零样本强化学习的改进算法框架

An Enhanced Framework for Zero-Shot Reinforcement Learning

(申请清华大学工学硕士学位论文)

培 养 单 位： 计算机科学与技术系

学　　科： 计算机科学与技术

研　究　生： 叶 洛 涟

指 导 教 师： 张 亚 勤

副指导教师： 詹 仙 园

二〇二五年五月

An Enhanced Framework for Zero-Shot Reinforcement Learning

Thesis submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Master of Science

in

Computer Science and Technology

by

Lauriane Teyssier

Thesis Supervisor: Ya-Qin Zhang

Associate Supervisor: Xianyuan Zhan

May, 2025

学位论文指导小组、公开评阅人和答辩委员会名单

指导小组名单

张亚勤	教授	清华大学

公开评阅人名单

刘云新	教授	清华大学
詹仙园	副研究员	清华大学

答辩委员会名单

主席	段海新	教授	清华大学
委员	刘永进	教授	清华大学
	张勇	副研究员	清华大学
	王东	副研究员	清华大学
	詹仙园	副研究员	清华大学
秘书	蔡康辉		清华大学

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）按照上级教育主管部门督导、抽查等要求，报送相应的学位论文。

本人保证遵守上述规定。

作者签名：叶洛滨　　导师签名：

日　　期：2025.05.20　　日　　期：2025.05.20

摘 要

零样本学习（Zero-shot Learning）在语言建模和计算机视觉等领域取得了显著的成功，通过利用大规模数据集实现了可扩展的学习。然而，由于强化学习（Reinforcement Learning）基于奖励驱动的范式，其在泛化能力方面面临诸多挑战。尽管已有一些针对零样本强化学习的研究提出，但它们在性能和泛化能力方面的表现仍不尽如人意。为此，我们提出了一种新颖的零样本强化学习框架——FB-Performance Enhanced Regularized Learning (FB-PERL)，将现有的 FB 学习框架扩展至更高效、更稳健的场景。

我们的主要贡献包括：提出了一种新型的神经网络框架，将 FB 学习框架扩展到更具表达力和鲁棒性的环境；提出了一种正则化方法，以避免分布外值的查询，确保训练过程保持在真实数据范围内。

通过这些改进，我们的实验表明，在 ExoRL 基准测试中，FB-PERL 的性能可超过当前最先进方法。代码已在以下地址开源：https://github.com/arwen-c/FB_generalization

关键词：零样本强化学习；无奖励训练；神经网络鲁棒性；分布外正则化

ABSTRACT

Zero-shot learning has achieved significant success in domains like language modeling and computer vision by leveraging large datasets for scalable learning. However, reinforcement learning faces challenges with generalization due to its reward-driven paradigm. Some approaches have been proposed for zero-shot reinforcement learning, without achieving great performance or generalization. We propose Forward Backward Representations-Performance Enhanced Regularized Learning (FB-PERL), a novel framework for zero-shot reinforcement learning, extending the FB learning framework to a more performant and robust setting. Our main contributions are twofolds: a novel neural network framework that extends the FB learning framework to a more expressive and robust setting, and a regularization method to avoid querying out-of-distribution values, ensuring training remains grounded in realistic data. Through these improvements, we show that our method can surpass previous methods by 9% to 34% IQM performance depending on the tasks on the ExoRL benchmark for the RND dataset. Code is made available at https://github.com/arwen-c/FB_generalization

Keywords: zero-shot reinforcement learning; reward-free training; neural network robustness; out-of-distribution regularization

TABLE OF CONTENTS

摘　要.....I
ABSTRACT II
TABLE OF CONTENTS III
LIST OF FIGURES.....VI
LIST OF TABLES IX
LIST OF SYMBOLS AND ACRONYMS X
CHAPTER 1 INTRODUCTION..... 1
1.1 Motivation..... 1
1.2 Reinforcement Learning..... 2
1.3 Zero-Shot RL..... 6
1.4 Contribution of this Work 7
CHAPTER 2 BACKGROUND..... 9
2.1 Overview..... 9
2.2 Reinforcement Learning Generalization.....10
2.2.1 Reinforcement Learning Framework 10
2.2.2 Different Approaches for Generalization in RL 13
2.3 Zero-Shot RL Methods 17
2.3.1 Successor Representation 17
2.3.2 Forward-Backward Representation 21
2.3.3 Alternative Approaches to Zero-Shot RL 29
2.4 Summary.....30
CHAPTER 3 ENHANCING THE EXPRESSIVENESS OF FB REPRESENTATIONS33
3.1 Representation Learning in Zero-Shot Reinforcement Learning.....33
3.1.1 Why Learn M Instead of Q or V?.....33
3.1.2 Fundamental Tensions in Representation Learning.....34
3.2 Scaling F and B Representations 34
3.2.1 Scaling Strategies in Modern RL Architectures.....35
3.2.2 Transformers as Most Scalable Neural Networks.....37
3.2.3 B Learner Design.....38
3.2.4 F Learner Design 40

3.3 Summary....42
CHAPTER 4 OOD REGULARIZATION FOR FB REPRESENTATIONS....43
4.1 Out-Of-Distribution Regularization in Offline RL....43
4.1.1 Out-of-Distribution Challenge in FB Learning....43
4.1.2 Existing General Regularization Methods for OOD Issues....44
4.1.3 FB OOD Regularization in the Literature....48
4.2 Our Proposal: In-Sample FB OOD Regularization....49
4.2.1 Objective....49
4.2.2 Expectile Regression: Definition....49
4.2.3 Learning Function....50
4.2.4 Overall Learning Algorithm....50
4.2.5 Additional B Regularization....51
4.3 Summary....51
CHAPTER 5 COMPLETE DESIGN: FB-PERL FRAMEWORK....53
5.1 Neural Network Architecture....53
5.1.1 Architecture Evolution....53
5.1.2 Implementation Choices....54
5.1.3 Neural Network Hyperparameters....55
5.2 Final Algorithm....56
5.2.1 Summary of the Changes....56
5.2.2 Our Proposal....56
5.2.3 Implementation Design Choices....59
5.3 Implementation Choices and Computing Performance Improvement....60
5.3.1 Score Improvement and Design Choices....60
5.3.2 Computing Speed Improvements....61
5.4 Hyperparameter Summary....62
CHAPTER 6 NUMERICAL EXPERIMENTS....64
6.1 Experiments Objectives....64
6.2 Experiments Methodology....65
6.2.1 Tools and Libraries....65
6.2.2 Datasets....65
6.2.3 Baselines....66
6.2.4 Experimental Details....66

6.2.5 Evaluation Protocol 67
6.2.6 Performance Metrics 67
6.3 Performance Evaluation 68
6.4 Ablation Studies 75
6.5 Conservatism Comparison 78
6.6 Detailed Results: Averaged Training Curves 80
6.7 Summary 84
CHAPTER 7 CONCLUSION AND PERSPECTIVES 86
7.1 Summary of Contributions 86
7.2 Theoretical Implications 87
7.3 Future Directions 87
7.4 Challenges in Real-World Deployment 88
7.5 Ethics 89
7.5.1 The Importance of an Ethics Section 89
7.5.2 General Ethical Considerations in AI 89
7.5.3 Ethical Considerations Specific to This Research 90
7.5.4 Personal Considerations 91
7.6 Final Thoughts 92
REFERENCES 94
APPENDIX A FAILED DESIGN — OPERATOR AND AUTO-REGRESSIVE B STRUCTURE 106
APPENDIX B DETAILED RESULTS AND TRAINING CRUVES 114
APPENDIX C IMPLEMENTATION DETAILS 135
ACKNOWLEDGEMENTS 144
声　明 145
RESUME 146
COMMENTS FROM THESIS SUPERVISOR 147
RESOLUTION OF THESIS DEFENSE COMMITTEE 148

LIST OF FIGURES

Figure 1.1 Schematic description of reinforcement learning: given a state s, the agent selects an action a according to its policy π and receives a reward r from the environment together with the next state s'. 2
Figure 1.2 Small grid example of problem-solving with reinforcement learning. 4
Figure 1.3 Small grid example of reinforcement learning value computation for problem-solving..... 4
Figure 1.4 Example of zero-shot transfer across manipulation. Green arrows indicate new tasks while red arrows represent training tasks[20]. 6
Figure 2.1 QMP workflow[51] 14
Figure 2.2 Evolution of ZSRL methods (2017-2025): From manually engineered features in successor features to our generalized FB framework. 17
Figure 2.3 RaMP[20]: Depiction of our proposed method for transferring behavior across tasks by leveraging model-free learning of random features. At training time, Q-basis functions are trained on accumulated random features. At test time, adaptation is performed by solving linear regression and recombining basis functions, followed by online planning with MPC 20
Figure 2.4 The transfer setting for generalized occupancy models. Given an unlabeled offline dataset, we learn a generalized occupancy model that models both "what can happen?" $p(\psi|s)$ and "how can we achieve a particular outcome?" $p(a|s,\psi)$. This is used for quick adaptation to new downstream tasks without test-time policy optimization.[72] 20
Figure 2.5 Function encoders workflow[65] 21
Figure 2.6 RL Zero framework: A video trajectory is synthesized from a text prompt, projected into the agent's observation space, and subsequently aligned with real agent observations. Observation-only imitation learning then grounds the generated trajectory into a policy that mimics the demonstrated behavior.[87] 27

Figure 2.7 FB-CPR architecture: Critic network (left) evaluates state-action pairs while discriminator (right) aligns latent policy distributions with demonstration data. Forward (F) and backward (B) networks form the FB representation core[83] 28

Figure 2.8 *left* Training of a mapping $\phi : S \to Z$ that maps temporally similar states to spatially similar latent states ($d*$ denotes the temporal distance). *right* Training of a latent-conditioned policy$\pi(a|s, z)$[17] 29

Figure 2.9 FRE architecture[64] 30

Figure 3.1 BroNet architecture[93]. Uses N=2 for actor and critic 35

Figure 3.2 Offline Q-learning network architecture[99] 36

Figure 3.3 Proposed Attention architecture for scaling offline actor-critic[101] 37

Figure 3.4 Backward learner architecture 40

Figure 3.5 Transformer block architecture 40

Figure 3.6 Forward learner 2nd architecture showing dual processing branches, with updated MLP architectures 41

Figure 3.7 Forward learner architecture, with shared Transformer Block structure with B, as per Figure 3.5 42

Figure 4.1 (Left) Offline RL methods must train on a dataset collected by a behavior policy optimizing against task $z_{collect}$, yet generalize to new tasks z_{eval}. Both tasks have associated optimal value functions $Q^*_{z_{collect}}$ and $Q^*_{z_{eval}}$. (Middle) Unregularized methods overestimate the value of actions not in the dataset. (Right) CQL regularization constrains the value of actions not in the dataset. Black dots represent state-action samples present in the dataset. Illustration from Jeen and al.[79] 46

Figure 5.1 Original FB neural network architecture 54

Figure 5.2 FB-PERL architecture 55

Figure 6.1 ExoRL evaluation environments: Walker, quadruped, point mass maze. In the Maze domain (right), we show an example of an initial state (yellow point) and the 20 test goals (red circles). Images from Touati and al.[16] 65

Figure 6.2 Score distribution interpretation, from Agarwal and al.[152] For a non-negative random variable X, IQM corresponds to the red-shaded region while an optimality gap corresponds to the orange shaded region 68

Figure 6.3 Maximum IQM performance achieved by the algorithms, trained on RND dataset in the walker environment on the three robots walker, quadruped, point mass maze after 10M steps. 69
Figure 6.4 Performance profile on the small dataset 73
Figure 6.5 Performance profile on the full dataset 74
Figure 6.6 Ablation studies on the large dataset (10M transitions) for the walker environment 75
Figure 6.7 Ablation studies averaged over walker, quadruped, and point mass maze (10M transitions)..... 76
Figure 6.8 Ablation studies on the small dataset (100k transitions) for the walker environment 77
Figure 6.9 Average performance of different models throughout training 78
Figure 7.1 Equivalent CO2 emissions to running thesis experiments 91

LIST OF TABLES

Table 2.1 Comparison of RL generalization approaches 16
Table 2.2 Comparison of Zero-Shot RL method characteristics..... 29
Table 4.1 Summary of Offline RL Regularization Approaches..... 45
Table 5.1 Speed Improvements (in multiplication factors) brought by the different optimizations 62
Table 5.2 Hyperparameter comparison (changes in **bold**) 63
Table 6.1 Domains and evaluation tasks 65
Table 6.2 Baselines 66
Table 6.3 Comparison of our FB averaged final performance over 5 seeds, with standard deviation..... 71
Table 6.4 Comparison of FB, VC-FB, MC-FB, and FB-PERL averaged across DIAYN, RND, and RANDOM big datasets..... 72
Table 6.5 Comparison of FB, VC-FB, MC-FB, and FB-PERL averaged across DIAYN, RND, and RANDOM small datasets 73

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
A2PR	Adaptive Advantage Guided Policy Regularization
AWAC	Advantage Weighted Actor Critic
AWR	Advantage Weighted Regression
BFM	Behavioral Foundation Model
CQL	Conservative Q-Learning
DP	Dynamic Programming
DRL	Deep Reinforcement Learning
DSM	Distributional Successor Measure
EQL	Exponential Q-Learning
FB	Forward-Backward
FRE	Functional Reward Encoding
GDT	Generalized Decision Transformers
GCRL	Goal-Conditioned Reinforcement Learning
GOM	Generalized Occupancy Model
HILPS	Hilbert Foundation Policy framework
HPO	Hyperparameter Optimization
IL	Imitation Learning
IQL	Implicit Q-Learning
IQM	Interquartile Mean
LLM	Language Learning Model
MCFB	Measure Conservative Forward-Backward
MDP	Markov Decision Process
MLP	Multi-Layer Perceptron
MMD	Maximum Mean Discrepancy
MoE	Mixture of Experts
NLP	Natural Language Processing
NN	Neural Network
OOD	Out-Of-Distribution
PSM	Proto Successor Measure
RaMP	Random Features for Model-Free Planning
RL	Reinforcement Learning

SF	Successor Feature
SOTA	State-Of-The-Art
SQL	Sparse Q-Learning
SR	Successor Representation
TD	Temporal Difference learning algorithm
USF	Universal Successor Feature
USFA	Universal Successor Features Approximators
UVFA	Universal Value Function Approximator
VCFB	Value Conservative Forward-Backward
ZSRL	Zero-Shot Reinforcement Learning

CHAPTER 1 INTRODUCTION

1.1 Motivation

Reinforcement Learning (RL) is a branch of artificial intelligence (AI) that has enabled systems to achieve superhuman performance in complex domains such as games and robotics. Unlike other machine learning approaches that require extensive labeled datasets, RL's strength lies in its ability to **learn from experience** in real time and adapt to unpredictable environments.

From AlphaGo's mastery of the ancient Chinese game of Go to robotic arms performing intricate manipulation tasks, RL has demonstrated its ability to solve narrowly defined problems with remarkable precision. However, RL struggles to generalize learned behaviors to new tasks without retraining. For instance, while AlphaGo[1-2] excels at Go, it cannot directly be used for playing chess. This task-specific nature limits RL's applicability in dynamic, real-world scenarios where adaptability is essential. Consider a home assistant robot capable of performing diverse household chores—washing dishes, folding laundry, cooking meals, and cleaning floors. Traditional RL approaches require separate training for each task, which makes them costly and inefficient.

The paradigm of zero-shot reinforcement learning (ZSRL) enables agents to generalize across tasks without retraining, addressing RL's limitations. **Zero-Shot RL** represents a paradigm shift: by leveraging fixed datasets of manipulations during the learning phase, it develops representations of the environment that can be leveraged to instantly adapt to new tasks. This approach could allow RL to catch up with fields like computer vision and natural language processing that have successfully adapted to a wide range of tasks in a zero-shot manner taking advantage of large, unlabeled datasets ([3-5]). By eliminating the need for task-specific training, ZSRL has the potential to transform AI applications and create systems capable of operating effectively in diverse and dynamic environments. Beyond robotics, RL's decision-making capabilities have demonstrated significant real-world impact across diverse sectors. In healthcare, RL algorithms can optimize treatment plans and drug protocols to improve patient outcomes[6-7]. In environmental applications, RL has already proven its value, for example, in optimizing cooling in buildings[8] or in

data centers[9-10].

Despite the significant advancements and potential of reinforcement learning, the challenges of generalization and scalability remain critical barriers to its broader applicability. Addressing these challenges requires a deeper understanding of how agents can efficiently learn representations that enable adaptability across diverse tasks. This leads to the central research questions of this thesis: How can we design agents that are both efficient and capable of generalizing across tasks without retraining? What representations and training methodologies are most effective for enabling zero-shot policy generation?

By exploring these questions, this work aims to contribute to the development of more versatile and scalable AI systems for decision-making processes, for ultimately reducing global environmental impact and enhancing quality of life.

1.2 Reinforcement Learning

Reinforcement Learning has its roots in the study of how humans and animals learn through **trial and error**. This natural learning process involves interacting with the environment, taking actions, and adjusting them based on feedback. Inspired by this, RL has been formalized as a computational framework in which an **agent** learns to make decisions by interacting with an **environment** (cf. Figure 1.1). The agent's goal is to maximize cumulative rewards over time by learning an optimal **policy**, i.e., an optimal mapping from states to actions.

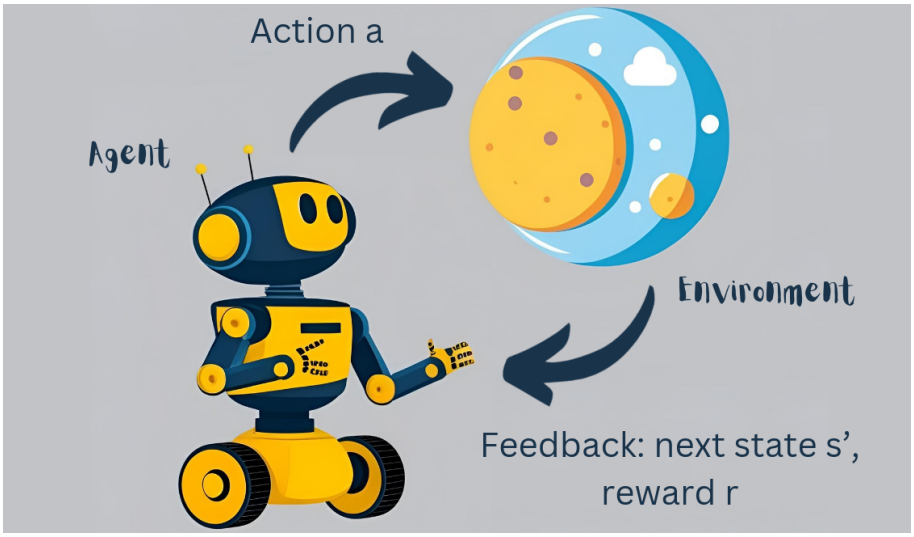


Figure 1.1 Schematic description of reinforcement learning: given a state s, the agent selects an action a according to its policy π and receives a reward r from the environment together with the next state s'.

Key Components of RL. The RL framework uses its own terminology to describe the key components of the learning process. At its core is the **agent**, which functions as the decision-maker that learns from its interactions with the world around it. This world the agent operates in is called **environment**. To guide the agent's learning process, a **reward signal** provides essential feedback indicating the quality of the agent's chosen actions, helping it distinguish beneficial decisions from harmful ones. The agent develops a **policy**, which represents a strategy that defines how it selects actions based on its perception of the current state. Complementing this decision-making strategy is the **value function**, which serves as a measure of the expected long-term reward from a given state or state-action pair, allowing the agent to evaluate different possible futures. Together, these components create a framework that enables autonomous systems to learn optimal behaviors through reinforcement.

With this formalization, RL enables agents to autonomously learn complex behaviors without explicit supervision. More generally said, RL dedicates to solve any sequential decision-making problem, where an agent is interacting with an unknown environment to find an optimal policy.

A naive approach for solving RL problems is to use a tabular representation of the value function. This means going through all the possible states and actions, and computing the value of each state-action pair. In the small grid example figure 1.2, where the goal is to reach the diamond using the shortest path, a human can intuitively draw the arrows representing the optimal path (optimal *policy*). However, humans would struggle with much bigger grids, and need more complex reasoning for tackling infinite size environments, so the necessity of algorithms. But how to teach an algorithm to find the optimal policy? In this small grid example, we could start by initializing the value of each state to 0, except the goal state having a reward of 1. From the goal state, we can look at all neighbour states and update their value (maximum reward that we can obtain by going to this state) to $\text{reward}_s = \gamma * \text{reward of the best neighboor})$, with γ the penalty for taking more time reaching the goal, then compute the value of their neighbors, and so on until full grid completion, as per Figure 1.3. Once all values are learned, the agent policy would be to go from any given start state to the next state choosing the state with the highest value.

However, this approach is not scalable to high-dimensional or continuous state spaces that cannot be fully explored. To address this, deep reinforcement learning (DRL) has emerged, combining deep learning with RL to learn complex policies from high dimen-

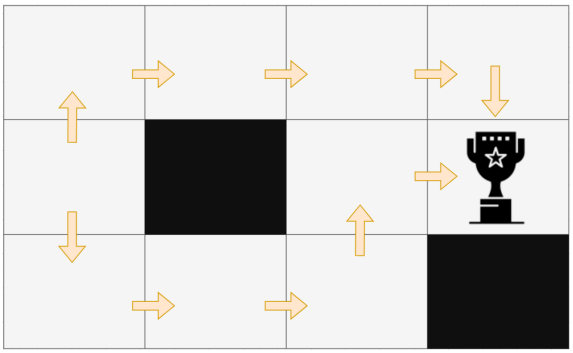


Figure 1.2 Small grid example of problem-solving with reinforcement learning.

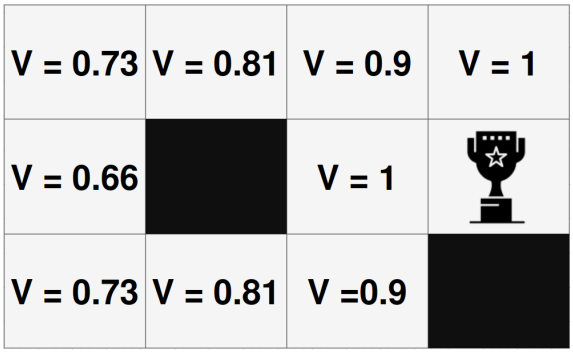


Figure 1.3 Small grid example of reinforcement learning value computation for problem-solving

sional inputs.

Recent Achievements in RL. RL has revolutionized problem-solving across diverse fields in recent years, achieving groundbreaking milestones through its adaptive learning capabilities. In **gaming**, RL algorithms have surpassed human expertise, exemplified by systems like AlphaGo[1-2] and AlphaZero[11], which mastered complex strategies in games such as Go, chess, or by DQN[12] on Atari. Beyond games, these techniques have also advanced **character animation**, enabling more lifelike virtual interactions, as demonstrated by Peng and al.[13] **Robotics** has similarly benefited from RL algorithms. They now empower machines to perform intricate tasks like dynamic object manipulation, adaptive locomotion, and navigation in unpredictable environments[14]. Meanwhile, **healthcare** has seen transformative applications, from optimizing personalized treatment plans to speeding up drug discovery by deriving effective strategies from sparse datasets. **Autonomous systems**, too, have been reshaped by RL, particularly in training self-driving vehicles[15] to navigate safely and make split-second decisions in chaotic traffic scenarios. Together, these achievements underscore RL's versatility in addressing real-world challenges through continuous interaction and reward-driven learning. The significance of reinforcement learning and its achievements has been further underscored by the pres-

tigious 2025 ACM A.M. Turing Award bestowed upon two of its pioneers, Andrew G. Barto and Richard S. Sutton, for their foundational contributions to the field.

These achievements highlight the potential of RL to tackle problems that require sequential decision-making in complex environments.

Challenges in RL. Despite its successes, traditional RL faces several challenges that limit its applicability. One major issue is **task specificity**, as RL agents are typically trained for a particular task and cannot generalize to other environments or objectives. Additionally, **data inefficiency** is a significant concern, as training often requires extensive interaction with the environment, which can be costly or impractical in real-world scenarios. **Computational complexity** further exacerbates these limitations, as many RL algorithms are resource-intensive and difficult to scale to high-dimensional or continuous environments. Finally, the **exploration-exploitation** trade-off remains a fundamental challenge, requiring a careful balance between exploring new strategies and exploiting known successful ones.

These limitations have motivated the development of more efficient and generalizable approaches. Recent advances in offline reinforcement learning have shown promise in addressing data efficiency by learning from pre-collected datasets. Additionally, the integration with large language models has opened new possibilities for improving task generalization and instruction following.

Various alternative approaches have been explored to enhance the capabilities of RL agents, including multi-task learning and transfer learning. However, these methods often encounter limitations in terms of scalability and the breadth of behaviors they can acquire[16-17]. Similarly, fine-tuning has struggled to yield significant improvements in agent capabilities, further highlighting the challenges in advancing RL methodologies[18].

However, a more fundamental shift may be needed to truly overcome these challenges. This has given rise to zero-shot reinforcement learning, a framework designed to allow agents to train without being assigned a task, and adapt to any task subsequently. The following section introduces this promising direction and its potential to transform the field of reinforcement learning.

1.3 Zero-Shot RL

Definition. Zero-shot Reinforcement Learning (ZSRL) consists in uncoupling the learning of a policy from its reward function[19]. The agent learns to capture diverse, optimal behaviors from unlabeled offline data during training, storing them in a representation that enables direct policy generation for any reward function at test time, without additional training or fine-tuning[17].

The framework operates in two distinct phases: A **Training Phase**, first, where agent learns to capture diverse, optimal behaviors from unlabeled offline data. It is then followed by a **Deployment Phase**, where the agent leverages learned representations to instantly generate policies for new tasks.

For example, a robot trained on diverse movements without specific goals can immediately generate a policy for stacking blocks when presented with this new task, without retraining. More examples can be seen in Figure 1.4.

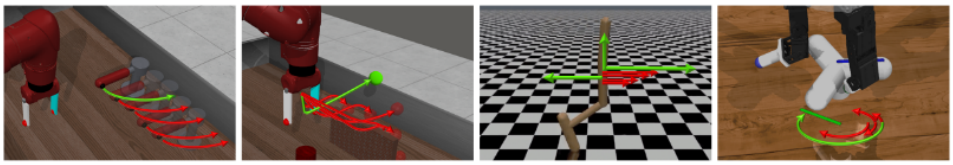


Figure 1.4 Example of zero-shot transfer across manipulation. Green arrows indicate new tasks while red arrows represent training tasks[20].

This approach has been enabled by advances in representation learning techniques such as successor features and Forward-Backward representations[16].

Specific Challenges for Zero-Shot RL. While promising approaches in ZSRL exist, their real-world applicability remains limited due to several fundamental challenges. One critical issue is **expressivity**, which refers to the model's performance. The goal is for the model to match or exceed the performance of current top models that are specifically trained for individual tasks, across multiple tasks. Current methods often struggle with **behavioral diversity**, particularly when trained on homogeneous datasets. **Scalability** is another concern. Unlike large language models, scaling ZSRL to high-dimensional environments remains computationally prohibitive, requiring methods to balance model complexity with generalization capabilities.

While trying to improve these three axes, one must keep in mind the compromise between performance, computational efficiency, and theoretical guarantees. By **compu-**

tational efficiency, we mean optimizing both computational resources and sample efficiency while maintaining the ability to learn diverse behaviors. Providing **performance guarantees**, including mathematical justification and model understanding, is crucial for deployment in real-world applications, particularly in safety-critical domains.

Offline Training Challenges. Unlike traditional RL where agents learn through environment interaction (online training), ZSRL uses offline training using fixed datasets.

The offline training approach offers key advantages. It eliminates the need for costly environment interactions during training, enabling **training efficiency**. Moreover, it enables learning from large-scale datasets spanning diverse environments, enhancing **scalability**.

However, this paradigm introduces several critical challenges due to the **limited amount of available data transitions** (state, action). First, **out-of-distribution (OOD) issues** arise when the new agent tries exploring a state or action that are not covered by the dataset. Because lacking reliable information about potential outcomes, it leads to unpredictable results. This issue is associated with **distributional shifts** that occur as the learned policy distribution frequently deviates from the training data distribution, resulting in poor generalization, particularly in regions with limited data coverage. Finally, the risk of **overfitting** remains significant, as models may become overly specialized to patterns within the offline dataset.

These challenges require the development of specialized training methods. While approaches such as conservative Q-learning[21] (and its derivative) or Dual RL approaches[22] have shown promise, work remains in developing further solutions for offline training and adapting these methods to the zero-shot paradigm.

1.4 Contribution of this Work

This thesis makes the following key contributions to the field of Zero-Shot Reinforcement Learning (ZSRL): We propose **FB-Performance Enhanced and Regularized Learning (FB-PERL)**, a novel framework that enhances the scalability and robustness of ZSRL by extending existing representation learning methods to more complex environments. We develop a specialized regularization approach for offline RL, addressing OOD issues. We demonstrate the effectiveness of our methods through extensive experiments on the ExoRL benchmark, and show significant performance gains over state-of-the-art

approaches.

To present our contributions and findings in a clear and systematic manner, this thesis is organized into six further chapters:

- **Background (chapter 2)**: Provides a comprehensive overview of reinforcement learning fundamentals and current state-of-the-art methods, establishing the theoretical foundation for our contributions.
- **ZSRL Generalization (chapter 3)**: Details our proposed enhancements to the existing ZSRL framework, focusing on improving scalability and performance across diverse tasks.
- **Offline RL Regularization Methodology (chapter 4)**: Introduces our novel regularization framework designed for ZSRL, addressing key challenges in offline learning and policy generalization.
- **FB-PERL framework (chapter 5)**: Presents our integrated FB-PERL framework, combining scalable representation learning with offline RL regularization to enhance ZSRL capabilities.
- **Numerical Experiments (chapter 6)**: Presents empirical evaluations of our proposed methods, including comprehensive analyses of their performance.
- **Conclusion (chapter 7)**: Summarizes our key findings and contributions while discussing promising directions for future research in ZSRL.

These contributions address key challenges in expressivity, scalability, and robustness, advancing ZSRL towards practical real-world applications.

CHAPTER 2 BACKGROUND

2.1 Overview

The core challenge of autonomous decision-making systems is achieving *zero-shot generalization*—the ability to solve diverse tasks without additional training. This requires enabling the system to handle arbitrary downstream tasks by leveraging pretrained environmental understanding without additional environment interaction. To achieve such generalization, we adopt a paradigm that involves training on an offline dataset of transitions and subsequently inferring optimal policies for downstream tasks based on task-specific rewards.

This chapter establishes the theoretical foundations for zero-shot generalization by pursuing three interconnected aims. First, it formalizes the concept through a rigorous mathematical framework that delineates its boundaries from related paradigms like transfer learning and meta-reinforcement learning. Secondly, it provides a methodological historiography of zero-shot reinforcement learning (ZSRL), tracing the evolution from early techniques such as successor features to contemporary representations. For each technique, it critically analyzes their underlying theoretical premises and practical performance compromises. Finally, the analysis crystallizes specific limitations in current approaches – particularly regarding expressivity and environmental assumptions – that create both the necessity and opportunity for the generalized framework developed in later chapters.

This chapter is structured around four key axes:

- **Foundations of Generalization:** Explore fundamental RL concepts and the challenges of generalization, contrasting traditional task-specific training with reward-agnostic approaches.
- **RL Generalization Contextualization:** Review the different approaches to solve markovian decision-making problems, including imitation learning and transfer methods, while highlighting the specificities of zero-shot RL.
- **Zero-Shot Method Taxonomy:** Examine leading ZSRL methods through their mathematical formulations and practical implementations.
- **Framework Selection Justification:** Justify the adoption of the Forward-Backward (FB) representation paradigm as the foundation for our approach.

This chapter emphasizes why existing methods are inherently limited in addressing real-world complexity—limitations that our proposed architectural innovations aim to overcome.

2.2 Reinforcement Learning Generalization

2.2.1 Reinforcement Learning Framework

We formalize reinforcement learning as a Markov Decision Process (MDPs)[23], building on the foundational work of Bellman[24-26]. The "Markov" term refers to the Markov property, which states that the future state of a process depends only on its current state and action, not on its past. Following the usual notation[27], we define an MDP as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ with:

- $\mathcal{S}$: Measurable state space (complete separable metric space—can be discrete or continuous)
- $\mathcal{A}$: Action space
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$: Markovian transition dynamics.
- $r : \mathcal{S} \to \mathbb{R}$: Bounded reward function, describing the task to achieve.
- $\gamma \in [0, 1)$: Temporal discount factor, balancing immediate versus future rewards.

In this work, we focus on the general scenario of infinite-horizon environments. This encompasses finite-horizon environments as well, as they can be transformed into infinite-horizon settings by introducing an augmented state space that includes a terminal state. This terminal state loops back onto itself indefinitely with zero rewards, effectively capturing the finite-horizon behavior within an infinite-horizon framework.

Reinforcement Learning Formal Definition. The agent interacts with $\mathcal{M}$ through a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ mapping states to action distributions. For the initial state s_0, the policy generates trajectories $(s_t, a_t)_{t\geq 0}$ with $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. The optimization objective is the expected sum of rewards over time (averaged or not).
The average expected reward over time, also called discounted return, is defined as:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right] \tag{2.1}$$

Two operators are fundamental to RL:

- **State-action value function**[27] (also called Q-function or expected cumulative re-

ward) :

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \Big| s_0 = s, a_0 = a, \pi\right] \tag{2.2}$$

Q represents the average return starting from a state-action pair (s, a) and following the policy π thereafter.

- **Successor measure**[19] (stochastic dynamics encoding):

$$M^{\pi}(s,a,X) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[(s_t,a_t) \in X | s_0 = s, a_0 = a, \pi] \tag{2.3}$$

The successor measure captures the long-term dynamics of the system under policy π. Some works, such as Wiltzer and al.[28], use a normalized version of the successor measure:

$$M^{\pi}(s_0,a_0,X) = \sum_{t\geq 0} (1-\gamma)\gamma^t \mathbb{P}((s_t,a_t) \in X \mid s_0,a_0,\pi) \tag{2.4}$$

This normalized measure describes the proportion of time spent in the state-action pairs and forms a probability distribution over X.

These functions satisfy the duality relationship:

$$Q^{\pi}(s,a) = \sum_{(s',a')\in\mathcal{S}\times\mathcal{A}} r(s',a') M^{\pi}(s,a,s',a') \tag{2.5}$$

We additionally define the **value function**[27] (V-function):

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \Big| s_0 = s, \pi\right] = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi}(s,a)] \tag{2.6}$$

These functions also have continuous formulations involving integrals over state-action spaces. While these definitions are written in the tabular setting, equivalent function approximator variants (e.g., with neural networks) can be instantiated to model Q or V with stochastic optimization techniques.

Learning Paradigms. Computing the optimal policy π^* can be done through different optimization methods, often relying on the Bellman optimality equation[24]:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}\left[\max_{a'} Q^*(s',a')\right] \tag{2.7}$$

This equation is the direct consequence of the Q-function definition. It states that the expected cumulative reward for a transition (s, a) is equal to the sum of the reward obtained if (s, a) is the first transition of a trajectory, and the rewards obtained if getting to this transition from a previous state. As an example, temporal-difference learning (TD)

methods[29], such as Q-learning[30] or SARSA (State–Action–Reward–State–Action, first introduced under the name "Modified Connectionist Q-Learning" (MCQ-L)[31]), use the Bellman equation to iteratively update the Q-function based on observed transitions:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{2.8}$$

On-Policy Versus Off-Policy Learning. Notable algorithms in reinforcement learning are categorized into **on-policy** and **off-policy** approaches. **On-policy** algorithms, such as SARSA[31], REINFORCE[32] or A2C[33], require direct environment interactions to update the policy based on the current data. In contrast, **off-policy** algorithms like Q-learning[30] (and its more modern derivatives like DQN[12]), DDPG[34], or SAC[35] learn from experiences gathered with a different policy (the behavior policy) from the one they will ultimately deploy (the target policy).

Offline methods, a subset of off-policy methods, seek to learn from a fixed dataset of transition $\mathcal{D} = \{(s_i, a_i, s'_i)\}_{i \in \mathbb{N}}$ generated using one or several behavior policies. Because of our offline setting, we are restricted to using offline algorithms.

Model-Free Versus Model-Based RL. Reinforcement learning can be broadly categorized into two approaches: model-free and model-based methods.

Model-free RL algorithms learn policies or value functions directly from interactions with the environment, without explicitly modeling the environment's dynamics. These methods have shown great potential in solving single-task sequential decision-making problems, particularly in environments with high-dimensional observations and long horizons. However, model-free RL struggles to generalize across tasks, as it lacks a structured representation of the environment that can be reused for new tasks[20].

On the other hand, **model-based** RL explicitly learns a dynamics model of the environment, which captures the transition probabilities and reward structure[36-37]. This task-agnostic representation naturally enables transfer across different reward functions, as the learned model can be reused for planning in new tasks. However, model-based RL faces significant challenges in scaling to complex environments[20]. Specifically, autoregressive model rollouts suffer from compounding errors—small approximation errors in the dynamics model accumulate over long horizons, making it ineffective for long-horizon problems[20].

The idea behind zero-shot RL is to combine the strengths of both approaches: leverag-

ing the generalization capabilities of model-based methods while avoiding the scalability issues associated with autoregressive rollouts. By learning a compact and reusable representation of the environment, zero-shot RL aims to enable agents to solve any task without further training, even in complex environments.

2.2.2 Different Approaches for Generalization in RL

Alternatives to reinforcement learning for creating generalist agents are being explored by the research community. As an example, several approaches[38-41] leverage vision and language foundation models to predict actions based on large dataset. These methods rely heavily on statistical patterns in data, limiting their ability to handle tasks outside their training distribution. Therefore, they often lack the decision-making framework required for true generalization. Moreover, open research challenges remain, particularly regarding the scarcity of robot-relevant training data, the need for safety guarantees and uncertainty quantification, and the demands of real-time execution[42]. In contrast, reinforcement learning provides a principled framework for decision-making, enabling agents to learn and adapt through interaction. Among RL approaches, zero-shot RL (introduced in Section 1.3) stands out as the most promising for building generalist agents, as it decouples policy learning from task-specific rewards and eliminates the need for fine-tuning at test time.

This section examines various RL approaches to generalization, analyzing their strengths and limitations. Through it, we justify our focus on zero-shot RL for developing more scalable and flexible frameworks.

Meta-Learning. Meta-learning, or "learning to learn", aims to train agents that can quickly adapt to new tasks with minimal data[43]. More specifically, meta-reinforcement learning (Meta-RL) involves training on a distribution of tasks and learning a policy that can adapt to new tasks from the same distribution.

In meta-reinforcement learning, tasks are typically assumed to be drawn from a known distribution, which can limit its applicability when faced with entirely novel tasks. A core feature of Meta-RL is its ability to perform **adaptation** at test time, where the agent leverages a small amount of task-specific data or interaction to refine its policy. Different works have shown its potential[44-46]. However, achieving such rapid adaptation comes at the cost of **scalability**, as Meta-RL often requires extensive training across a diverse range of tasks, making the computational demands particularly high.

While meta-learning has shown promise in settings with well-defined task distributions, it often struggles with tasks that fall outside the training distribution. Additionally, the need for task-specific adaptation at test time makes it less suitable for truly zero-shot generalization[43].

Multi-Task Learning. Multi-task learning trains agents on multiple tasks simultaneously, enabling them to learn shared representations that can be transferred to new tasks[47]. This setting is particularly effective when tasks share common structures or dynamics, like demonstrated by multiple works[47-50].

By learning shared features across multiple tasks, multi-task learning enhances **sample efficiency**[51] and improves **generalization**[52], as the agent benefits from knowledge gained across different environments. However, a significant challenge is **task interference**[53], where the learning of one task may negatively impact the performance of another, particularly when tasks are dissimilar or conflicting. Additionally, multi-task learning poses **scalability**[47] concerns, as it demands significant training data and computational resources, with these requirements increasing with the number of tasks being learned.

Recent work demonstrated its potential in various domains. For example, Hendawy and al.[53] pretrain agents on large-scale video datasets and fine-tune them on robot data to predict both actions and videos. Similarly, Zhang and al.[51] identify common subtasks across a set of tasks, enabling the sharing of knowledge between them (Figure 2.1). To achieve this, their QMP[51] identify common behaviors by looking at learned behaviors having the highest returns on the new task, evaluated by the Q-switch, as shown on Figure 2.1. It then incorporates shareable behavior by using training data generated by the interesting policies to train the new policy. Ishfaq and al.[54] use the low rank MDP ap-

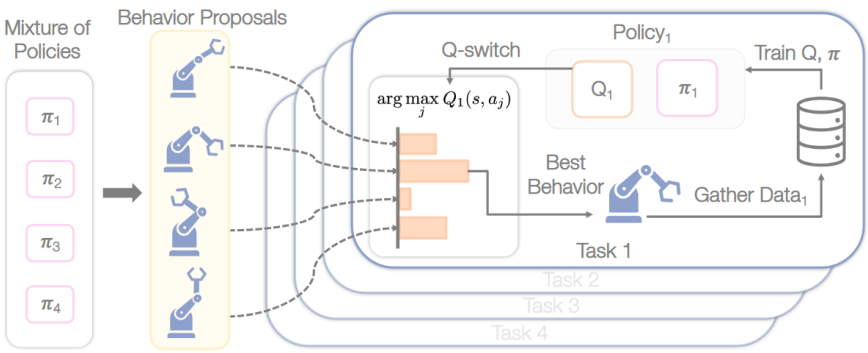


Figure 2.1 QMP workflow[51]

proximation to learn a multi-task representation. They also show theoretical benefits of learning common tasks representation. However, if the intuition that shared tasks' representation could benefit RL models, like shared multi-language representation benefits LLMs, the practical implementation on how to learn this latent space remains an open problem, especially as the number of tasks increases.

Despite recent multi-task advancements, existing methods require training data from each task and struggles with tasks that differ significantly from those seen during training. This limitation makes it less suitable for the "any task" generalization target of zero-shot RL.

Transfer Learning. Transfer learning leverages knowledge to improve performance on a target task[55]. Unlike multi-task learning, where tasks are trained simultaneously, transfer learning between tasks focuses on sequential learning: leveraging an agent already trained on a source task to facilitate training on the target task.

Transfer learning revolve around **knowledge transfer**[55]. In knowledge transfer, the agent leverages prior experience by transferring learned elements such as policies, value functions, or dynamics models from a source task to a target task. This process significantly reduces the need for extensive training on the target task, speeding up learning and improving efficiency[55].

A survey[55] presents methods to model long-term state occupancy, enabling efficient policy evaluation under new tasks. Transfer learning has been recently applied to many domains including cooling[56], vehicle energy management[57], autonomous-driving[58] or IoT[59]. You and al.[60] achieve cross-domain transfer learning by leveraging a shared latent space to align source and target domains. These methods provide valuable insights for zero-shot RL—some of which are described below as foundational to zero-shot RL methods—as their generalization approaches and infrastructure underpin many zero-shot techniques. However, a key distinction between transfer learning and zero-shot RL lies in their problem settings: in zero-shot RL, the agent is not trained on any task-specific reward function and does not allow any further training at test time. This makes zero-shot RL a more challenging but also more generalizable framework.

