robot used in the experiments, and the screenshots of the two designed tasks.

the angular velocity on the z -axis is $(-1.0, 1.0)$ rad/s. For *LimX-Oli-31dof-Mimic*, it requires the robot to imitate different pre-recorded human animations. We use a set of 20 pre-recorded human motions, each with a maximum length of 43 seconds and 4,300 frames. The two tasks encompass the primary categories of evaluation benchmarks in current humanoid robot research [204].

b) Algorithmic Baselines: As introduced in Section 5.3.3, we use PPO as the backbone RL algorithm and combine it with different SRL approaches. For SRL approaches, we implement SimSiam [31], SPR [174], and VAE [90]. By default, these SRL loss terms are applied to the policy encoder. We also conduct a hyperparameter

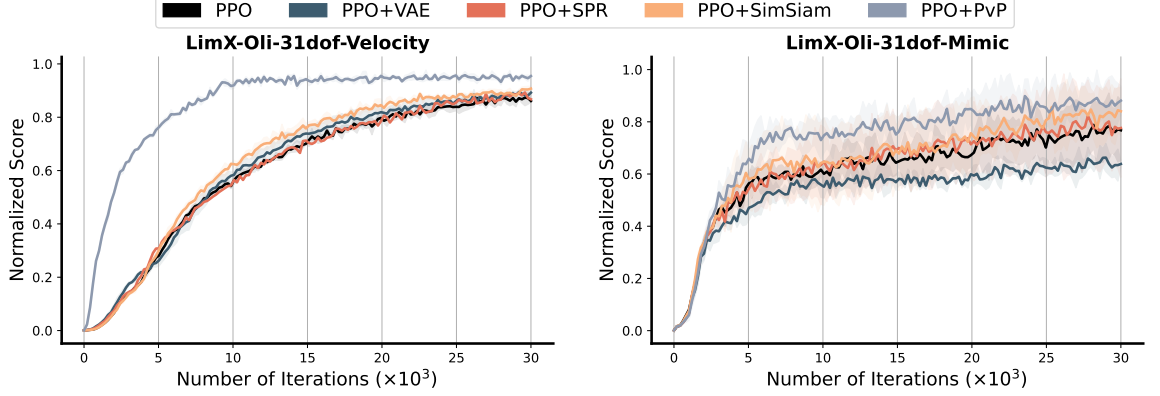


Figure 5.27: Training progress comparison between the vanilla PPO agent and its combination with four SRL methods on the two humanoid WBC tasks. The solid line and shaded region denote the mean and standard deviation, respectively.

search for these methods to determine the initial hyperparameters, such as data augmentation operations (*e.g.*, Gaussian noise and random masking) and loss coefficients. To make a fair comparison, all the methods share the same network architectures and training steps.

c) Evaluation Metrics: The evaluation metrics are fourfold. First, we compare the overall task performance of all the methods, which are computed as the weighted summation of all the sub-reward functions. Second, we compare the key performance indicators (KPIs) for each task, including velocity tracking accuracy in *LimX-Oli-31dof-Velocity* and position alignment accuracy in *LimX-Oli-31dof-Mimic*. Furthermore, we compare the training efficiency of all the methods, *i.e.*, if the method can achieve a higher convergence speed. Finally, we also consider the effectiveness of learned policies in real-world deployment. For example, the agent must effectively optimize the action smoothness reward term to prevent the robot from moving violently during real-time control.

5.3.4.2 Result Analysis

The following results analysis is performed based on the pre-defined research questions.

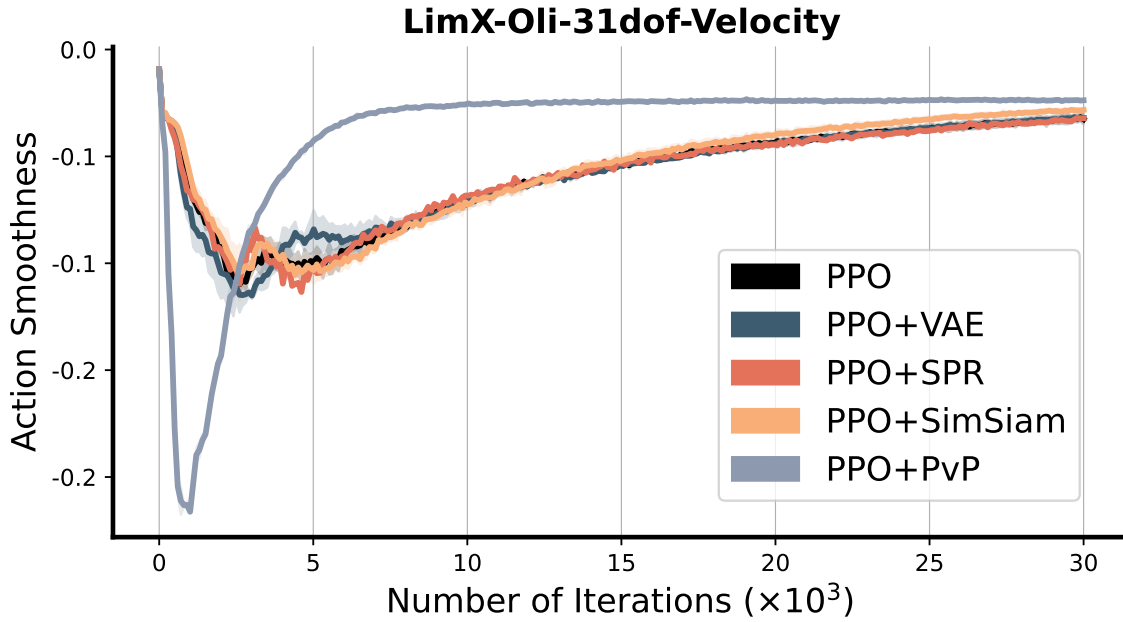


Figure 5.28: The comparison of action smoothness optimization between the vanilla PPO agent and its combinations with the four SRL methods. The solid line and shaded region denote the mean and standard deviation, respectively.

a) Task Performance Comparison: We first analyze the overall effect of integrating SRL into the WBC learning process, *i.e.*, the accumulation of all reward terms. Figure 5.27 illustrates the overall reward comparison between the vanilla PPO agent and its combination with the four SRL methods on the two humanoid WBC tasks. For the velocity tracking task, our PvP method significantly accelerates the learning process, while other SRL methods produce marginal improvements in learning speed. This demonstrates the advantage of leveraging privileged information to enhance SRL, allowing the agent to extract more informative features from noisy and redundant sensory inputs, thereby accelerating the learning process. In con-

5.3. PvP: Data-Efficient Humanoid Robot Learning with Proprioceptive-Privileged Contrastive Representations

trast, for the motion imitation task, two of the SRL methods outperform the vanilla PPO agent, with PvP achieving the highest performance. However, the VAE method shows performance degradation, indicating that simply reconstructing sensory data is insufficient to enhance the robot’s learning efficiency. These results demonstrate that learning high-quality features can improve both learning efficiency and the final performance in humanoid WBC tasks.