Goal-Conditioned Reinforcement Learning. Goal-conditioned reinforcement learning (GCRL) refers to a specialized subset of reinforcement learning frameworks designed to train agents capable of achieving diverse aims under specific scenarios[61]. Unlike con-

ventional RL methods, which rely solely on state or observation-based policies, GCRL introduces goal-dependent decision-making mechanisms, requiring the agent to condition its actions on designated goals[61].

Zeng et al.[62] explore the potential of sequence modeling — a method that learns a representation of the environment dynamics — as a way to condense trajectories into effective representations that improve policy learning. Their work aligns with FB representations in that they decouple learning an environment dynamic from learning a policy (cf. subsection 2.3.2). However, their approach goals diverge from our zero-shot learning setting. Instead of emphasizing zero-shot capabilities, their method strictly trains with Q-learning, distinguishing it from our zero-shot approach that uses M-learning.

Eysenbach et al.[63] similarly aim to enhance RL performance through improved representation learning. Their approach introduces contrastive learning techniques, which involve training representations such that neighboring states share similar encodings while unreachable states are mapped to distinctly different representations. This method builds upon the same foundational idea used later by Park et al.[17] in the development of their HILP framework. Their model architecture employs fully connected neural networks.

Several foundational works have played a significant role in shaping contemporary GCRL approaches. Notably, contributions by Park et al.[17], Frans et al.[64], and Ingebrand et al.[65] provide essential insights that have informed the development of our framework. The difference with our approach, however, is in the possible rewards enabled: Zero-shot RL allows for any reward function, while GCRL is limited to a specific goal state.

Approach	**Training**	**Inference**	**Available Reward**
Traditional RL	On 1 task	Direct	Fixed beforehand
Multi-Task RL	On multiple tasks	Direct	Fixed beforehand
Meta-RL	On multiple tasks	Fine-tuning	Any, from the same distribution
Transfer Learning	On source tasks	Fine-tuning	Any, in transfer space
Goal-Conditioned RL	On any transitions	Direct	A specific state (goal)
Zero-Shot RL	On any transitions	Direct	Any

Table 2.1 Comparison of RL generalization approaches

2.3 Zero-Shot RL Methods

In zero-shot RL, we want to learn the reward-free MDP $\mathcal{M} = (S, \mathcal{A}, \mathbb{P}, \gamma)$ relying on a dataset of transitions $\mathcal{D}$. Ideally, we would like to directly learn representations of the Q-function, value function (V), or successor measure (M) using Bellman updates, and then use these representations to deduce the optimal policy for any given reward at inference time. In the tabular case, where the number of states $| S |$ and actions $| A |$ are finite, algorithms exist that achieve policies with an optimality gap bounded by $O(\sqrt{(} | S | \cdot | A |))$[66]. However, this approach becomes intractable for large or continuous state and action spaces due to the *curse of dimensionality*, i.e. due to the exponential increase of the complexity and computational demands as the number of dimensions or features grows.

To address this challenge, zero-shot RL relies on approximations to make the problem tractable. Two primary strategies have emerged for implementing zero-shot RL: **Successor Representations (SRs)** and **Forward-Backward (FB) Representations**. SRs learn relationships between reward functions and Q-functions, often using predefined basis functions like Laplacian eigenfunctions. FB Representations learn a factorization of the successor measure (defined inEquation 2.3), enabling policy retrieval for arbitrary reward functions without task-specific fine-tuning.

This section explores the key methods for zero-shot RL, starting with successor representations and low-rank FB representations, followed by alternative approaches like Hilbert representations and functional reward encoding, as illustrated Figure 2.2.

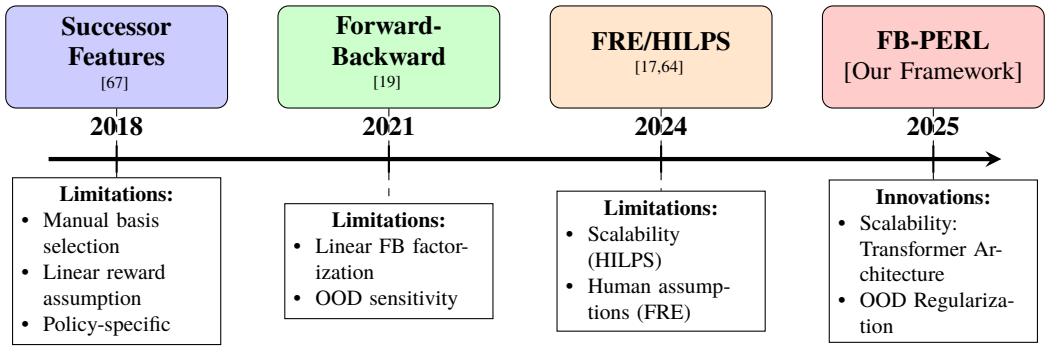


Figure 2.2 Evolution of ZSRL methods (2017-2025): From manually engineered features in successor features to our generalized FB framework.

2.3.1 Successor Representation

The concept of successor representations (SR) was introduced by Dayan[68] as a way to model the expected future occupancy of states under a fixed policy. The core idea is to decompose the action-value function into two interpretable components: a set of

handcrafted or learned basis features $\phi(s, a)$ that describe the state-action space, and a reward weight vector $\mathbf{w}_r$ that parameterizes the reward function. Specifically, the reward can be expressed as a linear function:

$$R(s, a) = \phi(s, a)^\top \mathbf{w}_r.$$

This leads to the decomposition of the Q-function as:

$$Q(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,\middle|\, s_0 = s, a_0 = a\right] = \psi(s, a)^\top \mathbf{w}_r,$$

where $\psi(s, a)$, the successor features, represent the expected discounted sum of future features:

$$\psi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right].$$

This formulation implies that for any linear reward function, computing the Q-function — and hence the optimal policy — reduces to a simple dot product between successor features and the reward weights. As a result, once $\psi(s, a)$ is learned, the agent can efficiently adapt to any new reward function within the linear span of ϕ by updating only $\mathbf{w}_r$, enabling fast policy transfer across tasks.

Successor Representation for Generalization. The successor representation framework was initially developed for transfer learning. Schaul and al.[69] introduced **Universal Value Function Approximators** (UVFA), which generalizes value functions to be conditioned on both states and goals. Mathematically, they extend the value function $V(s)$ to $V(s, g)$, where g represents a goal or task, enabling transfer across tasks with different reward functions. This allows the agent to compute optimal behavior for new tasks by reusing previously learned representations.

Building on this, Barreto and al.[67] formalized the use of **successor features** (SF) for transfer learning in RL. Their work builds on two key ideas. First, a generalization of the successor representation to continuous spaces, allowing for approximation in high-dimensional environments. Their second contribution is a generalization of Bellman's policy improvement theorem to multiple decision policies, enabling the reuse of learned policies across tasks.

Using the linearity of the reward within the Q-function, successor features decompose the Q-function as:

$$Q(s, a, r, \pi) = \psi_\pi(s, a)^\top \mathbf{w}_r,$$

where $\psi_\pi(s, a)$ represents the successor features under policy π.

This formulation was later extended to **Universal Successor Features** (USFs)[70], which further generalizes the framework to handle unseen tasks by conditioning on goal embeddings g. Unlike prior SF methods requiring explicit reward specifications, USFs implicitly encode tasks through learned goal embeddings g, handling **nonlinear reward structures** by replacing the linear reward decomposition $r = \phi(s)^\top w_g$ with $\phi(s, g)^\top w_g$.

Successor Measures[19] build upon this idea, but switch from Q (defined inEquation 2.2) to M (defined in Equation 2.3) for learning the paradigm. More details on successor measures and their learning framework will be provided in subsection 2.3.2

Adapting Successor Features for Zero-Shot RL. The transition from transfer learning to zero-shot RL was achieved by combining successor features with goal-conditioned policies. Borsa and al.[71] introduced **Universal Successor Features Approximators** (USFA), which extend successor features and UVFA to handle unseen tasks by conditioning on goal embeddings g. They define universal successor features $\psi(s, a, \pi) = \psi^\pi(s, a)$ This allows double dependency between the reward decomposition factor w: $Q^{\pi_w}(s, a, w)$. USFAs enable zero-shot policy evaluation for both seen and unseen goals, making them a powerful tool for generalization. However, they still rely on manually selected basis features $\phi(s, g)$, which limits their flexibility.

Recent Advances and Limitations. Recent work has sought to address the limitations of successor features by introducing more flexible and scalable methods. For example, Chen and al.[20] proposed **RaMP** (Random Features for Model-Free Planning), which learns a set of random Q-basis functions in a self-supervised manner. RaMP (Figure 2.3) eliminates the need for manual feature selection and improves long-horizon planning by modeling the long-term expected reward for many random functions. However, this idea is not yet ZSRL, as it still requires solving an optimization problem at test time to project new rewards onto the learned basis functions.

Another notable advancement is the **Distributional Successor Measure** (DSM) introduced by Wiltzer and al.[28] DSM models the distribution of successor states rather than their expectation, providing additional robustness to stochastic environments. However, like other successor feature methods, DSM is limited to tasks where rewards can be expressed as linear combinations of basis functions.

Last year, Zhu and al.[72] introduced **Generalized Occupancy Models** (GOMs),

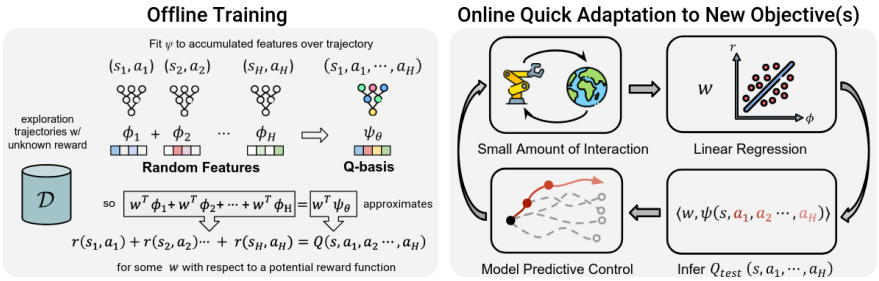


Figure 2.3 RaMP[20]: Depiction of our proposed method for transferring behavior across tasks by leveraging model-free learning of random features. At training time, Q-basis functions are trained on accumulated random features. At test time, adaptation is performed by solving linear regression and recombining basis functions, followed by online planning with MPC

which model the distribution of successor features under the behavior policy. GOMs encode all possible outcomes that appear in the dataset starting from a designated state, enabling quick adaptation to new downstream tasks without test-time policy optimization. However, GOMs require manual selection of features $\phi(s)$ and limit rewards to linear combinations, as shown in Figure 2.4.

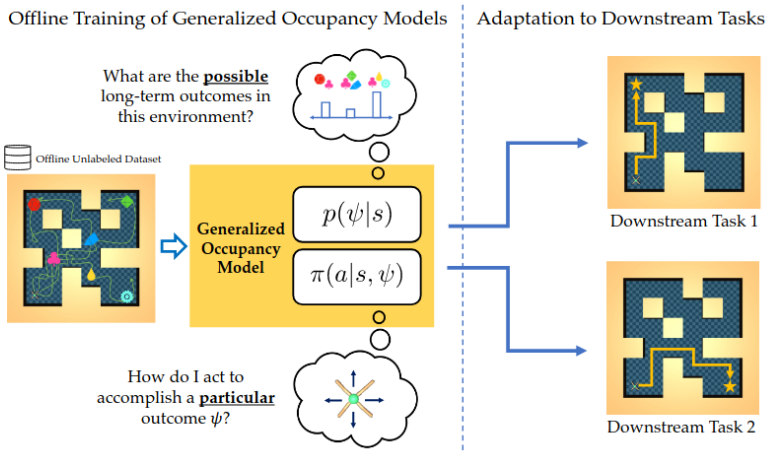


Figure 2.4 The transfer setting for generalized occupancy models. Given an unlabeled offline dataset, we learn a generalized occupancy model that models both "what can happen?" $p(\psi|s)$ and "how can we achieve a particular outcome?" $p(a|s,\psi)$. This is used for quick adaptation to new downstream tasks without test-time policy optimization.[72]

In parallel, Ingebrand and al.[65] proposed **Function Encoders**, which extend the idea of successor features by encoding the task/reward as a weighted combination of learned, non-linear basis functions. This approach allows the policy to transfer to new tasks by representing similar tasks with similar coefficients. However, it assumes the existence of a finite set of basis functions, which is a strong limitation. The workflow of *function encoders* is illustrated in Figure 2.5. Their setting enables zero-shot policy evaluation as

well. If *function encoders* and RaMP use similar function encoding ideas, they differ in the way they use these function encodings for computing Q.

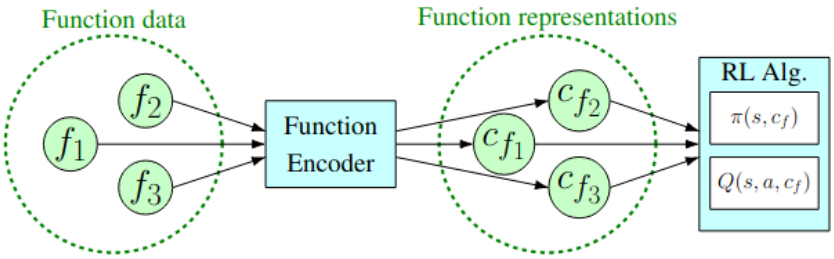


Figure 2.5 Function encoders workflow [65]

Despite their strengths, successor feature methods face two key challenges. First, they rely on manually selected basis features $\phi(s, a)$, which limit their flexibility and scalability. Second, they assume that rewards are linear in the feature space, which restricts their ability to handle more complex reward structures.

These limitations have motivated the development of alternative approaches, such as **Forward-Backward Representations**, which we discuss in the next section.

2.3.2 Forward-Backward Representation

To simplify computation in zero-shot reinforcement learning, we can use the successor measure, which is less dependent on the reward, rather than relying on the Q and V functions. However, learning the successor measure directly is intractable in real-life scenarios. The Forward-Backward representation addresses this by using a low-rank model for M, decoupling transition dynamics from policy learning.

Low-Rank Model. Low-rank Markov Decision Processes offer a way to break the curse of dimensionality in reinforcement learning by factorizing transition dynamics into a low-rank structure [66]. This factorization captures meaningful patterns in the data while reducing computational complexity. For a state s, an action a and a next action s', the transition dynamic is expressed as:

$$P(s'|s, a) = \sum_h \phi_h(s, a)\psi_h(s'),$$

where $\phi_h(s, a)$ and $\psi_h(s')$ represent state-action features and next-state features, respectively. This enables efficient learning and planning in high-dimensional environments.

Low-rank MDPs demonstrate greater expressiveness than alternatives (block

MDPs…), while maintaining remarkable sample efficiency guarantees[73].

Forward-Backward Representations. FB representations, introduced by Touati and al.[19], rely on a low-rank structure to express successor measures, that model the expected discounted time spent in each state-action pair.

For a start transition (s_0, a_0) and another transition X starting from s', the FB representation factorizes the successor measure $M^\pi(s_0, a_0, X)$ (cf. Equation 2.3) under policy π into two parts: a forward matrix F and a backward matrix B.

$$M^{\pi_z}(s_0, a_0, \mathrm{d}s') \approx F(s_0, a_0, z)^\top B(s')\rho(\mathrm{d}s') \tag{2.9}$$

This factorization enables zero-shot policy evaluation by expressing the value function as an expectation over the reward:

$$V_r^\pi(x) = (1-\gamma)^{-1}\mathbb{E}_{X'\sim M^\pi(\cdot|x)}[r(X')].$$

This formulation separates the value function into transition information (encoded in M^π) and reward information, as demonstrated by Blier and al.[74]

The policy can then be retrieved in a zero-shot manner by computing:

$$z_R = \int_{s,a} r(s,a)B(s,a)\rho(ds, da)$$
$$\pi_r(s) = \arg\max_a F(s, a, z_R)^\top z_R$$

where z_R is the latent representation of the reward.

The successor measure is learned using its Bellman update:

$$\left\| F^\top B\rho - \left(P + \gamma P_{\pi_z}\overline{F}^\top\overline{B}\rho\right)\right\|_\rho^2.$$

The FB loss $\mathcal{L}_{FB}(F, B)$, obtained after developing the Bellman update, can be written as $\min_{F,B}\mathcal{L}_{FB}(F, B)$, where:

$$\mathcal{L}_{FB}(F,B) := \mathbb{E}_{(s_t,a_t,s_{t+1},s')\sim\rho}\left[\left(F(s_t,a_t,z)^\top B(s') - \gamma\overline{F}(s_{t+1},\pi_z(s_{t+1}),z)^\top\overline{B}(s')\right)^2\right]$$
$$- 2\,\mathbb{E}_{(s_t,a_t,s_{t+1})\sim\rho}\left[F(s_t,a_t,z)^\top B(s_{t+1})\right] + \text{Const} \tag{2.10}$$

This FB framework is learned using the soft actor-critic (SAC) method[75], one state-of-the art model-free RL learning framework. SAC learns the policy through an actor-critic approach, where the actor generates actions (based on M) while the critic computes the corresponding Q-values to guide policy improvement, enabling efficient offline training

without interacting with the environment.

FB Theoretical Strength and Connection to Successor Features. Low-rank MDPs, and therefore FB, offer several theoretical advantages, particularly in terms of **reward diversity** (spare and dense rewards[66]), **compact representation**[76], **expressiveness**[66], and **sample efficiency**[73,77]. Unlike successor features, which are limited to covering rewards within the linear span of ϕ, FB representations provide broader coverage, accommodating any reward structure. The low-rank factorization also enables a compact representation of the transition dynamics, significantly reducing the number of parameters to be learned[76], breaking the curse of dimensionality. Furthermore, as demonstrated by Agarwal et al.[73], low-rank MDPs exhibit greater expressiveness compared to block MDPs and other structured models, while maintaining computational tractability. They also achieve strong sample efficiency guarantees, scaling as $O(d^4)$ with the rank parameter d, making them well-suited for large-scale applications. Notably, if the successor measure can be accurately approximated as a product, the resulting policy is guaranteed to be optimal for any reward function, as supported by Touati et al.[16].

Low rank MDP can be seen as a generalization of successor features[16,19]. The FB definition implies that $\psi(s, a, z) := F(s, a, z)$ are the successor features of $\phi(s) := (E_\rho BB^\top)^{-1} B(s)$. This follows from multiplying the Equation 2.9 by $B^\top(\mathbb{E}_\rho BB^\top)^{-1}$ on the right and integrating over $s' \sim \rho$. Thus, FB can be used to produce both ϕ and ψ in successor features, although the training process is different. This connection is one-directional: Equation 2.9 is a stronger condition. In particular, $F = B = 0$ is not a solution, unlike $\psi = \phi = 0$ for successor features. No additional criterion to train ϕ is required: F and B are trained jointly to provide the best rank-d approximation of the successor measures M^π. This summarizes an environment by selecting the features that best describe the relationship $Q_r^\pi = M^\pi r$ between rewards and Q-functions.

Experimental results[16] show that FB representations outperform successor feature models, even when the latter are optimized on different bases (e.g., Laplacian eigenfunctions, autoencoders, inverse curiosity, low-rank transition matrices, contrastive learning, and diversity methods).

The FB algorithm constructs a structured latent space that aligns states and rewards, enabling the direct computation of the value function V at any given state s without the

need to explicitly model V. Instead, it derives $V(s)$ as:

$$V_z(s) = \mathbb{E}_a[F(s,a,z)]^\top z = \mathbb{E}_a[F(s,a,z)]^\top B(s) r_z(s) \tag{2.11}$$

where $B(s)$ represents the learned basis functions, effectively encoding the environment's transition dynamics, weighted by the reward signal r_z.

A key theoretical foundation for FB stems from its relationship with successor measures, which form a simplex as discussed by Eysenbach and al.[78] This structure provides strong guarantees on the learnability and convergence properties of FB models, positioning them as a theoretically sound approximation method within reinforcement learning.

Summarizing its key advantages, FB are powerful tractable approximations, offering several compelling properties:

- **Minimal reliance on human priors**: Unlike successor features, FB does not require selecting a base policy, making it more adaptable across different environments.
- **Theoretical guarantees**: The FB framework is mathematically tractable and enjoys well-established theoretical guarantees on its expressivity and convergence[74,78].
- **Interpretability**: The decomposition into F and B provides meaningful insights—F aligns with policy retrieval, while B encapsulates Markov Decision Process dynamics, offering an interpretable representation of environment dynamics.
- **Empirical robustness**: Despite numerous proposed improvements, FB remains difficult to surpass in performance, highlighting its efficiency and robustness as a representation-learning method.

Forward-Backward Representations Limitations. Despite their advantages, FB representations face several practical challenges. Their performance is sensitive to dataset size and diversity, with degradation observed on small or homogeneous datasets as demonstrated by Jeen et al.[79] The successor measure factorization relies on low-rank assumptions. Its effectiveness is constrained by the dimensionality of the latent space (dimension of z, latent space representation), and the inherent approximation errors from this low-rank approach may hinder learning in environments with complex dynamics. Additionally, these representations require expansive and varied datasets for optimal performance. Although more efficient than full-rank models, the computational demands of training and inference in low-rank MDPs remain considerable, particularly when scaling to real-world applications.

The decomposition $M = F^\top B$ imposes a restrictive low-rank assumption on dynamics, limiting the expressivity of the learned representations. The neural networks used for F and B are typically multi-layer perceptrons (MLPs), which do not scale efficiently to high-dimensional state-action spaces.

These limitations highlight the need for further research into scalable and robust methods for zero-shot RL.

Addressing these limitations is the core objective of our work. We aim to develop scalable and robust zero-shot RL methods that enhance FB's generalization capabilities, ultimately advancing its real-world applicability.

Existing Work on Improving FB. Various approaches have sought to enhance FB's capabilities, focusing on:

- **Regularization**: These methods enhance FB representation robustness through loss function regularization, mitigating sensitivity to dataset quality by addressing out-of-distribution challenges and distributional shifts.
  - Methods like VCFB and MCFB[79] adapt conservative Q-learning (CQL)[21] to FB loss, improving performance by adding conservatism to either the value function or the measure.

$$\mathcal{L}_{\text{VC-FB}} = \alpha \cdot (\mathbb{E}[F(s,a,z)^\top z] - \mathbb{E}[F(s,a,z)^\top z] - \mathcal{H}(\mu)) + \mathcal{L}_{\text{FB}} \tag{2.12}$$

$$\mathcal{L}_{\text{MC-FB}} = \alpha \cdot (\mathbb{E}[F(s,a,z)^\top B(s_+)] - \mathbb{E}[F(s,a,z)^\top B(s_+)] - \mathcal{H}(\mu)) + \mathcal{L}_{\text{FB}} \tag{2.13}$$

    These variants improve robustness by reducing out-of-distribution (OOD) errors, with the measure-conservative FB representation outperforming the value-conservative variant on small datasets. However, this conservative variant should be used with caution, as it may lead to overly conservative policies.
  - FB-AWARE[80] proposes regularizing OOD actions using advantage-weighted regression loss[81-82] to improve the robustness of FB representations.

$$\arg\max_\theta \mathbb{E}_{(a_{1:n},s_{1:n})\sim B}\left[\sum_{i=1}^{n} w(s_i,a_i)\log \pi_\theta(a_i|s_i)\right], \tag{2.14}$$

$$\text{where } w(s_i,a_i) = \frac{\exp(A_\phi(s_i,a_i)/\beta)}{\sum_{j=0}^{n}\exp(A_\phi(s_j,a_j)/\beta)}, \tag{2.15}$$

$$\text{and } A_\phi(s,a) = Q_\phi(s,a) - \mathbb{E}_{a'\sim\pi}[Q_\phi(s,a')] \tag{2.16}$$

$$\arg\max_{\theta} \mathbb{E}_{(a_{1:n},s_{1:n})\sim B}\Big[\sum_{i=1}^{n} w(s_i,a_i)\log \pi_\theta(a_i|s_i)\Big], \tag{2.17}$$

where:

$$w(s_i,a_i) = \frac{\exp(A_\phi(s_i,a_i)/\beta)}{\sum_{j=0}^{n}\exp(A_\phi(s_j,a_j)/\beta)}, \tag{2.18}$$

$$A_\phi(s,a) = Q_\phi(s,a) - \mathbb{E}_{a'\sim\pi}[Q_\phi(s,a')] \tag{2.19}$$

The author reports performance improvements, however, the regularization performance is challenged by Tirinzoni and al.[83], reporting limited performance improvement for the same method. The FB-AWARE author did not publish their code as today.

- **Algorithmic Enhancements**:
  - **Auto-regressive B structure** A recent work from Cetin and al.[80] proposes FB-AWARE, which introduces autoregressive features in B to capture progressively refined task representations:

$$z_1 = \mathbb{E}[r(s)B_1(s)], \tag{2.20}$$

$$z_2 = \mathbb{E}[r(s)B_2(s,z_1)]. \tag{2.21}$$

    However, the author's implementation of this approach provides little improvement over plain FB.
  - **Breaking Linearity**: Proto Successor Measure (PSM)[84] extends FB's linear relationship into an affine transformation:

$$M = \sum_{i}^{d} \phi_i w_i^\pi + b, \tag{2.22}$$

    This affine extension is justified by the affine relation of the Bellman equation:

$$M^\pi(s,a,s^+,a^+) = (1-\gamma)\mathbb{I}[s=s^+,a=a^+] \tag{2.23}$$

$$+\gamma\sum P(s^+|s',a')M^\pi(s,a,s',a')\pi(a^+|s^+). \tag{2.24}$$

    If their extension is promising, the training method—sampling policies π and computing w^π separately—may not be optimal.

FB Representations Derivative Works. Several works have built upon FB representations to extend their applicability and improve performance, highlighting both the versatility of the FB framework and the motivation to further enhance its efficiency. While many

of these approaches introduce additional model complexity, they underscore the potential of FB as a foundation for more capable yet streamlined learning systems.

Some of these recent works[80,85] use the term **Behavioral Foundation Models** (BFMs) as an alternative to the term "zero-shot RL models"[86]. The term "behavior" emphasizes that these models focus on controlling agents in dynamic environments, distinguishing them from foundation models used in other domains such as images, videos, and language[85].

- **RL Zero**

  Sikchi and al. propose a framework to translate human language instructions into policies (Figure 2.6). It achieves it by mapping the human language to a reward, before using the FB model to obtain policy from the reward.

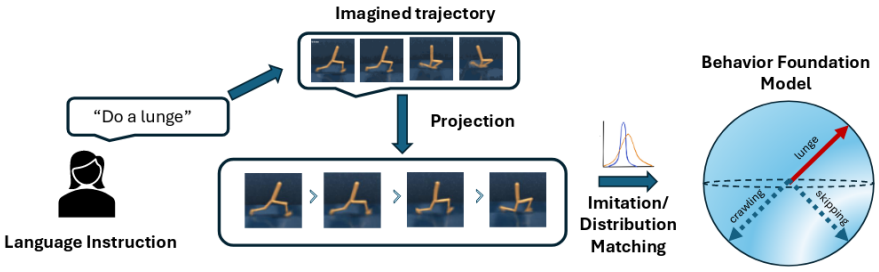


Figure 2.6 RL Zero framework: A video trajectory is synthesized from a text prompt, projected into the agent's observation space, and subsequently aligned with real agent observations. Observation-only imitation learning then grounds the generated trajectory into a policy that mimics the demonstrated behavior.[87]

- **Unsupervised ZSRL via dual-value Forward-Backward representation**

  FB representations have also been extended to online learning paradigms, as demonstrated by Sun and al.[88] However, its practical deployment is hindered by its computational overhead. It requires an extensive pretraining phase followed by fine-tuning. Moreover, the framework introduces additional computational complexity, such as double Q-value learning for π and contrastive losses for reward learning.

- **Zero-Shot RL from Zero-Shot Whole-Body Humanoid Control via Behavioral Foundation Models** The FB-CPR framework[83] enhances behavioral foundation models by integrating Forward-Backward representations into an adversarial framework (GAN type). The framework is detailed on Figure 2.7. This approach trains their architecture on large demonstration datasets to enable zero-shot whole-body control. The core innovation lies in an added GAN type loss providing extra latent space regularization - FB-CPR minimizes the $\mathrm{KL}\big(p_{\text{data}}(\pi) \parallel p_z(\pi)\big)$ divergence between dataset-induced policy distributions and latent task distributions using a

discriminator trying to distinguish between the FB generated data and the dataset.

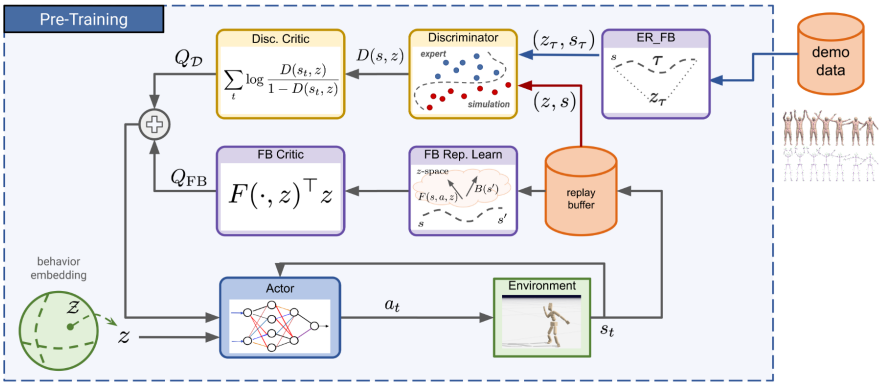


Figure 2.7 FB-CPR architecture: Critic network (left) evaluates state-action pairs while discriminator (right) aligns latent policy distributions with demonstration data. Forward (F) and backward (B) networks form the FB representation core[83].

While effective with high-quality demonstrations, the model's expressivity remains bounded by the neural parameterization of $F(s, a, z)$ and $B(s)$ networks. The models often struggle to generalize beyond the behaviors demonstrated in the training data, as highlighted by Brandfonbrener and al.[89] Additionally, while individual components use standard MLP architectures, their combination into critic-discriminator networks creates a computationally intensive framework of rigid components that individually scales poorly if we try to increase the expressivity of the model.

- **Fast Imitation via Behavior Foundation Models**

  This framework[85] relies on FB representations to combine imitation learning (IL) and zero-shot RL. Unlike traditional imitation learning, which requires expert demonstrations to replicate each specific behavior, the framework aims to generalize new tasks without task-specific training. The core innovation lies in leveraging IL for designing the reward to input to the FB model.

  In experimental evaluations, the method matches or surpasses state-of-the-art offline imitation learning algorithms[85]. However, this improvement comes at the cost of increased computational complexity: as the method is task agnostic, it requires learning a full environment model.

Despite their potential, these methods encounter notable practical limitations. These models face generalization challenges when applied to tasks that substantially deviate from their training distribution, limiting their adaptability to novel scenarios. Additionally, their performance depends critically on access to high-quality demonstration data for training—

a requirement that may prove difficult to fulfill in many real-world applications. The computational demands present another barrier: BFM architectures often employ complex, large-scale designs that incur significant resource costs during both training and inference phases, particularly when deploying these models in practical settings. These combined constraints of generalization capacity, data dependency, and computational intensity underscore key challenges in BFM implementation.

These challenges highlight the need for further research into scalable and robust methods.

In the next section, we discuss alternative approaches, such as Hilbert representations and functional reward encoding, comparing their strengths and limitations in the context of zero-shot RL.

Table 2.2 Comparison of Zero-Shot RL method characteristics

Method	**Reward Flexibility**	**Policy Coverage**	**Key limitations**
SF	Linear in predefined basis	Discrete & Continuous action spaces	Manual basis $\phi(s)$
FB	Nonlinear via latent encoding	Discrete & Continuous action spaces	Poor empirical scaling

2.3.3 Alternative Approaches to Zero-Shot RL

Hilbert Representations. Hilbert representations[17] use Hilbert spaces, which are complete metric spaces, to encode temporal distances as spatial distances through isometry, as shown on Figure 2.8. This method allows the retrieval of policies using standard RL training algorithms applied to the Hilbert space rather than the original dataset. The

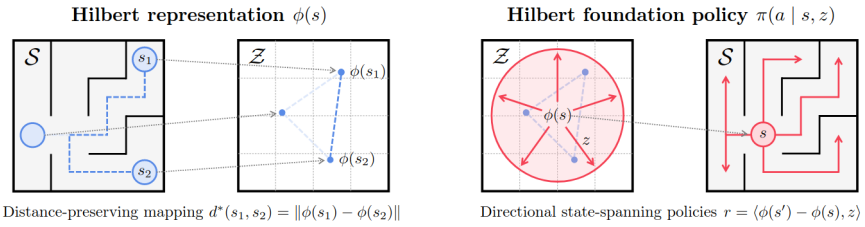


Figure 2.8 *left* Training of a mapping $\phi : S \to Z$ that maps temporally similar states to spatially similar latent states ($d*$ denotes the temporal distance). *right* Training of a latent-conditioned policy$\pi(a|s, z)$[17]

primary advantage of this approach is its explainability. However, it faces challenges in handling complex state relationships and scaling to more intricate problems. For instance, the symmetry of spatial distances may not accurately reflect temporal relations, and the

absence of an isometry between the environment and an Hilbert space can be a limitation. Despite these issues, the explainability of Hilbert representations makes them valuable for further online fine-tuning, as demonstrated by Kim and al.[90], which shows that unsupervised pretraining of diverse policies from offline data can effectively serve as a foundation for online RL.

Functional Reward Encoding (FRE). Functional Reward Encoding (FRE)[64] focuses on learning representations based on reward functions rather than environmental dynamics. This method discretizes the reward function space by evaluating the function on a finite set of states. A Transformer encoder is used to encode state-reward pairs into a latent

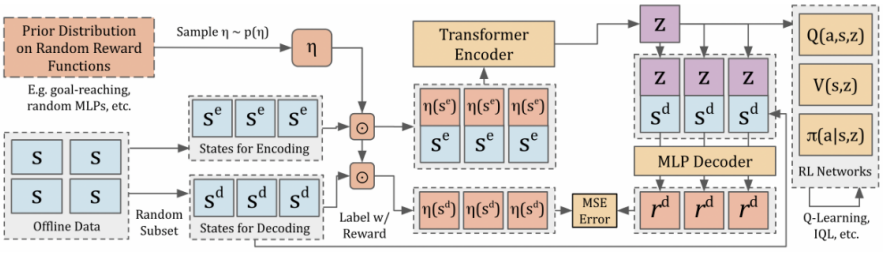


Figure 2.9 FRE architecture[64].

space, followed by a multi-layer perceptron (MLP) for decoding, as shown on Figure 2.9. The architecture is trained using two losses:

- A mean-square error loss between the decoded and actual rewards on a set of evaluation states.
- A loss that structures the latent space by minimizing a lower bound of the mutual information between reward evaluations on different state-action tuples.

This approach is combined with an off-the-shelf offline learning method, implicit Q-learning[91], to enhance the agent's generalization capabilities. While FRE allows the use of arbitrary rewards, it relies on human assumptions for the reward distribution for encoding.

2.4 Summary

This chapter provided an overview of generalization in reinforcement learning, with a particular focus on zero-shot RL (ZSRL) methods. We explored key representations such as successor features and Forward-Backward representations, highlighting their advantages and limitations. The discussion emphasized the necessity of moving beyond

handcrafted features and predefined reward structures to enable scalable, robust zero-shot policy learning.

From the literature, we can draw several key insights.

First, **Zero-shot RL remains an open challenge:** Successor representations provide a structured approach to policy transfer, but their reliance on predefined features limits their adaptability. Other methods face limitations in scalability, expressiveness, or computational efficiency, making them impractical for real-world use.

Then, **FB representations is a tractable alternative** to solving ZSRL. FB models offer a powerful yet computationally reasonable framework by aligning state and reward representations. Unlike SF, they do not require predefined policies, making them well-suited for zero-shot learning scenarios. However, their expressivity remains constrained by the FB learners implementation (MLP).

Current ZSRL methods struggle with complex reward compositions, high-dimensional state spaces, and robustness to OOD scenarios. FB models, despite their theoretical guarantees, suffer from limitations in compositional tasks and are sensitive to dataset quality.

Further improvements to FB representations could enhance the efficiency and applicability of these approaches, particularly in reducing computational costs while preserving their strong generalization capabilities. Such advancements would also facilitate broader adoption in BFMs, ultimately paving the way for real-world deployment of zero-shot RL agents.

Advancing ZSRL representation learning requires next-generation methods that systematically address three interconnected challenges: efficient computation, expressivity, and robustness. First, **efficient computation** demands to optimize FB algorithm and implementation for reducing the overhead. Second, **improved expressivity** necessitates developing models capable of generalizing across diverse tasks without extensive retraining, potentially via modular designs or meta-learning frameworks that adapt to new goals using shared latent structures. Third, **robustness to OOD** values calls for integrating regularization techniques—such as invariance constraints or uncertainty-aware training—to reduce sensitivity to distributional mismatches between training data and deployment scenarios.

This thesis aims to build upon FB representations by introducing structural improvements that address their key limitations. Specifically, we propose a novel framework to tackle expressivity and OOD issues, while aiming at being the foundation for further scal-

able models.

$$\text{Our framework} \xrightarrow{\text{addresses}} \begin{cases} \underbrace{\text{Expressivity}}_{\text{new FB architecture}} \\ \underbrace{\text{Dataset vulnerability}}_{\text{conservatism}} \end{cases}$$

By leveraging a new neural network structure and mechanism for stabilizing FB updates, our approach aims to enhance zero-shot RL's scalability and robustness, ultimately paving the way for real-world applications of BFMs.

CHAPTER 3 ENHANCING THE EXPRESSIVENESS OF FB REPRESENTATIONS

3.1 Representation Learning in Zero-Shot Reinforcement Learning

For zero-shot reinforcement learning to be effective in practical applications, it is essential to develop scalable algorithms capable of learning complex, tractable representations. This requires the ability to encode intricate dynamics while maintaining computational feasibility. To achieve this, the learned representations should satisfy the following criteria: They must rely on large enough neural network (NN) structures to capture complex dynamics without overfitting. They should be adaptable to both small and large datasets to ensure robustness and generalization. Computational costs—both in terms of time and resources—must remain manageable. The loss function should effectively guide the learning process, ensuring meaningful optimization.

A direct attempt to learn the function M using the Bellman update is infeasible due to the high dimensionality of the problem. Indeed, the Bellman loss functions fail to provide sufficient guidance to neural networks in capturing the problem's dynamics. To address these challenges, it is necessary to simplify the problem, while minimizing the loss of generalization.

Existing approaches within the FB (Forward-Backward) framework, as well as most representation learning methodologies in RL, focus on learning the function $M^{\pi^z}(s_0, a_0, s', a')$. However, this formulation presents several inherent challenges that make the problem both difficult and intriguing. The first challenge lies in the **trade-off between expressiveness and tractability:** Larger models tend to be more expressive but also introduce increased computational costs, complexity in training, and difficulty in interpretability. The second challenge lies in the **dataset size limitations:** The methodology must be general enough to remain expressive with large datasets while also being adaptable to small datasets to avoid overfitting.

3.1.1 Why Learn M Instead of Q or V?

As defined in Section 2.2.1, M, Q and V are all different ways of learning the MDP dynamic, and enable retrieving of the policy from them. The choice of M over Q or V is

motivated by its reduced dependency on the reward function. The successor measure M encapsulates the transition probabilities, allowing it to model the underlying dynamics of the environment more effectively. In contrast, the value function V aggregates rewards, making it more dependent on specific reward structures and therefore less generalizable. Given its higher dimensionality, M is better suited to capturing the detailed environment's dynamics in a way that supports zero-shot generalization. However, it's important to note that M still depends on the reward, as the transition probabilities are conditioned on the reward function—evident in its formulation through the dependency on z.

3.1.2 Fundamental Tensions in Representation Learning

Our framework's core methodology—decoupling dynamics encoding (M^{π}) from reward-driven policy optimization—creates inherent tensions among three interdependent considerations. First, the representation must achieve sufficient *expressiveness* to model intricate system dynamics, a requirement that risks overfitting if unconstrained. Second, the model needs *generalization* capabilities robust enough to extrapolate beyond the behavioral policies observed in the dataset $\mathcal{D}$, demanding careful regularization against distributional shifts. Third, these goals must coexist with *computational tractability*, requiring algorithmic designs that maintain efficiency even as representational complexity increases. Balancing these competing priorities—rich dynamic modeling, policy-agnostic generalization, and practical scalability—forms the central challenge in our representation learning paradigm.

As discussed in Section 2.3.2, a core principle of FB representations is the decoupling of transition learning from policy learning. Our approach retains this fundamental idea while extending it into a more expressive framework. Specifically, as presented in previous works[16,19], we aim to learn a general function of the form:

$$M(s, a, s', z) = F(s, a, z)^{\top} \cdot B(s') \tag{3.1}$$

3.2 Scaling F and B Representations

The default F and B structures, as proposed in previous works[16,19,79], are primarily based on multilayer perceptrons (MLPs), which do not scale effectively with increasing problem size. The next step in scaling the methods involves improving the expressiveness and efficiency of these representations by designing better F and B neural architectures.

3.2.1 Scaling Strategies in Modern RL Architectures

In this subsection, we examine recent advancements in neural architectures for both online and offline reinforcement learning, highlighting key works that have influenced our approach to scaling F and B representations.

Online RL Architectures. Recent architectural innovations in online reinforcement learning integrate supervised learning advancements with neural architecture optimization. Bjorck et al.[92] demonstrate improved training stability through normalization layers and skip connections adapted from computer vision architectures. Building on this, Nauman et al.[93] propose BroNet, combining dense layers with residual connections, layer normalization, and ReLU activations (Figure 3.1). Concurrently, Schwarzer et al.[94] implement ResNet[95] and CNN variants within the IMPALA framework[96], advocating width-based scaling strategies, aligned with observations by Ota et al.[97].

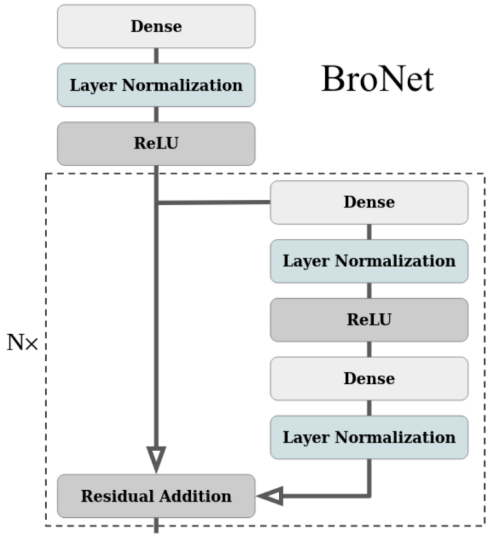


Figure 3.1 BroNet architecture[93]. Uses N=2 for actor and critic

Ota and al.[98] proposes learning better representations by working on the feature extraction to give as an input of the model. However, in our simulation, the features are imposed by the physics simulator.

These advancements align with our goal of designing scalable FB representations that maintain computational efficiency while capturing complex dynamics. However, while these strategies have shown promise in online RL, their direct application to offline RL and FB representations is limited.

Offline RL Architecture. Offline RL research has yielded several notable architectural innovations:

Kumar et al.[99] propose a modified ResNet architecture for feature extraction, combined with feedforward layers and linear projections (Figure 3.2). They demonstrate that their approach, incorporating group normalization, scales effectively with increasing parameters, while previous architectures like Impala degrade. Meta[82] builds upon the architecture proposed by Bjorck et al.[92], integrating regularization techniques from IQL[91] and AWAC[100]. Spangenberg et al.[101] propose multiple Attention Layers architecture for offline actor-critic methods (Figure 3.3). Their structure adapts the Transformer paradigm, incorporating an encoder and decoder to suit their problem. However, due to the high dimensionality of RL problems, they modify the architecture. Instead of an Attention-based decoder, they employ a distributional Q-function, following a single Cross-Attention Layer to combine the action and the observation information encoded into a latent space. This methodology aligns with the paradigm we propose. Cheng et al.[102] (JOWA) employ a Transformer backbone for joint world modeling and Q-value estimation, though their focus on aligning world models with Q-values diverges somewhat from our approach.

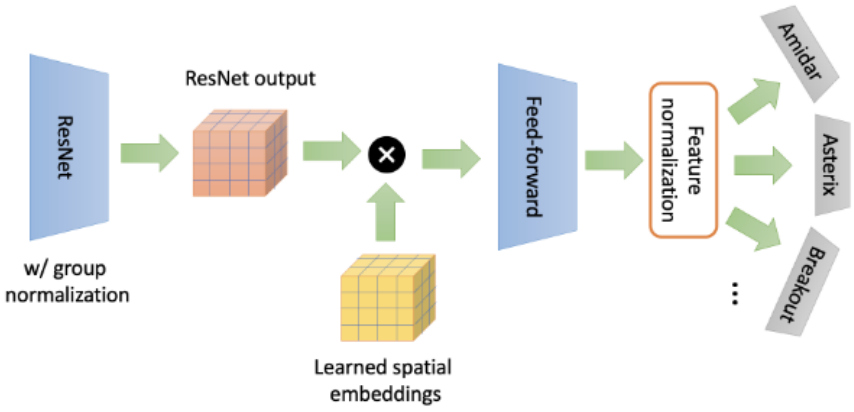


Figure 3.2 Offline Q-learning network architecture[99]

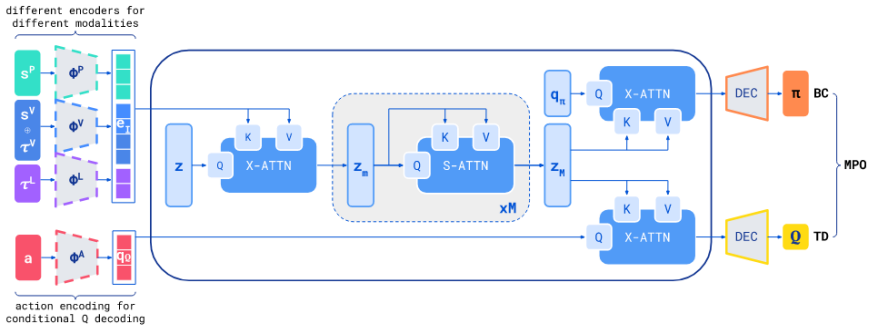


Figure 3.3 Proposed Attention architecture for scaling offline actor-critic[101].

An interesting observation from Nauman et al.[93] suggests that in actor-critic architectures, the critic's performance is the primary determinant of overall model effectiveness, with **actor scaling having limited impact**.

Building on these insights, our contribution extends these scaling strategies by adapting them specifically to the FB framework. We propose a modular Transformer-based architecture that balances expressiveness and computational efficiency, enabling scalability across diverse problem sizes. Unlike prior works, our design leaves flexibility, such as modular blocks and adjustable depth, to ensure adaptability to both small and large datasets. This approach not only addresses the limitations of MLPs but also leverages the strengths of modern RL architectures to enhance the robustness and generalization capabilities of FB representations.

3.2.2 Transformers as Most Scalable Neural Networks

Transformer Architecture. The application of Transformer architectures to reinforcement learning (RL) is motivated by parallels between RL and natural language processing (NLP) in learning sequential data representation. Our approach focuses on two key components: the Backward model (B) and the Forward model (F). The B learner network captures efficient representations from unsupervised data to classify sequential observations and predict actions, resembling NLP's sequence modeling. The F learner network predicts next-action probabilities, analogous to next-word prediction in NLP.

We adopt Transformers[103] to address scaling challenges in multilayer perceptrons, mirroring NLP's solution to similar issues. The Transformer's ability to model long-range dependencies and its scalability (demonstrated by numerous works, including BERT[104] or GPT[105]) make it a promising architecture for our problem.

While RL applications of Transformers exist, they primarily focus on Q-function

modeling for autoregressive state/action generation[106-107]. On the contrary, our approach relies on learning FB representations. Recent studies[108-117] demonstrate various Transformer implementations in RL, but their efficacy remains debated[118-119]. Implementation considerations include architectural choice (decoder-only, full Transformer, or encoder-decoder)[120] and problem-specific optimizations.

The ongoing research into Transformer efficiency promises future improvements in computational performance[121], further justifying their adoption. This potential for enhanced efficiency, coupled with the Transformer's inherent scalability, positions it as a promising foundation for future RL architectures.

Scaling MLPs is computationally cheap but yields limited performance. As an alternative to complex neural networks from the literature, we also tried to keep a lightweight structure, by scaling the original MLP structure.

The literature proposes insights, for example, ResNet[95] or DenseNet[122] structures, using residual connections for avoiding forgetting. Ota and al.[97] also propose increasing the width of the network, rather than increasing the depth, for having a much regular loss.

We attempted to improve the original FB model beyond its current performance by increasing the depth and width of the F and B neural networks, changing the learning rate or the NN architecture (Densenet, Resnet, MLP). However, these improvements do not help get better performance. This reveals that MLPs and residual connections are too simple to capture further complexity of F and B.

We observed, in addition, that the change of the actor size does not impact the performance of the model, as described before by Nauman and al.[93]
For finding more expressive and scalable models, we want to combine the input in different ways for learning more complex relationship, leading to the use of Attention layers and Transformers.

3.2.3 B Learner Design

Approaches to Learning the Backward Model. Training the backward model B can be achieved using two distinct methodological paradigms. The **Independent Learning (HILPS-style)** approach trains B as a standalone component with dedicated loss functions, leveraging techniques from goal-conditioned RL and representation learning akin to Park et al.[17] Conversely, **Joint Learning with** F co-optimizes B alongside the forward model through M.

Our focus is on the second approach, as it maintains generality while ensuring that the learned representation remains both expressive and computationally efficient.

B as an Environmental Model. The role of B is to model the environment's dynamics, capturing the transition structure within the system. As such, techniques from representation learning and model-based RL—particularly goal-conditioned RL (cf. paragraph 2.2.2)—can be leveraged, since B depends only on state transitions. To scale the model, we experimented with MLP structures, modifying the width and depth of the neural networks defining F and B. We notice better scaling in width than in depth, as per Ota and al.[97] As this approach did not yield significant improvements, we turned to more complex architectures, such as Transformers.

Transformer-Based Design for B. We experimented with various Transformer architectures to optimize B: Encoder-only, Encoder-Decoder and Decoder-only. The Encoder-Decoder structure, while expressive, proved computationally heavy without yielding significant gains. The encoder and decoder differ primarily in their Attention Mechanisms, particularly their masking strategies. Since our F and B modules do not use autoregressive masks, we are closer to an Encoder structure.

Given our modular approach, where F and B all rely on Transformers, we maintain consistency by reusing Transformer Blocks for efficiency. However, the PyTorch Transformer encoder was too rigid for our needs, so we implemented a custom Transformer Block, inspired by Karpathy's decoder-only open-source implementation[123-124].

To refine our model, we systematically tested different configurations of number of layers, number of Attention heads, hidden dimension size, dropout rates and activation functions.

Our final optimized B structure consists of an encoder-only Transformer with 8 Attention heads, 2 hidden layers of size 512 and no dropout outside the MLP.

The use of eight Attention heads enables the model to capture diverse dependencies in high-dimensional state spaces, which is critical for modeling complex dynamics in zero-shot RL.

The architectural design of the backward learner, including its Transformer-based implementation, is elaborated in Appendix C.3. The architectural design is illustrated in Figure 3.4.

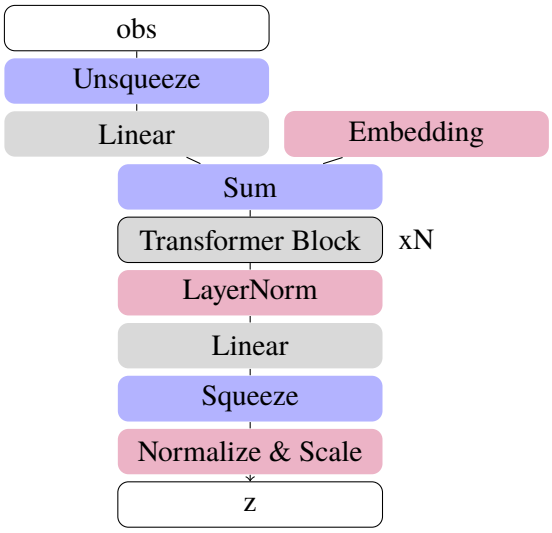


Figure 3.4 Backward learner architecture

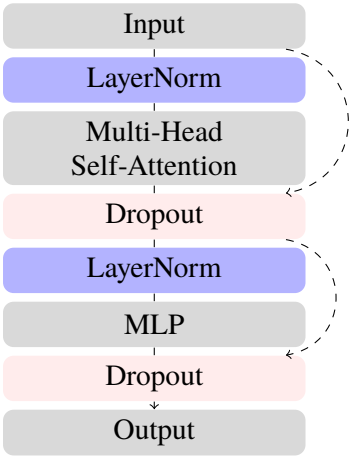


Figure 3.5 Transformer block architecture

3.2.4 F Learner Design

The first intuition for expanding generality was to work on F inputs, because of the limitations imposed by their simple MLP-based interactions. We want to capture more precisely the complex dynamic between actions and observations. Therefore, we think about breaking the linearity using Attention Layers. A first intuitive model has been deployed, using intuitive Attention Layers. Instead of using a simple MLP after preprocessing, we proposed combining updated MLP blocks with Attention Layers into a new structure (Figure 3.6). As per the original structure, the observations obs, the action a, and the latent variable z are preprocessed separately before being concatenated. They then pass through a series of Self-Attention and Feedforward Layers. Finally, an MLP is used to generate the outputs F.

However, such an approach is difficult to further scale or adapt in size depending on the size of the problem. The ideal approach would be to have a block-based architecture that can be combined, and whose number can be adjusted depending on the size and complexity of the problem. This enables avoiding overfitting in small environments, while having a good expressivity in more complex environments. Moreover, we would ideally have a structure that is efficient in time computing, so that it could be used in real-time applications with reasonable computing resources.

To effectively implement F, we leverage the same Transformer-based architecture as B, while optimizing specific hyperparameters to enhance efficiency. Its design training regularization elements, such as dropout, which has been shown in our experiments to be

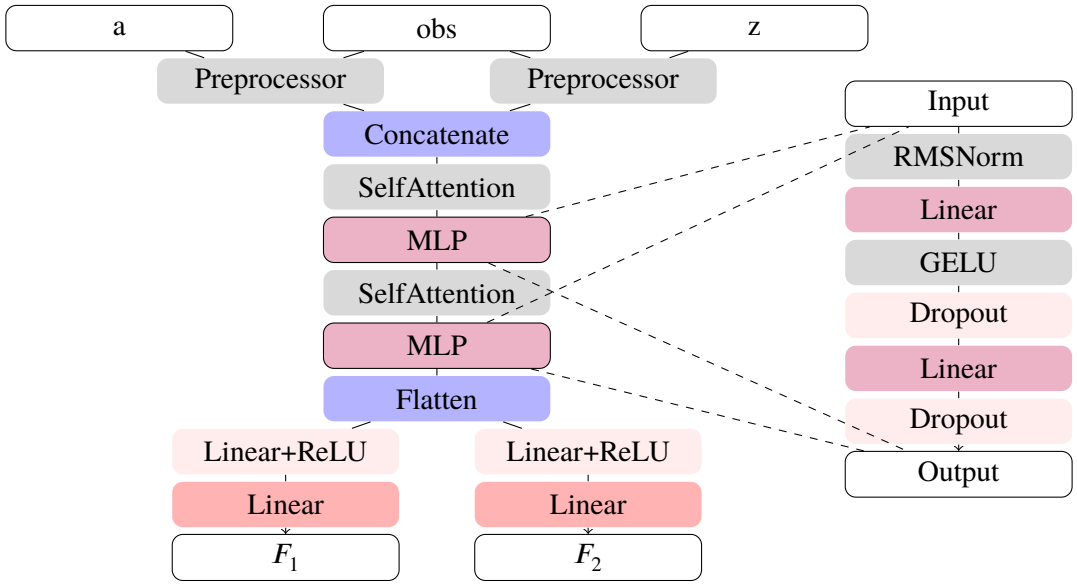


Figure 3.6 Forward learner 2nd architecture showing dual processing branches, with updated MLP architectures

more beneficial for F's structure than B's inherently noisy data distribution.

As input, we receive the observation-action pair and the observation-latent variable z pair. Different experiments have been conducted on how to combine these inputs efficiently, for example, using a simple sum, a product or cross-attention. The most effective approach was to concatenate the observation-action pair and the latent variable z along the last axis.

The architecture follows a structured pipeline, beginning with preprocessing and embedding transformations applied separately to observation-action pairs and observation-latent space pairs. These embeddings are subsequently concatenated to form a unified representation, which then passes through a series of Transformer Blocks. Each block maintains a standard configuration—comprising layer normalization, Self-Attention Mechanisms, and MLP layers—while fine-tuned hyperparameters ensure optimal performance for F's specific needs.

To maintain computational efficiency and scalability, we adjust the number of Transformer layers and optimize projection dimensions. If the performance of the model tends to increase with the number of Transformer layers, especially with large and diverse datasets, the number of blocks is designed as a compromise between performance and training time. Details on the hyperparameters can be found in Chapter 5. The resulting architecture, visualized in Figure 3.7, presents a streamlined yet effective approach to forward representation modeling. For a detailed implementation of the forward learner, refer to Appendix C.4, which provides the complete code and parameterization.

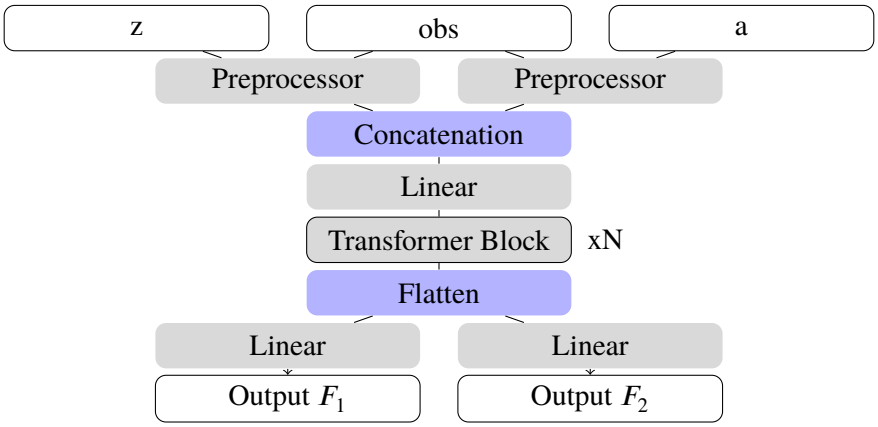


Figure 3.7 Forward learner architecture, with shared Transformer Block structure with B, as per Figure 3.5

3.3 Summary

In this chapter, we have advanced the FB framework by addressing its structural limitations and enhancing its scalability, robustness, and overall performance. Specifically, we transitioned the F and B representations to a Transformer-based architecture, allowing for greater expressiveness and scalability. The performance of the proposed framework will be later evaluated, together with our further improvements, in the chapter 6.

These modifications directly respond to the challenges outlined in the introduction. By refining the representation learning process, we have aimed to balance expressiveness with tractability, ensuring adaptability to both small and large datasets.

Despite these advancements, challenges remain—most notably, the need for greater robustness in OOD settings. As previously discussed, traditional loss functions often struggle to regularize against OOD values, leading to instability when generalizing beyond observed data distributions. The next chapter will focus on addressing this limitation through targeted regularization strategies.

CHAPTER 4 OOD REGULARIZATION FOR FB REPRESENTATIONS

4.1 Out-Of-Distribution Regularization in Offline RL

In reinforcement learning, the objective is to learn a policy that surpasses the performance of the behavior observed during training. This poses significant challenges for offline RL algorithms, which must rely solely on a fixed dataset without direct interaction with the environment.

One fundamental issue is **overfitting**[21,125]. Since offline RL lacks the ability to collect new data through exploration, models risk becoming too specialized to the training dataset. This over-specialization reduces the ability to generalize effectively in real-world scenarios, where conditions may differ from those encountered during training.

Another major difficulty is the **quality and diversity of the training data**[126]. For zero-shot RL to succeed, the pretraining dataset must be large and diverse enough to capture a broad spectrum of behaviors and scenarios. If the dataset lacks optimal transitions—i.e., those leading to the highest rewards—the agent cannot learn truly optimal policies[127]. This limitation is particularly problematic in high-stakes environments, where data collection is costly or impractical.

The most critical challenge, however, is the presence of **out-of-distribution** (OOD) actions. When faced with state-action pairs that are absent from the training data, offline RL algorithms assign arbitrary value estimates, leading to significant errors. This phenomenon, known as distributional shift, arises when the distribution of actions encountered during policy execution differs from that of the training dataset[125-129]. Such mismatches result in incorrect value predictions, ultimately degrading policy performance.

This chapter introduces our approach to regularization, aiming to mitigate these issues within our framework.

4.1.1 Out-of-Distribution Challenge in FB Learning

FB learning consists in minimizing the FB loss $\mathcal{L}_{FB}(F, B)$ proposed by Touati and al.[16,19], direct consequence of M Bellman update. With ρ the data distribution, P the transition operator, F and B the forward and backward representations, $\overline{F}$ and $\overline{B}$ their target networks, the loss can be written:

$$\begin{aligned}\mathcal{L}_{FB}(F, B) &:= \left\| F^\top B\rho - \left(P + \gamma P_{\pi_z} \overline{F}^\top \overline{B}\rho\right)\right\|_\rho^2 \\ &= \mathbb{E}_{(s_t, a_t, s_{t+1}, s')\sim\rho}\left[\left(F(s_t, a_t, z)^\top B(s') - \gamma \overline{F}(s_{t+1}, \pi_z(s_{t+1}), z)^\top \overline{B}(s')\right)^2\right] \\ &\quad - 2\,\mathbb{E}_{(s_t, a_t, s_{t+1})\sim\rho}\left[F(s_t, a_t, z)^\top B(s_{t+1})\right] + \text{Const}\end{aligned} \tag{4.1}$$

Our current learning objective $\mathcal{L}_{FB}(F, B)$ is an out-of-sample objective—meaning it relies on values during training that are not present in the dataset. It inherently suffers from OOD issues when computing $\pi_z(s_{t+1})$ because of the state s_{t+1} not being guaranteed to have any following actions in the dataset. When trained on a restricted or non-representative dataset, the policy's prediction for this out-of-distribution state becomes unreliable leading to a suboptimal update. This issue is not unique to our zero-shot RL (ZSRL) setting but is a well-documented challenge in offline RL more broadly.

Offline RL (also known as batch RL) is a problem setting where an agent learns a policy from a pre-collected dataset without direct interaction with the environment. This paradigm has the potential to scale RL by leveraging large datasets, improving sample efficiency, and enabling deployment in safety-critical applications such as autonomous driving, where online exploration is impractical or unsafe. However, the lack of environment interaction makes handling OOD states and actions a fundamental challenge, often leading to overestimation or underestimation of values in unseen regions of the state space.

Numerous approaches have been proposed to address OOD errors in offline RL, and we will review some of the most relevant methods in the next section. Some of these techniques have also been applied to our FB framework, yet they remain overly conservative, often restricting policy learning excessively. To overcome these limitations, we introduce a novel regularization method. This approach, detailed in later sections, aims to provide more effective regularization in offline RL settings.

4.1.2 Existing General Regularization Methods for OOD Issues

To address distributional shift, various regularization strategies have been proposed. Some approaches minimize the divergence between the learned policy and the original data distribution by incorporating a KL divergence penalty into the policy optimization objective[130]. Other methods, such as Conservative Policy Iteration (CPI)[129], dynamically update the reference policy used for KL divergence measurement to improve stability. However, these techniques assume a well-defined reference policy, which is not

always available—especially in cases where the dataset originates from multiple policies. Furthermore, since our approach focuses on learning policy-agnostic representations, such regularization strategies do not directly apply. While they could serve as complementary techniques within policy optimization in future work, our primary focus remains on developing a robust successor measure M learning algorithm.

Other methods could be associated with our framework. The following table summarizes key categories of offline RL regularization approaches, some representative methods, and the core strategies they employ to mitigate distributional shift and overestimation.

Table 4.1 Summary of Offline RL Regularization Approaches

Category	**Representative Methods**	**Core Strategy**
Combining multiple Q-function approximations	[131-132]	Learn several Q-functions independently and aggregate (e.g., via minimum or averaging) to reduce overestimation.
Constraining Policy Actions to the Dataset	[126,128]	Prevent selection of out-of-distribution actions.
Penalizing Q-values of OOD actions	[21,133]	Penalize high Q-values for unseen or unlikely actions.
Limiting Q-Value Estimation to In-Distribution Actions	[91,134-135]	Avoid estimating values for OOD actions by using in-distribution-only updates.

The following paragraphs provide a deeper analysis of each category summarized above.

Combining multiple Q-function approximations. The original FB framework[16,19] uses double Q-learning[131] to mitigate overestimation errors. This technique mitigates overestimation errors by maintaining two independent Q-functions and using their minimum for value updates, i.e. $Q = \min(Q_1, Q_2)$.

Another approach to offline RL regularization is Random Ensemble Mixture (REM)[132], which introduces Q-function averaging as a way to mitigate overestimation errors. While effective in reducing variance, REM does not directly tackle the underlying issue of distributional shift, leading to suboptimal performance when the dataset lacks diversity.

Constraining Policy Actions to the Dataset. BCQ[128] and BEAR[126] introduce explicit constraints on action selection. BCQ restricts the learned policy to actions present

in the dataset, ensuring theoretical guarantees under certain conditions. However, this assumption is often violated in real-world settings where the dataset does not fully cover the state-action space. BEAR refines this idea by constraining action selection through Maximum Mean Discrepancy (MMD) regularization, ensuring the learned policy stays within the behavior distribution. While BEAR improves stability, its conservative updates limit policy exploration, leading to reduced performance in certain environments.

Penalizing Q-values of OOD actions. CQL[21] addresses the issue of overestimated Q-values in OOD regions by modifying the standard Q-learning objective to introduce a penalty term. This prevents the agent from assigning high value estimates to actions outside the dataset, effectively keeping the learned policy close to the behavior policy (cf. Figure 4.1). One of CQL's main advantages is its simplicity—it only requires adding a term to the Q-learning update, making it easy to implement. However, this method involves a trade-off: stronger regularization improves stability but can overly constrain the learned policy, limiting its ability to explore better strategies. Because of this, CQL regularization is known to be overly conservative[136-139], hindering learning efficiency and performance.

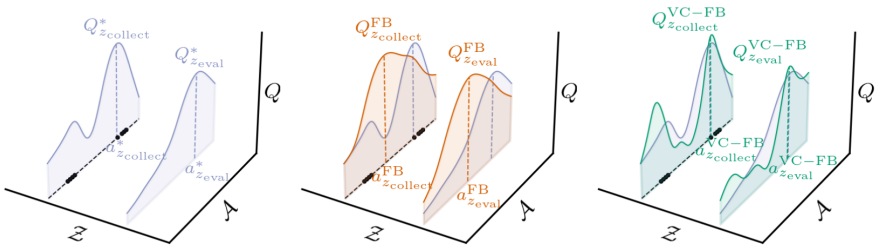


Figure 4.1 (Left) Offline RL methods must train on a dataset collected by a behavior policy optimizing against task $z_{collect}$, yet generalize to new tasks z_{eval}. Both tasks have associated optimal value functions $Q^*_{z_{collect}}$ and $Q^*_{z_{eval}}$. (Middle) Unregularized methods overestimate the value of actions not in the dataset. (Right) CQL regularization constrains the value of actions not in the dataset. Black dots represent state-action samples present in the dataset. Illustration from Jeen and al.[79]

Building on CQL, CalQL[133] introduces a constraint to prevent excessive conservatism. By ensuring that the learned Q-function remains above the Q-values of a reference policy, CalQL achieves a balance between stability and performance. However, this method is primarily designed for fine-tuning tasks and is less suited for full offline learning scenarios, where a well-defined or reliable reference policy may not be available.

Limiting Q-Value Estimation to In-Distribution Actions. Implicit Q-learning (IQL)[91] provides an alternative approach by eliminating the need to estimate Q-values for OOD actions altogether. Instead, IQL focuses only on state-action pairs available in the dataset. To do so, they replace the Q-learning update problematic term $\max_{a\sim\rho} Q(s,a)$ with the value function $V(s)$:

$$\min_{Q} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left(r(s,a) + \gamma V(s') - Q(s,a) \right)^2 \right] \tag{4.2}$$

This method then leverages *expectile regression* (cf. subsection 4.2.2) to estimate the value function relying on dataset actions only. Mathematically, the method obtains V by solving the optimization problem:

$$\min_{V} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[|\tau - \mathbb{1}\left(Q(s,a) - V(s) < 0\right)| \left(Q(s,a) - V(s)\right)^2 \right] \tag{2}$$

where $\mathbb{1}$ is the indicator function.

After learning Q and V, IQL extracts the policy using advantage-weighted regression[81,100]:

$$\min_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\exp\left(\beta \left(Q(s,a) - V(s)\right)\right) \log \pi(a|s) \right]. \tag{3}$$

IDQL[134] extends IQL by incorporating diffusion-based techniques, leading to a 10% performance improvement in offline RL tasks, albeit with increased computational complexity. Another variant, AlignIQL[135], modifies policy generation from the Q-function through an optimization approach. However, empirical results show that AlignIQL does not significantly outperform standard IQL or IDQL.

A more recent in-sample learning paradigm, introduced by Xu and al.[140], presents Implicit Value Regularization, a general framework that unifies CQL and Advantage-Weighted Regression (AWR)[81]. This framework provides a theoretical foundation for balancing value estimation and policy constraints based on Dual RL[141]. SQL and EQL, two learning algorithms derived from this framework, address the core OOD problem in the Q-learning update using V replacement, like shown in Equation 4.2, and incorporating an additional regularization term:

$$V(s) = \mathbb{E}_{a\sim\pi} \left[Q(s,a) - \alpha f\left(\frac{\pi(a|s)}{\mu(a|s)} \right) \right]. \tag{4.3}$$

Using dual optimization, they deduce the optimal policy and value functions, leading to a more robust learning framework. However, a similar transformation using their dual optimization technique cannot apply to the FB framework.

While IQL and its extensions provide strong performance in offline RL, all existing

methods remain dependent on the dataset's quality, particularly the diversity of its action space. As highlighted by Levine and al.[127], designing scalable and efficient offline RL methods that generalize well remains an open challenge. Future work should explore new exploration strategies for constructing diverse datasets while maintaining computational feasibility.

4.1.3 FB OOD Regularization in the Literature

The default regularization approach in FB learning, proposed by Touati and al.[16,19] and based on Double Q-learning[131], is pessimistic and does not completely avoid overestimations (cf. paragraph 4.1.2).

Building upon FB, Jeen and al. introduces VCFB/MCFB, extending FB with additional regularization mechanisms, inspired by CQL[21]. Indeed, as discussed earlier (cf. paragraph 4.1.2), CQL regularizes by forcing any OOD actions value to be below the value of in-distribution actions, which prevent the policy from learning better strategies than strategies relying on the dataset states and action. These modifications are layered on top of Double Q-learning, which already introduces an implicit form of regularization. The paper does not explicitly discuss the rationale for combining these techniques, leaving open questions about their necessity and impact.

Some other papers have proposed alternatives to Q-learning and Double Q-learning, for example, combining two Q-functions using their mean[93]. However, we find out experimentally that using a mean is as efficient as Single Q-learning, while less computationally efficient.

Cetin and al.[80] integrates AWAC[100] into the FB framework. However, the effectiveness of AWAC remains debated[82,90,136], and the authors do not provide publicly available code, making it challenging to validate their findings. In our own experiments, AWAC has not demonstrated significant improvements over the standard FB framework.

Rather than adapting these conservative regularization methods, we propose to adapt in-sample learning techniques to the FB framework. The advantage of in-sample regularization over out-of-sample regularization is that it enables less strong conservatism, leaving more room for the policy to learn better strategies.

4.2 Our Proposal: In-Sample FB OOD Regularization

4.2.1 Objective

To avoid computing $\pi_z(s_{t+1})$, and eventually obtain out-of-distribution actions, we define $O(s, z)$ such as:

$$O(s, z) \cdot z = \max_a \left[F(s, a, z)^T z \right] \tag{4.4}$$

This definition of O enables to rewrite the learning objective:

$$\begin{aligned} \mathcal{L}_{FB}(F, B) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \rho} \Big[\big(F(s_t, a_t, z)^\top B(s') - \gamma O(s_t, z) B(s') \big)^2 \Big] \\ - 2\mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \rho} \Big[F(s_t, a_t, z)^\top B(s_{t+1}) \Big] + \text{Constant} \end{aligned} \tag{4.5}$$

The loss expression now exclusively relies on dataset samples.

The idea behind this definition, rather than $O(s, z) = \mathbb{E}_{a \in \mathcal{D}} \left[F(s, a, z)^T B(s') \right]$, for example, is to emphasize the training of $F \cdot z$ to better retrieve π later. In the current model, F and B are exclusively trained using the Bellman equation, with conjoint training, which is suboptimal for two reasons. First, because we don't take advantage of the π definition and properties within the training. Secondly, because we systematically train FB together, without having $F \cdot z$ appearing, which could lead to finding a suboptimal F - a good FB conjoint results that do not lead to the best F representation possible.

We learn O using its definition, by minimizing the difference:

$$||O(s, z)z - \max_a \left[F(s, a, z)^T z \right] ||$$ (4.6)

The goal is to rewrite the expression of O to sample a from the dataset rather than from the goal distribution π. While learning O, we want to avoid learning F overestimated values. For doing so, we propose to use the τ-expectile regression[91] to learn O.

4.2.2 Expectile Regression: Definition

Because of the F being subject to outliers, we want to avoid querying the actual maximum of F—that would lead to overestimation, considering this maximum being an outlier. Instead, we prefer querying the τ best value, as per Kostrikov and al.[91] This means, for example, querying the top 70% of F score instead of the maximum. These 70% provide a better approximation of F's theoretical maximum because they are less subject to overestimation. This new F approximation, defined as the τ proportion of the original F, is called τ-expectile regression.

Expectile regression is formally defined by solving the optimization problem:

$$\arg\min_{m_\tau} \mathbb{E}_{x\sim X}\left[L_2^\tau(x - m_\tau)\right]$$

where the loss function $L_2^\tau(u)$ is defined as:

$$L_2^\tau(u) = |\tau - \mathbf{1}(u < 0)|u^2$$

In this formulation, $u = x - m_\tau$ represents the deviation of a data point x from the regression target m_τ, while $\mathbf{1}$ is the indicator function, returning 1 if $u < 0$ and 0 otherwise. The parameter $\tau \in [0, 1]$ controls the asymmetry of the regression, allowing more emphasis on over- or underestimation depending on its value. When $\tau = 0.5$, the method reduces to least-square regression, capturing the conditional mean of the data distribution.

By selecting an appropriate τ, the regression framework mitigates the influence of extreme values (often artifacts of noise or data sparsity) that could distort the computed maximum values of $F(s, a, z) \cdot z$.

4.2.3 Learning Function

Given the defined function $O(s, z)$ and using expectile regression, the learning framework seeks to minimize the following objective:

$$\mathcal{L}_O(O) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(O(s, z)z - \max_a F(s, a, z)^T z\right)\right],$$

where (s, a) represents samples from the dataset $\mathcal{D}$ and z is the reward latent representation ($z = B \cdot r_z$)

This objective mitigates the sensitivity to outliers inherent in direct maximization of $F(s, a, z)$. Doing so, we avoid overestimating outlier values in the function $F(s, a, z)$, ensuring improved stability and robustness in the learned function $O(s, z)$.

4.2.4 Overall Learning Algorithm

To summarize, we conjointly minimize these two losses:

$$\begin{cases} \mathcal{L}_{FB}(F, B) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\rho}\left[\left(F(s_t, a_t, z)^\top B(s') - \gamma O(s, z)B(s')\right)^2\right] \\ \qquad\qquad - 2\mathbb{E}_{(s_t,a_t,s_{t+1})\sim\rho}\left[F(s_t, a_t, z)^\top B(s_{t+1})\right] + \text{Constant} \\ \mathcal{L}_O(O) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(O(s, z)z - \max_a F(s, a, z)^T z\right)\right] \end{cases}$$

4.2.5 Additional B Regularization

Through experimentation, we found that introducing an auxiliary loss, originally designed to mimic the Q-loss (defined in Equation (4.7)) as proposed in the original work[16], significantly improves training stability when **reformulated to jointly regularize both F and B**.

Expression of B within the Q-Loss. The *auxiliary loss* proposed in the original paper is expressed as:

$$\mathcal{L}'(F) \coloneqq \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\rho}\left[\left(F(s_t,a_t,z)^\top z - r_{\text{implicit}}(s_{t+1}) - \gamma \bar{F}(s_{t+1},\pi_z(s_{t+1}),z)^\top z\right)^2\right] \tag{4.7}$$

Incorporating this Q-loss requires the definition of a reward function, which is not provided during our reward-free training. However, it can be inferred from B and z, following Touati and al.[16]:

$$r_{\text{implicit}}(s') = B(s') \cdot \Sigma_B^{-1} \cdot z \tag{4.8}$$

where $\Sigma_B = \frac{1}{N} B(s')^T B(s')$ and $\Sigma_B^{-1} = \left(B(s')^T B(s')/N\right)^{-1}$

Regularization of B. While the original Q-loss is primarily designed to optimize F by focusing training on the diagonal case $z' = z$, our findings suggest that modifying this loss to directly regularize B—i.e., without detaching the reward—substantially enhances training stability. This modification implicitly influences the covariance structure of B, which, while difficult to formally characterize, appears to introduce beneficial constraints on its representation.

Although the precise mathematical mechanism underlying this improvement remains complex due to the expression of r_{implicit} as a function of B including an inverse covariance term, our empirical results demonstrate that this additional regularization leads to more stable learning.

4.3 Summary

In this chapter, we explored the impact of different regularization strategies on FB learning, focusing on their effect on stability, training efficiency, and overall policy performance. Our findings show that incorporating regularization on top of Double Q-learning

can enhance performance, but the improvements are not always consistent across different datasets and task complexities.

A key insight from our work is that directly applying regularization to single Q-learning yields superior performance compared to Double Q-learning, particularly in large-scale datasets. Since larger datasets inherently provide better coverage of the state-action space, they require less conservative regularization, allowing single Q-learning with regularization to achieve both stability and improved learning efficiency. Our results indicate that this approach leads to faster convergence and greater training stability. However, one trade-off is that while it enhances overall robustness, it may slightly reduce peak performance compared to methods that focus on maximizing the best observed reward—a common practice in some studies when reporting results.

To validate the effectiveness of our proposed method, we conducted extensive evaluations across different datasets, ranging from simple to highly complex environments, as well as across varying dataset sizes. This allowed us to demonstrate how our approach scales and adapts to different learning scenarios. Additionally, we benchmarked our regularization method on the original FB framework and our enhanced FB architecture against previously proposed regularization techniques, showing that our enhanced architecture consistently outperforms existing approaches in terms of stability and efficiency.

Our findings suggest that effective regularization, when carefully designed, can significantly improve offline RL performance without unnecessary conservatism.

CHAPTER 5 COMPLETE DESIGN: FB-PERL FRAMEWORK

5.1 Neural Network Architecture

In this section, we present our final neural network architecture and its evolution from the original FB architecture. The primary objective was to improve the learning of environment representations while keeping the actor architecture unchanged. The modifications therefore primarily focus on enhancing the forward network F learner and the backward representation network B learner. Below, we describe the initial architecture and the characteristics of the new architecture.

5.1.1 Architecture Evolution

Original Architecture. The original architecture, denoted as the FB architecture, consists of multiple feedforward neural networks with a single or double hidden layers[16,19].

Forward Network $F(s, a, z)$: The network takes as input the state s, action a, and a latent variable z. The inputs are preprocessed using two separate feedforward networks with one hidden layer containing 1024 units, each reducing the dimensionality to a 512-dimensional space. The outputs are concatenated and passed through two additional feedforward network heads, each with a single hidden layer of 1024 units, ultimately producing a d-dimensional vector. The F learner neural network is illustrated in Figure 5.1.

Backward Representation Network $B(s)$: The backward network B is a feedforward network with two hidden layers, each comprising 256 units. It receives a state s as input and outputs a d-dimensional embedding. The embedding is L2-projected onto a sphere of radius d and then batch-normalized to ensure that $E_{s\sim\rho}[B(s)] = 1$, ensuring invariance to reward translation. The B learner neural network is illustrated in Figure 5.1.

Policy Network $\pi(s, z)$: Similar to the forward network, the policy network processes state s and latent variable z using two separate feedforward networks with one hidden layer (1024 units each). The results are concatenated and passed through a single hidden layer feedforward network with 1024 units. A Tanh activation is applied to the final output to map actions within the action space $[-1, 1]^{d_A}$, with d_A the dimension of the action space.

The architecture is illustrated in Figure 5.1, providing a visual representation of the original FB architecture.

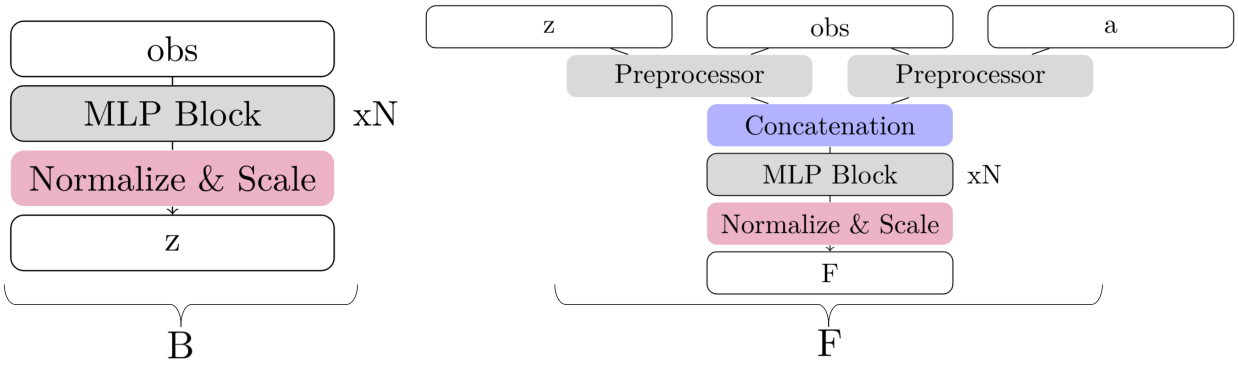


Figure 5.1 Original FB neural network architecture

FB-PERL Architecture. To address the limitations of the original architecture, we implemented a Transformer-based neural network architecture. This shift enhances the model's ability to capture long-range dependencies and improves its generalization capacity. If we do not modify F preprocessors, the previous MLP-based networks in F and B were replaced with Transformer-based networks, which inherently excel in capturing complex patterns within the data. The new FB architecture is depicted in Figure 5.2.

5.1.2 Implementation Choices

For all the architectures, we apply a layer normalization and `Tanh` activation in the first layer in order to standardize the states and actions. We switch from using `ReLu` to `GeLu` for the MLP Layers within Transformers. We also pre-normalize $z : z \leftarrow \sqrt{d}\frac{z}{\|z\|_2}$ in the input of F, π and ψ after sampling from a normal distribution to make sure that the metric space of sampled z-vectors aligns with the codomain of B.

Additionally, dropout regularization is applied within the forward network F to prevent overfitting, while the backward network B omits dropout due to the inherent noisiness of its data. Contrary to MLPs in RL context – that scale in width, but are challenging to scale in depth – our Transformer-based framework empirically demonstrates superior scalability along the depth axis. While F network performance exhibits computationally

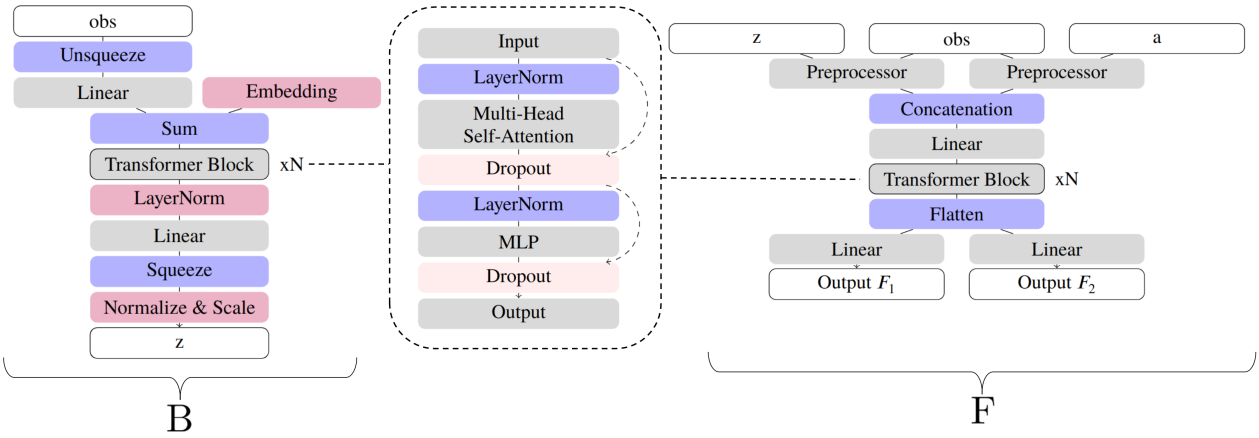


Figure 5.2 FB-PERL architecture

efficient scaling with added layers, component F manifests diminishing marginal utility beyond five layers, which is attributable to overfitting. For balancing performance and computational efficiency, we settled on a three-layer Transformer architecture for F and a two-layer Transformer for B.

5.1.3 Neural Network Hyperparameters

Table 5.2 summarizes the key hyperparameter changes between the original and new architectures. Bolded values represent adjusted parameters. Notable changes include:

- **Increasing the hidden dimension of F and B to 512** to enhance representational capacity while keeping reasonable computing time. Further increases lead to significantly reduced learning. As an example, we observe a x4 slowdown of the whole training for a single modification of B hidden dimension to 2048.
- **Introduction of Transformer heads in F and B** with a count of 8 for B and 4 for F, providing the ability to model complex interactions. The number of heads is designed to capture the more complex information while avoiding redundancy. While increasing the number of heads from 1 to 2 or 4 brings improvement to F and B performances, further increase, for example, to 16 for B, does not provide any additional benefit, or even slightly reduces the performance.