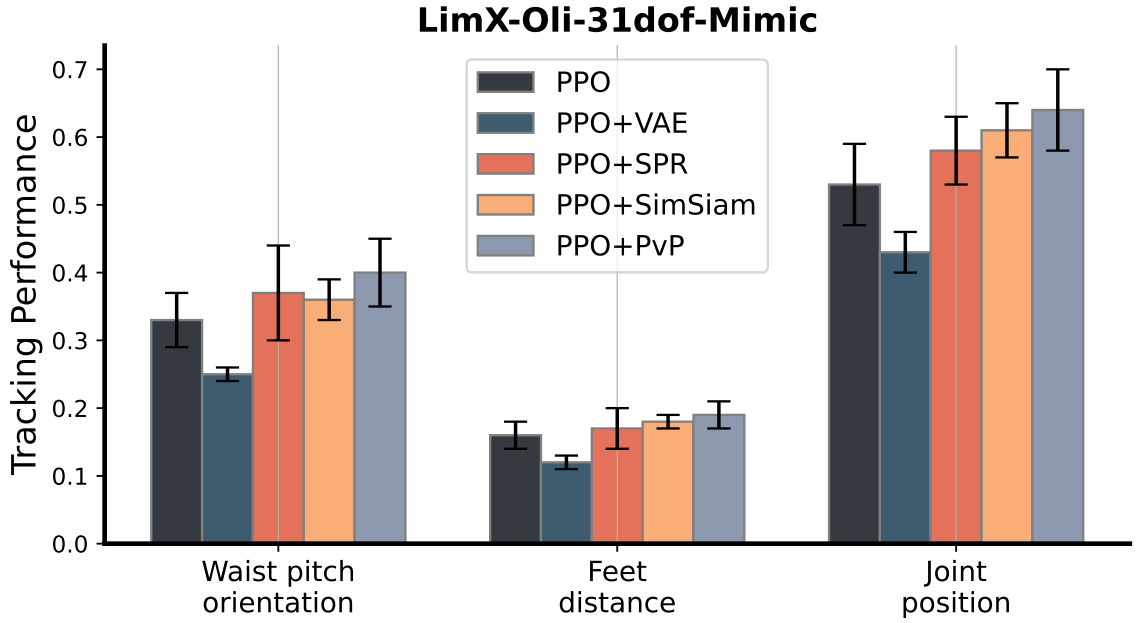


Figure 5.29: The tracking performance comparison between the PPO agent and its combinations with the four SRL methods. Our PvP achieves the highest performance in terms of the three key tracking metrics.

Next, we evaluate the performance of these methods based on specific reward terms. For the velocity tracking task, we compare the optimization of the action smoothness term, defined as the difference between the robot’s actions across three consecutive frames. This term helps prevent the robot from making abrupt movements, ensuring smoother, more controlled motions during real-world deployment. As shown in Figure 5.28, PvP significantly accelerates the convergence of this penalty term. This indicates that our PvP method can not only accelerate the policy learning in sim-

ulation but also guarantee the reliability of real-world deployment. In contrast, the motion imitation task places higher demands on control precision. Figure 5.29 illustrates the comparison of PvP and other baseline algorithms across three key metrics. It is clear that the PvP approach not only improves overall performance but also optimizes critical KPIs, providing reliable increments across diverse tasks.

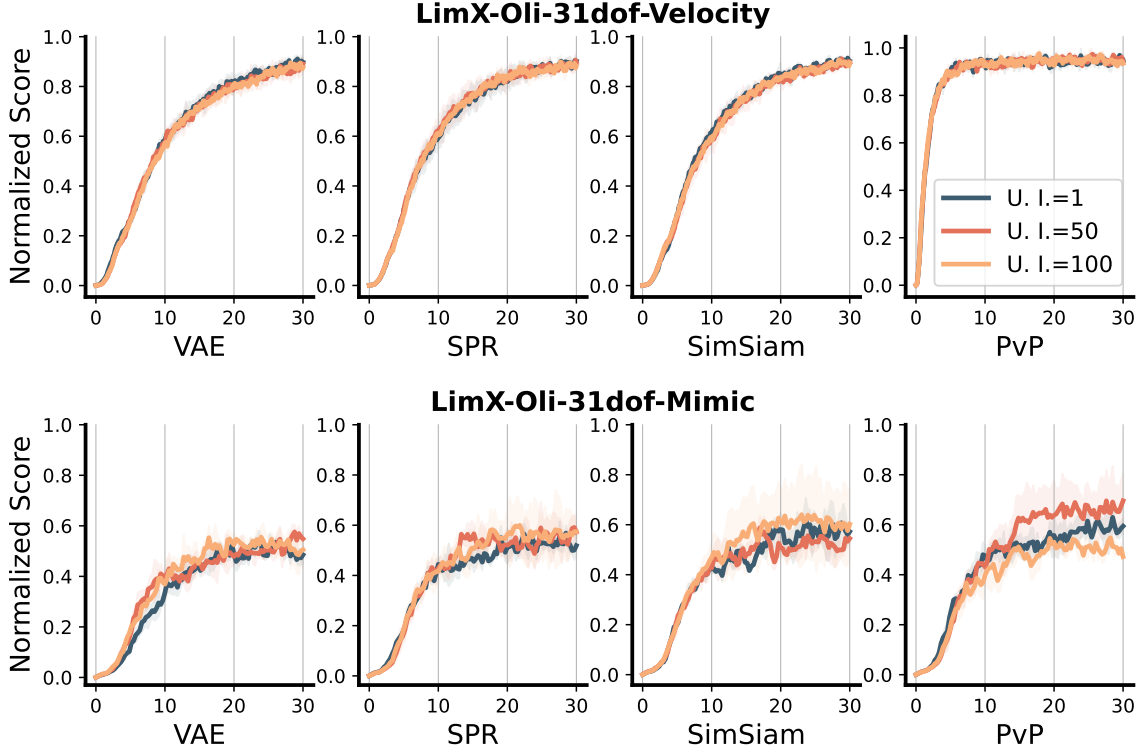


Figure 5.30: Training progress comparison of the four SRL methods with different training time proportions on the two humanoid WBC tasks. The solid line and shaded region denote the mean and standard deviation, respectively.

b) Impact of Training Time Proportion: All the SRL methods are designed to be updated synchronously with the RL part as an auxiliary task. In the early stages of training, SRL helps the agent quickly understand its environment by providing informative state representations. However, as training progresses and the agent becomes more adept at the task, the need for continuous SRL updates diminishes. Additionally, during the initial stages, large amounts of homogeneous and low-quality

data in the early stage are likely to cause the SRL module to prematurely fall into local optima. Thus, it is worthwhile to investigate how the proportion of training time impacts SRL’s performance.

We evaluate different update intervals (1, 50, and 100) for each SRL method. As shown in Figure 5.30, adjusting the update intervals has minimal effect on the velocity tracking task, but a clear impact on the motion imitation task. Specifically, an update interval of 50 is generally optimal for all the SRL methods. These results indicate that carefully selecting the update interval can improve the performance of SRL, prevent premature convergence to local optima, and boost overall training efficiency by reducing computational overhead.