5.2 Final Algorithm

5.2.1 Summary of the Changes

Our proposed framework FB-PERL consists of two algorithms based on the original FB framework. **Algorithm 1**: A simplified version using single Q-learning. **Algorithm 2**: An extended version that builds upon Algorithm 1 by incorporating a regularization term for the backward representation B and introducing a new representation network for the $F \cdot z$ model O. Algorithm 1 proves effective on large and diverse datasets, where ample data helps correct learning errors, whereas Algorithm 2 targets the performance challenges that arise in smaller datasets, where OOD errors are more pronounced. The additional regularization enhances generalization and robustness, while requiring more computational resources.

Our FB-PERL framework algorithm proposal relies on three innovations. We first use **Single Q-Learning** instead of the original author double Q-learning to reduce unnecessary conservatism. For small datasets, we also introduce **Backward Network Regularization** by introducing a regularization term to mitigate OOD errors, especially in smaller datasets (cf. subsection 4.2.5). In large environments, we relax this guidance, as greater freedom in B representation leads to better performance. Finally, for small datasets, we implement an **OOD Representation Network** that learns to approximate the forward network F for better OOD handling.

5.2.2 Our Proposal

The simple algorithm FB-PERL-light is as follows:

Algorithm 5.1 FB-PERL-light

Require: Replay buffer $\mathcal{D}$, Polyak coefficient α, ν a probability distribution over $\mathbb{R}^d$, randomly initialized networks F_θ and B_ω, learning rate η, mini-batch size b, number of episodes E, number of gradient updates N, temperature τ, and regularization coefficient λ.

for $m = 1, \dots, M$ **do**
  /* Perform N stochastic gradient descent updates for $\mathcal{L}_{FB}$ */
  for $n = 1, \dots, N$ **do**
    Sample a mini-batch of transitions $\{(s_i, a_i, s_{i+1})\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$
    Sample a mini-batch of target state-action pairs $\{(s'_i, a'_i)\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$
    Sample $z \in I \sim \nu$
    Set $\pi_{z_i}(a \mid s_{i+1}) = \text{softmax}(F_\theta(s_{i+1}, a, z_i)^\top z_i / \tau)$
    $\mathcal{L}(\theta, \omega) \leftarrow \frac{1}{b} \sum_{i \in I} \left(F_\theta(s_i, a_i, z_i)^\top B_\omega(s'_i, a'_i) - \gamma F(s_{i+1}, \pi(s_{i+1}), z_i)^\top B_\omega(s_{i+1}, a) \right)^2 -$
    $\frac{1}{b} \sum_{i \in I} F_\theta(s_i, a_i, z_i)^\top B_\omega(s_i, a_i)$
    /* Compute orthogonality regularization loss */
    $\mathcal{L}_{\text{reg}}(\omega) \leftarrow \frac{1}{b} \sum_{i \in I} B_\omega(s_i, a_i)^\top \text{stop-gradient}(B_\omega(s'_i, a'_i)) \cdot$
    $\left(\frac{1}{b} \sum_{j \in I} B_\omega(s_j, a_j)^\top \text{stop-gradient}(B_\omega(s'_j, a'_j)) \right)$
    Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \omega)$ and $\omega \leftarrow \omega - \eta \nabla_\omega (\mathcal{L}(\theta, \omega) + \lambda \cdot \mathcal{L}_{\text{reg}}(\omega))$
  end for
  /* Update F, B target network parameters */
  $\theta^- \leftarrow \alpha \theta^- + (1 - \alpha)\theta$
  $\omega^- \leftarrow \alpha \omega^- + (1 - \alpha)\omega$
end for

For taking OOD regularization into account, we propose FB-PERL-regularized. The changes from algorithm 5.1 to Algorithm 5.2 have been highlighted in green for added part, and red for modified part.

Algorithm 5.2 FB-PERL-regularized

Require: Replay buffer $\mathcal{D}$, Polyak coefficient α, ν a probability distribution over $\mathbb{R}^d$, randomly initialized networks F_θ and B_ω, learning rate η, mini-batch size b, number of episodes E, number of gradient updates N, temperature τ, and regularization coefficient λ.

for $m = 1, \dots, M$ **do**
  /* Perform N stochastic gradient descent updates for $\mathcal{L}_{FB}$ */
  for $n = 1, \dots, N$ **do**
    Sample a mini-batch of transitions $\{(s_i, a_i, s_{i+1})\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$
    Sample a mini-batch of target state-action pairs $\{(s'_i, a'_i)\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$
    Sample $z \in I \sim \nu$ of size $|I| = b$
    Set $\pi_{z_i}(a \mid s_{i+1}) = \text{softmax}(F_\theta(s_{i+1}, a, z_i)^\top z_i / \tau)$
    $\mathcal{L}(\theta, \omega) \leftarrow \frac{1}{b} \sum_{i \in I} \left(F_\theta(s_i, a_i, z_i)^\top B_\omega(s'_i, a'_i) - \gamma O(s_{i+1}, z_i)^\top B_\omega(s_{i+1}, a) \right)^2 -$
    $\frac{1}{b} \sum_{i \in I} F_\theta(s_i, a_i, z_i)^\top B_\omega(s_i, a_i)$
    /* Compute orthogonality regularization loss */
    $\mathcal{L}_{\text{reg}}(\omega) \leftarrow \frac{1}{b} \sum_{i \in I} B_\omega(s_i, a_i)^\top \text{stop-gradient}(B_\omega(s'_i, a'_i)) \cdot$
    $\left(\frac{1}{b} \sum_{j \in I} B_\omega(s_j, a_j)^\top \text{stop-gradient}(B_\omega(s'_j, a'_j)) \right)$
    Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \omega)$ and $\omega \leftarrow \omega - \eta \nabla_\omega (\mathcal{L}(\theta, \omega) + \lambda \cdot \mathcal{L}_{\text{reg}}(\omega))$
    /* Compute F and B regularization loss */
    $inv\_cov \leftarrow \left(\frac{1}{b} B_\omega^\top B_\omega \right)^{-1}$
    implicit reward $\leftarrow (B_\omega \cdot inv\_cov) \cdot z_i$
    targets $\leftarrow$ implicit reward $+ \gamma \cdot \text{detach}(O(s_{i+1}, z_i)^\top \cdot z_i)$
    $\mathcal{L}_{reg} \leftarrow \frac{1}{b} \sum (F_\theta(s, a, z_i) \cdot z_i - \text{targets})^2$ *// B regularization through implicit reward*
  end for
  /* Update F, B target network parameters */
  $\theta^- \leftarrow \alpha \theta^- + (1 - \alpha)\theta$
  $\omega^- \leftarrow \alpha \omega^- + (1 - \alpha)\omega$
  /* Perform N_z stochastic gradient descent updates for $\mathcal{L}_O$ */
  for $n = 1, \dots, N_z$ **do**
    Sample a mini-batch of transitions $\{(s_i, a_i, s_{i+1})\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$
    Sample a mini-batch of target state-action pairs $\{(s'_i, a'_i)\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$
    Sample a mini-batch of $\{z_i\}_{i \in I} \sim \nu$ of size $|I| = b$
    $\mathcal{L}_\mathcal{O}(\phi) \leftarrow L_2^\tau \left(O(s, z) z - \max_a F(s, a, z)^T z \right)$
    Update $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_\mathcal{O}(\phi)$
  end for
end for

In our implementation, we fixed N_z to 1, as it leads to the best performance. The overall changes to the algorithm lead to the extra training of one variable 0 and the design of one extra hyperparameter τ. Our experiments on the RND dataset and walker environment show that the two regularized and non-regularized algorithms asymptotically converge to the same value on the big datasets. FB-PERL-regularized can still be used in large datasets; however, the learning of an extra representation slows down the training, leading to a slower convergence (more than 10M iteration before convergence for the regularized training, against 200k iterations for the non-regularized one). By introducing two

algorithms tailored to large and small datasets, we achieve an effective balance between computational efficiency and performance.

5.2.3 Implementation Design Choices

In designing our regularization approach, we prioritize simplicity and scalability. Cetin and al.[82] highlight the need for effective regularization without excessive conservatism while cautioning against overly complex algorithms that hinder practical deployment. Based on these principles, we structure our design to remain efficient and mathematically grounded.

To determine the optimal model configuration, we identify several key aspects that require experimental evaluation:

Architectural Separation of O. A critical consideration is whether the OOD network O should be implemented as an entirely independent network with its own optimizer, following the actor-critic paradigm, or integrated within the FB representation. Our experiments indicate that a shared structure, in which O is part of the FB class, is profitable for both computational efficiency and network stability. Moreover, code maintainability and readability are improved through this integrated design.

Neural Network Architecture for O. Another key question is whether the O network should inherit structural components from F, particularly its preprocessing layers, to enhance representation learning. Our findings show that sharing preprocessors does not yield performance improvements. We implement O learner using a basic MLP structure, which provides adequate performance with minimal computational overhead.

There are two main reasons that explain why a simplified representation network for O is effective experimentally. First, because of its **lower dimensionality:** Unlike F, the O network does not require action-dependent inputs, resulting in a reduced dimensional space. Since it models the maximum of F, the function O does not need to be as complex as B or F learners. An additional explanation can be found by drawing parallels with **knowledge distillation**. Similar to findings in NLP[142-144], where knowledge distillation enables simple architecture to learn complex representation by mimicking the output of complex models, O effectively mimics F outputs with a simpler network. This allows for efficient learning without sacrificing accuracy.

Hyperparameter Tuning. Choosing appropriate hyperparameters is essential to ensure model stability, fast convergence, and strong generalization. For OOD regularization, we apply a coefficient τ of 0.7, as often used by previous IQL-based works. While higher regularization coefficients can improve performance in some scenarios (for example, walker), they tend to completely degrade training performance in more complex or irregular tasks (for example quadruped).

DR3 and Feature Co-Adaptation. The DR3 regularizer, proposed by Kumar and al.[145], mitigates feature co-adaptation by reducing the dot product between consecutive state embeddings. While this regularizer has proven beneficial in supervised learning tasks, its utility in our setting is limited due to the absence of explicit rewards.
Additionally, since we work with a larger state representation space and rely on the M function instead of the traditional Q function, the concern about feature orthogonality is less pronounced. Our experimental results confirm that incorporating the DR3 regularizer does not yield significant performance gains. Therefore, we exclude it from our final algorithm.

5.3 Implementation Choices and Computing Performance Improvement

5.3.1 Score Improvement and Design Choices

To optimize the performance of our algorithm, we made several design choices focusing on stability, training efficiency, and computational robustness.

Unlike Cetin and al.[80], we chose to employ a deterministic actor instead of a Gaussian actor. This choice was motivated by empirical evidence showing superior performance in deterministic settings, as demonstrated by Touati and al.[19]

To improve performance, additional design adjustments included a learning rate scheduler, a weight initialization strategy, and an optimizer change. We employ CosineAnnealingWarmRestarts as the **FB learning rate scheduler**. Its cyclic adjustment of the learning rate helps the model escape local minima, promoting more stable and reliable convergence during training. A **weight initialization** function complements it by establishing optimal initial conditions: it breaks symmetry between neurons, ensures stable gradient flow, and aligns with activation function dynamics, collectively enabling efficient learning from the outset of training. The update of the **optimizer** from Adam to AdamW

better weight decay and gradient management, resulting in an improved convergence.

These cumulative improvements resulted in about 10% performance enhancements (measured by comparing the performance at the end of the training, after 10M steps) and better training stability, mostly brought by the use of AdamW.

The use of Transformer-based architectures in our work positions us to benefit from ongoing research on Transformer computation speed and performance optimization, reinforcing our architectural choices from both theoretical and implementation perspectives.

Building on the core improvements that enhance algorithmic performance and stability, we now focus on the complementary optimizations aimed at significantly improving computing efficiency and reducing training time.

5.3.2 Computing Speed Improvements

In addition to improving algorithmic performance, we implemented several optimizations to enhance computing efficiency. These changes aimed to reduce computation time without impacting model accuracy.

A significant improvement in our implementation was achieved through **profiling and code optimization**. By conducting detailed performance profiling, we identified computational bottlenecks and applied targeted optimizations to enhance execution speed. Additionally, we enabled **bfloat16** training, which substantially reduced computation time. The autocast bfloat16 has been widely used for experiments on the RND dataset with walker and quadruped domains. However, this optimization is applicable only in configurations without regularization, as low-precision computations are unsuitable for reliable matrix inverses. Therefore, it cannot be used together with our small dataset evaluation, which computes an inverse covariance within the regularization. To further enhance efficiency, we used PyTorch's **torch.compile** for Just-In-Time (JIT) compilation, which optimizes the model graph for faster runtime performance. Furthermore, we took advantage of TensorFloat32 (TF32) matrix multiplication on compatible GPUs, significantly speeding up large-scale matrix operations with minimal precision loss. These optimizations led to significant speedups without degrading model performance.

We experimented with the Muon optimizer, a recent adaptive optimizer, as a potential alternative to AdamW. This optimizer was designed to improve training speed and has proven being efficient for LLMs. However, it exhibited both lower performance and slower convergence, making it unsuitable for our use case.

The table below summarizes the relative speedups achieved through the applied op-

timizations:

Table 5.1 Speed Improvements (in multiplication factors) brought by the different optimizations

Optimization Step	**Speed**	**Notes**
Baseline (Normal)	1x	Standard training setup without any optimizations.
+ `torch.compile`	2x	Compilation for improved runtime performance.
+ TF32 Matrix Product	5x	Utilizes TensorFloat32 for matrix multiplications on compatible GPUs.
+ bfloat16	10x	Autocast to bfloat16, applicable without regularization.
- Double Q-Learning	Additional Gain	Elimination of Double Q-learning

While most of the improvements are significant, removing Double Q-learning only provides a minor speedup. In our tests, the transition from conservative Double Q-learning to conservative Single Q-learning increased the iteration rate from 17 iterations per second to 18.5 iterations per second — a modest but measurable 7% speedup.

Although this gain is smaller compared to other optimizations, it supports our decision to remove Double Q-learning when it does not yield additional accuracy benefits. This demonstrates the advantage of streamlining the algorithm when possible, particularly in scenarios where computational resources are constrained.

5.4 Hyperparameter Summary

In addition to neural network parameters, additional parameters have been added for computational efficiency options (such as allow_tf32 for efficient matrix product on cuda cores, or autocast to enable the use of bfloat16, cf. subsection 5.3.2), and for the proposed regularization (such as τ).

Hyperparameter	**Reference**	**Ours**
Latent dimension d	50	50
F hidden layers	2	**3**
F hidden dimension	256	**512**
B hidden layers	3	**2**
B hidden dimension	256	**512**
P_F hidden layers	2	2
P_F hidden dimension	1024	1024
P_π hidden layers	2	2
P_π hidden dimension	1024	1024
Std. deviation for policy smoothing σ	0.2	0.2
Truncation level for policy smoothing	0.3	0.3
Learning steps	1,000,000	1,000,000
Batch size	512	512
Optimizer	Adam	AdamW
Learning rate	0.0001	0.0001
Discount γ	0.98	0.98
Activations	ReLU	ReLU (GeLu in Transformers)
Polyak coefficient τ	0.01	0.01
z-inference steps	10,000	10,000
z mixing ratio	0.5	0.5
Orthonormalization coefficient	1	1
New Parameters	**Reference**	**Ours**
Transformer heads (B)	-	8
Transformer heads (F)	-	4
regularization τ	-	0.7
regularization	-	true (in small environments)
allow_tf32	-	True
autocast	-	None

Table 5.2 Hyperparameter comparison (changes in **bold**)

CHAPTER 6 NUMERICAL EXPERIMENTS

6.1 Experiments Objectives

In this chapter, we present an extensive empirical evaluation of our proposed enhancements to the FB framework for ZSRL. Our primary goal is to assess the **performance and scalability** of our approach across various environments and to validate the impact of our **novel architecture and regularization method**.

Experimental Goals. To rigorously evaluate the impact of these modifications, we structure our experiments to answer the following key research questions: Does our improved architecture lead to **higher average returns** compared to previous FB implementations and SOTA methods? How does our regularization technique affect **training stability** and **convergence speed**? How does the full model (FB + regularization) perform **compared to existing zero-shot RL baselines**, including PSM and HILPS? What is the performance contribution of each proposed enhancement?

Section Organization. The remainder of this section is organized as follows:

- **Section 6.2** describes our experimental setup, datasets, and evaluation protocol.
- **Section 6.3** presents a comprehensive evaluation of the full model (FB + regularization), including comparisons with previous methods from the literature.
- **Section 6.4** evaluates the individual impacts of our neural network and algorithmic modifications to FB.
- **Section 6.5** analyzes the efficiency of our regularization technique compared to out-of-the-shelves regularization methods.
- **Section 6.6** provides additional results and training curves for all experiments.
- **Section 6.7** summarizes our key findings.

In this section, we aim to provide clear and conclusive evidence of our method's effectiveness. We will therefore demonstrate its advantages in terms of **stability and performance** over existing zero-shot RL techniques.

6.2 Experiments Methodology

6.2.1 Tools and Libraries

We use Python as a language for our implementation, with PyTorch as the main deep learning framework. We built upon a codebase provided by Jeen and al.[79] for the FB framework, and reuse their FB, and conservative FB implementations for baselines.

6.2.2 Datasets

ExoRL Dataset. For evaluating our ZSRL method, we rely on the DeepMind Control Suite[146] environment, as it is widely used in the literature by similar works and provides a diverse set of agents and tasks. It is built upon the MuJoCo physics engine[147] and provides a set of continuous control tasks with a high degree of realism. The chosen simulation in these dataset agents are point mass maze, walker and quadruped, as shown in Figure 6.1. We use ExoRL[148] datasets, a state-of-the-art data generation method for offline RL.

Figure 6.1 ExoRL evaluation environments: Walker, quadruped, point mass maze. In the Maze domain (right), we show an example of an initial state (yellow point) and the 20 test goals (red circles). Images from Touati and al.[16]

For our evaluation, we use a collection of 3 environments and 12 tasks, as shown in Table 6.1.

Domain	**Eval Tasks**	**Dimensionality**	**Type**	**Reward**
Walker	stand, walk, run, flip	Low	Locomotion	Dense
Quadruped	stand, roll, roll fast, jump, escape	High	Locomotion	Dense
Point Mass Maze	reach top left, reach top right, reach bottom left, reach bottom right	Low	Goal-reaching	Sparse

Table 6.1 Domains and evaluation tasks

For convenience, we will later simply refer to point mass maze tasks as “top left”, “top right”, “bottom left”, and “bottom right”. In tables, we will refer to point mass maze as maze.

For comprehensive testing, we consider **two dataset sizes**: 100k transitions and 10M transitions, and three data collection algorithms: RANDOM, RND[149] and DIAYN[150]. The trajectories' datasets can be downloaded following the instructions in the GitHub repository of Yarats et al.[148]https://github.com/denisyarats/exorl. The dataset of walker diayn that we used, as provided by Jeen and al., contains only 8,928,000 transitions. We will use this dataset with its full number of transitions as full dataset, and report this result together with other 10M transitions datasets for simplicity.

6.2.3 Baselines

To evaluate the effectiveness of our proposed method, we conduct a comprehensive comparison against state-of-the-art approaches in ZSRL.

Table 6.2 summarizes the baselines considered in our study, highlighting whether the method is a derivative of the FB framework and detailing the specific improvements proposed by each.

Method	**Source**	**FB Derivative**	**Improvement over FB**
FB	[19]	NA	NA
PSM	[84]	Yes	Affine M expression, new training framework
HILPS	[17]	No	NA
VCFB/MCFB	[17]	Yes	add CQL type conservatism

Table 6.2 Baselines

6.2.4 Experimental Details

We conduct our experiments on an A800 GPU.

To ensure reproducibility, we provided in Section 5 comprehensive documentation of our method, including network architectures and optimization settings. More details and environment specifications can be found in the project repository. All random seeds, software versions, as well as the original code are as well recorded and made available in our code repository https://github.com/arwen-c/zero-shot-RL.

Hyperparameter Fine-Tuning. Hyperparameter optimization (HPO) is more difficult in RL than other machine learning fields. Most papers in the fields only tune two or

three hyperparameters using hyperparameter sweeps or grid searches[151]. The reason behind this includes the high variability of the results on the learning curve and the strong sensitivity to the random seed. Although HPO has been shown to be effective in online RL[151], it has not yet proven effective in offline RL. Therefore, we manually tune the hyperparameters for each environment and task.

We finetune the hyperparameters of FB models once for all domains by performing hyperparameter sweeps over the batch size, learning rates, representation dimensions, or layers type like dropout and normalization. We evaluate our model using its reward-based performance on downstream tasks for each domain. We select the hyperparameters that lead to the best averaged performance over each domain's tasks. We re-train the final FB models with the selected hyperparameters for five different random seeds

6.2.5 Evaluation Protocol

We follow the best practices in Deep Reinforcement Learning evaluation[152] for reporting our experimental results.

We conduct experiments across many configurations. We use 3 different environments (walker, quadruped, point mass maze), 4 different tasks per environment — specific to each environment, 2 different dataset sizes (small 100k transitions and large 10M transitions), $N = 5$ different random seeds and different dataset exploration algorithms: RND[149], DIAYN[150], RANDOM.

For demonstrating the performance of our method, we compare it to the state-of-the-art methods on the same datasets.

Because of the high computational cost of training, we choose not to run ablations and additional experiments on all seeds and over all configurations. The specific settings used for each experiment will be detailed together with the results in the following sections.

6.2.6 Performance Metrics

Interquartile Mean (IQM). The Interquartile Mean (IQM) is a robust measure of central tendencies that is less sensitive to outliers than the mean. It is defined as the mean of the middle 50% of the data, excluding the lowest and highest 25% of the data. For a random variable X with cumulative distribution function F_X, the IQM is formally defined as:

$$\text{IQM}(X) = \int_0^1 F_X^{-1}(\tau)d\tau - \int_0^{0.25} F_X^{-1}(\tau)d\tau - \int_{0.75}^1 F_X^{-1}(\tau)d\tau \tag{6.1}$$

Score Distribution. In addition, we provide score distributions, as suggested by Agarwal and al.[152] The score distributions $\hat{F}_X$ are plot of the fraction of runs above a certain normalized score:

$$\hat{F}_X(\tau) = \hat{F}(\tau; x_{1:M,1:N}) = \frac{1}{M}\sum_{m=1}^{M}\hat{F}_m(\tau) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}[x_{m,n} > \tau] \tag{6.2}$$

They allow us to visualize on a single plot different metrics like IQM and optimality gap as areas, as illustrated on Figure 6.2[152].

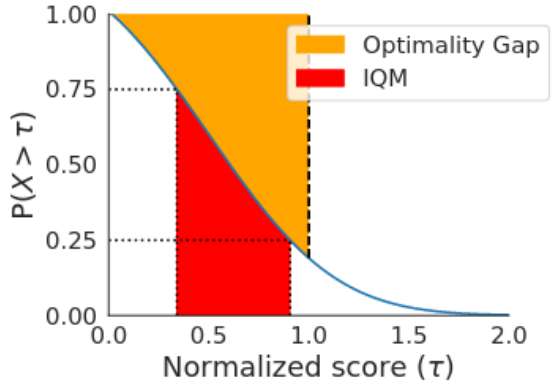


Figure 6.2 Score distribution interpretation, from Agarwal and al.[152] For a non-negative random variable X, IQM corresponds to the red-shaded region while an optimality gap corresponds to the orange shaded region

The τ value where the profiles intersect y = 0.5 shows the median, while for a non-negative random variable, the area under the performance profile corresponds to the mean.

Learning curves. We provide complete averaged learning curves for the most important results with confidence bands to illustrate the training dynamics of our models in Section 6.6. We also provide the non-averaged learning curves in Appendix B Section B.1.

6.3 Performance Evaluation

As running baselines are resource intensive (in terms of computing power, money, and time), we decided to rely on previous works baselines. As two of the most recent and performant methods (PSM and VCFB/MCFB) use different aggregations, we present our results compared to theirs in different tables.

Aggregated Performance

As a first overview of our method results, we plot the maximum IQM achieved by our method on the RND environment for the walker task, compared to previous state-of-the-art methods, presented in Figure 6.3. All results are averaged over five different seeds. The standard deviation at the average maximum performance is also reported.

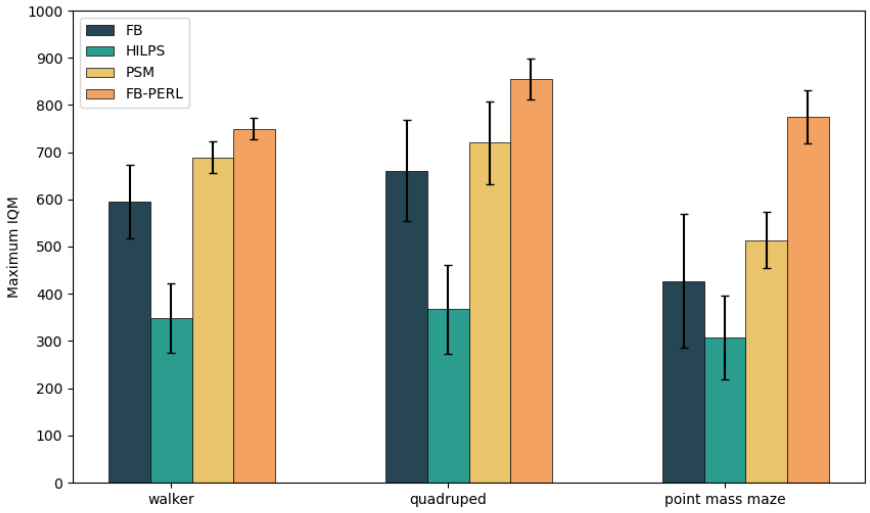


Figure 6.3 Maximum IQM performance achieved by the algorithms, trained on RND dataset in the walker environment on the three robots walker, quadruped, point mass maze after 10M steps.

Our method outperforms the state-of-the-art methods in all tasks. In addition to a higher performance, our method shows a lower variance than the baselines. The complete results of our method across all datasets can be found in Appendix in Table B.1. The training curves associated with these results can be found in Appendix 6.6.

Comparison with the State-Of-The-Art

Performance Comparison with PSM and HILPS. To provide a comprehensive evaluation of our model's performance, we present results across various environments and tasks in Table 6.3. In line with the evaluation protocol of PSM, the most recent and performant method at the time of writing, we compute the performance of our model using **five different random seeds** across **three distinct domains**. The evaluation datasets are generated with the RND policy [149].

It is important to note that recently published works, such as Farebrother et al. [153], which appeared less than a month prior to the submission deadline, have not been included

in our comparative analysis.

On the large-scale dataset comprising 10M transitions, our method demonstrates superior performance over existing state-of-the-art approaches across all evaluated environments and tasks. This highlights the effectiveness of our proposed scaling strategy, in contrast to the reliance on algorithmic refinements introduced in prior work such as PSM[84].

However, we acknowledge certain limitations in the standard reporting methodology. Specifically, current benchmarks often report the average of the highest scores achieved across tasks, rather than the highest average score over tasks. This practice can artificially inflate performance metrics and is statistically less robust than alternative approaches, such as reporting the maximum of the task-averaged performance. Due to constraints in computational resources, which prevent us from re-running baseline methods under standardized conditions, we adopt the same reporting convention as PSM for the sake of comparability.

Furthermore, the PSM authors do not provide detailed information on how their uncertainty intervals were calculated. Based on our analysis of their published results and on our experiment plots, we infer that they computed the error as the standard deviation over these best scores. Consequently, we follow the same procedure in our reporting. Nonetheless, this approach does not capture the variance inherent in the training process itself—a critical aspect that we discuss in detail in Section 6.3.

Additional performance data, including the full set of numerical results and training curves, is provided in Appendix 6.6.

Performance Comparison with VCFB/MCFB. Large Dataset Performance. As illustrated by our results on the large dataset (Table 6.4), our architecture slightly outperforms existing baselines overall, demonstrating competitive performance across multiple environments. In particular, our model achieves strong outcomes in the walker environment. However, performance on the quadruped task is slightly lower than that of the original FB architecture. This can be attributed to the increased complexity of the quadruped environment, which imposes higher demands on model regularization and stability. This can be mitigated by adding conservatism to the big-sized model; however, this conservatism would decrease the performance of other domains.

In environments such as DIAYN and especially the Random environment—where the datasets are inherently less diverse—our model, characterized by a higher number of parameters, encounters greater learning difficulty. This performance reduction is a mani-

Environment	Task	FB	HILP	PSM	FB-PERL
Walker	Stand	902.6±38.9	607.1±165.3	872.6±38.8	970.1±8.2
	Run	392.8±31.3	107.8±34.2	351.5±19.5	443.1±41.6
	Walk	877.1±81.0	399.7±39.3	891.4±46.8	935.9±15.5
	Flip	206.2±162.3	277.9±59.6	640.7±31.9	651.6±21.5
	Average	594.7	348.1	689.1	**750.4**
Quadruped	Stand	740.0±107.1	409.5±97.6	842.9±82.9	953.7±53.8
	Jump	581.3±107.4	325.5±93.1	596.4±94.2	755.7±34.8
	Average	660.7	367.5	719.6	**854.7**
Maze	Top Left	897.8±35.8	944.5±12.9	831.4±69.5	883.7±129.8
	Top Right	274.9±197.9	96.0±166.3	730.3±58.1	622.1±5.9
	Bottom Left	517.2±302.6	192.3±177.5	451.4±73.5	820.3±10.3
	Bottom Right	19.4±33.5	0.2±0.3	43.3±38.4	574.7±79.6
	Average	427.3	308.2	514.1	**775.2**

Table 6.3 Comparison of our FB averaged final performance over 5 seeds, with standard deviation

festation of the trade-off between expressiveness in data-rich settings and learning efficacy in data-poor or low-diversity environments. This explains a lower increase of our method performance compared to baselines in this comparison compared to the PSM/HILPS baselines that are only computed on the RND dataset.

Despite these challenges, the overall consistency of performance across complex domains underscores the adaptability and scalability of our architecture. A comprehensive breakdown of results by dataset type is provided in Appendix 6.6.

Small Dataset Performance. In the low-data regime, our model outperforms state-of-the-art baselines (Table 6.5). Our method continues to demonstrate solid performance in the walker environment and, more notably, in the quadruped environment—despite the increased task complexity, thanks to regularization. These findings affirm the model's capacity to extract meaningful behavioral patterns, even in limited-data contexts, provided the task structure allows for informative signal extraction. Because of the inability of all models to learn on point mass maze small size datasets, we choose not to present them.

All associated training curves supporting this analysis, along with detailed quantitative results broken down by dataset type, are included in Appendix 6.6.

Environment	Task	FB	VC-FB	MC-FB	FB-PERL
Walker	Walk	476.0	523.3	495.0	587.3
	Stand	574.7	549.7	548.3	730.0
	Run	139.7	178.7	176.7	219.7
	Flip	305.7	299.7	334.0	407.3
	Average	374.0	412.8	388.5	**486.1**
Quadruped	Stand	712.3	699.3	728.0	625.0
	Roll	449.3	619.3	651.3	438.0
	Jump	550.7	520.3	551.0	531.0
	Escape	56.3	46.3	61.7	57.3
	Average	442.2	471.3	**498.0**	410.3
Maze	Top Left	525.0	566.3	631.3	321.0
	Top Right	0.0	212.5	135.0	283.5
	Bottom Left	218.5	92.5	117.5	188.0
	Bottom Right	0.0	0.0	0.0	60.5
	Average	185.9	217.8	**220.9**	213.0
All	Average	334.0	367.3	369.1	**369.8**

Table 6.4 Comparison of FB, VC-FB, MC-FB, and FB-PERL averaged across DIAYN, RND, and RANDOM big datasets

Score Distribution

Following the methodology of Agarwal and al.[152], we provide the score distribution (or performance profile) of our method compared to the baselines (cf. paragraph 6.2.6 for explanations on the graph). We decided not to normalize the performance of our method to the maximum performance of the baselines, since the environment is similar, and we wanted to discuss the difficulty of the tasks. We used the IQM scores to plot the performance profile.

Environment	Task	FB	VC-FB	MC-FB	FB-PERL
Walker	Walk	118.0	277.0	209.0	214.3
	Stand	431.5	434.0	371.0	490.5
	Run	79.0	100.7	73.0	98.0
	Flip	134.3	205.7	147.0	187.7
	Average	190.7	**254.3**	200.0	247.6
Quadruped	Stand	290.3	360.3	267.0	480.7
	Roll	135.0	162.0	154.0	351.3
	Jump	186.3	182.3	172.3	233.0
	Escape	19.0	13.3	17.7	40.7
	Average	157.7	179.5	152.7	**276.7**
All	Average	174.2	216.9	176.35	**262.15**

Table 6.5 Comparison of FB, VC-FB, MC-FB, and FB-PERL averaged across DIAYN, RND, and RANDOM small datasets

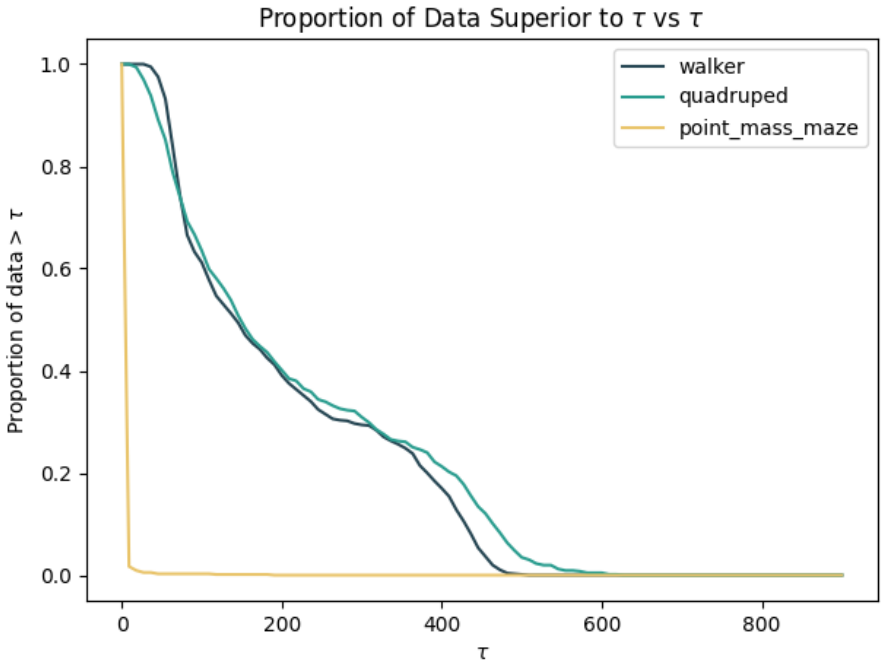


Figure 6.4 Performance profile on the small dataset

In both performance profiles (Figure 6.4 and Figure 6.5), the point mass maze environment emerges as the most challenging, as evidenced by the consistently low performance across all methods. In addition to low maximum scores, the average performance of our model—as well as that of the baselines— on point mass maze remains significantly low, with many simulations achieving zero rewards. This underscores the limitations of current

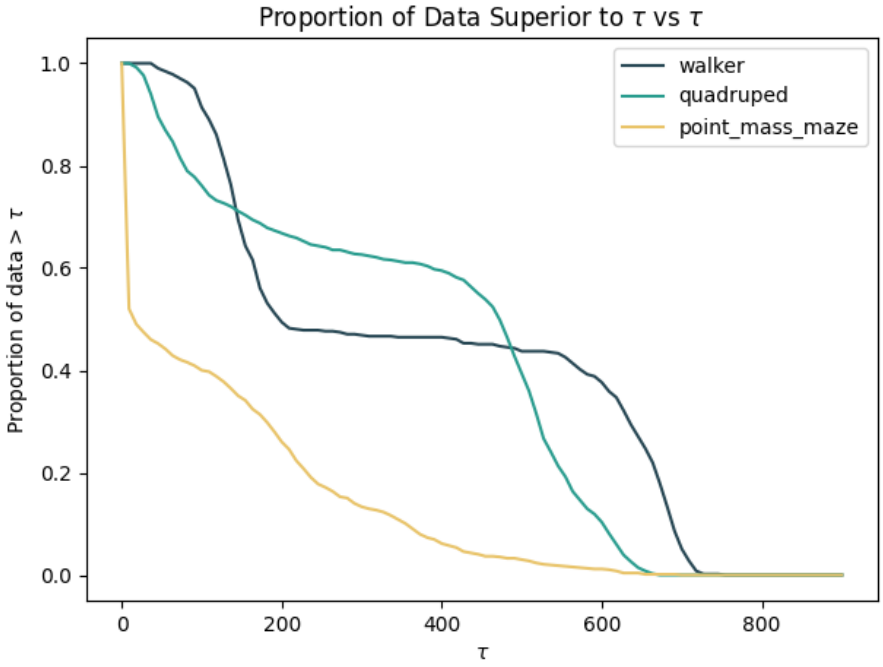


Figure 6.5 Performance profile on the full dataset

representation learning techniques in handling environments with sparse rewards.

In contrast, the quadruped and walker environments exhibit characteristic S-shaped training curves. These patterns correspond to performance jumps during training, with scores rising sharply from the 200-point range to peaks exceeding 400. This behavior mirrors the evaluation scores, which were measured every 200k training steps and demonstrate significant variance.

While some trajectories achieve high peak performance, it is important to note that the average performance in both small and big datasets remains noticeably lower. Although much of the current research landscape prioritizes maximizing peak performance, this variance poses a significant limitation for real-world deployment. In practical scenarios, it is rarely doable to halt training precisely at a performance peak, and relying solely on maximum-reported metrics may lead to overestimations of actual capability.

The variability of the IQM indicates a potential overstatement of real-world effectiveness unless such a variance is explicitly accounted for.

More extensive quantitative results and detailed training curves are available in Appendix 6.6.

6.4 Ablation Studies

To evaluate the contribution of each component in our architecture, we conducted a series of ablation studies comparing the performance of the full model against variants with individual components removed or altered.

The model configurations are defined as follows:

- **Original FB:** The baseline architecture from the original paper.
- **Original FB + X:** The original FB model with our newly proposed neural architecture for component X.
- **Single Q Learning with Original FB:** The original FB model modified to use single Q-learning instead of the default double Q-learning.
- **Single Q Learning + Conservatism:** The original FB with single Q-learning, augmented by our proposed conservative regularization.
- **FB-PERL:** Our final proposed method combining all improvements.

Ablation on the Walker Environment (Large Dataset). In the first ablation study, we evaluated all model variants on a dataset of 10^7 transitions from the RND policy, within the walker environment. As shown in Figure 6.6, performance metrics averaged over three random seeds are represented with colored bars (indicating standard deviation) and a black bar (representing the mean). Single-seed runs are shown as individual bars.

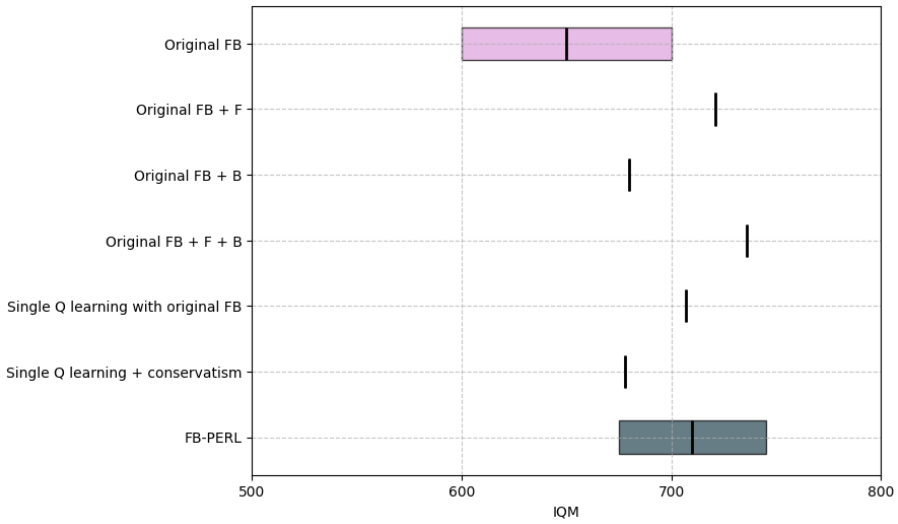


Figure 6.6 Ablation studies on the large dataset (10M transitions) for the walker environment

The results highlight the significant contribution of the new F and B components to overall performance. The F component, in particular, exhibits a more significant impact, and its design proved easier to optimize due to its more direct influence on learning. How-

ever, the combined performance gain from F and B is sub-additive, suggesting interaction effects.

As discussed, the addition of conservative regularization degrades performance in the walker large dataset setting because of unfinished training convergence. Given computational constraints and a focus on efficiency, our final model omits conservatism in large-scale training, and nonetheless demonstrates strong overall performance.

Generalization Across Environments. To test generalizability, we extended the ablation to include the walker, quadruped, and point mass maze environments. Results were averaged across environments and are reported using maximum IQM scores (Figure 6.7). The black bar indicates the mean performance, and the color bar indicates standard deviation.

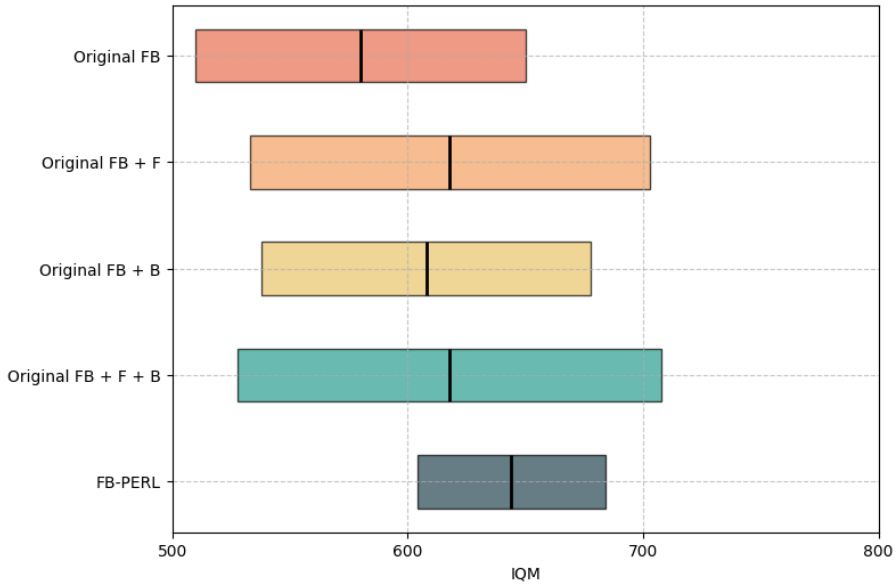


Figure 6.7 Ablation studies averaged over walker, quadruped, and point mass maze (10M transitions)

These findings confirm that, while the inclusion of both F and B Transformers does not drastically outperform F alone, the B architecture still demonstrates benefits in data-rich environments with minimal additional computational cost. This justifies its inclusion in the final model.

Ablation on the Walker Environment (Small Dataset). In a final set of ablation experiments, we focused on a smaller dataset of 10^5 transitions within the walker environment (Figure 6.8). Here, we emphasize the impact of conservative regularization in conjunction with architectural improvements. Performance is reported using the maximum IQM score

(black bar), and variance is quantified as the range between the highest and lowest score within the final 20% of training (colored bar).

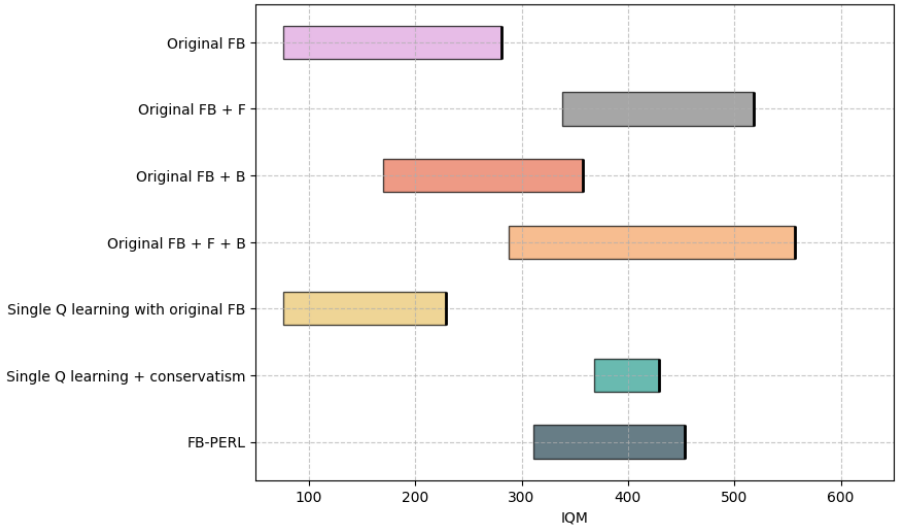


Figure 6.8 Ablation studies on the small dataset (100k transitions) for the walker environment

The results demonstrate that our architectural modifications yield performance gains even under data-constrained conditions. NN design choices have effect on variance, with the *original* $FB + F + B$ configuration exhibiting the highest variability. Switching to single Q-learning alone significantly deteriorates performance. However, combining it with conservative regularization not only mitigates variance but also markedly enhances performance. When paired with the full F and B architecture, this setup offers both stability and improved outcomes across environments. If the maximum score reached with full F and B architectural improvements is higher than with the improvements plus conservatism, it is because of a higher training variance. It results in a high maximum peak score at the start of the training, while the regularized model presents smaller peaks but more stable performance.

Of note, the single Q-learning variant showed instability in the walker environment, initially reaching a score of 345 within 100k steps but then declining to a range between 200 and 229 for the remainder of training.

The training curves corresponding to all ablation studies are provided in Section 6.6.

Training Evolution Analysis. To further compare architectural variants, Figure 6.9 presents the average performance of different models over the training process in the RND setting, aggregated over all tasks, domains, and dataset sizes.

Interestingly, the architecture using F as a Transformer and B as an MLP consis-

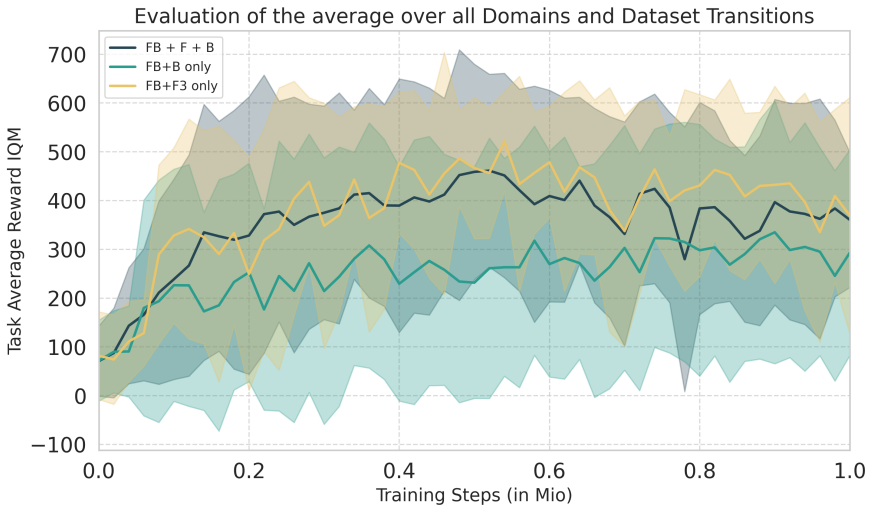


Figure 6.9 Average performance of different models throughout training

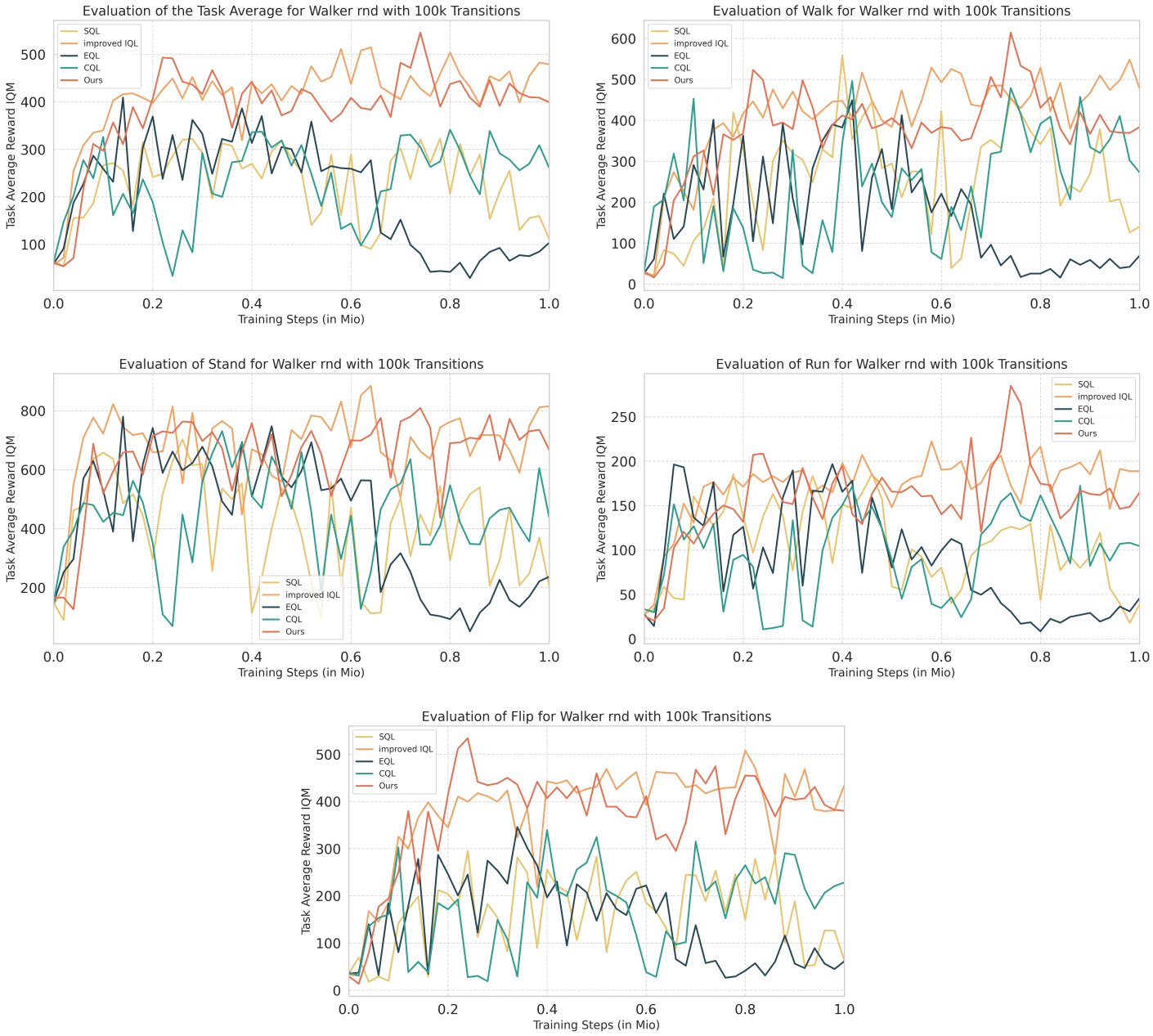
tently outperforms the full Transformer-based FB variant. This suggests potential overfitting issues when deploying Transformer architectures in low-data regimes. Nonetheless, when disaggregated by dataset size and environment, the full FB model proves superior in quadruped for the large dataset, while F-only performs better in the small dataset for quadruped. Performance in the walker environment remains similar across both configurations. Due to high variance and seed sensitivity, performance in the point mass maze environment is harder to assess conclusively.

Considering our primary objective—scaling to more complex and larger environments—we opt for the full FB architecture in our final model.

6.5 Conservatism Comparison

To further assess the robustness and generalization capacity of our proposed model, we compared our conservatism method to some of the best alternatives from the offline regularization literature.

Comparison of the Different Regularization Methods Implemented over FB-PERL, and the Original FB. To substantiate the rationale behind our regularization strategy, we evaluate the performance of several prominent offline regularization techniques from the literature, implemented within our FB framework. Specifically, we compare our approach against CQL[21,79], SQL, EQL[140] and a Q-function based version of our regularization named “improved IQL”. For clarity and improved readability, results for stan-



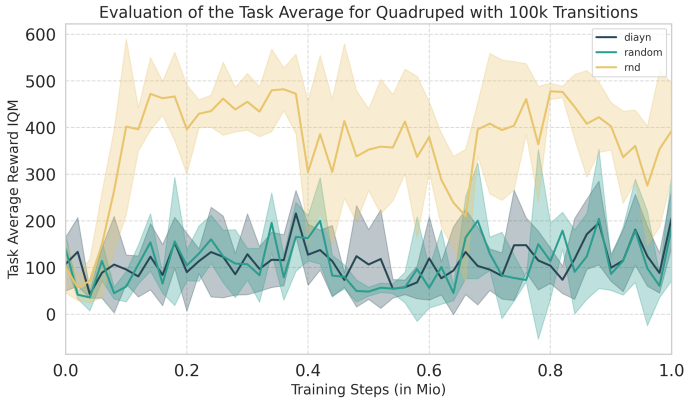
dalone IQL, which performed substantially worse than the other methods, are omitted from the figures.

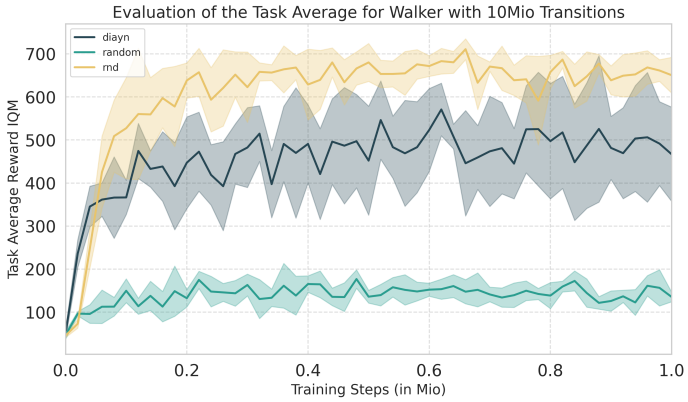
Experimental results on the small dataset demonstrate a clear performance advantage of the enhanced IQL variants—both our proposed method and IQL with $\boldsymbol{B}$ regularization—over other conservative strategies. Although both top-performing methods achieve similar final performance, our proposed regularization offers slightly faster training and lower memory usage by avoiding the need to compute Q-values. Therefore, the proposed regularization framework is designed to operate on M rather than Q.

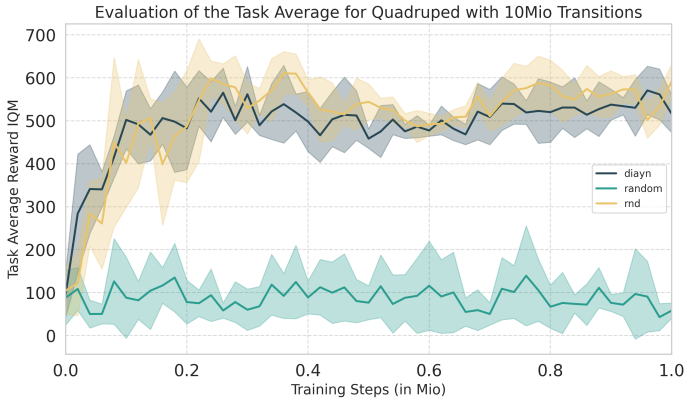
6.6 Detailed Results: Averaged Training Curves

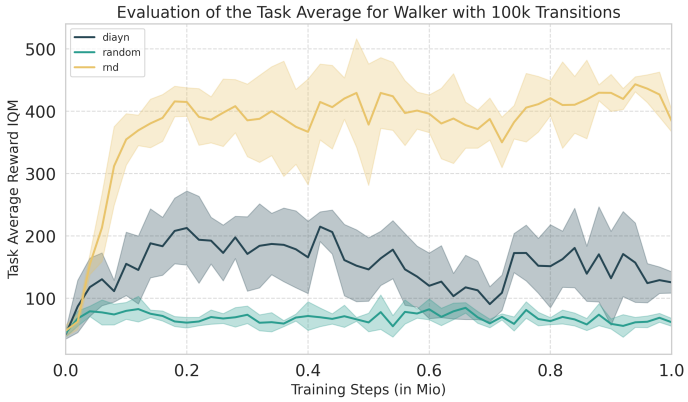
We present here below our raw results: the training curves of our model for the different tasks and datasets. During the training process, running for 1 million steps, we evaluate the performance of the model every 200,000 steps (50 evaluation steps). We average the results obtain over the five seeds.

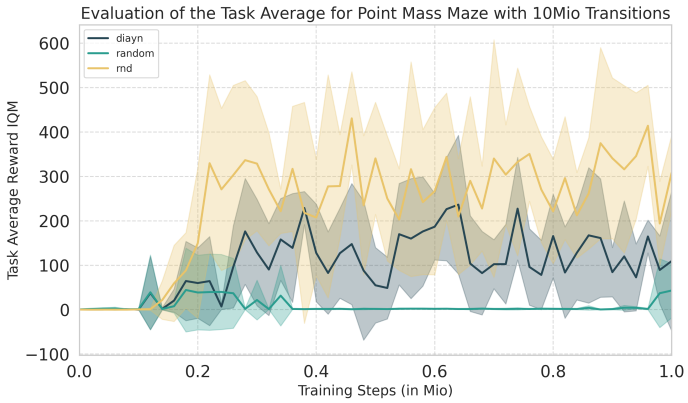
We report here below the averaged training curves of our model for the different tasks and datasets. The "Task Average" corresponds to the average performance over the four tasks used for evaluation for a given domain (walker, quadruped, point mass maze). The standard deviation is also reported as a shape, to show the variance of the training process. More details plots (non averaged) can be found in Appendix B.

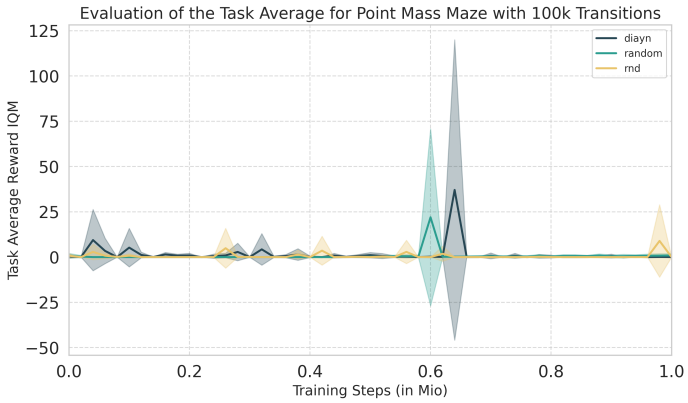

Evaluation of the Task Average for Quadruped with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

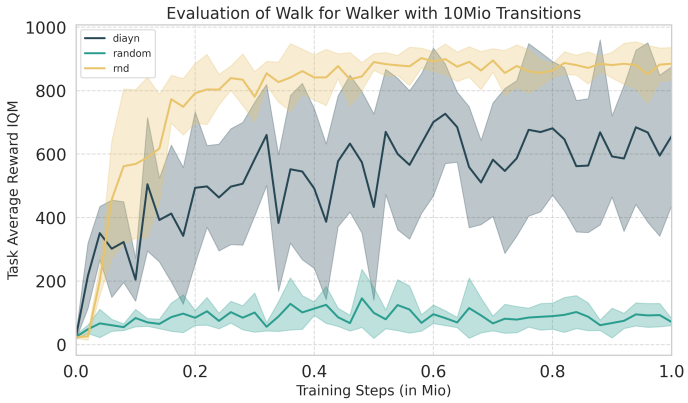

Evaluation of the Task Average for Walker with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

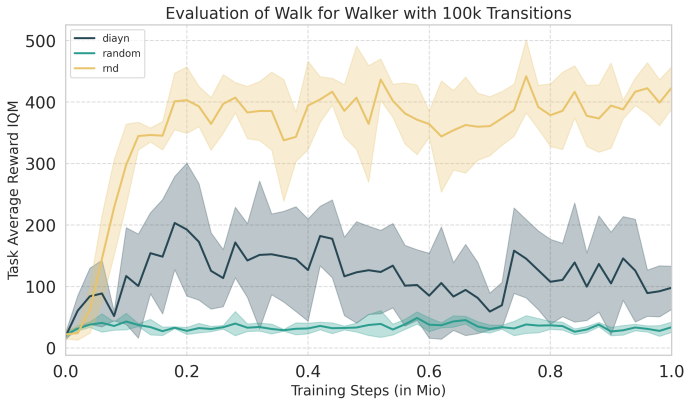

Evaluation of the Task Average for Quadruped with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

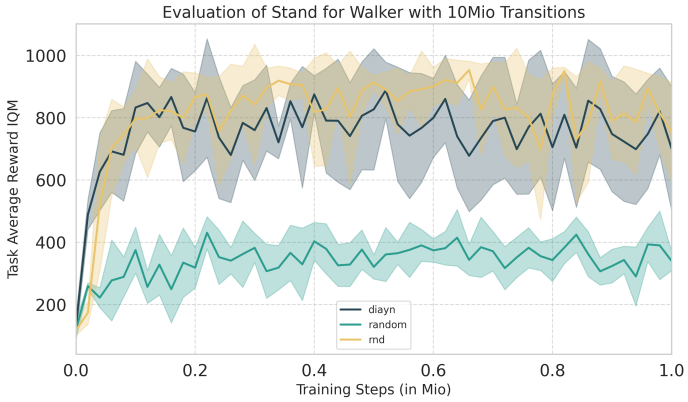

Evaluation of the Task Average for Walker with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

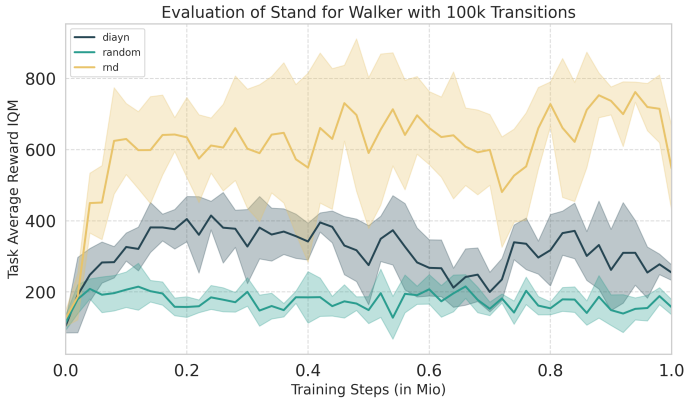

Evaluation of the Task Average for Point Mass Maze with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

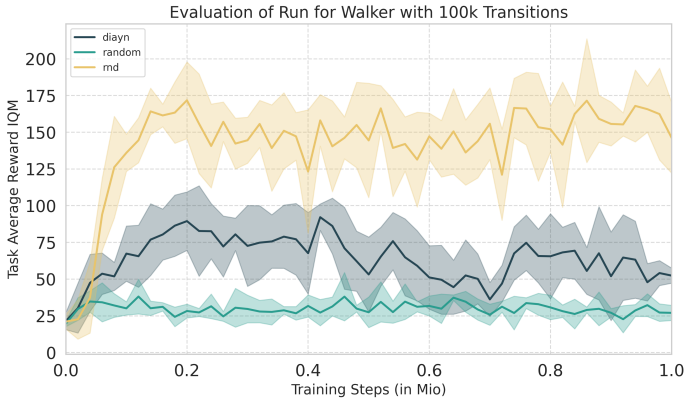

Evaluation of the Task Average for Point Mass Maze with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

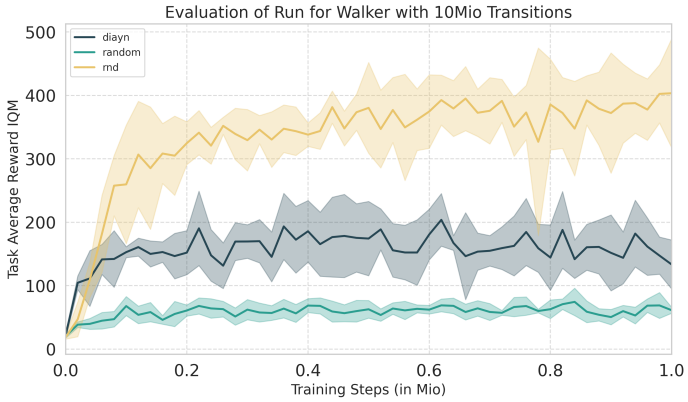

Evaluation of Walk for Walker with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

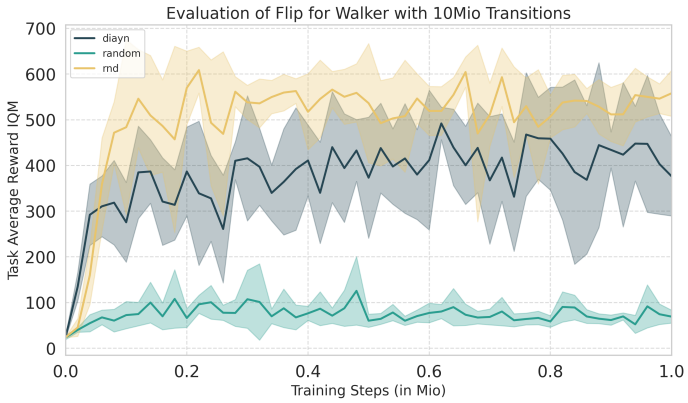

Evaluation of Walk for Walker with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

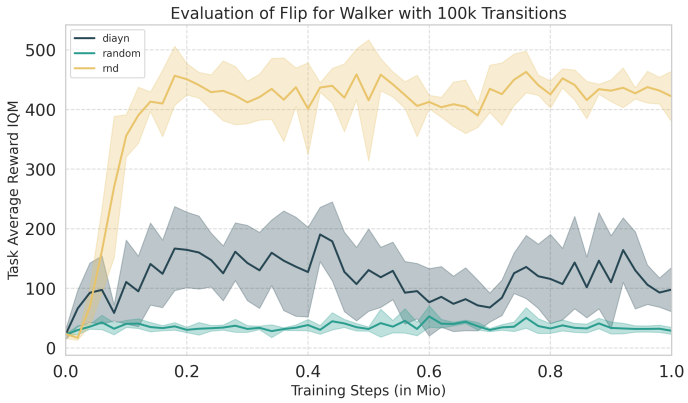

Evaluation of Stand for Walker with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

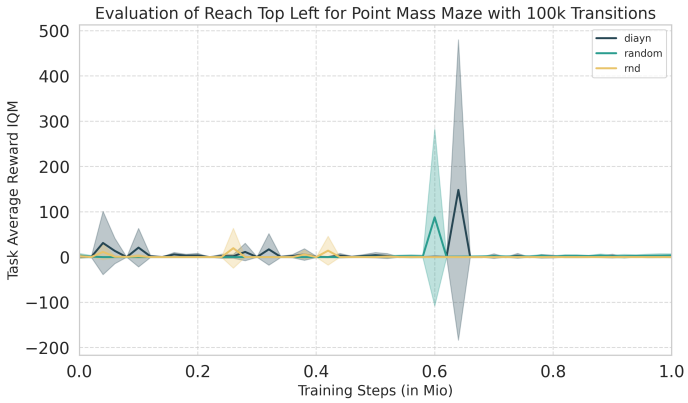

Evaluation of Stand for Walker with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

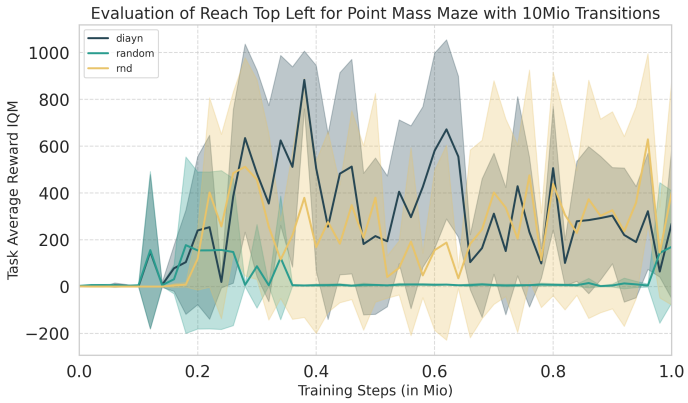

Evaluation of Run for Walker with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

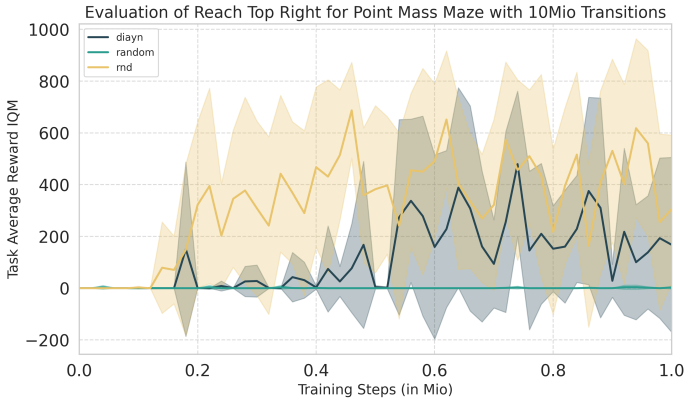

Evaluation of Run for Walker with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

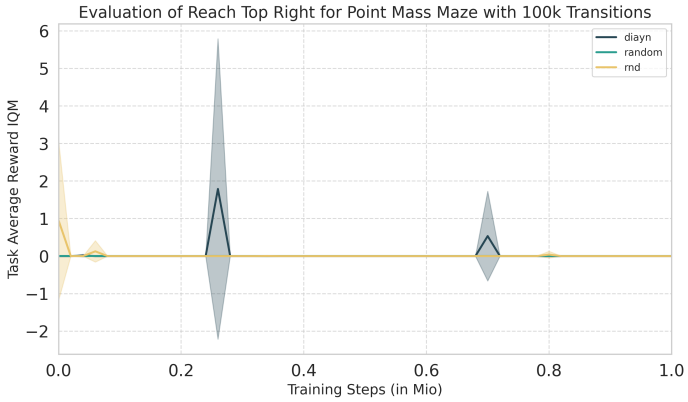

Evaluation of Flip for Walker with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

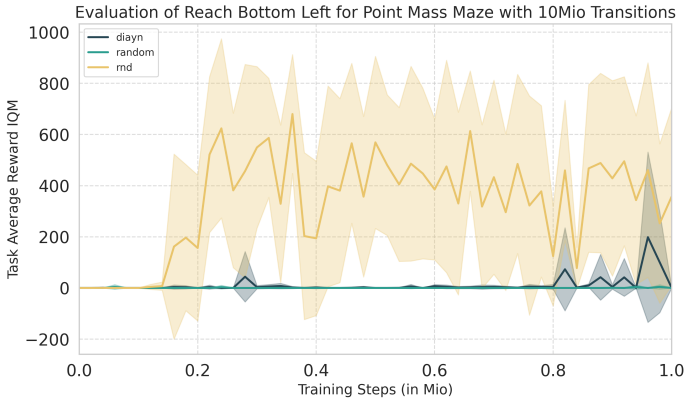

Evaluation of Flip for Walker with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

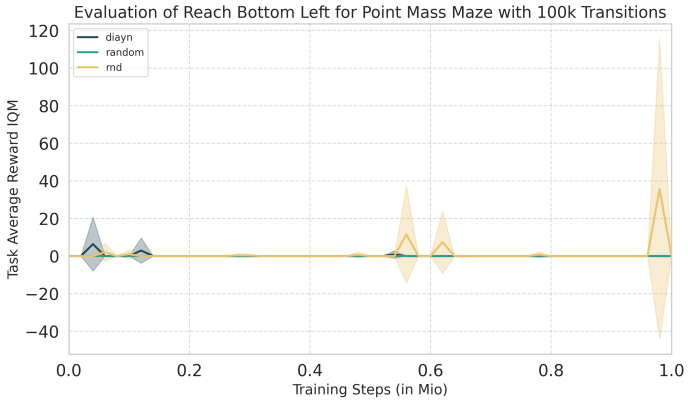

Evaluation of Reach Top Left for Point Mass Maze with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

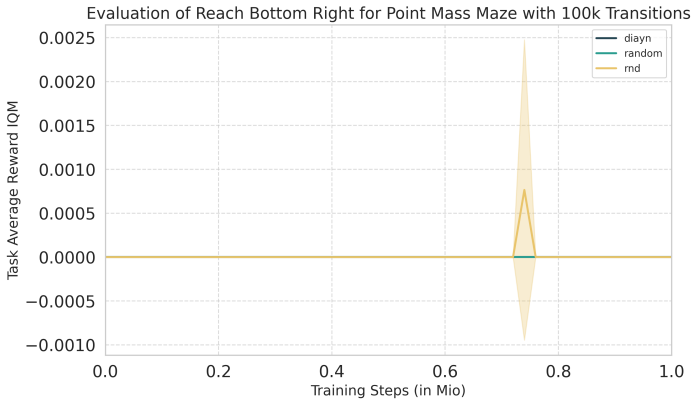

Evaluation of Reach Top Left for Point Mass Maze with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

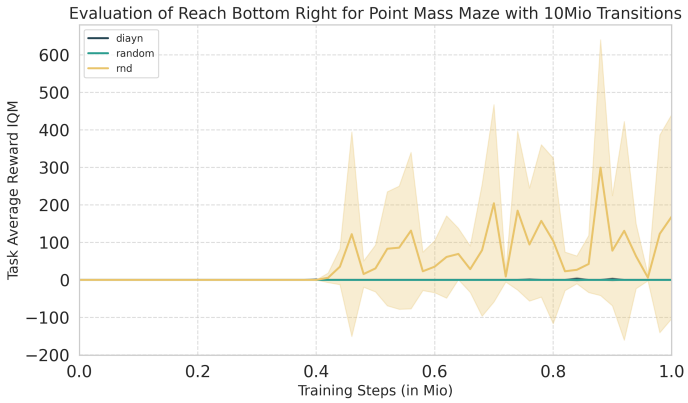

Evaluation of Reach Top Right for Point Mass Maze with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

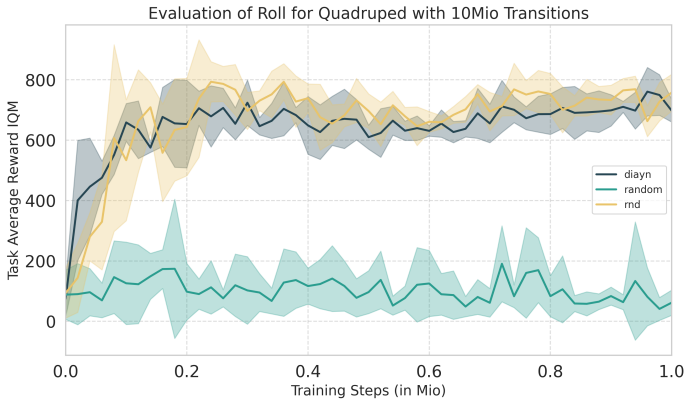

Evaluation of Reach Top Right for Point Mass Maze with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

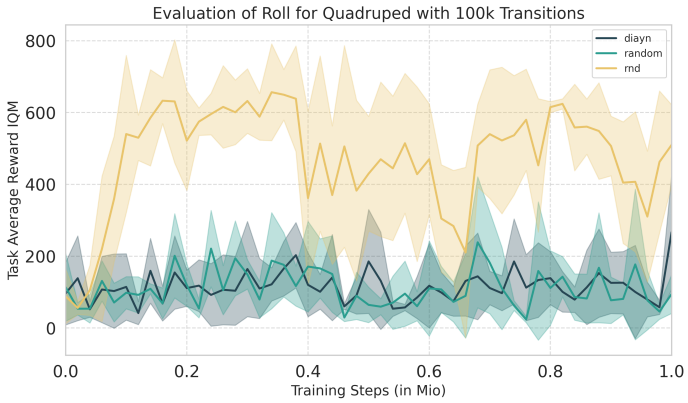

Evaluation of Reach Bottom Left for Point Mass Maze with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd

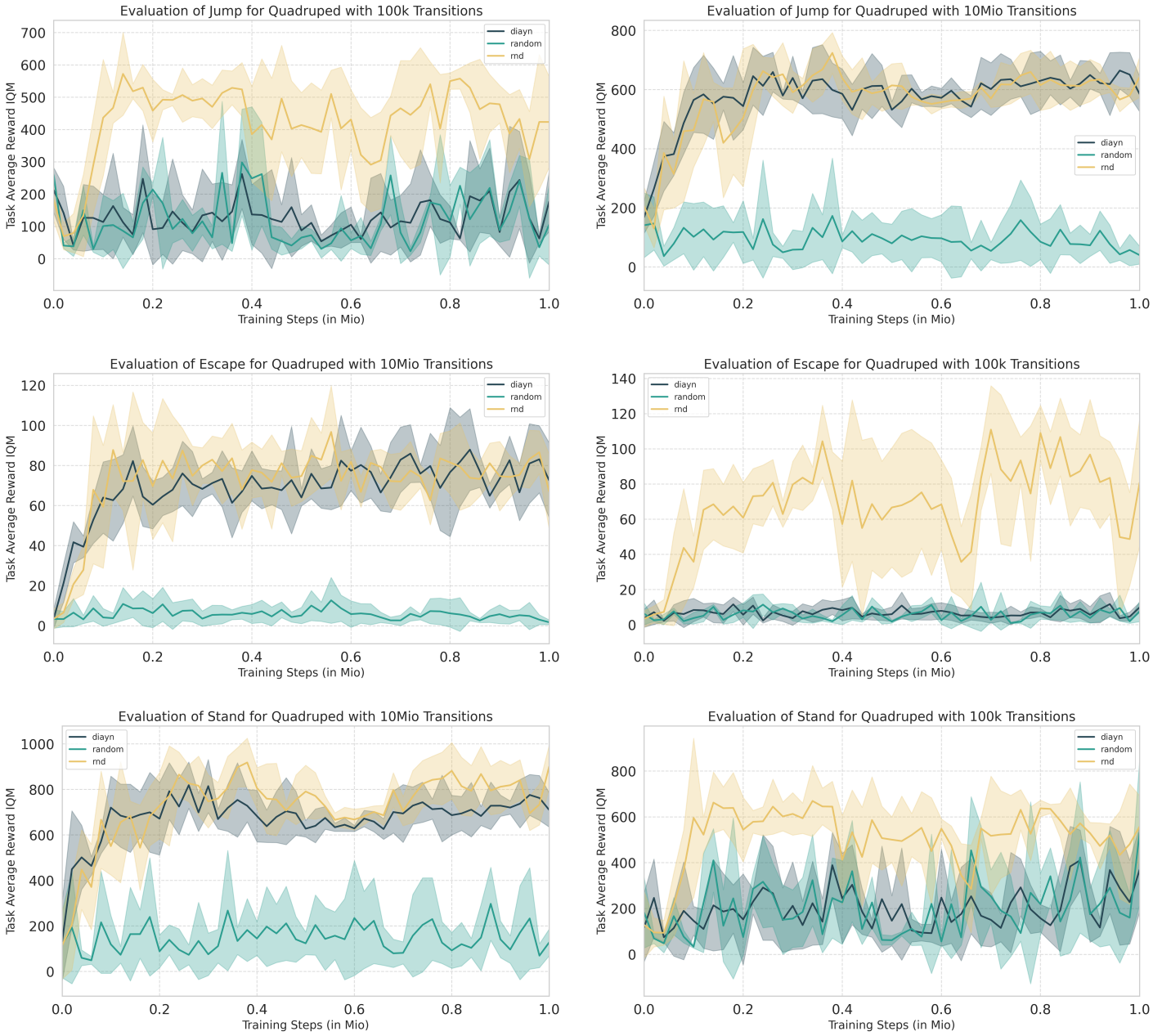

Evaluation of Reach Bottom Left for Point Mass Maze with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd


Evaluation of Reach Bottom Right for Point Mass Maze with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd


Evaluation of Reach Bottom Right for Point Mass Maze with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd


Evaluation of Roll for Quadruped with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd


Evaluation of Roll for Quadruped with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
diayn
random
rnd



6.7 Summary

Our empirical investigation demonstrates that the proposed architectural and regularization advancements deliver performance improvements across ZSRL benchmarks. The enhanced FB model, particularly when integrating Transformer-based F and B modules, exhibits superior expressivity, achieving state-of-the-art results in high-dimensional and data-rich environments such as walker and quadruped. Importantly, our framework not only surpasses existing baselines in average return but also displays enhanced stability and reduced variance in most large-scale settings.

Moreover, this performance increase comes together with a training time reduction from 17 to 11 hours for the 10M step training.

However, this improved expressivity comes at the cost of reduced generalization capabilities on poor datasets. Notably, while the Transformer-based architecture excels in large-scale settings, it tends to overfit or underperform in environments characterized by limited data diversity, such as the point mass maze or small-scale walker domains. This observation underscores a fundamental trade-off: higher architectural complexity and representational capacity enhance performance in expressive environments but compromise adaptability in constrained regimes. Similarly, regularization techniques, particularly conservative strategies, improve stability and robustness in low-data settings but may hinder performance or convergence speed in high-data scenarios if not appropriately tuned.

Overall, our method marks a meaningful step forward in the field of Zero-Shot Reinforcement Learning, offering a more expressive and computationally efficient training paradigm. By balancing architectural innovation with practical performance considerations, it provides a solid foundation for future advancements—whether as a standalone approach or as a building block within more complex, modular RL frameworks.

CHAPTER 7 CONCLUSION AND PERSPECTIVES

7.1 Summary of Contributions

In this work, we proposed two key enhancements to the FB framework, making significant progress toward a more practical and scalable approach to zero-shot Reinforcement Learning.

First, we developed an **expressive FB architecture** for reinforcement learning. This involved improving the neural network design using Transformers for the F and B models, leading to better representation, learning and generalization.

Second, we introduced a **novel regularization strategy** to mitigate out-of-distribution errors and stabilize training. Our contributions include an in-sample learning method inspired by Implicit Q-learning that ensures robust policy learning without excessive conservatism, and **B-regularization**, a stabilization technique that enhances training consistency and improves the B representation.

Our experimental results demonstrate that these improvements lead to significant gains in performance and efficiency. We have shown that our method can surpass previous methods by 9% to 34% IQM performance depending on the tasks on the ExoRL benchmark for the RND dataset. Moreover, we achieved improvements across most of the different datasets and tasks on the ExoRL benchmark compared to previous methods. Specifically, the neural network architecture enhancements alone accounted for most of the performance boost, while the regularization techniques contributed additional small gains coupled with more learning stability (lower variance). Additionally, the framework showed significantly faster convergence, reducing the number of training iterations required for stable performance. Implementation optimizations also resulted in reduced training time, improving computational efficiency.

Beyond numerical performance, our approach introduces a method that is **practical, scalable, and easy to use**, requiring minimal additional hyperparameter tuning compared to the original FB framework.

7.2 Theoretical Implications

Our work aligns with existing research in offline RL and zero-shot RL, introducing key insights into regularization strategies. Rather than contradicting prior findings, we questioned the necessity of excessive regularization. Through systematic experimentation, we validated and compared various theoretical insights that have been individually proposed in separate works.

Ultimately, our results emphasized that while regularization is crucial for offline RL, excessive conservatism can hinder performance, requiring a well-balanced approach to achieve both stability and adaptability.

7.3 Future Directions

While our work significantly enhances FB learning, several open challenges and areas for improvement remain.

Further performance enhancements are possible by developing methods to design optimal rewards for real-world applications where explicit reward functions are challenging to define. Improving the final policy extraction frameworks can also lead to better decision-making.

Since the methods are very sensitive to datasets, further results improvement needs to be combined with better dataset. In terms of dataset preprocessing, recent approaches such as those proposed by Mao et al.[154] and Li et al.[155] involve dataset augmentation and verification to mitigate the impact of out-of-distribution (OOD) values. Methods like DOGE[155] and SCAS[154] aim to align the distribution of new states with in-distribution data, reducing the effects of OOD states. Gathering large datasets in robotics applications is challenging, and the quality of the dataset is crucial for training. Simulation, and works on bridging the gap from simulation to reality[156-160] can help improve future performance.

Other improvements can be made by emphasizing on increasing the performance on sparse rewards. Indeed, if current works measure their model performance in the case of dense rewards, the model can theoretically be used in a sparse reward environment. Given that daily life situations like folding clothes or cooking are often sparse reward environments, it is important to push the boundaries of our model to work in these situations.

Regarding model scalability, further enhancements could be achieved by incorporating advancements in reinforcement learning, neural network architectures, and opti-

mization techniques. However, these improvements introduce additional computational complexity and trade-offs. A promising direction involves **distributional reinforcement learning**[161-163], which learns the full probability distribution of returns rather than just their expectation. This can enhance risk-aware decision-making, as explored by Wiltzer et al.[28] through their concept of the *distributional successor measure (DSM)*.

Another option is integrating **diffusion-based models** for more flexible representations, explored by Ho and al.[164] and recently used work by Zhu and al.[55]

Additional architectural refinements could further improve model performance and stability. **Mixture of Experts (MoE) models**, as explored by Obando and al.[165] provide scalable alternatives by dynamically selecting specialized subnetworks for different tasks. Similarly, classification-based frameworks, such as the approach proposed by Farebrother et al.[166], may lead to further performance gains. Although these methods are promising, their implementation will require careful consideration of computational efficiency, interpretability, and scalability.

Computational efficiency can also be further enhanced. Transferring the code from PyTorch to JAX, which is known for its speed, is one possible solution. Additionally, optimizing the dataset loading process can significantly reduce memory overhead and training time. Further improvements could involve using distributed computing frameworks to leverage multiple GPUs or TPUs, allowing for more efficient large-scale data training.

7.4 Challenges in Real-World Deployment

While our approach moves zero-shot Reinforcement Learning closer to real-world usability, several challenges remain for practical deployment, particularly in robotics. A better training stability would be one of the most important improvements to be made, on the algorithm design side. Sensor data processing remains a challenge, requiring the translation of raw inputs from vision, sound, and other sensory modalities into structured observations interpretable by the model. Designing robust reward functions that allow effective learning without explicit supervision is another challenge. Additionally, ensuring the security and reliability of AI-driven agents is essential, particularly for deployment in uncertain environments.

Addressing these challenges will be critical for transitioning zero-shot RL from research to practical applications, enabling autonomous systems to generalize effectively across diverse tasks.

7.5 Ethics

7.5.1 The Importance of an Ethics Section

The inclusion of an ethics section is essential to demonstrate that this research has been conducted responsibly and that time has been dedicated to consideration of its broader impact on society and the environment. When researching or developing AI application, it is not enough to focus solely on technical advancements; we must also critically evaluate the implications of our work. Ethical awareness fosters accountability and ensures that innovation contributes positively to the world. Especially in a context where laws and policies evolve slower than technological development[167].