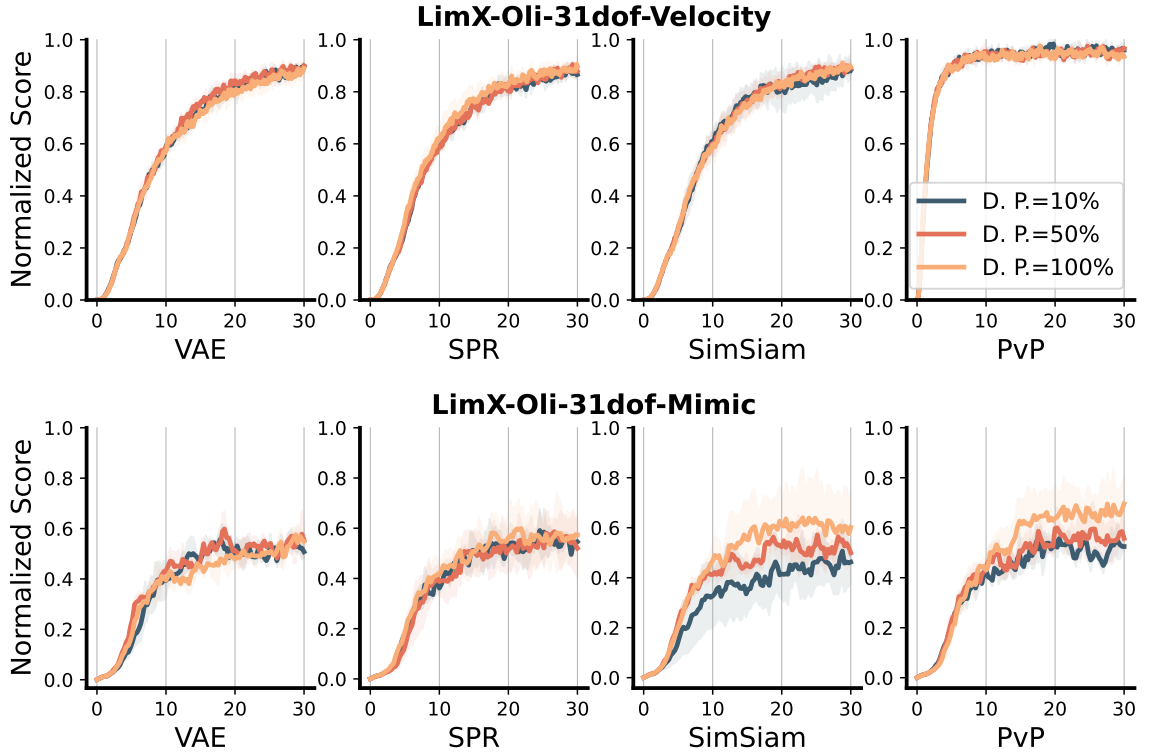


Figure 5.31: Training progress comparison of the four SRL methods with different training data proportions on the two humanoid WBC tasks. The solid line and shaded region denote the mean and standard deviation, respectively.

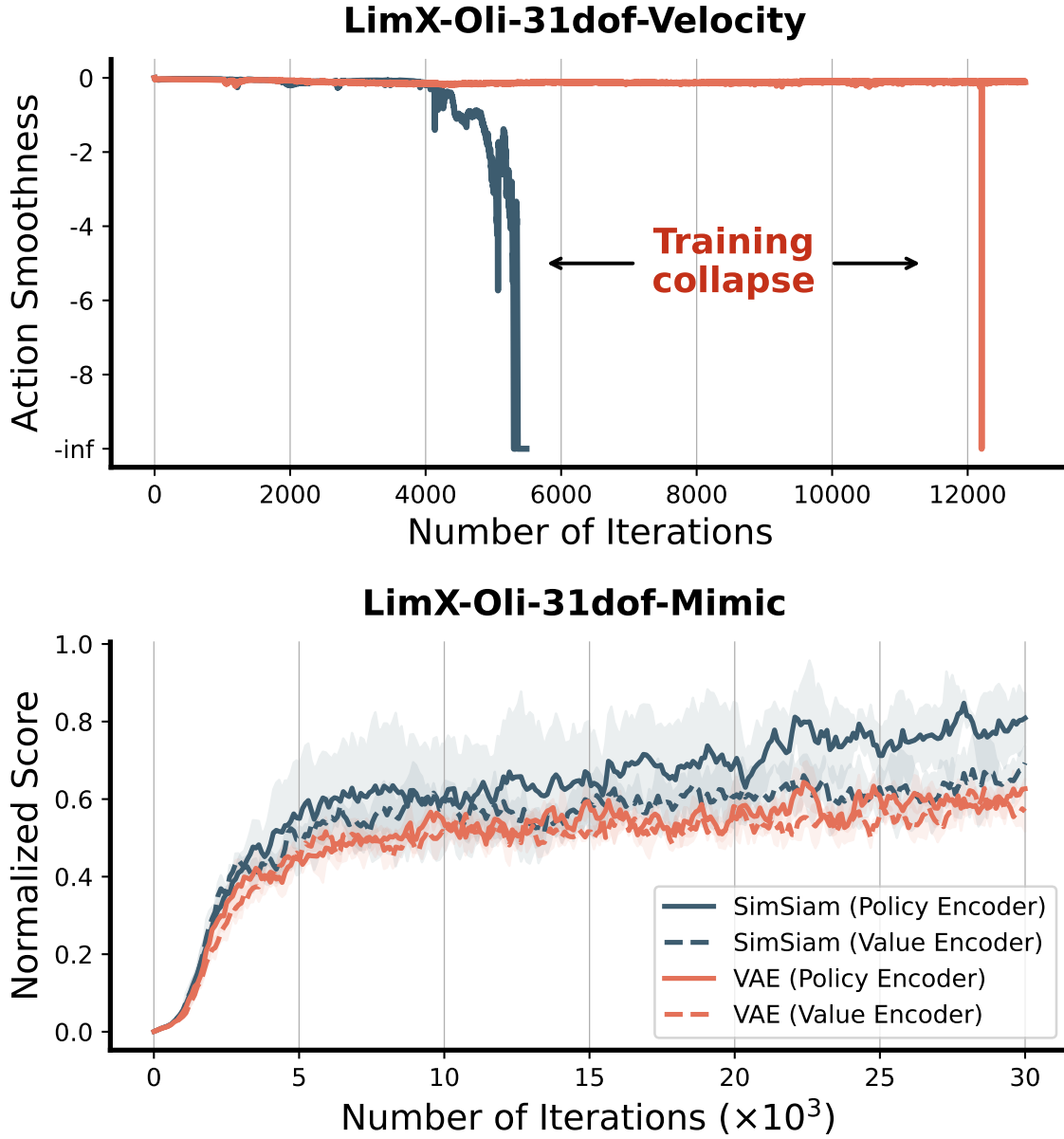


Figure 5.32: Learning curves of applying the SRL to the value encoder. The solid line and shaded region denote the mean and standard deviation, respectively.

c) Impact of Training Data Proportion: Likewise, we examine the impact of the proportion of training data on SRL’s performance, inspired by [212], which investigates the impact of the training data proportion on intrinsic rewards. We train each SRL method using 10%, 50%, and 100% of the rollouts data sampled by the agent in

each episode, with data resampled through a random masking operation. Figure 5.31 illustrates the training progress comparison of the four SRL methods across different training data proportions. For the velocity tracking task, using different proportions results in nearly identical training curves. In contrast, increasing the proportion generally enhances performance, particularly for the SimSiam and PvP methods. These results demonstrate that allocating an appropriate proportion of training data accelerates learning and improves performance, especially in the motion imitation task.

d) Impact of SRL on Value Encoder: Previous experiments have focused on applying SRL to the policy encoder to improve the understanding of the proprioceptive state. Correspondingly, it is worth exploring the impact of SRL on the value encoder. We implement this functionality in SRL4Humanoid and conduct ablation experiments using combinations of the PPO agent and two SRL methods (SPR requires state-action pairs for training). As shown in Figure 5.32, applying SRL loss to the value encoder leads to slower convergence compared to applying SRL loss to the policy encoder. Specifically, in the velocity tracking task, we observe a collapse in training when SRL is applied to the value encoder, as indicated by the sharp drop in action smoothness before recovery. These findings suggest that applying SRL to the policy encoder yields a more stable learning process and enhanced performance.

e) Impact of the λ : Furthermore, we investigate the impact of the weighting coefficients on the performance of our PvP method. As shown in Figure 5.33, $\lambda = 0.1$ achieves the best final performance and sample efficiency, indicating that PvP benefits from a relatively smaller λ .

f) Comparison with Teacher-Student Distillation: Both teacher-student distillation (TSD) [93, 104] and PvP aim to leverage privileged information to guide representation learning, thereby reducing the complexity of learning for high-dimensional humanoid control. However, TSD has several critical limitations. The student’s performance ceiling is strictly constrained by the teacher’s quality and inductive biases,

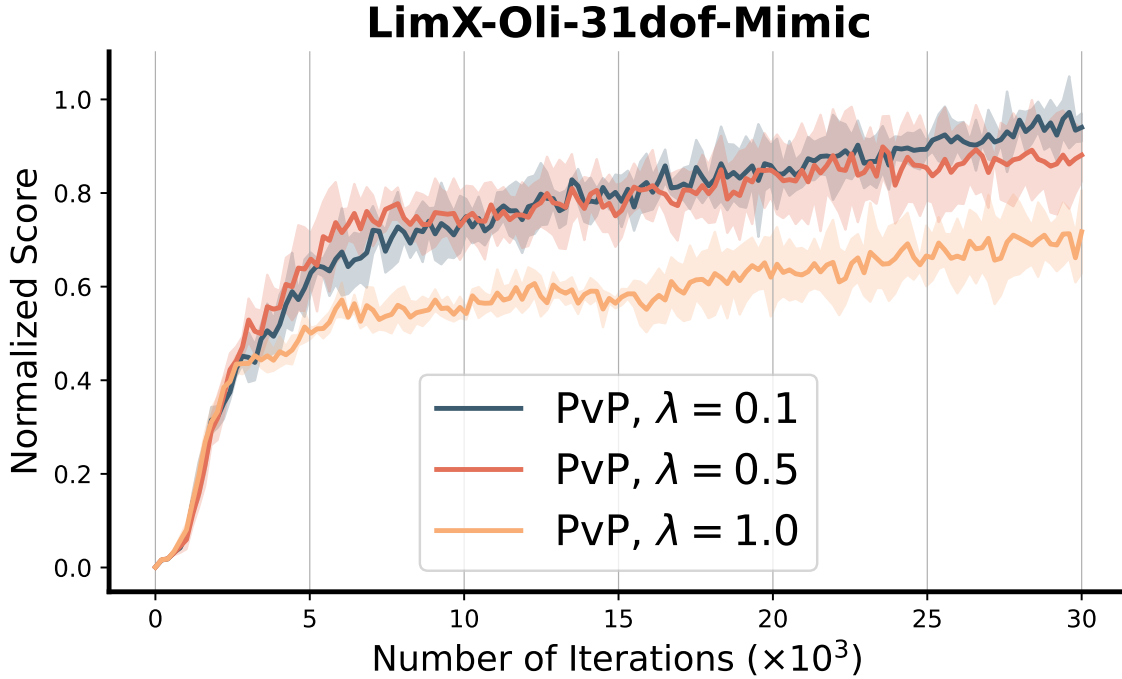


Figure 5.33: Performance comparison of the PvP method with different weighting coefficients on the *LimX-Oli-31dof-Mimic* task. The solid line and shaded region denote the mean and standard deviation, respectively.

and any sub-optimality or observation mismatches in the teacher are directly inherited by the student. As shown in the experiments below, the student fails to match the teacher’s performance and exhibits a significant gap after distillation. Moreover, strict alignment objectives can over-regularize the student and suppress exploration, causing the student to collapse toward conservative or averaged behavior rather than discovering alternative (potentially better) solutions. Finally, the teacher pre-training stage is often compute-intensive, limiting reproducibility and rapid iteration as tasks scale up.

In contrast, PvP facilitates a synergy between representation learning and policy optimization. Unlike the disjointed two-stage pipeline of TSD, PvP integrates contrastive representation learning directly into the RL framework. This enables the latent space to evolve alongside the policy’s exploration, rather than merely replicating a static

5.3. PvP: Data-Efficient Humanoid Robot Learning with Proprioceptive-Privileged Contrastive Representations

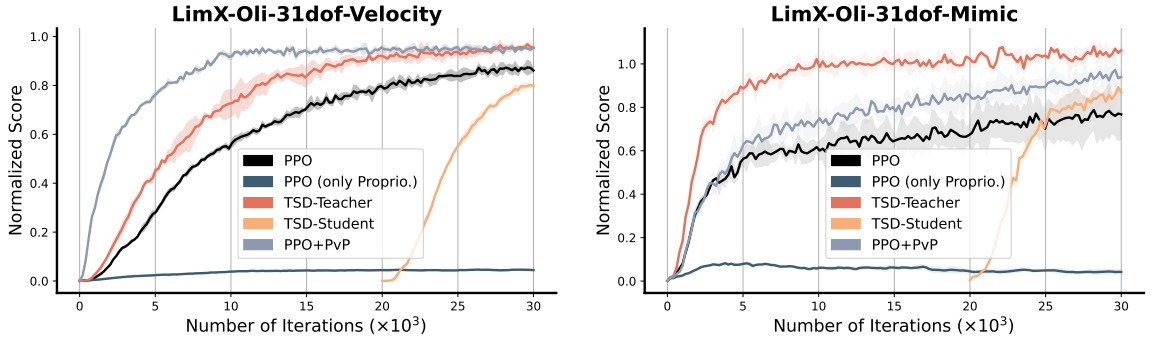


Figure 5.34: Training progress comparison between the teacher-student distillation method and PvP in the *LimX-Oli-31dof-Velocity* task and *LimX-Oli-31dof-Mimic* task. The solid line and shaded region denote the mean and standard deviation, respectively.

teacher. By optimizing a contrastive objective that captures structural invariants, the representation and policy mutually enhance each other. The policy benefits from a more discriminative latent space, while the representation adaptively prioritizes task-relevant features discovered during training.

To demonstrate PvP’s advantage over the TSD method, we conduct ablation experiments on the two designed humanoid WBC tasks. As illustrated in Figure 5.34, there is a significant performance gap between the teacher and student policies, and consistently outperforms the student policy. Notably, the PPO agent that relies solely on proprioceptive state information fails to learn in both tasks, highlighting the need for privileged information to support learning.

g) Evaluation on More Diverse Tasks:

To evaluate PvP on more diverse scenarios and robot platforms, we introduce two new tasks: *LimX-Oli-31dof-Velocity-Rough* and *Unitree-G1-29dof-Velocity*. The former is a variant of the previously defined velocity-tracking task that introduces rough terrain, while the latter uses a different robot platform. As illustrated in Figure 5.35, PvP excels in the two tasks in terms of final performance and time cost.

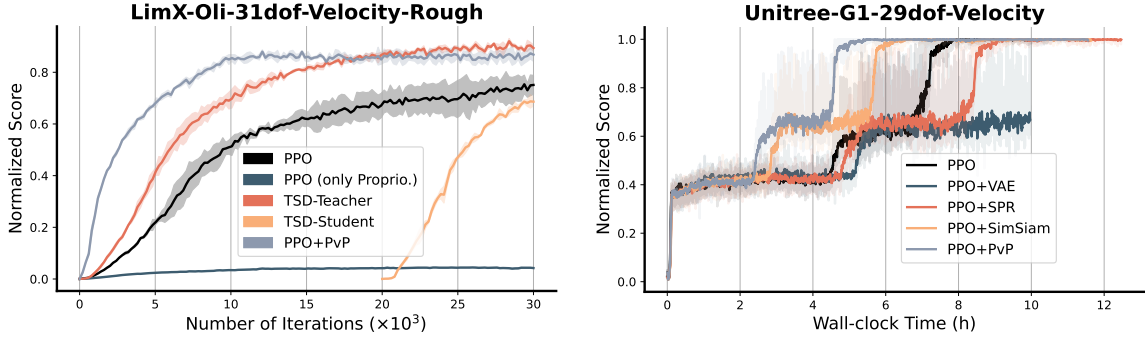


Figure 5.35: (Left) Training progress comparison between the teacher-student distillation method and PvP in the *LimX-Oli-31dof-Velocity-Rough* task. (Right) Training progress comparison between the vanilla PPO agent and its combination with four SRL methods on the *Unitree-G1-29dof-Velocity* task. The solid line and shaded region denote the mean and standard deviation, respectively.

h) Computational Efficiency Comparison: Furthermore, we investigate the computational efficiency of these SRL methods. Equipped with IsaacLab [140], all experiments can be run on a single GPU, and a RTX4090 with 24GB memory is used for all our experiments. Our implementation enables the SRL module to run entirely on the GPU, without affecting overall training efficiency. These results demonstrate that SRL4Humanoid can effectively accelerate humanoid WBC tasks with minimal computational resource cost.

i) Real-world Evaluation: To assess the effectiveness of our approach in real-world scenarios, we first conduct a thorough Sim2Sim evaluation on the MuJoCo platform [195], which provides simulation precision closer to real-world conditions than IsaacLab [134]. Figure 5.36 provides a clear demonstration of the robot’s ability to execute complex tasks with the learned policy. Following the simulation evaluation, we proceed with real-robot testing on the LimX Oli humanoid robot, as shown in Figure 5.23. More demonstrations can be found in the Supporting Videos in the supplementary materials.