7.5.2 General Ethical Considerations in AI

The increasing autonomy of AI systems raises fundamental ethical challenges, requiring clear boundaries and regulatory frameworks. Fictional constructs, such as Asimov's Three Laws of Robotics, provide a starting point for discussing AI governance, but real-world AI ethics must address complex social, economic, political, and environmental considerations.

Social Impact

AI systems inherit biases from their training data, which can reinforce discrimination and inequality. For example, an autonomous vehicle might behave differently depending on the perceived age or demographic of a pedestrian, leading to unintended ethical dilemmas. Additionally, AI influences human interactions—both with machines and among people—shaping societal norms and behaviors in ways that require careful examination.

Economic Impact

The automation of tasks traditionally performed by humans will reshape the job market. While AI could eliminate certain entry-level roles, it may also create demand for highly skilled workers and more human-centric jobs. Moreover, AI-driven productivity gains are expected to significantly impact economic growth, with some estimates predicting a 33% increase in aggregate productivity over the next two decades[168].

Political and Regulatory Impact

The governance of AI requires policies that address transparency, intellectual property, privacy, security, and human rights[169]. AI regulation is specific to each region:

- **Europe:** GDPR for data protection, copyright laws, and ethical AI guidelines, but no specific military AI policies.
- **United States:** Sector-specific privacy laws, copyright regulations, and dedicated AI policies for military applications.
- **China:** The Personal Information Protection Law (PIPL) and Data Security Law (DSL), unclear copyright regulations regarding AI-generated content.

These differing strategies underscore the need for international cooperation in AI governance[168].

Ethical and Bio-ethical Considerations

AI's growing role in decision-making raises questions about fairness, accountability, and human oversight. Bioethics emphasize the need to ensure that AI serves humanity rather than the reverse[170]. Researchers must consider not only the technical performance of AI but also its alignment with human values.

7.5.3 Ethical Considerations Specific to This Research

Environmental Impact. AI research, particularly deep learning, has a significant environmental footprint due to high computational demands.

In this thesis, experiments were run on A800 GPU servers in China, consuming 300 W[171]. We ran about 800 experiments, each taking an average of 22 hours, so about 18,000 hours in total. The GPU cluster being located in China, we estimate the carbon footprint at 0.5 kgCO2/kWh[172].

The carbon footprint of this research project equates to approximately 2.2 metric tons of CO_2. This is comparable to the annual sustainable carbon footprint for human consumption to equate planet resource renewal speed. It also represents 16,000 km driving a gasoline-powered car or a year of heating, as shown on Figure 7.1. While computation is necessary for research, optimizing resource usage and considering alternative energy sources should be integral to AI development.

Getting statistical meaning results needing running experiments on 3 to 10 random seeds significantly impacts the research in terms of time, cost, and energy consumption.

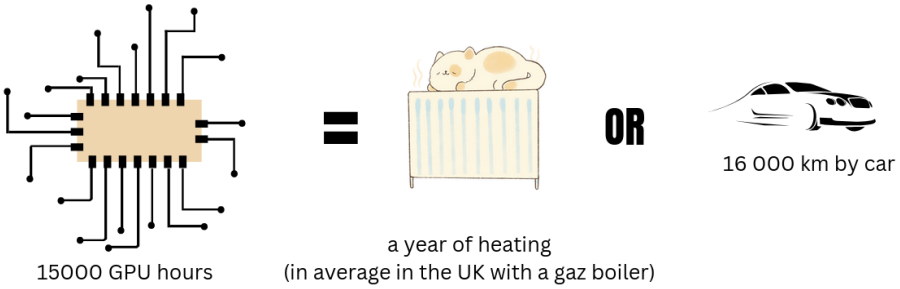


Figure 7.1 Equivalent CO2 emissions to running thesis experiments

Societal Impact of More General RL Models. The broader applicability of RL raises concerns about control and predictability. Neural networks are often black-box models, making it difficult to ensure reliable behavior in high-stakes applications such as autonomous driving. More advanced RL models could significantly impact society by enabling highly autonomous decision-making systems.

Risks: Reinforcement Learning (RL) presents certain risks that must be addressed. **Unpredictability** remains a significant concern, as RL models often lack strong guarantees for safety in complex scenarios. Additionally, **bias** in RL decision-making can lead to ethically problematic outcomes, particularly in scenarios like autonomous vehicles where decisions may not align with human moral intuitions.

Opportunities: On the other hand, RL offers promising opportunities. **Informed decision-making** can be enhanced through RL, particularly in fields like politics and medicine, by analyzing complex, interrelated factors beyond human cognitive limits. Furthermore, **value alignment** is an emerging application of RL, where it is used to align AI systems with human values. This is visible in the use of reinforcement learning from human feedback (RLHF) in natural language processing, promoting the development of more human-centered AI systems.

Striking a balance between benefits and risks is crucial. While RL can be misused[173], complete rejection of its development is neither realistic nor desirable. Instead, ensuring ethical alignment and responsible governance is key.

7.5.4 Personal Considerations

This thesis represents a small trial to contribute to the broader field of AI and robotics, serving as a year dedicated to learning and incremental advancement.

Methods are too nascent to pose immediate ethical dilemmas. However, scientific progress is a collective effort, built on countless minor contributions rather than singular

breakthroughs. Consequently, no individual researcher can be solely responsible for the ultimate applications—whether beneficial or harmful—of their work. If LLMs or RL agents are misused, for example, in autonomous weaponry, the blame cannot rest on a single contributor. Research, like any tool, is neutral; it can be harnessed for both constructive and destructive ends.

Preventing any possible misuse of AI is impossible—there will always be unforeseen applications, just as cybersecurity must continually evolve to counter new threats. The ethical challenge is thus to ensure that the positive impact of AI outweighs its risks. In the case of foundation models for RL, the potential societal transformation is immense. These models could enable autonomous decision-making in everyday robotics, bringing us closer to both the utopian and dystopian visions of AI.

However, AI also offers unprecedented opportunities. Intelligent systems could assist in evaluating environmental impacts, optimizing resource usage, and predicting the social and economic consequences of policy decisions. The ability to model and understand complex systems may ultimately lead to more informed, responsible decision-making at all levels of society.

Despite the risks, we believe the pursuit of AI knowledge is justified. The drive to develop AI is not merely about efficiency but about deepening our understanding of the world and our interactions within it. Just as humanity has historically sought to comprehend nature—through agriculture, medicine, and physics—AI represents another step in our ongoing quest for meaning and insight. While regulations and ethical frameworks will need to evolve to keep pace with technological advancements, the responsibility to guide AI development lies not just with policymakers but also with researchers in their choice of projects and methodologies. Balancing innovation with ethical foresight remains a fundamental challenge, but one worth undertaking.

7.6 Final Thoughts

This work contributes to advancing zero-shot Reinforcement Learning by enhancing the FB framework with improved scalability, regularization, and efficiency. Our method provides a practical, easy-to-use solution that achieves competitive performance on Ex-

oRL while maintaining computational efficiency.

Looking ahead, continued research will aim to generalize our approach towards developing **foundation models** for reinforcement learning—universal agents capable of learning and adapting to any task with minimal supervision. This goal will require further innovations in neural network design, optimization algorithms, and data-driven learning paradigms.

By refining these techniques and bridging the gap between theoretical reinforcement learning research and practical applications, we move closer to realizing reinforcement learning models capable of operating reliably in real-world scenarios across various industries.

REFERENCES

[1] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. nature, 2016, 529(7587): 484-489.

[2] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J/OL]. Nature, 2017, 550(7676): 354-359[2025-03-26]. https://www.nature.com/articles/nature24270. DOI: 10.1038/nature24270.

[3] Kojima T, Gu S S, Reid M, et al. Large Language Models are Zero-Shot Reasoners[Z].

[4] Wei J, Bosma M, Zhao V Y, et al. Finetuned language models are zero-shot learners: arXiv:2109.01652[M/OL]. arXiv, 2022[2024-05-30]. http://arxiv.org/abs/2109.01652.

[5] Pourpanah F, Abdar M, Luo Y, et al. A review of generalized zero-shot learning methods[J/OL]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022: 1-20[2024-05-30]. https://ieeexplore.ieee.org/document/9832795/. DOI: 10.1109/TPAMI.2022.3191696.

[6] Yu C, Liu J, Nemati S, et al. Reinforcement learning in healthcare: A survey[J]. ACM Computing Surveys (CSUR), 2021, 55(1): 1-36.

[7] Abdellatif A A, Mhaisen N, Mohamed A, et al. Reinforcement Learning for Intelligent Healthcare Systems: A Review of Challenges, Applications, and Open Research Issues[J/OL]. IEEE Internet of Things Journal, 2023, 10(24): 21982-22007[2025-03-26]. https://ieeexplore.ieee.org/document/10162185/. DOI: 10.1109/JIOT.2023.3288050.

[8] Wei T, Ren S, Zhu Q. Deep Reinforcement Learning for Joint Datacenter and HVAC Load Control in Distributed Mixed-Use Buildings[J/OL]. IEEE Transactions on Sustainable Computing, 2021, 6(3): 370-384[2025-01-24]. https://ieeexplore.ieee.org/document/8691496/?arnumber=8691496. DOI: 10.1109/TSUSC.2019.2910533.

[9] Li Y, Wen Y, Guan K, et al. Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning: arXiv:1709.05077[M/OL]. arXiv, 2018[2025-01-24]. http://arxiv.org/abs/1709.05077. DOI: 10.48550/arXiv.1709.05077.

[10] Ran Y, Hu H, Zhou X, et al. DeepEE: Joint Optimization of Job Scheduling and Cooling Control for Data Center Energy Efficiency Using Deep Reinforcement Learning[C/OL]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). 2019: 645-655[2025-01-24]. https://ieeexplore.ieee.org/document/8885255/?arnumber=8885255. DOI: 10.1109/ICDCS.2019.00070.

[11] Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play[J]. Science, 2018, 362(6419): 1140-1144.

[12] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. nature, 2015, 518(7540): 529-533.

[13] Peng X B, Abbeel P, Levine S, et al. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills[J/OL]. ACM Transactions on Graphics, 2018, 37(4): 1-14[2025-03-26]. https://dl.acm.org/doi/10.1145/3197517.3201311.

[14] Singh B, Kumar R, Singh V P. Reinforcement learning in robotic applications: A comprehensive survey[J]. Artificial Intelligence Review, 2022, 55(2): 945-990.

[15] Kiran B R, Sobh I, Talpaert V, et al. Deep reinforcement learning for autonomous driving: A survey[J]. IEEE transactions on intelligent transportation systems, 2021, 23(6): 4909-4926.

[16] Touati A, Rapin J, Ollivier Y. Does zero-shot reinforcement learning exist?[M/OL]. arXiv, 2023 [2024-04-07]. http://arxiv.org/abs/2209.14935. DOI: 10.48550/arXiv.2209.14935.

[17] Park S, Kreiman T, Levine S. Foundation policies with Hilbert representations: arXiv:2402.15567[M/OL]. arXiv, 2024[2024-04-08]. http://arxiv.org/abs/2402.15567.

[18] Wołczyk M, Cupiał B, Ostaszewski M, et al. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem: arXiv:2402.02868[M/OL]. arXiv, 2024[2024-06-15]. http://arxiv.org/abs/2402.02868.

[19] Touati A, Ollivier Y. Learning one representation to optimize all rewards[J/OL]. Advances in Neural Information Processing Systems, 2021, 34:13–23[2024-04-07]. http://arxiv.org/abs/2103.07945.

[20] Chen B, Zhu C, Agrawal P, et al. Self-supervised reinforcement learning that transfers using random features[A/OL]. 2023[2024-05-30]. arXiv: 2305.17250. http://arxiv.org/abs/2305.17250.

[21] Kumar A, Zhou A, Tucker G, et al. Conservative Q-learning for offline reinforcement learning [J/OL]. Advances in Neural Information Processing Systems, 2020[2024-05-22]. http://arxiv.org/abs/2006.04779.

[22] Nachum O, Dai B. Reinforcement learning via fenchel-rockafellar duality: arXiv:2001.01866 [M/OL]. arXiv, 2020[2024-08-29]. http://arxiv.org/abs/2001.01866.

[23] Bellman R. A Markovian decision process[J]. Journal of mathematics and mechanics, 1957: 679-684.

[24] Bellman R. Dynamic programming[J]. science, 1966, 153(3731): 34-37.

[25] Bellman R, Kalaba R E, et al. Dynamic programming and modern control theory: Vol. 81[M]. Citeseer, 1965.

[26] Bellman R, Lee E. History and development of dynamic programming[J]. IEEE Control Systems Magazine, 1984, 4(4): 24-28.

[27] Sutton R S, Barto A G. Reinforcement learning: An introduction[M/OL]. Cambridge, MA, USA: A Bradford Book, 2018. 10.5555/3312046.

[28] Wiltzer H, Farebrother J, Gretton A, et al. A distributional analogue to the successor representation[M/OL]. arXiv, 2024[2025-03-24]. http://arxiv.org/abs/2402.08530. DOI: 10.48550/arXiv.2402.08530.

[29] Sutton R S. Learning to predict by the methods of temporal differences[J]. Machine learning, 1988, 3: 9-44.

[30] Watkins C J, Dayan P. Q-learning[J]. Machine learning, 1992, 8: 279-292.

[31] Rummery G A, Niranjan M. On-line Q-learning using connectionist systems: Vol. 37[M]. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[32] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8: 229-256.

[33] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning [C]//International Conference on Machine Learning. PmLR, 2016: 1928-1937.

[34] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning [A]. 2015. arXiv: 1509.02971.

[35] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[C]//International Conference on Machine Learning. Pmlr, 2018: 1861-1870.

[36] Moerland T M, Broekens J, Plaat A, et al. Model-based reinforcement learning: A survey[J]. Foundations and Trends® in Machine Learning, 2023, 16(1): 1-118.

[37] Luo F M, Xu T, Lai H, et al. A survey on model-based reinforcement learning[J/OL]. Science China Information Sciences, 2024, 67(2): 121101[2025-04-22]. https://link.springer.com/10.1007/s11432-022-3696-5.

[38] Li X, Liu M, Zhang H, et al. Vision-language foundation models as effective robot imitators [A]. 2023. arXiv: 2311.01378.

[39] Brohan A, Brown N, Carbajal J, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control[A]. 2023. arXiv: 2307.15818.

[40] Kim M J, Pertsch K, Karamcheti S, et al. Openvla: An open-source vision-language-action model[A]. 2024. arXiv: 2406.09246.

[41] Rocamonde J, Montesinos V, Nava E, et al. Vision-language models are zero-shot reward models for reinforcement learning[A]. 2023. arXiv: 2310.12921.

[42] Firoozi R, Tucker J, Tian S, et al. Foundation models in robotics: Applications, challenges, and the future[J]. The International Journal of Robotics Research, 2023: 02783649241281508.

[43] Beck J, Vuorio R, Liu E Z, et al. A Survey of Meta-Reinforcement Learning: arXiv:2301.08028 [M/OL]. arXiv, 2024[2025-02-26]. http://arxiv.org/abs/2301.08028. DOI: 10.48550/arXiv.2301.08028.

[44] Mitchell E, Rafailov R, Peng X B, et al. Offline meta-reinforcement learning with advantage weighting[C]//International Conference on Machine Learning. PMLR, 2021: 7780-7791.

[45] Nagabandi A, Clavera I, Liu S, et al. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning[A]. 2018. arXiv: 1803.11347.

[46] Gupta A, Mendonca R, Liu Y, et al. Meta-reinforcement learning of structured exploration strategies[J]. Advances in neural information processing systems, 2018, 31.

[47] Vithayathil Varghese N, Mahmoud Q H. A survey of multi-task deep reinforcement learning[J]. Electronics, 2020, 9(9): 1363.

[48] Wilson A, Fern A, Ray S, et al. Multi-task reinforcement learning: A hierarchical bayesian approach[C]//Proceedings of the 24th International Conference on Machine Learning. 2007: 1015-1022.

[49] Teh Y, Bapst V, Czarnecki W M, et al. Distral: Robust multitask reinforcement learning[J]. Advances in neural information processing systems, 2017, 30.

[50] Andreas J, Klein D, Levine S. Modular multitask reinforcement learning with policy sketches [C]//International Conference on Machine Learning. PMLR, 2017: 166-175.

[51] Zhang G, Jain A, Hwang I, et al. Efficient multi-task reinforcement learning via selective behavior sharing: arXiv:2302.00671[M/OL]. arXiv, 2023[2024-06-15]. http://arxiv.org/abs/2302.00671.

[52] Zhang Y, Yang Q. A survey on multi-task learning[J]. IEEE transactions on knowledge and data engineering, 2021, 34(12): 5586-5609.

[53] Hendawy A, Peters J, D'Eramo C. Multi-task reinforcement learning with mixture of orthogonal experts[Z]. 2024.

[54] Ishfaq H, Nguyen-Tang T, Feng S, et al. Offline multitask representation learning for reinforcement learning[J]. Advances in Neural Information Processing Systems, 2024, 37: 70557-70616.

[55] Zhu Z, Lin K, Jain A K, et al. Transfer learning in deep reinforcement learning: A survey [A/OL]. 2023[2024-06-15]. arXiv: 2009.07888. http://arxiv.org/abs/2009.07888.

[56] Kadamala K, Chambers D, Barrett E. Enhancing HVAC control systems through transfer learning with deep reinforcement learning agents[J]. Smart Energy, 2024, 13: 100131.

[57] Huang R, He H, Su Q, et al. Enabling cross-type full-knowledge transferable energy management for hybrid electric vehicles via deep transfer reinforcement learning[J]. Energy, 2024, 305: 132394.

[58] Liang X, Liu Y, Chen T, et al. Federated transfer reinforcement learning for autonomous driving [M]//Federated and Transfer Learning. Springer, 2022: 357-371.

[59] Mai T, Yao H, Zhang N, et al. Transfer reinforcement learning aided distributed network slicing optimization in industrial IoT[J]. IEEE Transactions on Industrial Informatics, 2021, 18(6): 4308-4316.

[60] You H, Yang T, Zheng Y, et al. Cross-domain adaptive transfer reinforcement learning based on state-action correspondence[C]//Uncertainty in Artificial Intelligence. PMLR, 2022: 2299-2309.

[61] Liu M, Zhu M, Zhang W. Goal-conditioned reinforcement learning: Problems and solutions: arXiv:2201.08299[M/OL]. arXiv, 2022[2024-12-26]. http://arxiv.org/abs/2201.08299. DOI: 10.48550/arXiv.2201.08299.

[62] Zeng Z, Zhang C, Wang S, et al. Goal-Conditioned Predictive Coding for Offline Reinforcement Learning[Z]. 2023.

[63] Eysenbach B, Zhang T, Salakhutdinov R, et al. Contrastive Learning as Goal-Conditioned Reinforcement Learning: abs/2206.07568[A/OL]. 2022: null. https://www.semanticscholar.org/paper/4247b93593b6ecd70262a1b1c7021dcecc26c8e0. DOI: 10.48550/arXiv.2206.07568.

[64] Frans K, Park S, Abbeel P, et al. Unsupervised zero-shot reinforcement learning via functional reward encodings: arXiv:2402.17135[M/OL]. arXiv, 2024[2024-04-07]. http://arxiv.org/abs/2402.17135.

[65] Ingebrand T, Zhang A, Topcu U. Zero-shot reinforcement learning via function encoders: arXiv:2401.17173[M/OL]. arXiv, 2024[2024-05-22]. http://arxiv.org/abs/2401.17173.

[66] Zhang T, Ren T, Yang M, et al. Making linear mdps practical via contrastive representation learning[C]//International Conference on Machine Learning. PMLR, 2022: 26447-26466.

[67] Barreto A, Dabney W, Munos R, et al. Successor Features for Transfer in Reinforcement Learning[A/OL]. 2018[2024-05-22]. arXiv: 1606.05312. http://arxiv.org/abs/1606.05312.

[68] Dayan P. Improving generalization for temporal difference learning: The successor representation[J/OL]. Neural Computation, 1993, 5(4): 613-624[2024-05-28]. https://direct.mit.edu/neco/article/5/4/613-624/5736. DOI: 10.1162/neco.1993.5.4.613.

[69] Schaul T, Horgan D, Gregor K, et al. Universal value function approximators[C]//International Conference on Machine Learning. 2015.

[70] Ma C, Ashley D R, Wen J, et al. Universal successor features for transfer reinforcement learning: arXiv:2001.04025[M/OL]. arXiv, 2020[2024-05-22]. http://arxiv.org/abs/2001.04025.

[71] Borsa D, Barreto A, Quan J, et al. Universal successor features approximators: arXiv:1812.07626[M/OL]. arXiv, 2018[2024-05-22]. http://arxiv.org/abs/1812.07626. DOI: 10.48550/arXiv.1812.07626.

[72] Zhu C, Wang X, Han T, et al. Transferable reinforcement learning via generalized occupancy models[C]//ICML 2024 Workshop on Structured Probabilistic Inference {\&} Generative Modeling. 2024.

[73] Agarwal A, Kakade S, Krishnamurthy A, et al. FLAMBE: Structural complexity and representation learning of low rank MDPs[Z].

[74] Blier L, Tallec C, Ollivier Y. Learning successor states and goal-dependent values: A mathematical viewpoint[A/OL]. 2021[2024-04-07]. arXiv: 2101.07123. http://arxiv.org/abs/2101.07123.

[75] Haarnoja T, Zhou A, Hartikainen K, et al. Soft actor-critic algorithms and applications: arXiv:1812.05905[M/OL]. arXiv, 2019[2024-09-01]. http://arxiv.org/abs/1812.05905.

[76] Uehara M, Zhang X, Sun W. Representation learning for online and offline RL in low-rank MDPs: arXiv:2110.04652[M/OL]. arXiv, 2022[2024-06-05]. http://arxiv.org/abs/2110.04652.

[77] Papini M, Tirinzoni A, Pacchiano A, et al. Reinforcement learning in linear mdps: Constant regret and representation selection[J]. Advances in Neural Information Processing Systems, 2021, 34: 16371-16383.

[78] Eysenbach B, Salakhutdinov R, Levine S. The information geometry of unsupervised reinforcement learning[Z]. 2022.

[79] Jeen S, Bewley T, Cullen J. Zero-shot reinforcement learning from low quality data[J]. Advances in Neural Information Processing Systems, 2024, 37: 16894-16942.

[80] Cetin E, Touati A, Ollivier Y. Finer behavioral foundation models via auto-regressive features and advantage weighting: arXiv:2412.04368[M/OL]. arXiv, 2024[2024-12-16]. http://arxiv.org/abs/2412.04368. DOI: 10.48550/arXiv.2412.04368.

[81] Peng X B, Kumar A, Zhang G, et al. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning: arXiv:1910.00177[M/OL]. arXiv, 2019[2025-02-22]. http://arxiv.org/abs/1910.00177. DOI: 10.48550/arXiv.1910.00177.

[82] Cetin E, Tirinzoni A, Pirotta M, et al. Simple ingredients for offline reinforcement learning: arXiv:2403.13097[M/OL]. arXiv, 2024[2024-09-26]. http://arxiv.org/abs/2403.13097.

[83] Tirinzoni A, Touati A, Farebrother J, et al. Zero-Shot whole-body humanoid control via behavioral foundation models[C].

[84] Agarwal S, Sikchi H, Stone P, et al. Proto Successor Measure: Representing the space of all possible solutions of reinforcement learning: arXiv:2411.19418[M/OL]. arXiv, 2024[2024-12-04]. http://arxiv.org/abs/2411.19418. DOI: 10.48550/arXiv.2411.19418.

[85] Pirotta M, Tirinzoni A, Touati A, et al. Fast imitation via behavior foundation models[C]//2024.

[86] Yang S, Nachum O, Du Y, et al. Foundation models for decision making: Problems, methods, and opportunities: arXiv:2303.04129[M/OL]. arXiv, 2023[2024-12-16]. http://arxiv.org/abs/2303.04129. DOI: 10.48550/arXiv.2303.04129.

[87] Sikchi H, Agarwal S, Jajoo P, et al. RL Zero: Zero-Shot language to behaviors without any supervision: arXiv:2412.05718[M/OL]. arXiv, 2024[2024-12-16]. http://arxiv.org/abs/2412.05718. DOI: 10.48550/arXiv.2412.05718.

[88] Sun J, Tu S, Li H, et al. Unsupervised zero-shot reinforcement learning via dual-value forward-backward representation[C/OL]//The Thirteenth International Conference on Learning Representations. 2025. https://openreview.net/pdf?id=0QnKnt411O.

[89] Brandfonbrener D, Bietti A, Buckman J, et al. When does return-conditioned supervised learning work for offline reinforcement learning?[A/OL]. 2023[2025-03-03]. arXiv: 2206.01079. http://arxiv.org/abs/2206.01079.

[90] Kim J, Park S, Levine S. Unsupervised-to-online reinforcement learning[Z].

[91] Kostrikov I, Nair A, Levine S. Offline reinforcement learning with implicit Q-learning[C/OL]//Advances in Neural Information Processing SystemsDeep RL Workshop. arXiv, 2021[2024-05-22]. http://arxiv.org/abs/2110.06169.

[92] Bjorck J, Gomes C P, Weinberger K Q. Towards deeper deep reinforcement learning with spectral normalization[A/OL]. 2022[2024-10-19]. arXiv: 2106.01151. http://arxiv.org/abs/2106.01151.

[93] Nauman M, Ostaszewski M, Jankowski K, et al. Bigger, regularized, optimistic: Scaling for compute and sample-efficient continuous control: arXiv:2405.16158[M/OL]. arXiv, 2024[2024-12-23]. http://arxiv.org/abs/2405.16158. DOI: 10.48550/arXiv.2405.16158.

[94] Schwarzer M, Ceron J S O, Courville A, et al. Bigger, better, faster: Human-level atari with human-level efficiency[C]//International Conference on Machine Learning. PMLR, 2023: 30365-30380.

[95] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C/OL]//2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, 2016: 770-778[2025-04-25]. http://ieeexplore.ieee.org/document/7780459/. DOI: 10.1109/CVPR.2016.90.

[96] Espeholt L, Soyer H, Munos R, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures[C]//International Conference on Machine Learning. PMLR, 2018: 1407-1416.

[97] Ota K, Jha D K, Kanezaki A. Training larger networks for deep reinforcement learning: arXiv:2102.07920[M/OL]. arXiv, 2021[2024-09-12]. http://arxiv.org/abs/2102.07920.

[98] Ota K, Oiki T, Jha D, et al. Can increasing input dimensionality improve deep reinforcement learning?[C]//International Conference on Machine Learning. PMLR, 2020: 7424-7433.

[99] Kumar A, Agarwal R, Geng X, et al. Offline Q-Learning on diverse multi-task data both scales and generalizes: arXiv:2211.15144[M/OL]. arXiv, 2023[2024-10-23]. http://arxiv.org/abs/2211.15144.

[100] Nair A, Gupta A, Dalal M, et al. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets: arXiv:2006.09359[M/OL]. arXiv, 2021[2025-02-18]. http://arxiv.org/abs/2006.09359. DOI: 10.48550/arXiv.2006.09359.

[101] Springenberg J T, Abdolmaleki A, Zhang J, et al. Offline actor-critic reinforcement learning scales to large models: arXiv:2402.05546[M/OL]. arXiv, 2024[2024-10-28]. http://arxiv.org/abs/2402.05546.

[102] Cheng J, Qiao R, Xiong G, et al. Scaling offline model-based RL via jointly-optimized world-action model pretraining: arXiv:2410.00564[M/OL]. arXiv, 2024[2025-01-21]. http://arxiv.org/abs/2410.00564. DOI: 10.48550/arXiv.2410.00564.

[103] Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need: arXiv:1706.03762[M/OL]. arXiv, 2023[2025-02-18]. http://arxiv.org/abs/1706.03762. DOI: 10.48550/arXiv.1706.03762.

[104] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019: 4171-4186.

[105] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[M]. San Francisco, CA, USA, 2018.

[106] Chen L, Lu K, Rajeswaran A, et al. Decision Transformer: Reinforcement Learning via Sequence Modeling[Z].

[107] Chebotar Y, Vuong Q, Hausman K, et al. Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-functions[C/OL]//Proceedings of The 7th Conference on Robot Learning. PMLR, 2023: 3909-3928[2024-09-26]. https://proceedings.mlr.press/v229/chebotar23a.html.

[108] Wang K, Zhao H, Luo X, et al. Bootstrapped Transformer for Offline Reinforcement Learning [Z]. 2022.

[109] Wang Y, Yang C, Wen Y, et al. Critic-Guided Decision Transformer for Offline Reinforcement Learning[C/OL]//Proceedings of the AAAI Conference on Artificial Intelligence: Vol. 38. 2024: 15706-15714[2025-02-09]. https://ojs.aaai.org/index.php/AAAI/article/view/29499. DOI: 10.1609/aaai.v38i14.29499.

[110] Wu Y H, Wang X, Hamaya M. Elastic Decision Transformer[Z].

[111] Furuta H, Matsuo Y, Gu S S. Generalized Decision Transformer for Offline Hindsight Information Matching: arXiv:2111.10364[M/OL]. arXiv, 2022[2025-02-18]. http://arxiv.org/abs/2111.10364. DOI: 10.48550/arXiv.2111.10364.

[112] Correia A, Alexandre L A. Hierarchical Decision Transformer[C/OL]//2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2023: 1661-1666[2025-02-09]. https://ieeexplore.ieee.org/document/10342230/?arnumber=10342230. DOI: 10.1109/IROS55552.2023.10342230.

[113] Luu T M, Lee D, Yoo C D. Predictive coding for decision transformer[C]//2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2024: 7469-7476.

[114] Hu S, Fan Z, Huang C, et al. Q-value Regularized Transformer for Offline Reinforcement Learning: arXiv:2405.17098[M/OL]. arXiv, 2024[2025-02-09]. http://arxiv.org/abs/2405.17098. DOI: 10.48550/arXiv.2405.17098.

[115] Kotb M, Weber C, Hafez M B, et al. QT-TDM: Planning with transformer dynamics model and autoregressive Q-learning[A/OL]. 2024[2024-10-28]. arXiv: 2407.18841. http://arxiv.org/abs/2407.18841.

[116] Sudhakaran S, Risi S. Skill Decision Transformer: arXiv:2301.13573[M/OL]. arXiv, 2023[2025-02-09]. http://arxiv.org/abs/2301.13573. DOI: 10.48550/arXiv.2301.13573.

[117] Huang K, Shen L, Zhao C, et al. Solving Continual Offline Reinforcement Learning with Decision Transformer: arXiv:2401.08478[M/OL]. arXiv, 2024[2025-02-09]. http://arxiv.org/abs/2401.08478. DOI: 10.48550/arXiv.2401.08478.

[118] Siebenborn M, Belousov B, Huang J, et al. How Crucial is Transformer in Decision Transformer?: arXiv:2211.14655[M/OL]. arXiv, 2022[2025-02-09]. http://arxiv.org/abs/2211.14655. DOI: 10.48550/arXiv.2211.14655.

[119] Bhargava P, Chitnis R, Geramifard A, et al. WHEN SHOULD WE PREFER DECISION TRANSFORMERS FOR OFFLINE REINFORCEMENT LEARNING?[Z]. 2024.

[120] Fu Z, Lam W, Yu Q, et al. Decoder-Only or Encoder-Decoder? Interpreting Language Model as a Regularized Encoder-Decoder: arXiv:2304.04052[M/OL]. arXiv, 2023[2025-02-17]. http://arxiv.org/abs/2304.04052. DOI: 10.48550/arXiv.2304.04052.

[121] Agarwal P, Rahman A A, St-Charles P L, et al. Transformers in Reinforcement Learning: A Survey: arXiv:2307.05979[M/OL]. arXiv, 2023[2025-02-05]. http://arxiv.org/abs/2307.05979. DOI: 10.48550/arXiv.2307.05979.

[122] Huang G, Liu Z, Van Der Maaten L, et al. Densely Connected Convolutional Networks[C/OL]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE, 2017: 2261-2269[2025-04-25]. https://ieeexplore.ieee.org/document/8099726/. DOI: 10.1109/CVPR.2017.243.

[123] Andrej. Karpathy/nanoGPT[EB/OL]. 2025[2025-03-05]. https://github.com/karpathy/nanoGPT.

[124] Andrej. Karpathy/minGPT[EB/OL]. 2025[2025-03-05]. https://github.com/karpathy/minGPT.

[125] Fu J, Kumar A, Soh M, et al. Diagnosing bottlenecks in deep Q-learning algorithms[C]//International Conference on Machine Learning. 2019.

[126] Kumar A, Fu J, Soh M, et al. Stabilizing off-policy q-learning via bootstrapping error reduction [J]. Advances in neural information processing systems, 2019, 32.

[127] Levine S, Kumar A, Tucker G, et al. Offline reinforcement learning: Tutorial, review, and perspectives on open problems: arXiv:2005.01643[M/OL]. arXiv, 2020[2024-05-22]. http://arxiv.org/abs/2005.01643. DOI: 10.48550/arXiv.2005.01643.

[128] Fujimoto S, Meger D, Precup D. Off-policy deep reinforcement learning without exploration [C]//International Conference on Machine Learning. PMLR, 2019: 2052-2062.

[129] Ma Y, Hao J, Hu X, et al. Iteratively Refined Behavior Regularization for Offline Reinforcement Learning[C]//NeurIPS. 2024.

[130] Achiam J, Held D, Tamar A, et al. Constrained policy optimization[C]//International Conference on Machine Learning. PMLR, 2017: 22-31.

[131] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]// Proceedings of the AAAI Conference on Artificial Intelligence: Vol. 30. 2016.

[132] Agarwal R, Schuurmans D, Norouzi M. An optimistic perspective on offline reinforcement learning[C]//International Conference on Machine Learning. 2019.

[133] Nakamoto M, Zhai S, Singh A, et al. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning[J]. Advances in Neural Information Processing Systems, 2023, 36: 62244-62269.

[134] Hansen-Estruch P, Kostrikov I, Janner M, et al. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies: arXiv:2304.10573[M/OL]. arXiv, 2023[2024-09-04]. http://arxiv.org/abs/2304.10573.

[135] He L, Shen L, Tan J, et al. AlignIQL: Policy alignment in implicit Q-learning through constrained optimization: arXiv:2405.18187[M/OL]. arXiv, 2024[2024-12-13]. http://arxiv.org/abs/2405.18187. DOI: 10.48550/arXiv.2405.18187.

[136] Wu K, Zhao Y, Xu Z, et al. ACL-QL: Adaptive Conservative Level in Q-Learning for Offline Reinforcement Learning[J/OL]. IEEE Transactions on Neural Networks and Learning Systems, 2025: 1-15[2025-03-28]. http://arxiv.org/abs/2412.16848. DOI: 10.1109/TNNLS.2024.3497667.

[137] Shimizu Y, Hong J, Levine S, et al. Strategically Conservative Q-Learning: arXiv:2406.04534[M/OL]. arXiv, 2024[2025-03-28]. http://arxiv.org/abs/2406.04534. DOI: 10.48550/arXiv.2406.04534.

[138] Shao Z, Zhuang L, Yan J, et al. Conservative In-Distribution Q-Learning for Offline Reinforcement Learning[C/OL]//2024 International Joint Conference on Neural Networks (IJCNN). Yokohama, Japan: IEEE, 2024: 1-8[2025-03-28]. https://ieeexplore.ieee.org/document/10650768/. DOI: 10.1109/IJCNN60899.2024.10650768.

[139] Lyu J, Ma X, Li X, et al. Mildly Conservative Q-Learning for Offline Reinforcement Learning: arXiv:2206.04745[M/OL]. arXiv, 2024[2025-03-28]. http://arxiv.org/abs/2206.04745. DOI: 10.48550/arXiv.2206.04745.

[140] Xu H, Jiang L, Li J, et al. Offline RL with no OOD actions: In-sample learning via implicit value regularization[M/OL]. arXiv, 2023[2024-06-15]. http://arxiv.org/abs/2303.15810.

[141] Sikchi H, Zheng Q, Zhang A, et al. Dual RL: Unification and new methods for reinforcement and imitation learning: arXiv:2302.08560[M/OL]. arXiv, 2024[2024-11-03]. http://arxiv.org/abs/2302.08560.

[142] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network: arXiv:1503.02531 [M/OL]. arXiv, 2015[2025-04-03]. http://arxiv.org/abs/1503.02531. DOI: 10.48550/arXiv.1503.02531.

[143] Chen P, Liu S, Zhao H, et al. Distilling Knowledge via Knowledge Review[C/OL]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Nashville, TN, USA: IEEE, 2021: 5006-5015[2025-04-03]. https://ieeexplore.ieee.org/document/9578915/. DOI: 10.1109/CVPR46437.2021.00497.

[144] Cheng X, Rao Z, Chen Y, et al. Explaining Knowledge Distillation by Quantifying the Knowledge[C/OL]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA: IEEE, 2020: 12922-12932[2025-04-03]. https://ieeexplore.ieee.org/document/9157818/. DOI: 10.1109/CVPR42600.2020.01294.

[145] Kumar A, Agarwal R, Ma T, et al. DR3: Value-based deep reinforcement learning requires explicit regularization: arXiv:2112.04716[M/OL]. arXiv, 2021[2024-09-02]. http://arxiv.org/abs/2112.04716.

[146] Tassa Y, Doron Y, Muldal A, et al. DeepMind control suite[M/OL]. arXiv, 2018[2024-08-22]. http://arxiv.org/abs/1801.00690.

[147] Todorov E, Erez T, Tassa Y. MuJoCo: A physics engine for model-based control[C/OL]//2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura-Algarve, Portugal: IEEE, 2012: 5026-5033[2025-03-18]. http://ieeexplore.ieee.org/document/6386109/. DOI: 10.1109/IROS.2012.6386109.

[148] Yarats D, Brandfonbrener D, Liu H, et al. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning: arXiv:2201.13425[M/OL]. arXiv, 2022[2024-08-22]. http://arxiv.org/abs/2201.13425.

[149] Burda Y, Edwards H, Storkey A, et al. Exploration by Random Network Distillation: arXiv:1810.12894[M/OL]. arXiv, 2018[2025-03-10]. http://arxiv.org/abs/1810.12894. DOI: 10.48550/arXiv.1810.12894.

[150] Eysenbach B, Gupta A, Ibarz J, et al. Diversity is all you need: Learning skills without a reward function: arXiv:1802.06070[M/OL]. arXiv, 2018[2024-12-23]. http://arxiv.org/abs/1802.06070. DOI: 10.48550/arXiv.1802.06070.

[151] Eimer T, Lindauer M, Raileanu R. Hyperparameters in reinforcement learning and how to tune them[C]//International Conference on Machine Learning. PMLR, 2023: 9104-9149.

[152] Agarwal R, Schwarzer M, Castro P S, et al. Deep Reinforcement Learning at the Edge of the Statistical Precipice[J]. Advances in Neural Information Processing Systems, 2021, 34.

[153] Farebrother J, Pirotta M, Tirinzoni A, et al. Temporal Difference Flows: arXiv:2503.09817 [M/OL]. arXiv, 2025[2025-03-24]. http://arxiv.org/abs/2503.09817. DOI: 10.48550/arXiv.2503.09817.

[154] Mao Y, Wang Q, Chen C, et al. Offline reinforcement learning with OOD state correction and OOD action suppression: arXiv:2410.19400[M/OL]. arXiv, 2024[2024-12-18]. http://arxiv.org/abs/2410.19400. DOI: 10.48550/arXiv.2410.19400.

[155] Li J, Zhan X, Xu H, et al. When data geometry meets deep function: Generalizing offline reinforcement learning: arXiv:2205.11027[M/OL]. arXiv, 2023[2024-10-18]. http://arxiv.org/abs/2205.11027.

[156] Salvato E, Fenu G, Medvet E, et al. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning[J]. IEEE access : practical innovations, open solutions, 2021, 9: 153171-153187.

[157] Zhao W, Queralta J P, Westerlund T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey[C/OL]//2020 IEEE Symposium Series on Computational Intelligence (SSCI). 2020: 737-744[2025-04-28]. http://arxiv.org/abs/2009.13303. DOI: 10.1109/SSCI47803.2020.9308468.

[158] Rao K, Harris C, Irpan A, et al. Rl-cyclegan: Reinforcement learning aware simulation-to-real [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 11157-11166.

[159] Torne M, Simeonov A, Li Z, et al. Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation: arXiv:2403.03949[M/OL]. arXiv, 2024[2024-04-08]. http://arxiv.org/abs/2403.03949.

[160] Ju H, Juan R, Gomez R, et al. Transferring policy of deep reinforcement learning from simulation to reality for robotics[J]. Nature Machine Intelligence, 2022, 4(12): 1077-1087.

[161] Marc G. Bellemare, Will Dabney, and Mark Rowland. Distributional reinforcement learning [M/OL]. MIT Press, [2024][2024-05-29]. https://www.distributional-rl.org/.

[162] Bellemare M G, Dabney W, Rowland M. Adaptive Computation and Machine Learning series: Distributional Reinforcement Learning[M]. Cambridge: The MIT Press, 2023.

[163] Bellemare M G, Dabney W, Munos R. A distributional perspective on reinforcement learning [C]//International Conference on Machine Learning. PMLR, 2017: 449-458.

[164] Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models[J]. Advances in neural information processing systems, 2020, 33: 6840-6851.

[165] Obando-Ceron J, Sokar G, Willi T, et al. Mixtures of experts unlock parameter scaling for Deep RL: arXiv:2402.08609[M/OL]. arXiv, 2024[2024-12-23]. http://arxiv.org/abs/2402.08609. DOI: 10.48550/arXiv.2402.08609.

[166] Farebrother J, Orbay J, Vuong Q, et al. Stop Regressing: Training value functions via classification for scalable deep RL: arXiv:2403.03950[M/OL]. arXiv, 2024[2024-12-30]. http://arxiv.org/abs/2403.03950. DOI: 10.48550/arXiv.2403.03950.

[167] Auernhammer J. Human-centered AI: The role of Human-centered Design Research in the development of AI[EB/OL]. 2020[2025-03-11]. https://dl.designresearchsociety.org/cgi/viewcontent.cgi?article=1178&context=drs-conference-papers.

[168] Comunale M, Manera A. IMF Working Papers: The Economic Impacts and the Regulation of AI: A Review of the Academic Literature and Policy Actions[M/OL]. Washington, D.C: International Monetary Fund, 2024. DOI: 10.5089/9798400268588.001.

[169] European Commission. Directorate General for Economic and Financial Affairs. Artificial intelligence: Economic impact, opportunities, challenges, implications for policy.[M/OL]. LU: Publications Office, 2024[2025-03-11]. https://data.europa.eu/doi/10.2765/48272.

[170] Tai M T. The impact of artificial intelligence on human society and bioethics[J/OL]. Tzu Chi Medical Journal, 2020, 32(4): 339[2025-03-11]. https://journals.lww.com/10.4103/tcmj.tcmj_71_20.

[171] ThinkSystem NVIDIA A800 PCIe 4.0 GPUs Product Guide (withdrawn product)[J/OL]. Lenovo Press[2025-03-13]. https://lenovopress.lenovo.com/lp1813-thinksystem-nvidia-a800-pcie-gpu.

[172] China: Power sector carbon intensity 2023[J/OL]. Statista[2025-03-13]. https://www.statista.com/statistics/1300419/power-generation-emission-intensity-china/.

[173] Meyer T, Kaloudi N, Li J. A Systematic Literature Review on Malicious Use of Reinforcement Learning[C/OL]//2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS). 2021: 21-28[2025-03-11]. https://ieeexplore.ieee.org/document/9476057/?arnumber=9476057. DOI: 10.1109/EnCyCriS52570.2021.00011.