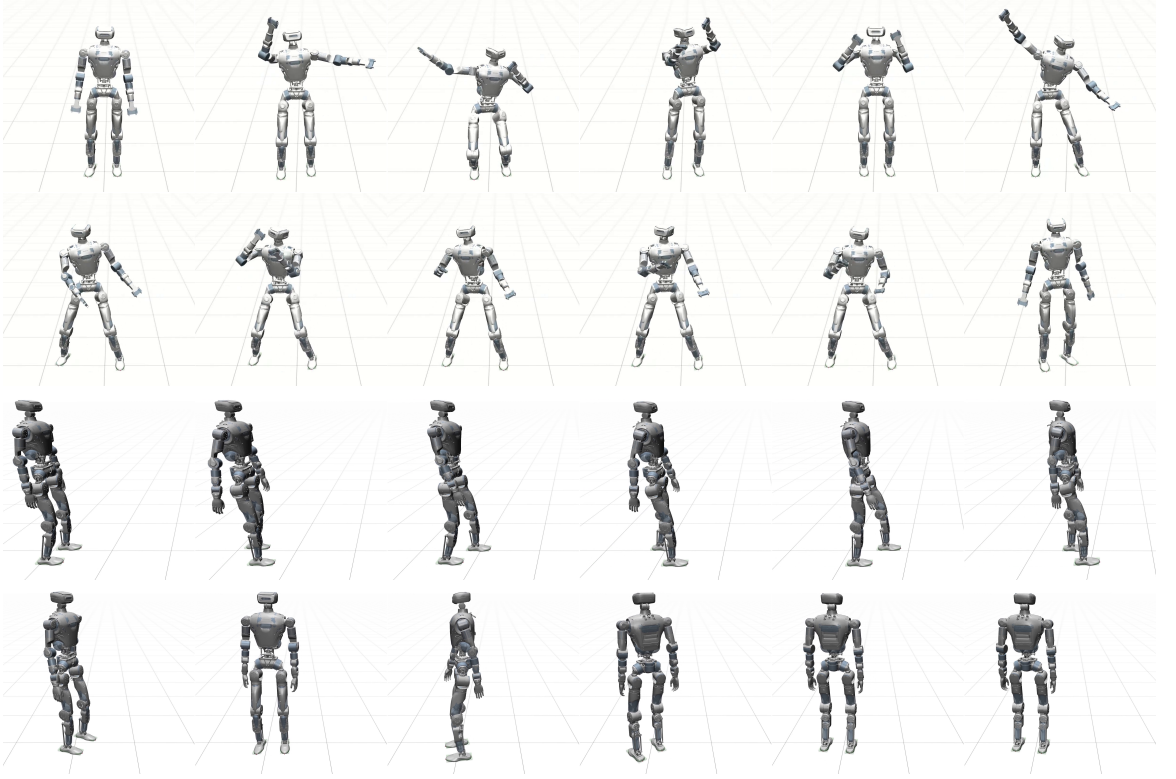


Figure 5.36: Sim2Sim evaluation on the MuJoCo simulator. The first two rows demonstrate motion imitation ability, and the last two rows show velocity tracking ability.

5.3.5 Conclusion

In this section, we propose PvP, a proprioceptive-privileged contrastive learning framework designed to enhance sample efficiency and performance in humanoid WBC tasks. By leveraging the intrinsic complementarity of the two state modalities, PvP enhances proprioceptive representations for policy learning without relying on hand-crafted data augmentations. We also introduce the SRL4Humanoid framework, which provides high-quality, modular implementations of representative SRL techniques for humanoid robot learning. Extensive experiments on the LimX Oli humanoid robot demonstrate the effectiveness of our approach, showing significant improvements over baseline methods. These results highlight the potential of SRL-empowered RL for

humanoid WBC tasks, providing valuable insights for future research in data-efficient humanoid robot learning.

Still, there are currently remaining limitations to this work. While we have demonstrated the efficacy of the PvP framework using several SRL methods, future research could explore integrating additional SRL techniques to further enhance policy learning. Additionally, recent advancements in perception-based humanoid research have shown the potential of incorporating multimodal data, such as RGB or depth images, into policy learning. We aim to extend our work to these settings, thereby expanding the capabilities of humanoid robots in more complex environments.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis investigates how to systematically accelerate RL by enhancing exploration efficiency and exploitation effectiveness. Motivated by the persistent challenges of sparse rewards, partial observability, and severe sample inefficiency in complex control tasks, we study intrinsic motivation, data exploitation scheduling, hyperparameter optimization, and state representation learning. Our unified objective is to enable RL agents to learn more quickly, stably, and reliably a variety of increasingly complex environments.

From the exploration perspective, this thesis conducts in-depth technical investigations into the intrinsically-motivated exploration mechanism. In the early chapters, we demonstrate that naive or poorly designed intrinsic rewards frequently suffer from vanishing incentives, biased optimization objectives, and substantial computational overhead. To address these limitations, we introduce a series of principled intrinsic reward mechanisms and frameworks. REVD provides a computation-efficient intrinsic reward by directly leveraging visitation discrepancies across different periods, offering strong exploration incentives without requiring auxiliary representa-

tion learning or complex model design. Furthermore, AIRS automatically and adaptively selects intrinsic reward functions across different learning stages, alleviating the biased-objective issue and ensuring reliable exploration. Finally, through the RL-eXplore framework, we consolidate intrinsic motivation research into a unified and reproducible benchmark, revealing critical implementation details and standardizing best practices. Together, these contributions significantly improve the robustness, efficiency, and reproducibility of intrinsically-motivated RL.

In addition to exploring efficiency, this thesis aims to investigate task-aware data exploitation mechanisms, motivated by the non-stationary nature of RL across different tasks and learning phases. To this end, we propose ADEPT, an adaptive data exploitation scheduling strategy that dynamically controls update intensity across learning stages and tasks. By formulating exploitation scheduling as a bandit problem, ADEPT automatically balances computational cost and performance gain, significantly improving data efficiency. Additionally, this thesis investigates the long-standing challenge of hyperparameter optimization in RL, with a particular focus on developing lightweight, efficient, and generally deployable optimization strategies. ULTHO reframes hyperparameter optimization as a hierarchical bandit problem, enabling efficient, stable, and low-overhead hyperparameter adaptation within single training runs. This perspective avoids the prohibitive costs and instability of conventional HPO methods while delivering consistent long-term performance improvements.

Beyond algorithm innovation and evaluation in simulated environments, this thesis further moves toward real-world applications to examine the practical impact of accelerated RL. In particular, we focus on humanoid WBC as a representative and challenging application domain, where learning efficiency, stability, and robustness are critical for practical use. In response, this thesis introduces PvP, a proprioceptive-privileged contrastive learning framework that leverages the intrinsic complementarity between proprioceptive and privileged states. PvP learns compact and task-relevant latent

representations without requiring hand-crafted data augmentations, enabling faster and more stable policy learning. We also introduce the SRL4Humanoid framework, the first unified and modular framework that provides high-quality implementations of representative SRL methods for humanoid robot learning. Experiments in simulation and real-world deployment demonstrate that PvP consistently improves sample efficiency and policy performance in humanoid WBC tasks such as motion imitation and velocity tracking, offering valuable guidance for data-efficient humanoid robot learning.

6.2 Future Work

The work in this thesis has opened up two new directions for future work.

Integrated and Adaptive Optimization of Exploration and Exploitation. A promising future direction lies in the systematic integration of the developed exploration and exploitation techniques into a unified and adaptive framework. While the current thesis advances each perspective individually, their synergistic combination could offer a pathway to break the fundamental exploration-exploitation dilemma. For instance, the aggregated system can not only apply REVD, AIRS, ADEPT, ULTHO, and PvP in parallel, but also orchestrate them through a meta-controller that dynamically balances resource allocation. The goal is to develop a holistic learning system that intelligently shifts its focus between seeking novel information and refining its policy based on a unified measure of learning progress, thereby achieving more robust, stable, and sample-efficient RL across a wider spectrum of environments.

Application to Large-Scale and Cost-Sensitive Real-World Scenarios. While the proposed algorithms consider general RL settings and are evaluated on standard benchmarks, a critical and high-impact direction is their application to large-scale, real-world problems where data efficiency and training stability are paramount. A

primary candidate is the post-training and alignment of LLMs, a critical process that is notoriously resource-intensive. Future work could investigate how the developed intrinsic reward mechanisms can be effectively integrated into the RL-based post-training of LLMs, where exploration incentives could help the model discover more nuanced or novel behavioral modes, especially for enhancing its reasoning capability. Meanwhile, as data collection is extremely expensive, exploitation enhancement techniques such as ADEPT and the ultra-lightweight HPO of ULTHO can be applied to maximize data utility and significantly reduce the overall number of training iterations. In summary, the systematic deployment of these RL acceleration techniques in complex and resource-intensive domains, such as LLM alignment, will effectively validate their practical effectiveness beyond simulated benchmarks and contribute to the development of a more efficient and scalable RL paradigm to support the next generation of large AI systems.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.
- [3] Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pages 421–438. PMLR, 2023.
- [4] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021.
- [5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Woj-

- ciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.
- [6] Remzi H Arpaci-Dusseau and Andrea C Arpaci-Dusseau. Operating systems: Three easy pieces. 2018.
- [7] Arthur Aubret, Laetitia Matignon, and Salima Hassas. An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey. *Entropy*, 25(2):327, 2023.
- [8] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of machine learning research*, 3(Nov):397–422, 2002.
- [9] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2020.
- [10] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, et al. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2018.
- [11] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.
- [12] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.
- [13] Juan Cruz Barsce, Jorge Palombarini, and Ernesto Martinez. A hierarchical two-tier approach to hyper-parameter optimization in reinforcement learning.

-
- SADIO Electronic Journal of Informatics and Operations Research*, 19(2):2–27, 2020.
- [14] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.
- [15] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Proceedings of Advances in Neural Information Processing Systems*, 29:1471–1479, 2016.
- [16] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.
- [17] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.
- [18] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.
- [19] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- [20] Nicolas Bougie and Narimasa Watanabe. Exploring beyond curiosity rewards: Language-driven exploration in RL. In *The 16th Asian Conference on Machine Learning (Conference Track)*, 2024.

- [21] Robert Briscoe, Rick Grush, and Alison Springler. *Action-based Theories of Perception*. Metaphysics Research Lab, Stanford University, Fall 2023 edition, 2023.
- [22] 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.
- [23] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *Proceedings of the International Conference on Learning Representations*, pages 1–17, 2019.
- [24] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *Proceedings of the International Conference on Learning Representations*, pages 1–17, 2019.
- [25] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *Proceedings of the 7th International Conference on Learning Representations*, pages 1–17, 2019.
- [26] Emil Carlsson, Devdatt Dubhashi, and Fredrik D. Johansson. Thompson sampling for bandits with clustered arms. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2212–2218. International Joint Conferences on Artificial Intelligence Organization, 8 2021.
- [27] Roger Creus Castanyer, Joshua Romoff, and Glen Berseth. Improving intrinsic exploration by creating stationary objectives. *arXiv preprint arXiv:2310.18144*, 2023.
- [28] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. *Advances in neural information processing systems*, 30, 2017.

-
- [29] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24, 2011.
- [30] Eric R Chen, Zhang-Wei Hong, Joni Pajarinen, and Pulkit Agrawal. Redeeming intrinsic rewards via constrained policy optimization. In *Advances in Neural Information Processing Systems*, 2022.
- [31] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15745–15753, 2021.
- [32] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.
- [33] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. In *Robotics: Science and Systems*, 2024.
- [34] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*, December 2023.
- [35] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic grid-world environment for gymnasium, 2018.
- [36] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

- [37] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.
- [38] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.
- [39] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. URL <http://pybullet.org>, 2016–2018.
- [40] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158, 1975.
- [41] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. Supervised learning. In *Machine learning techniques for multimedia: case studies on organization and retrieval*, pages 21–49. Springer, 2008.
- [42] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, volume 2, page 3, 2008.
- [43] J Russo Daniel, Van Roy Benjamin, Kazerouni Abbas, Osband Ian, and Wen Zheng. A tutorial on thompson sampling. *Foundations and trends® in machine learning*, 11(1):1–99, 2018.
- [44] Amodei Dario and Hernandez Danny. Ai and compute. *OpenAI blog*, 2018.
- [45] Kouros Darvish, Luigi Penco, Joao Ramos, Rafael Cisneros, Jerry Pratt, Eiichi Yoshida, Serena Ivaldi, and Daniele Pucci. Teleoperation of humanoid robots: A survey. *IEEE Transactions on Robotics*, 39(3):1706–1727, 2023.

-
- [46] Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.
- [47] Nicolai Dorka, Tim Welschhold, and Wolfram Burgard. Dynamic update-to-data ratio: Minimizing world model overfitting. In *The Eleventh International Conference on Learning Representations*, 2023.
- [48] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.
- [49] Mhairi Dunion, Trevor McInroe, Kevin Luck, Josiah Hanna, and Stefano Albrecht. Conditional mutual information for disentangled representations in reinforcement learning. *Advances in Neural Information Processing Systems*, 36:80111–80129, 2023.
- [50] Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V Albrecht. Temporal disentanglement of representations for improved generalisation in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.
- [51] Ayoub Echchahed and Pablo Samuel Castro. A survey of state representation learning for deep reinforcement learning. *Transactions on Machine Learning Research*, 2025. Survey Certification.
- [52] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *International conference on machine learning*, pages 9104–9149. PMLR, 2023.
- [53] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in

- deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.
- [54] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.
- [55] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.
- [56] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 1146–1155. PMLR, 2017.
- [57] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.
- [58] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.
- [59] Jörg K.H. Franke, Gregor Koehler, André Biedenkapp, and Frank Hutter. Sample-efficient automated deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

-
- [60] Anna Goldie, Azalia Mirhoseini, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. Addendum: A graph placement methodology for fast chip design. *Nature*, pages 1–2, 2024.
- [61] Richard Gross. *Psychology: The science of mind and behaviour 6th Edition*. Hodder Education, 2012.
- [62] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.
- [63] Zhaohan Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35:31855–31870, 2022.
- [64] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.
- [65] 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.
- [66] Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. *Advances in neural information processing systems*, 31, 2018.