APPENDIX A FAILED DESIGN — OPERATOR AND AUTO-REGRESSIVE B STRUCTURE

A.1 Enhancing Expressivity: Auto-Regressive B

To further improve expressivity, we explored the autoregressive B structure proposed by Cetin and al. [1] The idea is to transform $B(s)$ into $B(s, z)$, allowing more complex latent representations. The FB framework, like all successor-feature-based methods, relies on a linear encoding of tasks, meaning that at test time, each new reward function is projected onto a fixed set of pre-trained features. This limits expressivity.

To address this linearity limitation, Cetin and al. [1] propose an autoregressive B model, where a portion of B depends on its previous outputs and z. Although their results show limited performance improvements, our Transformer-based architecture is well-suited for this approach. As they do not publish their code, we aim to rebuild a structure based on their idea.

However, we do not notice any performance improvement using their technic. On the contrary, the performance tends to decrease, which can be explained by the fact that the masks reduced the knowledge provided as input for computing the parameters.

A.2 Operator

A.2.1 Motivation and Constraints

Learning the complete transition model M directly is computationally intractable for complex environments. To address this challenge, FB and later works propose decoupling the learning of a policy from understanding the environment dynamics [2,3,4,5].In FB, the problem is formulated as: $M^{\pi^z}(s_0, a_0, ds, da) = f_1(s, a, \pi_z) * f_2(s', a')$.

However, the expression of M as a product of F and B representations introduces a linearity constraint that limits the model's expressiveness. To overcome this limitation, we wanted to propose extending the FB framework to allow for non-linear interactions between F and B representations using an operator: $M^{\pi^z}(s_0, a_0, ds, da) = f(f_1(s, a, \pi_z), f_2(s', a'))$, where:

- $f_1(s, a, \pi_z)$ functions as a policy extraction mechanism,
- $f_2(s', a')$ provides representation learning of the environment dynamics.

In our FB framework, we would implement this decomposition through:

- A forward model F that serves as the policy extraction function $f1$.
- A backward model B that encodes rewards information—and by extension, future states—as a latent variable, effectively learning representations of environment dynamics (as $f2$).
- An operator o that combines F and B representations (as f).

The key constraint in the original FB formulation is the linearity assumption in the representation implied by the use of a product $F \times B$ to combine F and B. Indeed, using a matrix product to combine F and B representations implies some linear relationship between F and B representations rows. It therefore implies a linear relationship between a future state and action s', a (F row indices) and the different original states s (B row indices).

To overcome this limitation, we propose learning a more complex combination of F and B coefficient. For doing so, we propose leveraging neural networks to enable non-linear interactions between the forward and backward models:

$$M^{\pi^z}(s_0, a_0, ds, da) = F(s_0, a_0, z)^\top \bigcirc B(s, a)\rho(ds, da)$$

Where $\bigcirc$ represents an operation learned via a neural network.

With this formulation, we can retrieve the policy for any reward function r by computing:

$$z_R = B \times r$$

$$\pi_r(s) = \arg\max_a F(s, a, z_R) \bigcirc z_R$$

This proposed extension offers two significant advantages. First, it enhances the capacity for modeling complex dynamical systems. Second, it is able to capture non-linear relationships between states and actions.

By maintaining distributivity for the operator over the matrix product $\bigcirc$ (i.e. $F \bigcirc (B \cdot r)) = (F \bigcirc B) \cdot z$) while introducing non-linearities, our approach balances mathematical elegance with practical tractability, remaining compatible with the original FB framework while significantly extending its representational power.

A.2.2 Our proposed operator

After establishing the theoretical foundation for enhancing the expressiveness of the FB framework through a learned operator, we explored several architectural designs to implement this approach. Our goal was to develop an operator structure that balances expressiveness with computational efficiency while maintaining the theoretical properties of the original framework. If over-complex operators can render the problem intractable, proposing some neural networks close to the product while being slightly more general sounded promising. For staying close to the original product, we focused on architectures that are easily able to learn back the original matrix product.

A.2.2.1 Attention Mechanism

Our initial investigation focused on Attention Mechanisms, which provide flexible, context-dependent interactions between representations. This approach seemed promising as Attention can theoretically learn the original product operation while potentially discovering more powerful combinations of forward and backward representations.

Simple Attention Layers. We experimented with several Attention variants (Self-Attention, Multi-Head Attention, Cross-Attention) with different methods of combining F and B representations (concatenation, product, sum). Surprisingly, these Attention Mechanisms alone performed worse than the original product operator. Our analysis suggests that within the span of functions learnable by Attention, the dot product remains an optimal combination for the FB framework. The Attention Mechanisms appeared to approximate the product operation without discovering more effective alternatives.

Attention-Based Operator To address this limitation, we developed a more sophisticated architecture incorporating Cross-Attention Blocks within a deeper network (Figure A.1). Its full implementation is provided in Appendix C.1.

Limitations and Challenges. While this architecture demonstrated improved performance in some environments, it introduced several challenges.

Because of the non-modular structure and high level of parameters, **further scaling** of the operator is impossible. The operator structure contains 68,749 trainable parameters compared to 151,346 for B and 3,939,428 for F. While smaller than the primary representation networks, further scaling for improving performance or adapting to bigger en-

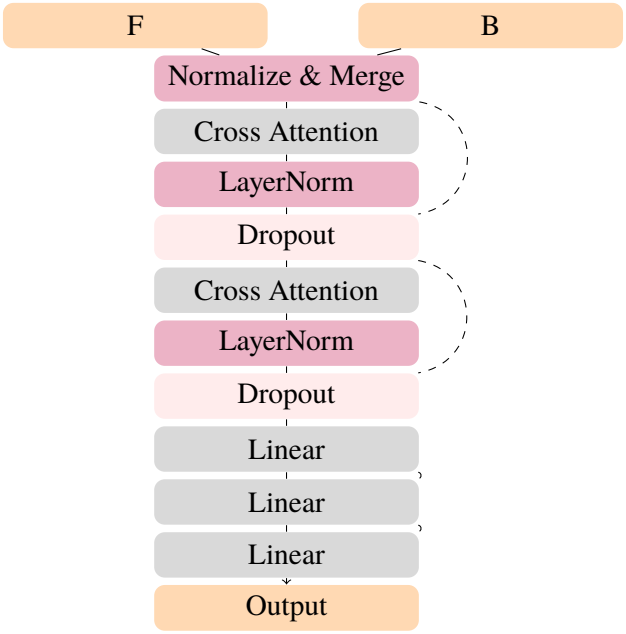


Figure A.1 Proposed Attention Architecture with updated residual connections

vironments is not feasible, without storing environment-specific information, potentially encroaching on the roles assigned to the representation networks themselves.

In addition, the operator is strongly **coupled** with specific F and B architectures. Because of it, its performance is degrading significantly when the representation networks change.

Moreover, the structure shows high **hyperparameter sensitivity**: The architecture demonstrates significant sensitivity to hyperparameter choices, compromising robustness across different environments.

These limitations highlight the need for further research to develop more scalable operator architectures. We seek to balance expressiveness with parameter efficiency while maintaining architectural independence from specific F and B implementations.

A.2.2.2 MLP-Based Approaches and Taylor Representations

Multilayer Perceptrons (MLPs) offer significant flexibility in modeling complex relationships. However, applying an MLP naively to combine F and B representations presents challenges, particularly in maintaining the core assumption of the FB framework, the distributivity of the matrix product over our operator $\bigcirc$: $Q = M \cdot r = F \bigcirc B \cdot r = F \bigcirc z$.

Challenges with Direct MLP Application. When an MLP is used to combine F and B, the linear relationship $Q = Fz$ no longer holds. To address this, we explored reconstruct-

ing Q from M, which requires estimating the reward r during training using the implicit reward (cf. Equation 4.8).

Using this implicit reward, we can reconstruct Q from M:

$$Q = (F \bigcirc B(s')) \cdot r_{\text{implicit}} \tag{A.1}$$

While this approach allows for more complex relationships between F and B, it introduces additional computational complexity and risks becoming intractable. It grants excessive freedom to F, potentially blurring the distinct roles of F and B in learning representations, and compromises the interpretability of the model.

DenseNet or ResNet as an Extension of MLP. Our experiments revealed that simpler MLP structures often outperformed more complex architectures like DenseNet or ResNet when used as operators. The increased complexity of these advanced architectures appeared to hinder learning rather than enhance it, confirming the intuition about the tractability of the problem.

Taylor's Representation as an Extension of MLP. To balance the expressiveness of MLPs with the stability of linear models, we explored neural network architectures explicitly designed to learn function decompositions, such as Fourier or Taylor expansions. Since we want our operator to easily learn back a product, and build around, Taylor decomposition is the best choice. This approach allows us to investigate the benefits of providing affine freedom to the model while maintaining a clear mathematical interpretation.

While some recent work has focused on modeling very high-order functions or generalizing complex existing neural networks [6,7], our requirements called for a simpler approach. We implemented a Taylor second order network structure (detailed in Appendix C.5) to approximate second-order Taylor expansions of the operator function.

The Taylor second order approach offers several advantages. Its main strength is **interpretability**: The network's structure directly corresponds to terms of a Taylor expansion, providing clear mathematical meaning to its operations. This enables to infirm or confirm the efficiency of giving affine freedom to the model [8].In addition, we have a **controlled complexity** over it: By limiting the expansion to second order, we maintain a balance between expressiveness and simplicity. Finally, it provides **efficient parameterization**: The Taylor-inspired structure allows for rich non-linear interactions with a relatively small number of parameters.

Results and Implications. Our experiments with the Taylor second order structure yielded promising results, demonstrating improvements over the simple $F \cdot B$ product while maintaining a compact parameter count. This suggests that allowing for controlled non-linear interactions between F and B representations can enhance the model's expressiveness without sacrificing stability or interpretability.

Since these NN structures are still limited in complexity of the learned relationships, choice has been made to scale to Transformers, a much more widespread and powerful architecture.

A.2.2.3 Transformer-Based Operator

Following the successful application of Transformers to F and B representations (detailed in Section 3.2), we explored scaling the operator to a Transformer architecture as well. This decision was motivated by the Transformers' demonstrated effectiveness in capturing complex relationships in various domains.

Affine Relationship Extension. Inspired by Agarwal et al. [8], we extended the operator to include an additional set of trainable parameters: $\bigcirc(F, B, c)$, where c is a constant. This modification allows the operator to express affine relationships, resulting in improved performance.

Input Combination Strategies. A key challenge in implementing a Transformer-based operator was determining the optimal method for combining F and B inputs. We experimented with several approaches:

- Concatenation
- Matrix product
- Cross-Attention
- Addition
- Element-wise product

Our experiments revealed that the element-wise product yielded the best results among these strategies.

Resulting Architecture. The final Transformer-based operator architecture (illustrated in Figure A.2) demonstrates enhanced scalability and expressiveness compared to previous iterations. This structure effectively balances the need for complex relationship

modeling with computational efficiency. Detailed implementation specifics can be found in Appendix C.2.

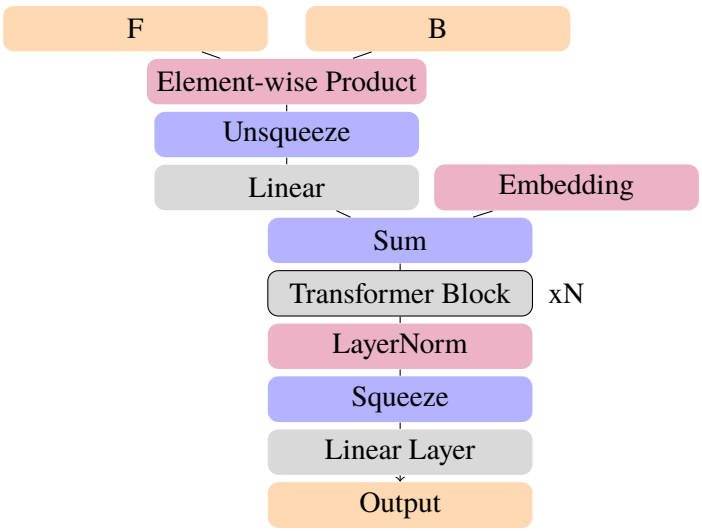


Figure A.2 Operator architecture

A.2.2.4 Scalability Difficulties

Our analysis of various operator architectures, including Attention Mechanisms, MLPs, and Transformers, highlights the Transformer-based approach as the most scalable and effective. Its element-wise input strategy and affine relationship extension provide significant performance advantages. However, the operator's effectiveness is ultimately constrained by the expressiveness of the F and B representations. To enhance the framework, we focused on scaling and refining these representations. Once they are more expressive, we were hoping the operator to be further optimized to fully harness their potential.

However, we notice experimentally that scaling F leads to much better results than obtained through adding an operator. Moreover, once using scaled F and B, then adding operator decreases the performance of the model. This phenomenon occurs for all the different types of operator that we developed. If the simplest architectures, like simple Attention, do not bring performance improvements, bigger architectures give too much freedom to the problem and render it intractable when scaling, for example, when relying on more expressive F and B structures. Our goal is to develop a framework that is more performant and can be used as a basis for more scalable models. For this aim, the operator does not provide good results enough, especially given the significant rise in computing cost that it implies. Therefore, we abandon the operator.

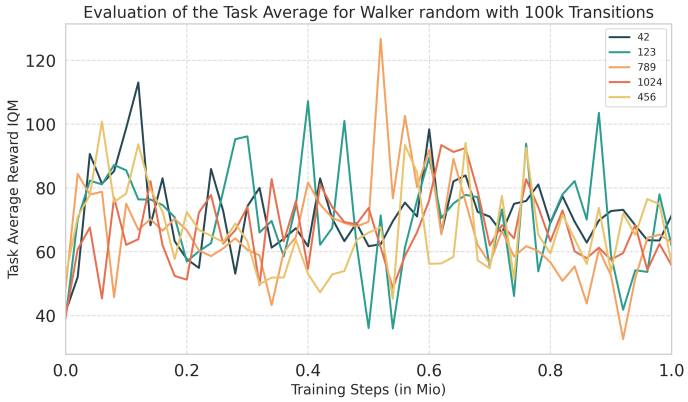
References

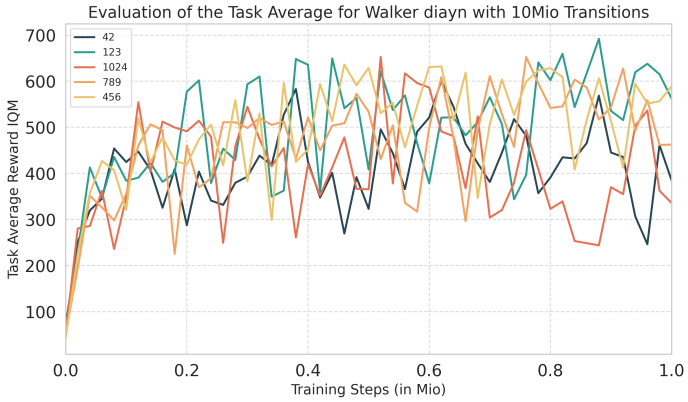
[1] Cetin E, Touati A, Ollivier Y. Finer behavioral foundation models via autoregressive features and advantage weighting: arXiv:2412.04368[M/OL]. arXiv, 2024[2024-12-16]. https://arxiv.or g/abs/2412.04368. DOI: 10.48550/arXiv.2412.04368 [2] Park S, Kreiman T, Levine S. Foundation policies with Hilbert representations: arXiv:2402.15567[M/OL]. arXiv, 2024[2024-04-08]. https://arxiv.org/abs/2402.15567.
[3] Touati A, Ollivier Y. Learning one representation to optimize all rewards[J/OL]. Advances in Neural Information Processing Systems, 2021, 34:13–23[2024-04-07]. https://arxiv.org/abs/2103.07945.
[4] Touati A, Rapin J, Ollivier Y. Does zero-shot reinforcement learning exist?[M/OL]. arXiv, 2023 [2024-04-07]. https://arxiv.org/abs/2209.14935. DOI: 10.48550/arXiv.2209.14935.
[5] Jeen S, Bewley T, Cullen J. Zero-shot reinforcement learning from low quality data[J]. Advances in Neural Information Processing Systems, 2024, 37: 16894-16942.
[6] TaylorNet: A Taylor-driven generic neural architecture[EB/OL]. [2024-12-29]. httpss://openreview.net/pdf?id=tDNGHd0QmzO.
[7] Xiao T, Zhang W, Cheng Y, et al. HOPE: High-order polynomial expansion of black-box neural networks[A/OL]. 2023[2024-12-29]. arXiv: 2307.08192. https://arxiv.org/abs/2307.08192.10
[8] Agarwal S, Sikchi H, Stone P, et al. Proto Successor Measure: Representing the space of all possible solutions of reinforcement learning: arXiv:2411.19418[M/OL]. arXiv, 2024[2024-12- 04]. https://arxiv.org/abs/2411.19418. DOI: 10.48550/arXiv.2411.19418.

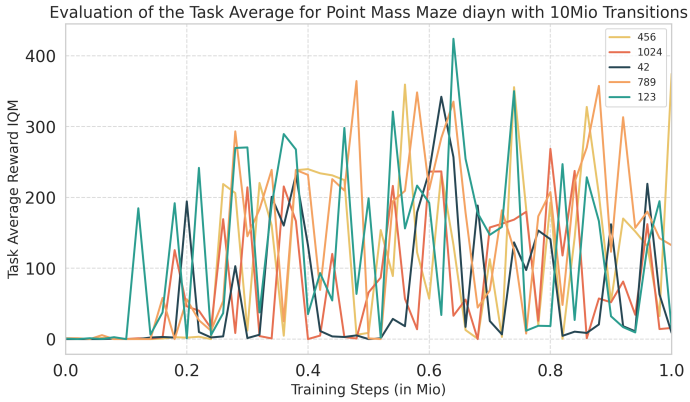
APPENDIX B DETAILED RESULTS AND TRAINING CRUVES

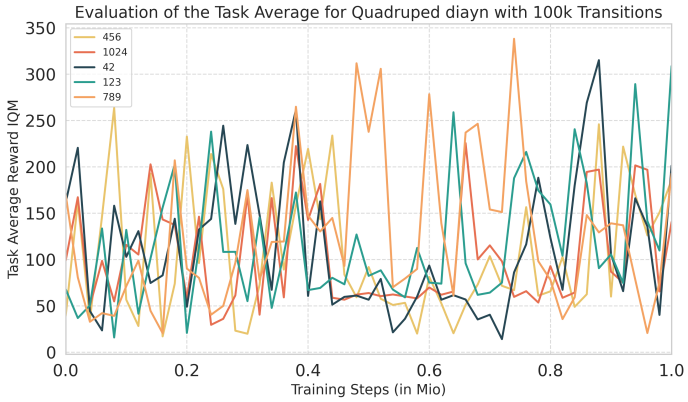
B.1 Final Model

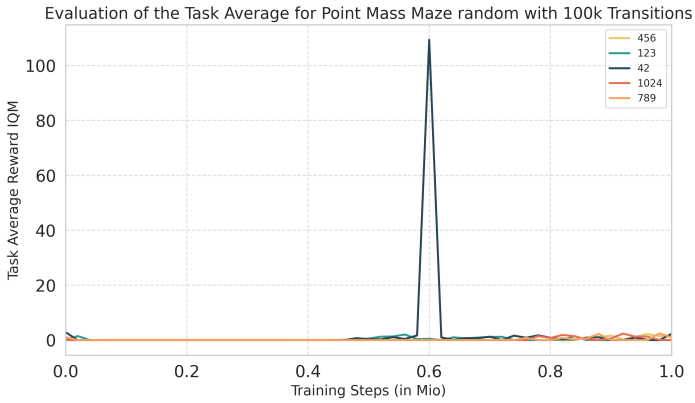
We report here below the training curves of our model for the different tasks and datasets. The “Task Average” corresponds to the average performance over the four tasks used for evaluation for a given domain. The training curves of the different seeds are presented to show in details the variance of the training process. Each curve corresponds to a different random seed, with the seed value indicated in the legend.

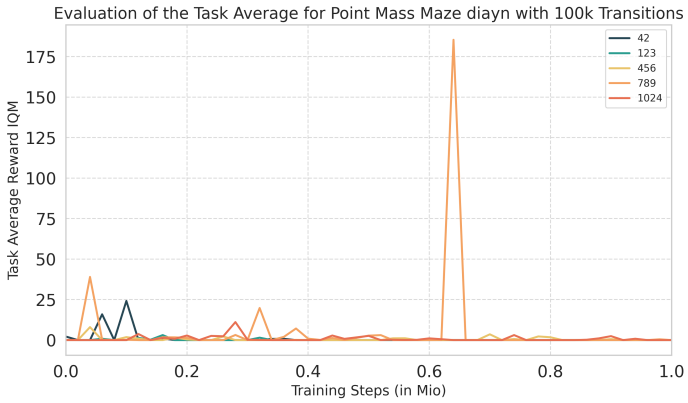

Evaluation of the Task Average for Walker random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
789
1024
456

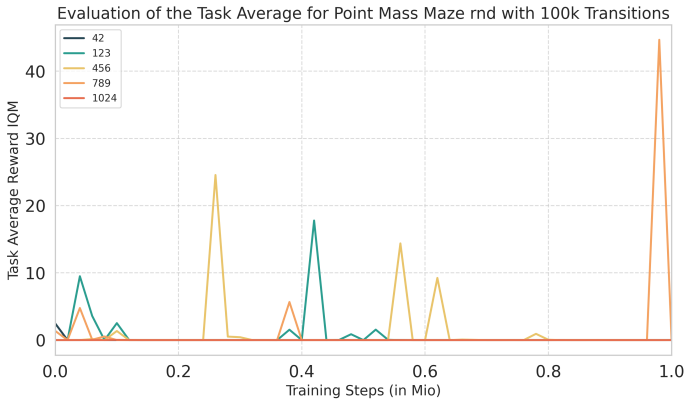

Evaluation of the Task Average for Walker diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
1024
789
456

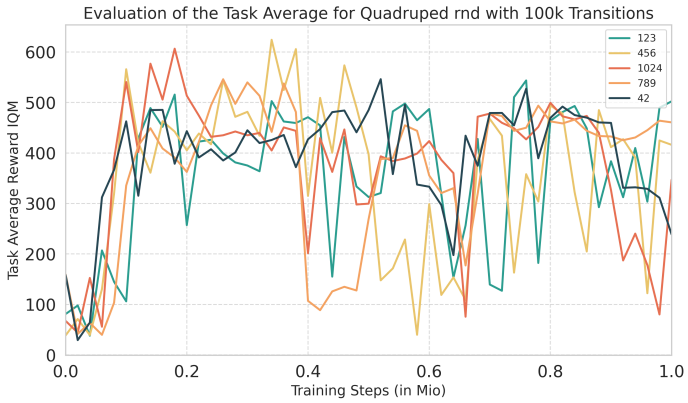

Evaluation of the Task Average for Point Mass Maze diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
789
123

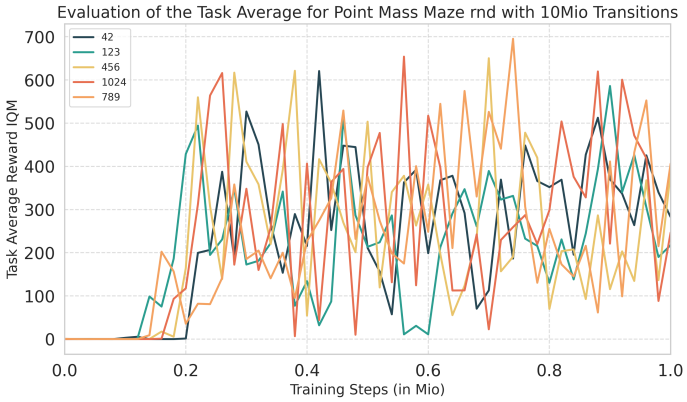

Evaluation of the Task Average for Quadruped diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
123
789

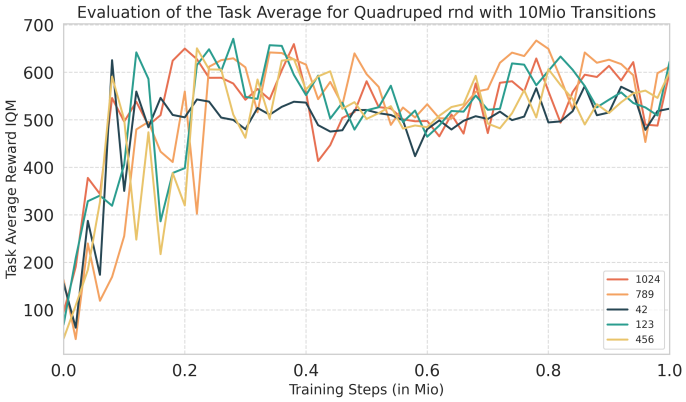

Evaluation of the Task Average for Point Mass Maze random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
123
42
1024
789

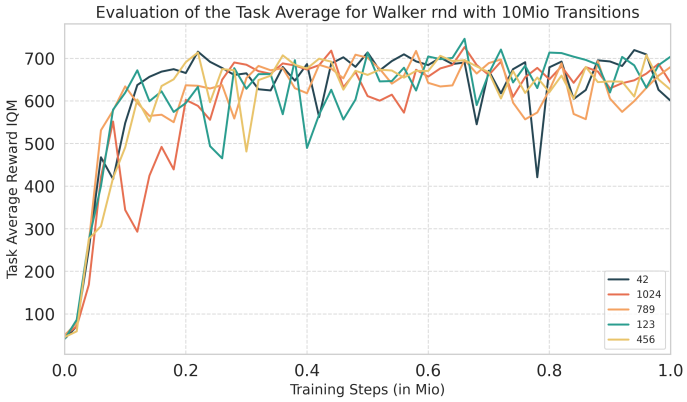

Evaluation of the Task Average for Point Mass Maze diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

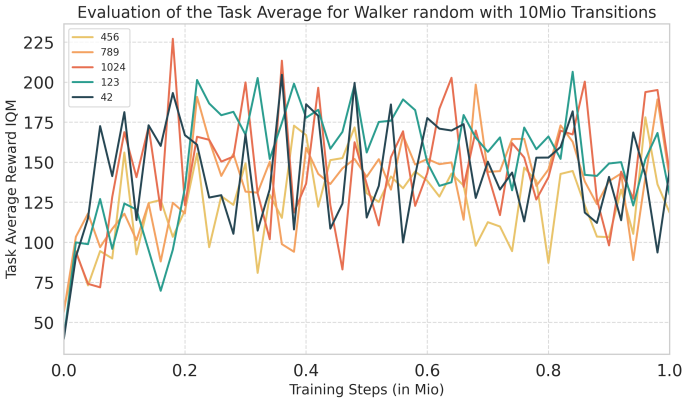

Evaluation of the Task Average for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

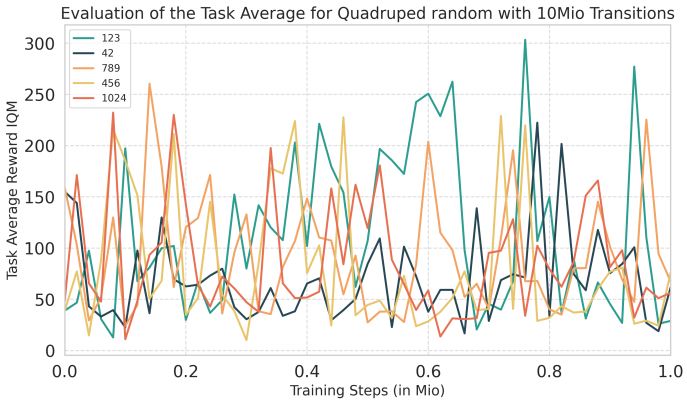

Evaluation of the Task Average for Quadruped rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
456
1024
789
42

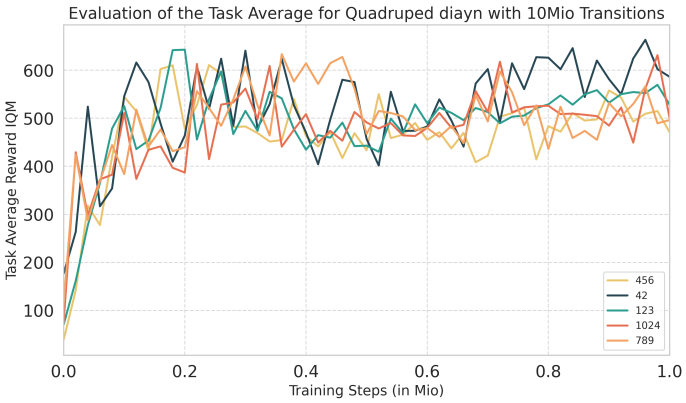

Evaluation of the Task Average for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
1024
789

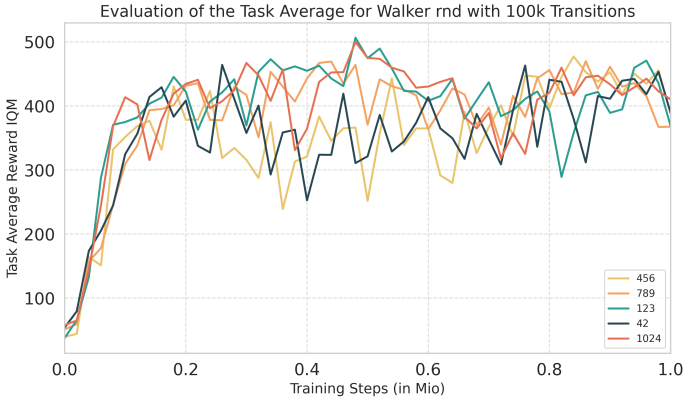

Evaluation of the Task Average for Quadruped rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
1024
789
42
123
456

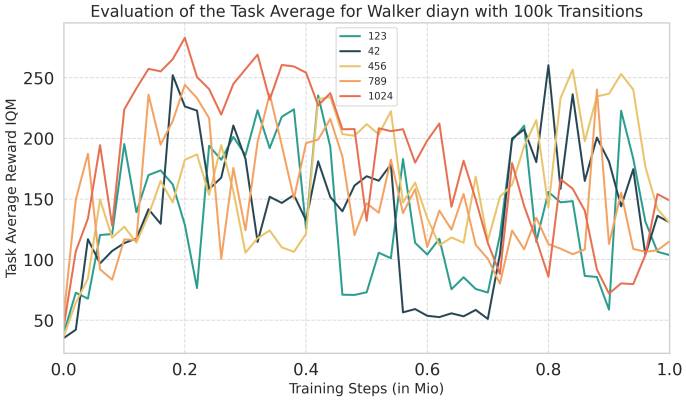

Evaluation of the Task Average for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789
123
456

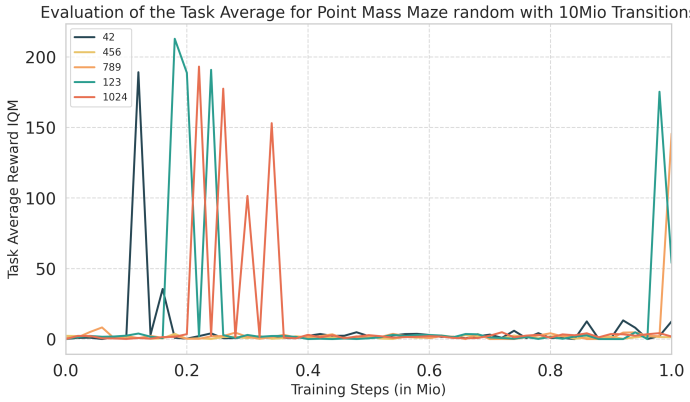

Evaluation of the Task Average for Walker random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
1024
123
42

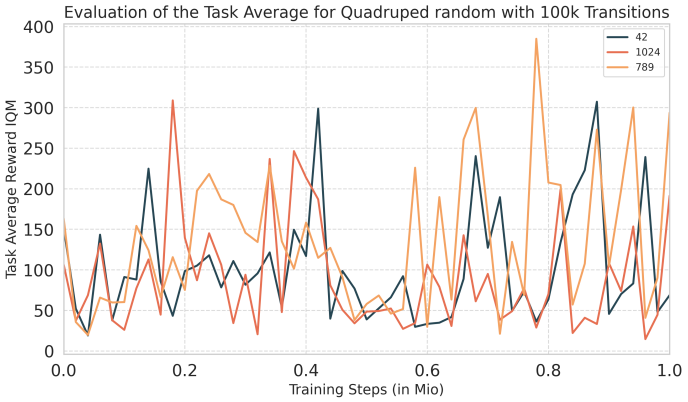

Evaluation of the Task Average for Quadruped random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
789
456
1024

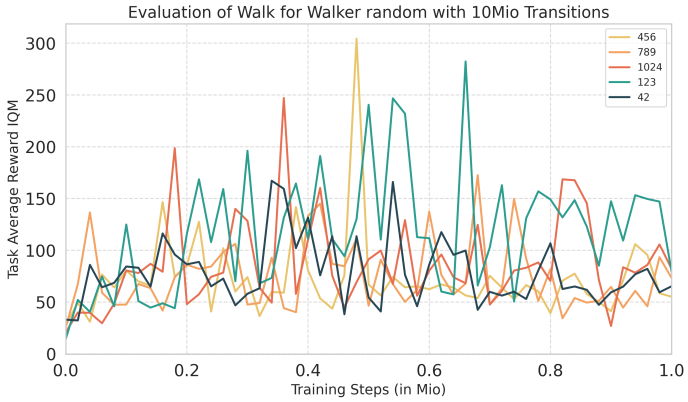

Evaluation of the Task Average for Quadruped diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
42
123
1024
789

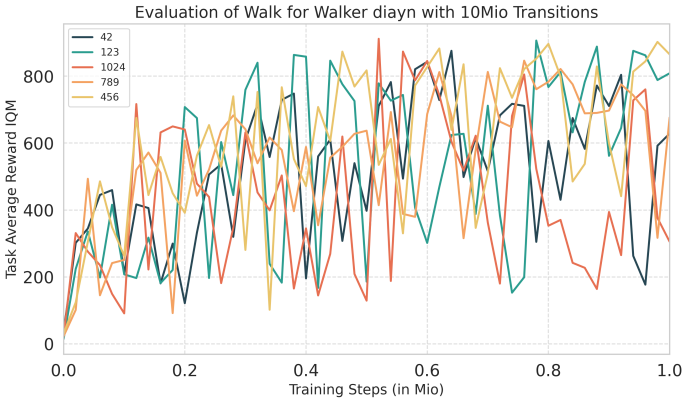

Evaluation of the Task Average for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
123
42
1024

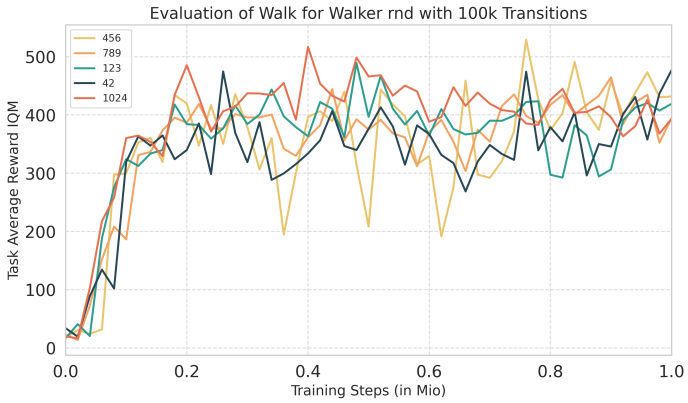

Evaluation of the Task Average for Walker diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
456
789
1024

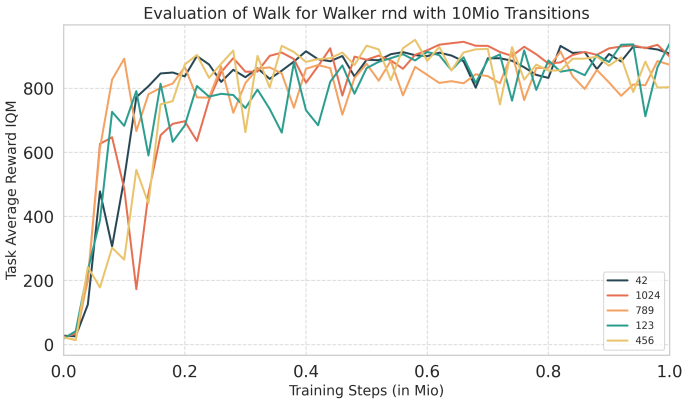

Evaluation of the Task Average for Point Mass Maze random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
456
789
123
1024

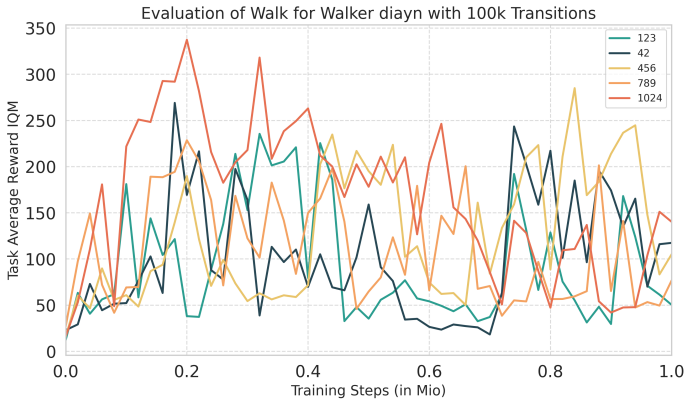

Evaluation of the Task Average for Quadruped random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789

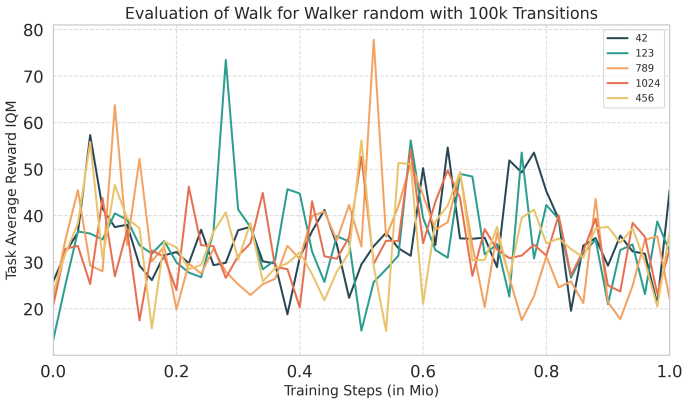

Evaluation of Walk for Walker random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
1024
123
42

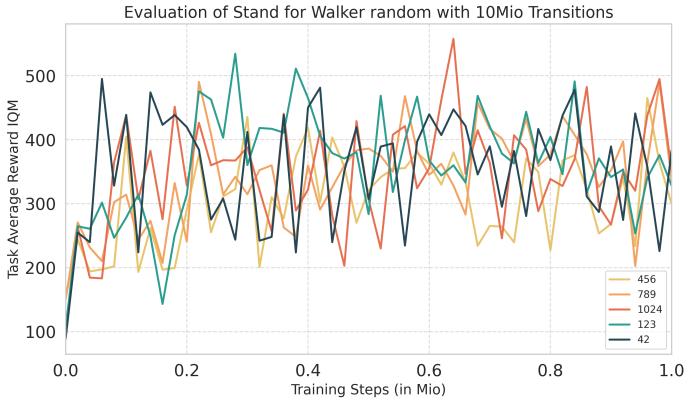

Evaluation of Walk for Walker diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
1024
789
456

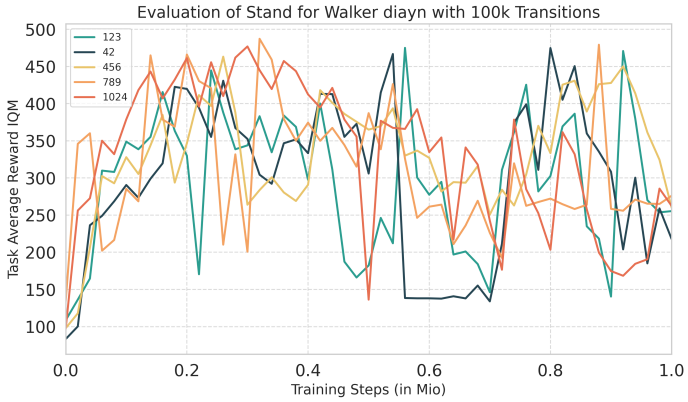

Evaluation of Walk for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
123
42
1024

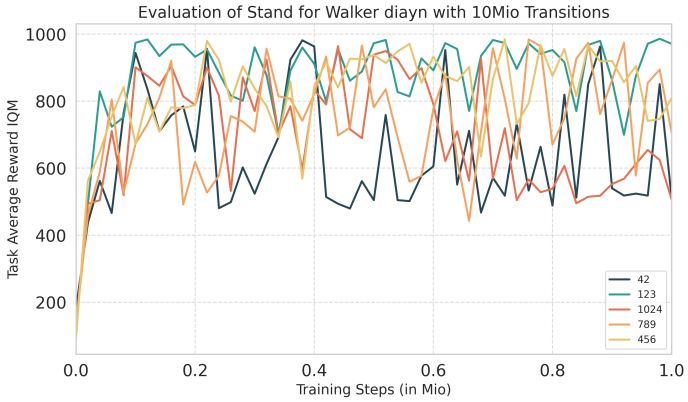

Evaluation of Walk for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789
123
456

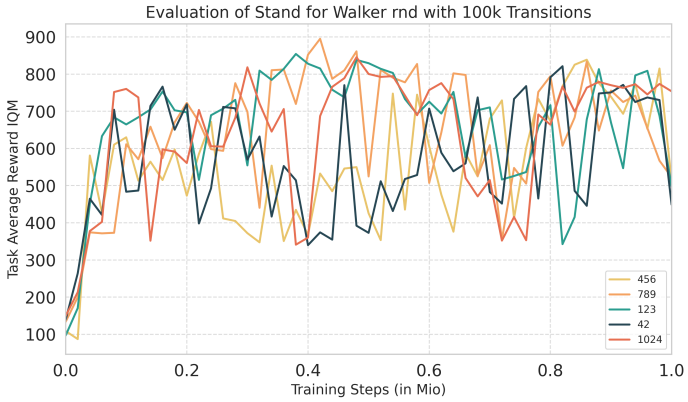

Evaluation of Walk for Walker diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
456
789
1024

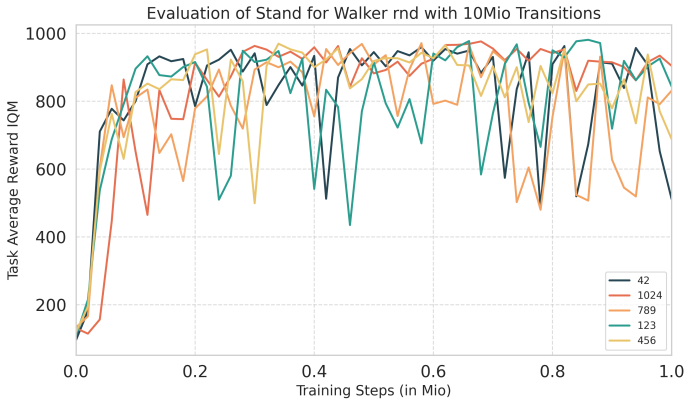

Evaluation of Walk for Walker random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
789
1024
456