- [67] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.
- [68] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Overview of supervised learning. In *The elements of statistical learning: Data mining, inference, and prediction*, pages 9–41. Springer, 2008.
- [69] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning: Data mining, inference, and prediction*, pages 485–585. Springer, 2008.
- [70] Mikael Henaff, Minqi Jiang, and Roberta Raileanu. A study of global and episodic bonuses for exploration in contextual mdps. *arXiv preprint arXiv:2306.03236*, 2023.
- [71] Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via elliptical episodic bonuses. *Advances in Neural Information Processing Systems*, 35:37631–37646, 2022.
- [72] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [73] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.
- [74] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

-
- [75] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.
- [76] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Proceedings of Advances in neural information processing systems*, 29:1109–1117, 2016.
- [77] Chloe Ching-Yun Hsu, Celestine Mendler-Dünnér, and Moritz Hardt. Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*, 2020.
- [78] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *International Conference on Learning Representations*, 2022.
- [79] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022.
- [80] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃGo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.
- [81] Tao Huang, Junli Ren, Huayi Wang, Zirui Wang, Qingwei Ben, Muning Wen, Xiao Chen, Jianan Li, and Jiangmiao Pang. Learning humanoid standing-up control across diverse postures. In *Robotics: Science and Systems*, 2025.
- [82] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

- [83] Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Rajiv Didolkar, Dipendra Misra, Xin Li, Harm Van Seijen, Remi Tachet Des Combes, et al. Principled offline rl in the presence of rich exogenous information. In *International Conference on Machine Learning*, pages 14390–14421. PMLR, 2023.
- [84] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.
- [85] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.
- [86] Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking exploration. *Royal Society Open Science*, 10(6):230539, 2023.
- [87] Daejin Jo, Sungwoong Kim, Daniel Nam, Taehwan Kwon, Seungeun Rho, Jongmin Kim, and Donghoon Lee. Leco: Learnable episodic count for task-specific intrinsic reward. *Advances in Neural Information Processing Systems*, 35:30432–30445, 2022.
- [88] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.
- [89] Steven Kapturowski, Alaa Saade, Daniele Calandriello, Charles Blundell, Pablo Sprechmann, Leopoldo Sarra, Oliver Groth, Michal Valko, and Bilal Piot. Un-

- locking the power of representations in long-term novelty-based exploration. In *Second Agent Learning in Open-Endedness Workshop*.
- [90] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.
- [91] Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew P Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3):196–210, 2022.
- [92] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.
- [93] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. 2021.
- [94] Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.
- [95] Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.
- [96] Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Rajiv Didolkar, Dipendra Misra, Dylan J Foster, Lekan P Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of control-endogenous latent states

- with multi-step inverse models. *Transactions on Machine Learning Research*, 2025.
- [97] Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library. <http://github.com/RobertTLange/gymnax>, 2022.
- [98] Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. *Advances in Neural Information Processing Systems*, 35:34478–34491, 2022.
- [99] Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 34478–34491, 2022.
- [100] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pages 5639–5650. PMLR, 2020.
- [101] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.
- [102] Misha Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.
- [103] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

-
- [104] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
 - [105] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *International Conference on Learning Representations*, 2020.
 - [106] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.
 - [107] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.
 - [108] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations*, 2023.
 - [109] Yixuan Li, Yutang Lin, Jieming Cui, Tengyu Liu, Wei Liang, Yixin Zhu, and Siyuan Huang. Clone: Closed-loop whole-body humanoid teleoperation for long-horizon tasks. *arXiv preprint arXiv:2506.08931*, 2025.
 - [110] Qiayuan Liao, Takara E Truong, Xiaoyu Huang, Guy Tevet, Koushil Sreenath, and C Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241*, 2025.
 - [111] 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. In *International Conference on Learning Representations*, 2016.

- [112] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.
- [113] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.
- [114] Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. Competitive experience replay. In *International Conference on Learning Representations*, 2019.
- [115] Xiao-Yang Liu, Zechu Li, Ming Zhu, Zhaoran Wang, and Jiahao Zheng. ElegantRL: Massively parallel framework for cloud-native deep reinforcement learning. <https://github.com/AI4Finance-Foundation/ElegantRL>, 2021.
- [116] Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849, 2022.
- [117] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. Finrl: Deep reinforcement learning framework to automate trading in quantitative finance. In *Proceedings of the second ACM international conference on AI in finance*, pages 1–9, 2021.
- [118] Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. *arXiv preprint arXiv:2306.03186*, 2023.

-
- [119] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model. *arXiv preprint arXiv:2411.14386*, 2024.
- [120] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.
- [121] Jieliang Luo and Hui Li. Dynamic experience replay. In *Conference on robot learning*, pages 1191–1200. PMLR, 2020.
- [122] Guozheng Ma, Zhen Wang, Zhecheng Yuan, Xueqian Wang, Bo Yuan, and Dacheng Tao. A comprehensive survey of data augmentation in visual reinforcement learning. *International Journal of Computer Vision*, pages 1–38, 2025.
- [123] Guozheng Ma, Linrui Zhang, Haoyu Wang, Lu Li, Zilin Wang, Zhen Wang, Li Shen, Xueqian Wang, and Dacheng Tao. Learning better with less: Effective augmentation for sample-efficient visual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:59832–59859, 2023.
- [124] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.
- [125] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu based physics simulation for robot learning. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

- [126] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.
- [127] Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.
- [128] Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. In *IJCAI*, 2017.
- [129] Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2024.
- [130] Trevor McInroe, Lukas Schäfer, and Stefano V Albrecht. Multi-horizon representations with hierarchical forward models for reinforcement learning. 2025.
- [131] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.
- [132] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

-
- [133] Mayank Mittal, Nikita Rudin, Victor Klemm, Arthur Allshire, and Marco Hutter. Symmetry considerations for learning task symmetric robot policies. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7433–7439. IEEE, 2024.
- [134] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.
- [135] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.
- [136] 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.
- [137] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [138] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.
- [139] Guido Novati and Petros Koumoutsakos. Remember and forget for experience replay. In *International Conference on Machine Learning*, pages 4851–4860. PMLR, 2019.

- [140] NVIDIA. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning, 2025.
- [141] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28, 2015.
- [142] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [143] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the International Conference on Machine Learning*, pages 2721–2730, 2017.
- [144] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.
- [145] Pierre-Yves Oudeyer and Frederic Kaplan. How can we define intrinsic motivation? In *the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Lund University Cognitive Studies, Lund: LUCS, Brighton, 2008.
- [146] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [147] Jack Parker-Holder, Vu Nguyen, Shaan Desai, and Stephen J Roberts. Tuning mixed input hyperparameters on the fly for efficient population based autorl. *Advances in Neural Information Processing Systems*, 34:15513–15528, 2021.

-
- [148] Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in neural information processing systems*, 33:17200–17211, 2020.
 - [149] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.
 - [150] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.
 - [151] Supratik Paul, Vitaly Kurin, and Shimon Whiteson. Fast efficient hyperparameter tuning for policy gradient methods. *Advances in Neural Information Processing Systems*, 32, 2019.
 - [152] Fernando Pérez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE International Symposium on Information Theory*, pages 1666–1670. IEEE, 2008.
 - [153] Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning. In *International Conference on Machine Learning*, pages 7652–7662. PMLR, 2020.
 - [154] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.
 - [155] André Quadros, Cassio Silva, and Ronnie Alves. Llm-driven intrinsic motivation for sparse reward reinforcement learning. *arXiv preprint arXiv:2508.18420*, 2025.

- [156] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of machine learning research*, 22(268):1–8, 2021.
- [157] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.
- [158] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on robot learning*, pages 1634–1644. PMLR, 2022.
- [159] Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 8787–8798. PMLR, 2021.
- [160] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5402–5415, 2021.
- [161] Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*, 2020.
- [162] Carlo Romeo, Girolamo Macaluso, Alessandro Sestini, and Andrew D. Bagdanov. SPEQ: Offline stabilization phases for efficient q-learning in high update-to-data ratio reinforcement learning. In *Reinforcement Learning Conference*, 2025.
- [163] Giulio Romualdi, Stefano Daffarra, Giuseppe L’Erario, Ines Sorrentino, Silvio Traversaro, and Daniele Pucci. Online non-linear centroidal mpc for humanoid robot locomotion with step adjustment. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10412–10419. IEEE, 2022.

-
- [164] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.
- [165] Vishnu Sarukkai, Brennan Shacklett, Zander Majercik, Kush Bhatia, Christopher Ré, and Kayvon Fatahalian. Automated rewards via llm-generated progress functions. *arXiv preprint arXiv:2410.09187*, 2024.
- [166] Laura Saunders and Melissa A Wong. Learning theories: Understanding how people learn. *Instruction in Libraries and Information Centers*, 2020.
- [167] Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphael Marinier, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *Proceedings of the International Conference on Learning Representations*, 2019.
- [168] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.
- [169] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.
- [170] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *Proceedings of the International Conference on Learning Representations*, 2015.
- [171] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

- [172] Clemens Schwarke, Victor Klemm, Matthijs van der Boon, Marko Bjelonic, and Marco Hutter. Curiosity-driven learning of joint locomotion and manipulation tasks. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2594–2610. PMLR, 2023.
- [173] Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.
- [174] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.
- [175] Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.
- [176] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE International Conference on Robotics and Automation*, pages 2641–2648. IEEE, 2006.
- [177] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9443–9454, 2021.
- [178] Jonah Siekmann, Yesh Godse, Alan Fern, and Jonathan Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7309–7315. IEEE, 2021.

-
- [179] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [180] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- [181] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.
- [182] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.
- [183] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.
- [184] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.
- [185] Mark A Sujan, Huayi Huang, and Jeffrey Braithwaite. Learning from incidents in health care: critique from a safety-ii perspective. *Safety science*, 99:115–121, 2017.
- [186] Wandong Sun, Baoshi Cao, Long Chen, Yongbo Su, Yang Liu, Zongwu Xie, and Hong Liu. Learning perceptive humanoid locomotion over challenging terrain. *arXiv preprint arXiv:2503.00692*, 2025.

- [187] Wandong Sun, Long Chen, Yongbo Su, Baoshi Cao, Yang Liu, and Zongwu Xie. Learning humanoid locomotion with world model reconstruction. *arXiv preprint arXiv:2502.16230*, 2025.
- [188] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [189] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [190] 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.
- [191] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
- [192] MuJoCo Team. Mujoco xla.
- [193] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.
- [194] Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirodda. Zero-shot whole-body humanoid control via behavioral foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [195] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

-
- [196] Yuchuang Tong, Haotian Liu, and Zhengtao Zhang. Advancements in humanoid robots: A comprehensive review and future prospects. *IEEE/CAA Journal of Automatica Sinica*, 11(2):301–328, 2024.
- [197] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.
- [198] Kirsten Lynx Valenzuela, Samantha Isabel Roxas, and Yung-Hao Wong. Embodying intelligence: Humanoid robot advancements and future prospects. In *Interacción*, 2024.
- [199] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.
- [200] Hongxi Wang, Haoxiang Luo, Wei Zhang, and Hua Chen. Cts: Concurrent teacher-student reinforcement learning for legged locomotion. *IEEE Robotics and Automation Letters*, 2024.
- [201] Qing Wang, Sanjeev R Kulkarni, and Sergio Verdú. A nearest-neighbor approach to estimating divergence between continuous random vectors. In *2006 IEEE International Symposium on Information Theory*, pages 242–246. IEEE, 2006.
- [202] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6, 2022.

- [203] Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, et al. Envpool: A highly parallel reinforcement learning environment execution engine. *Advances in Neural Information Processing Systems*, 35:22409–22421, 2022.
- [204] Yufei Xue, Wentao Dong, Minghuan Liu[^], Weinan Zhang, and Jiangmiao Pang. A unified and general humanoid whole-body controller for versatile locomotion. In *Robotics: Science and Systems*, 2025.
- [205] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.
- [206] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2021.
- [207] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.
- [208] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021.
- [209] Tao Yu, Cuiling Lan, Wenjun Zeng, Mingxiao Feng, Zhizheng Zhang, and Zhibo Chen. Playvirtual: Augmenting cycle-consistent virtual trajectories for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5276–5289, 2021.
- [210] Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. Mask-based latent reconstruction for reinforcement learning. *Advances in Neural Information Processing Systems*, 35:25117–25131, 2022.

-
- [211] Xingrui Yu, Yueming Lyu, and Ivor Tsang. Intrinsic reward driven imitation learning via generative model. In *Proceedings of the International Conference on Machine Learning*, pages 10925–10935, 2020.
- [212] Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Wenjun Zeng, and Glen Berseth. Rlexplore: Accelerating research in intrinsically-motivated reinforcement learning. *Transactions on Machine Learning Research*, 2025.
- [213] Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Rewarding episodic visitation discrepancy for exploration in reinforcement learning. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.
- [214] Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Automatic intrinsic reward shaping for exploration in deep reinforcement learning. In *International Conference on Machine Learning*, pages 40531–40554. PMLR, 2023.
- [215] Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Adaptive data exploitation in deep reinforcement learning. *arXiv preprint arXiv:2501.12620*, 2025.
- [216] Mingqi Yuan, Man-On Pun, and Dong Wang. Rényi state entropy maximization for exploration acceleration in reinforcement learning. *IEEE Transactions on Artificial Intelligence*, 2022.
- [217] Mingqi Yuan, Tao Yu, Wenqi Ge, Xiuyong Yao, Huijiang Wang, Jiayu Chen, Xin Jin, Bo Li, Hua Chen, et al. Behavior foundation model: Towards next-generation whole-body control system of humanoid robots. *arXiv preprint arXiv:2506.20487*, 2025.
- [218] Mingqi Yuan, Zequn Zhang, Yang Xu, Shihao Luo, Bo Li, Xin Jin, and Wenjun Zeng. Rllte: Long-term evolution project of reinforcement learning. *arXiv preprint arXiv:2309.16382*, 2023.

- [219] Mingqi Yuan, Zequn Zhang, Yang Xu, Shihao Luo, Bo Li, Xin Jin, and Wenjun Zeng. Rllte: Long-term evolution project of reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.
- [220] Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *Advances in neural information processing systems*, 33:20913–20924, 2020.
- [221] Yanjie Ze, Zixuan Chen, Joao Pedro Araujo, Zi ang Cao, Xue Bin Peng, Jiajun Wu, and Karen Liu. TWIST: Teleoperated whole-body imitation system. In *9th Annual Conference on Robot Learning*, 2025.
- [222] Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, 2021.
- [223] Junzi Zhang, Jongho Kim, Brendan O’Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10887–10895, 2021.
- [224] Zhikai Zhang, Jun Guo, Chao Chen, Jilong Wang, Chenghuai Lin, Yunrui Lian, Han Xue, Zhenrong Wang, Maoqi Liu, Huaping Liu, et al. Track any motions under any disturbances. *arXiv preprint arXiv:2509.13833*, 2025.
- [225] Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding. In *International Conference on Learning Representations*, 2024.
- [226] Han Zheng, Yi Cheng, Hang Liu, Jiayi Li, Yizhe Li, Linqi Ye, and Houde Liu. Transformer-based world interaction modeling for humanoid locomotion

- control. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 2766–2773. IEEE, 2025.
- [227] Qinqing Zheng, Mikael Henaff, Amy Zhang, Aditya Grover, and Brandon Amos. Online intrinsic rewards for decision making agents from large language model feedback. In *Reinforcement Learning Conference*, 2025.
- [228] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, 31, 2018.