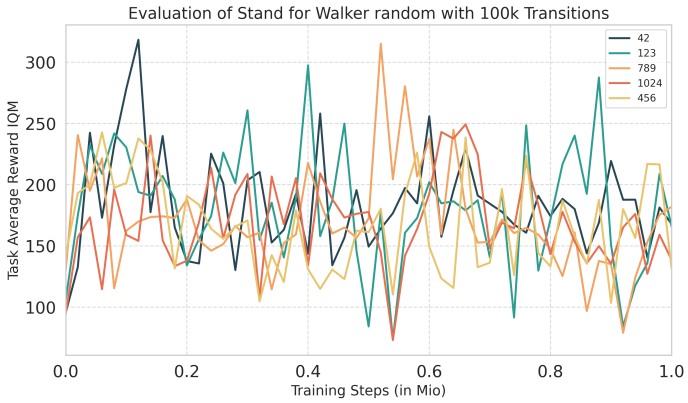

Evaluation of Stand for Walker random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
1024
123
42

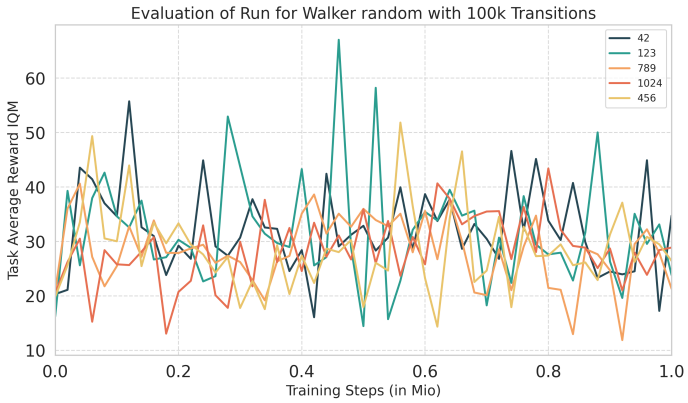

Evaluation of Stand for Walker diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
456
789
1024


Evaluation of Stand for Walker diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
1024
789
456

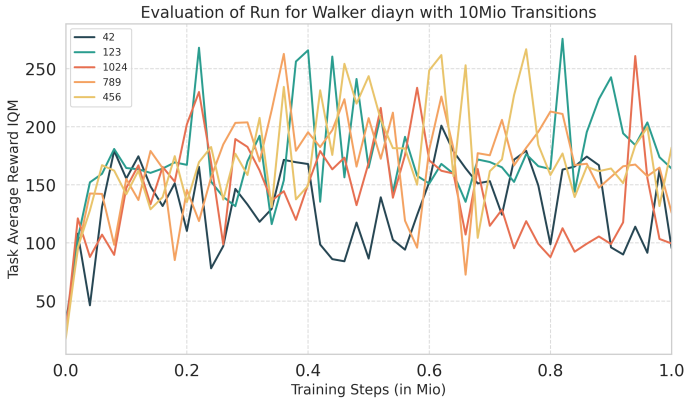

Evaluation of Stand for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
123
42
1024

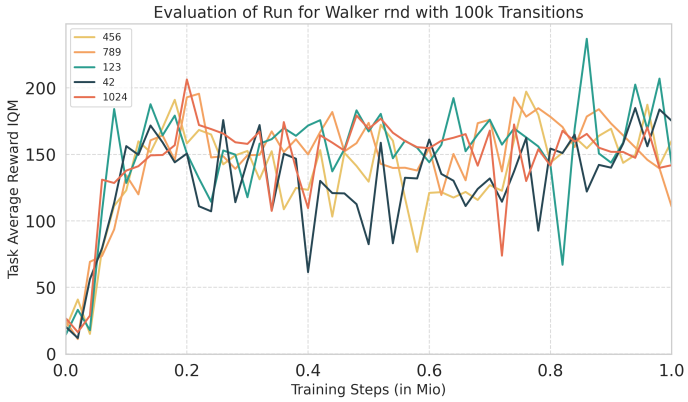

Evaluation of Stand for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789
123
456

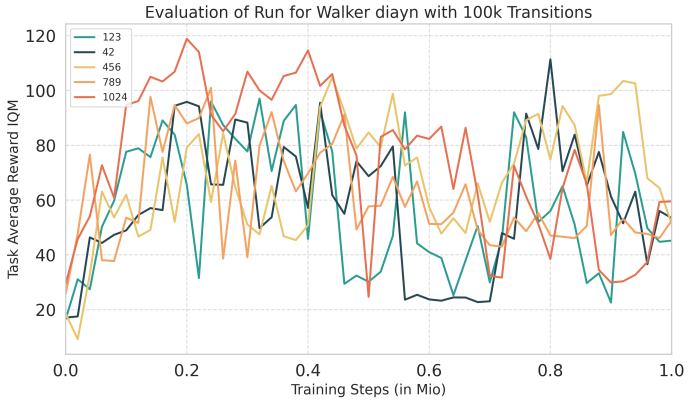

Evaluation of Stand for Walker random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
789
1024
456

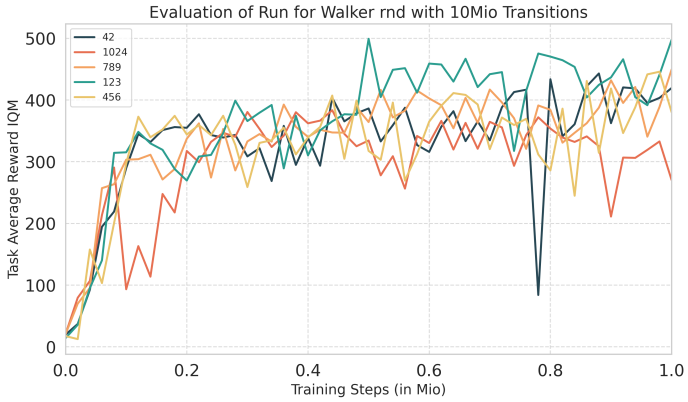

Evaluation of Run for Walker random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
789
1024
456

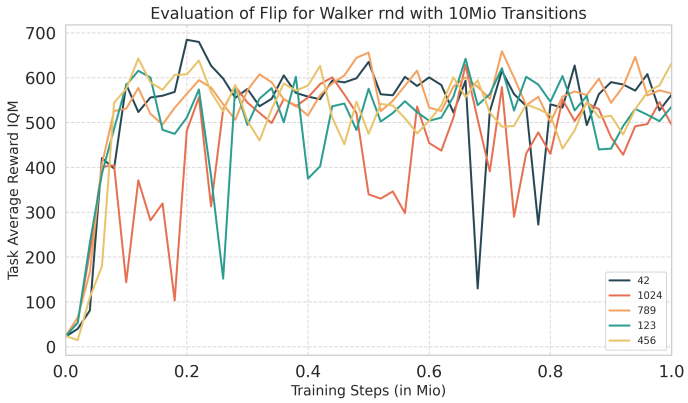

Evaluation of Run for Walker random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
1024
123
42

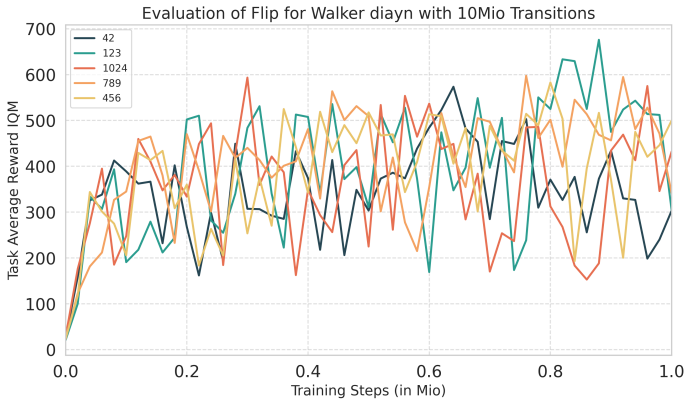

Evaluation of Run for Walker diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
1024
789
456

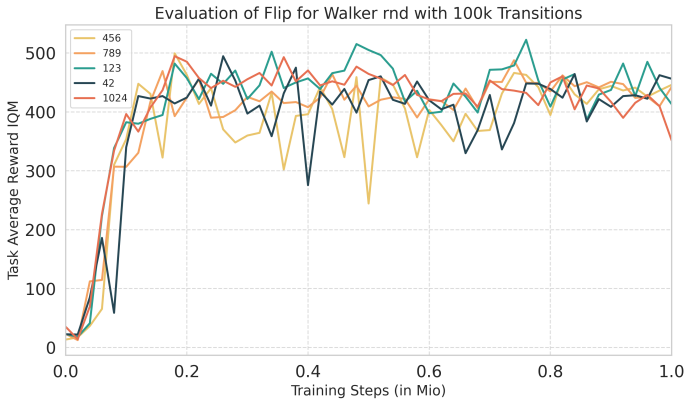

Evaluation of Run for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
123
42
1024

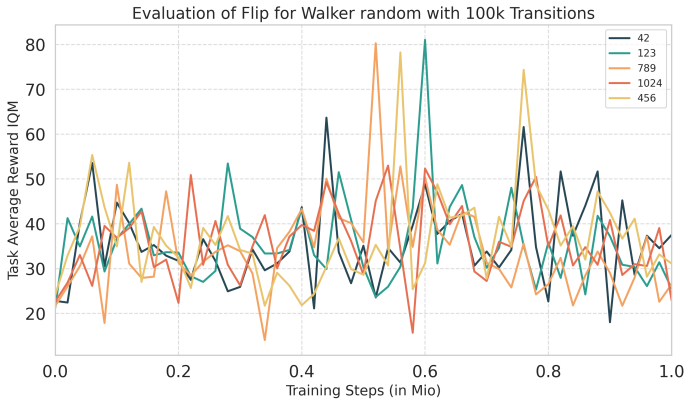

Evaluation of Run for Walker diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
456
789
1024

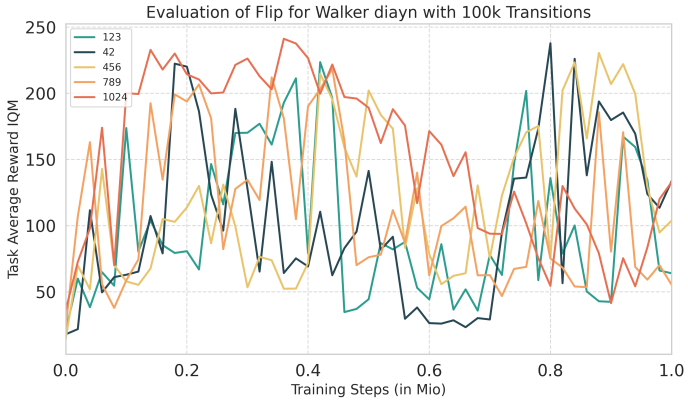

Evaluation of Run for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789
123
456

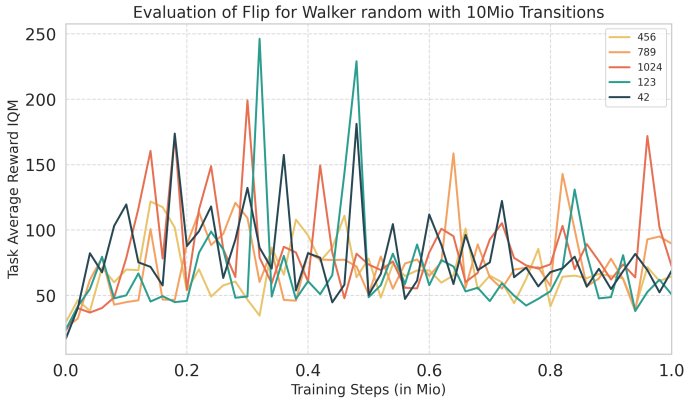

Evaluation of Flip for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789
123
456

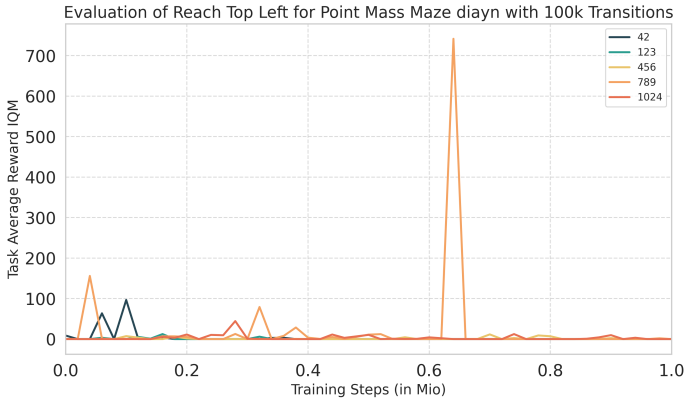

Evaluation of Flip for Walker diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
1024
789
456

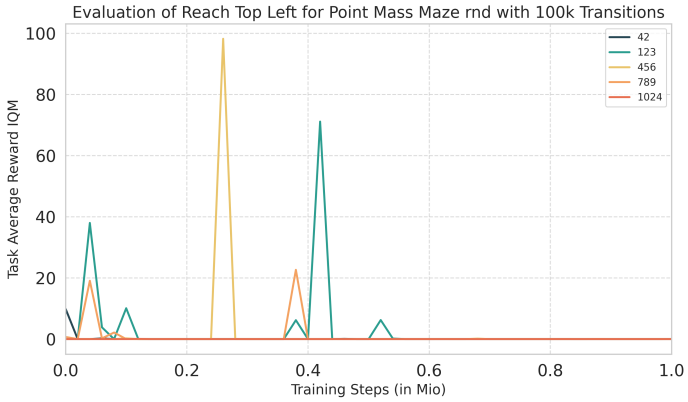

Evaluation of Flip for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
123
42
1024

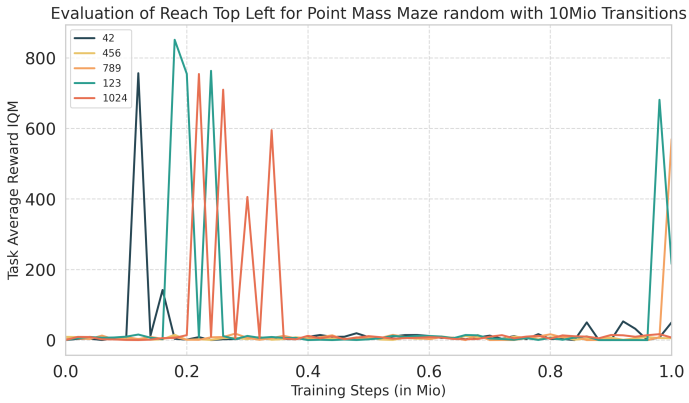

Evaluation of Flip for Walker random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
789
1024
456

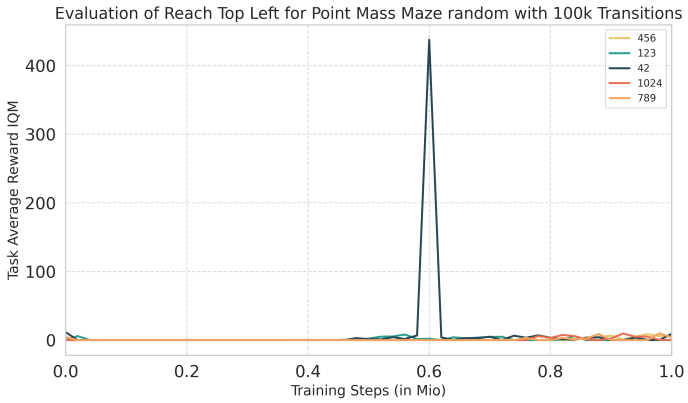

Evaluation of Flip for Walker diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
456
789
1024

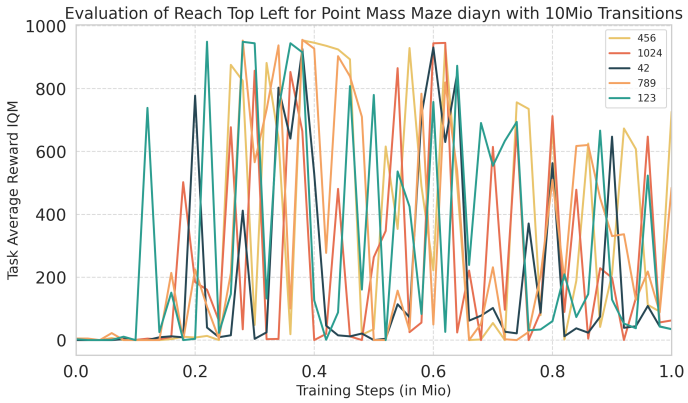

Evaluation of Flip for Walker random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
789
1024
123
42

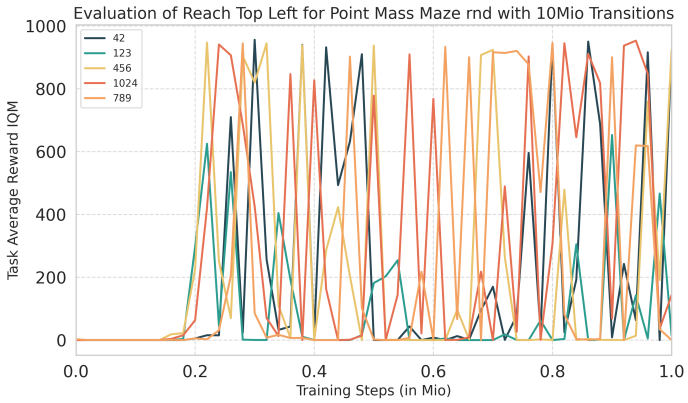

Evaluation of Reach Top Left for Point Mass Maze diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

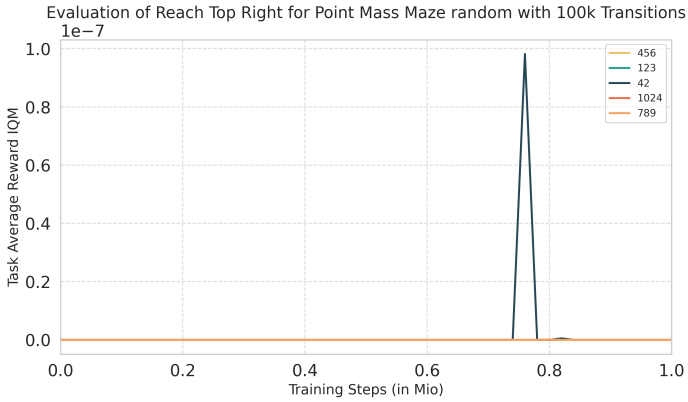

Evaluation of Reach Top Left for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

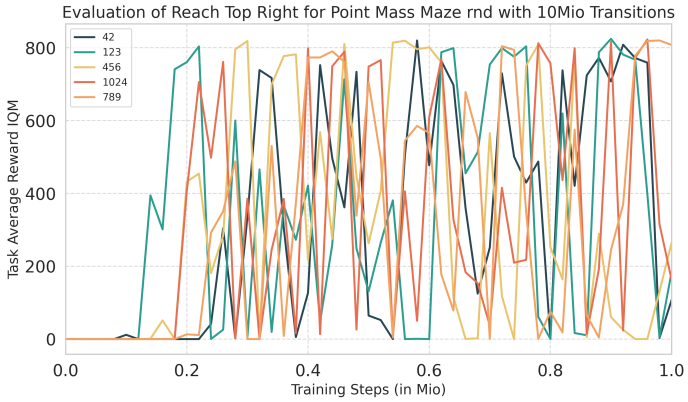

Evaluation of Reach Top Left for Point Mass Maze random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
456
789
123
1024

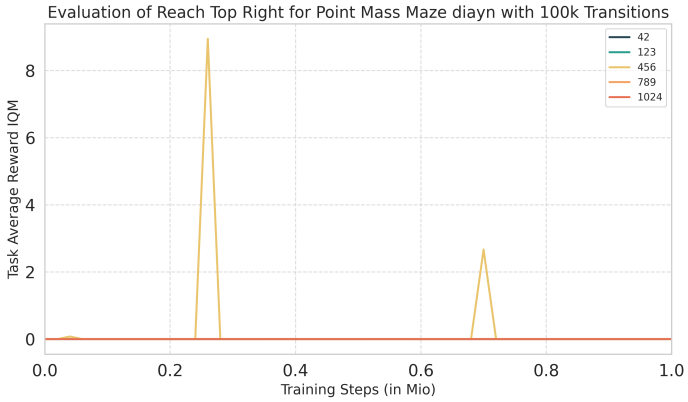

Evaluation of Reach Top Left for Point Mass Maze random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
123
42
1024
789

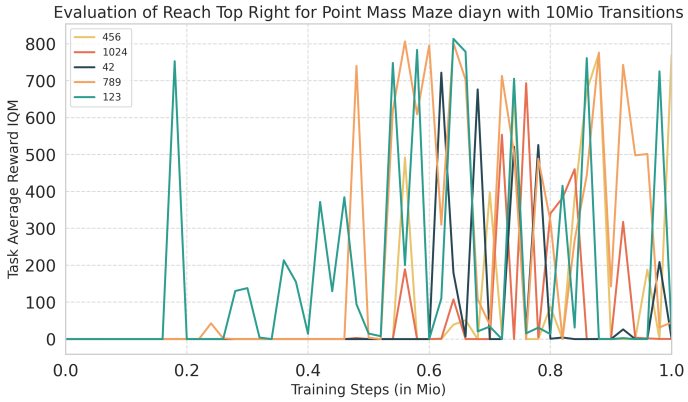

Evaluation of Reach Top Left for Point Mass Maze diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
789
123

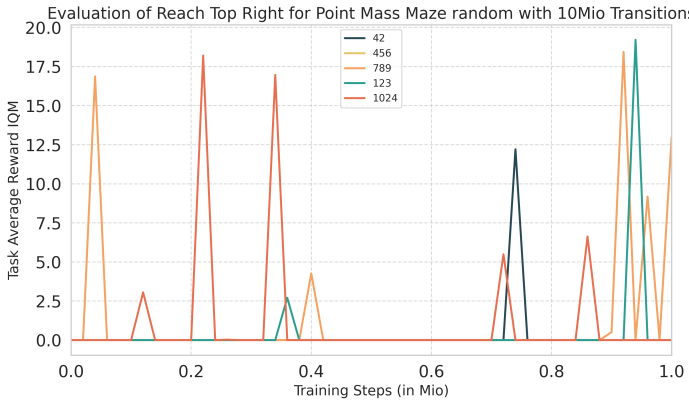

Evaluation of Reach Top Left for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
1024
789

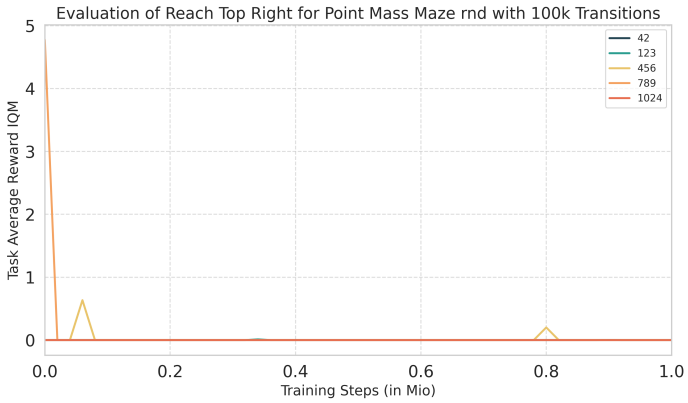

Evaluation of Reach Top Right for Point Mass Maze random with 100k Transitions
1e−7
Task Average Reward IQM
Training Steps (in Mio)
456
123
42
1024
789

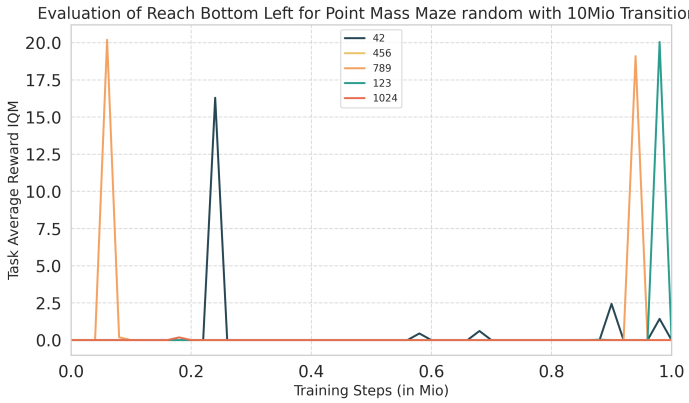

Evaluation of Reach Top Right for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
1024
789

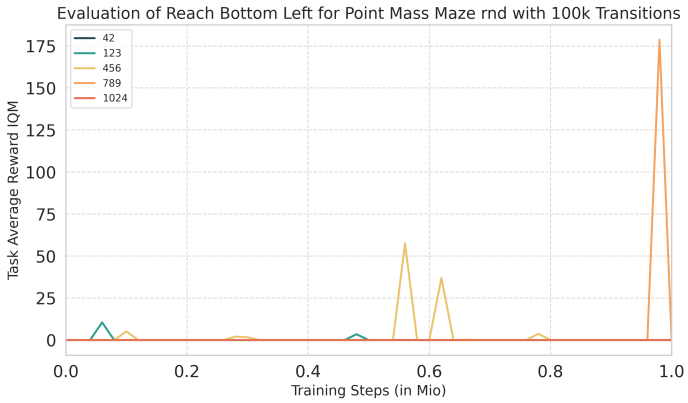

Evaluation of Reach Top Right for Point Mass Maze diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

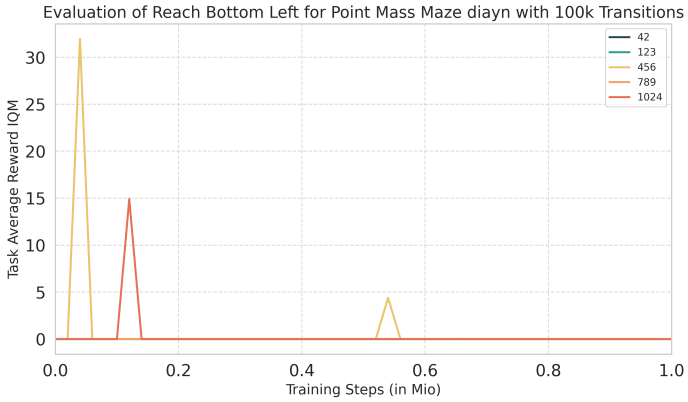

Evaluation of Reach Top Right for Point Mass Maze diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
789
123

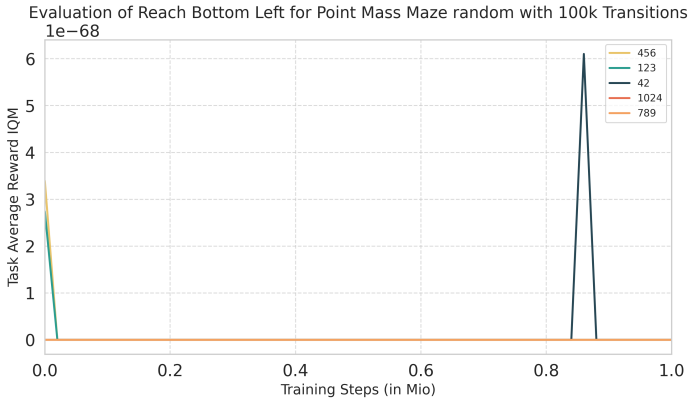

Evaluation of Reach Top Right for Point Mass Maze random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
456
789
123
1024

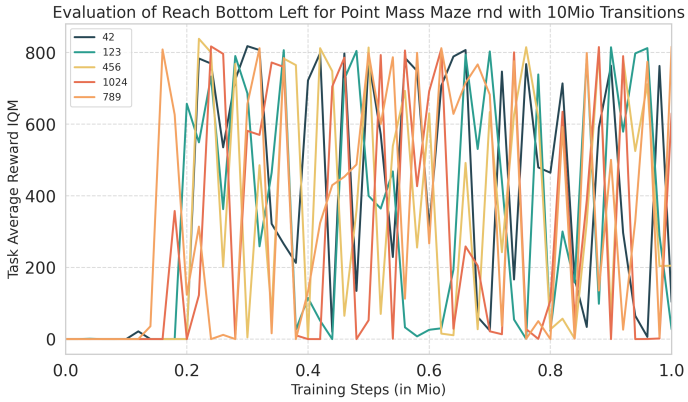

Evaluation of Reach Top Right for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

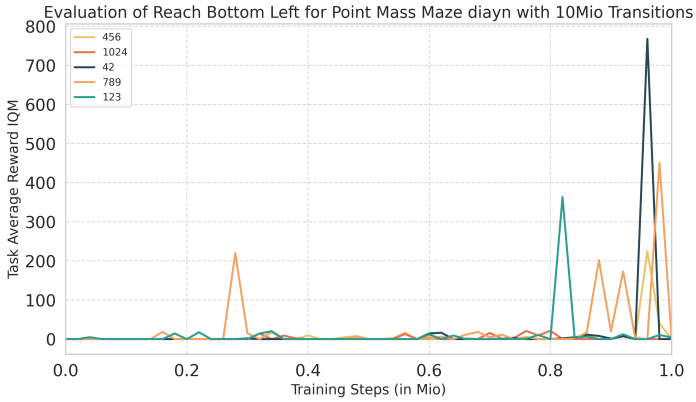

Evaluation of Reach Bottom Left for Point Mass Maze random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
456
789
123
1024

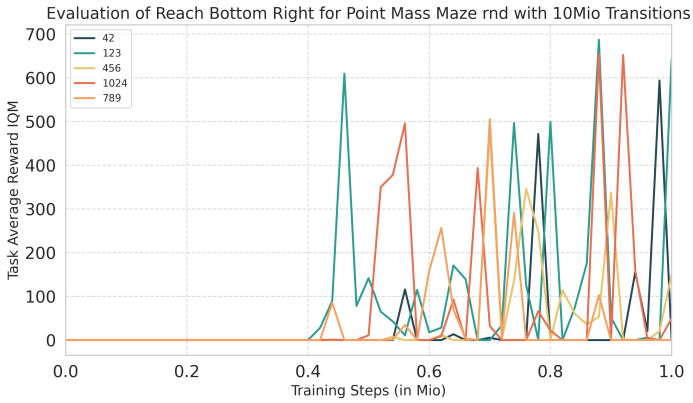

Evaluation of Reach Bottom Left for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

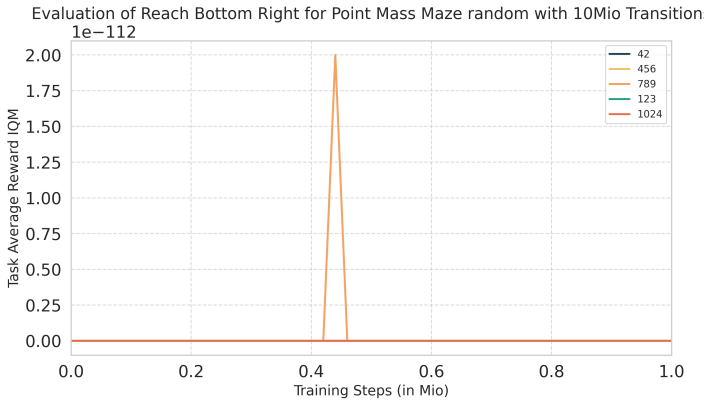

Evaluation of Reach Bottom Left for Point Mass Maze diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

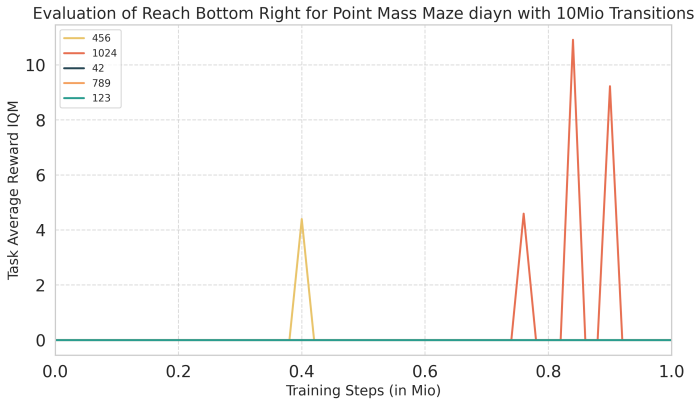

Evaluation of Reach Bottom Left for Point Mass Maze random with 100k Transitions
1e−68
Task Average Reward IQM
Training Steps (in Mio)
456
123
42
1024
789

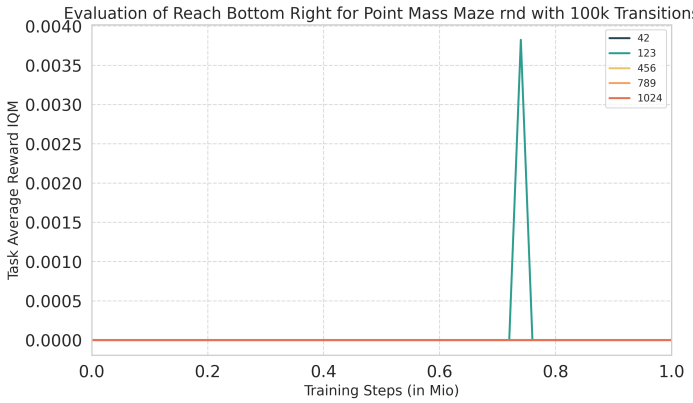

Evaluation of Reach Bottom Left for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
1024
789

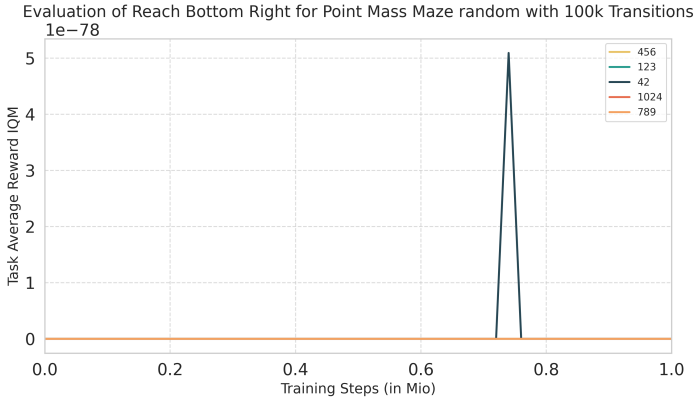

Evaluation of Reach Bottom Left for Point Mass Maze diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
789
123

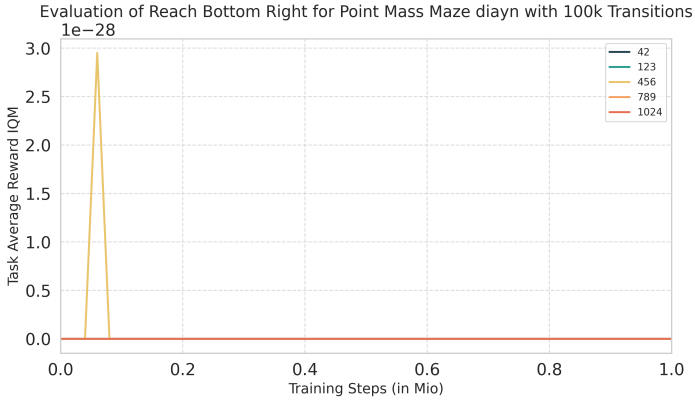

Evaluation of Reach Bottom Right for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
1024
789

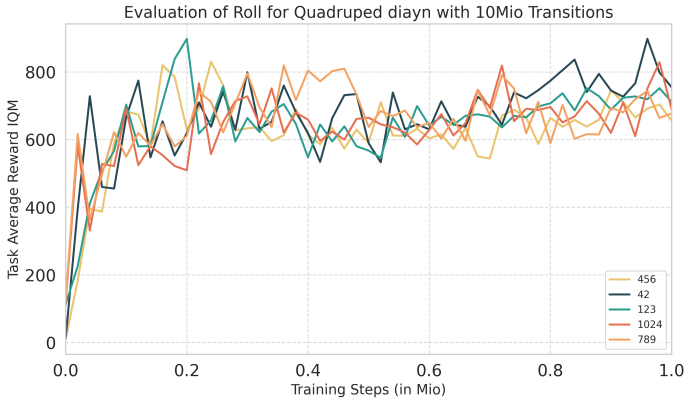

Evaluation of Reach Bottom Right for Point Mass Maze random with 10Mio Transition
1e−112
Task Average Reward IQM
Training Steps (in Mio)
42
456
789
123
1024

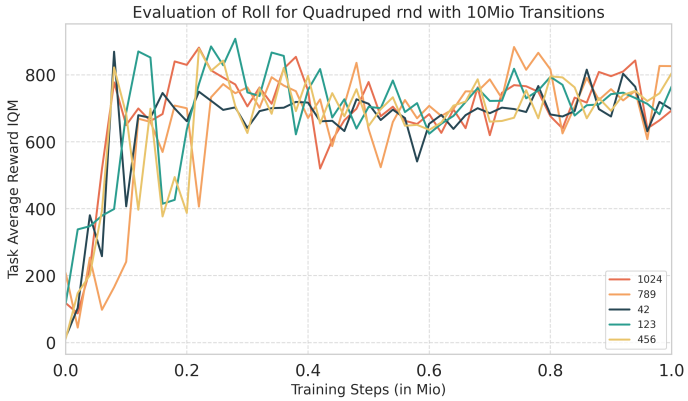

Evaluation of Reach Bottom Right for Point Mass Maze diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
789
123

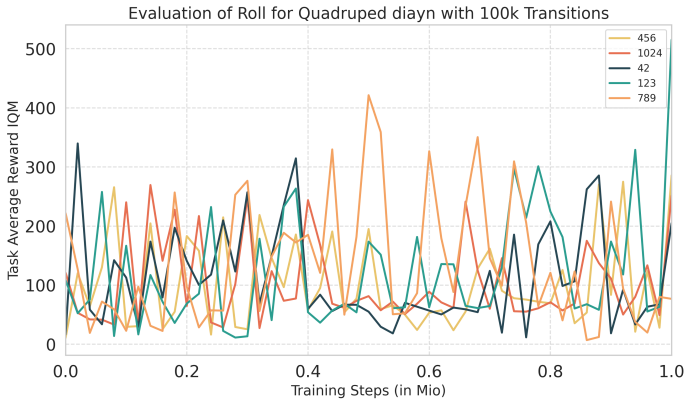

Evaluation of Reach Bottom Right for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

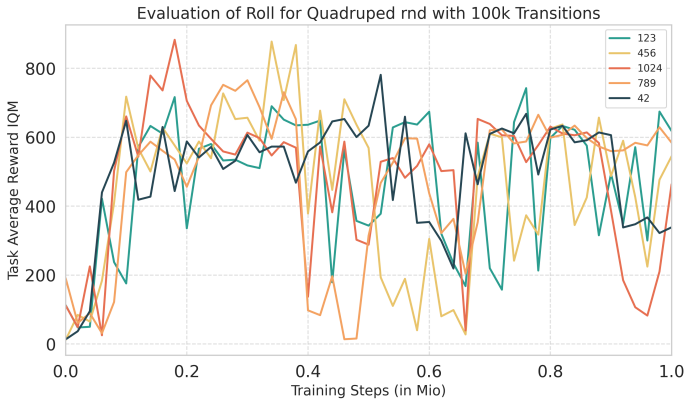

Evaluation of Reach Bottom Right for Point Mass Maze random with 100k Transitions
1e−78
Task Average Reward IQM
Training Steps (in Mio)
456
123
42
1024
789

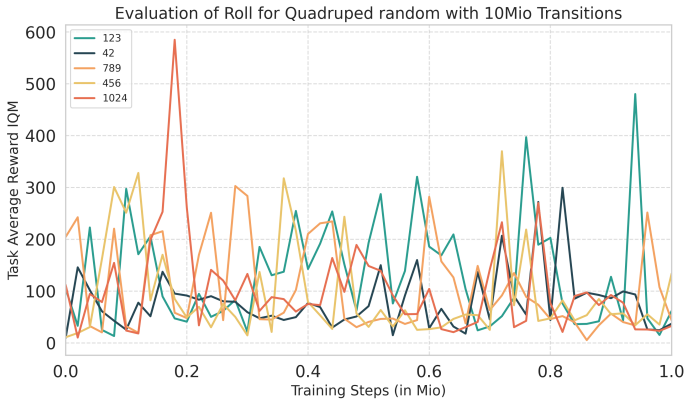

Evaluation of Reach Bottom Right for Point Mass Maze diayn with 100k Transitions
1e−28
Task Average Reward IQM
Training Steps (in Mio)
42
123
456
789
1024

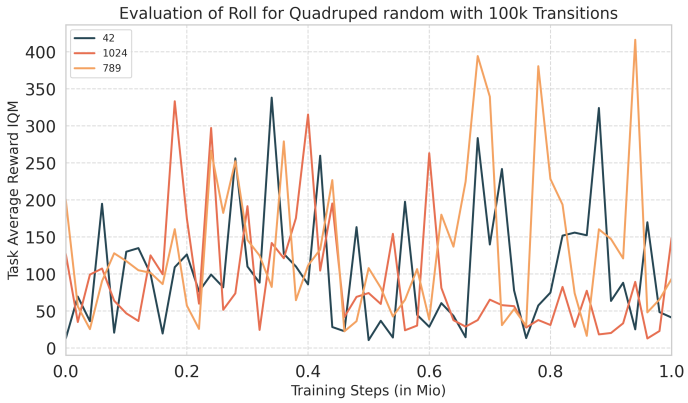

Evaluation of Roll for Quadruped diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
42
123
1024
789


Evaluation of Roll for Quadruped rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
1024
789
42
123
456

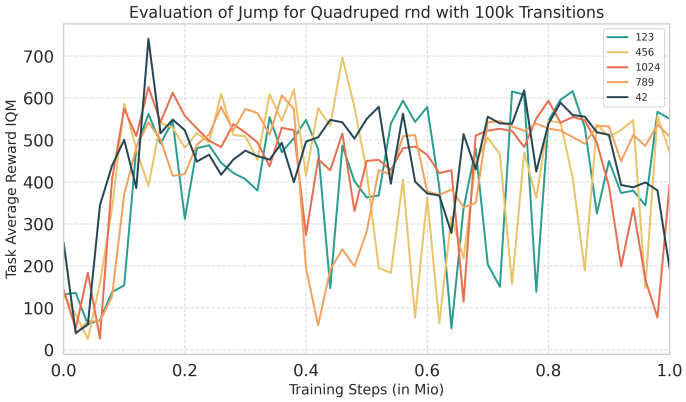

Evaluation of Roll for Quadruped diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
123
789

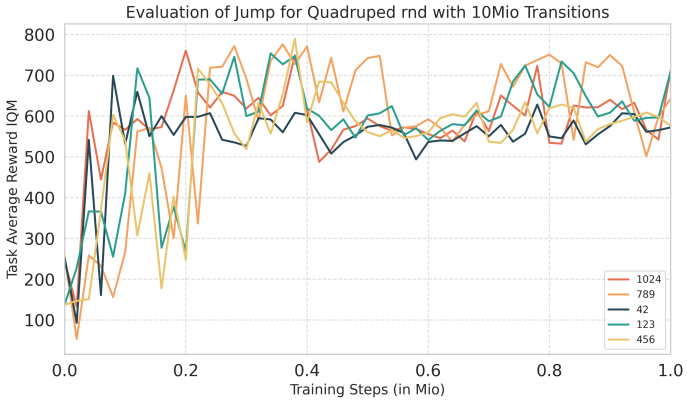

Evaluation of Roll for Quadruped rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
456
1024
789
42

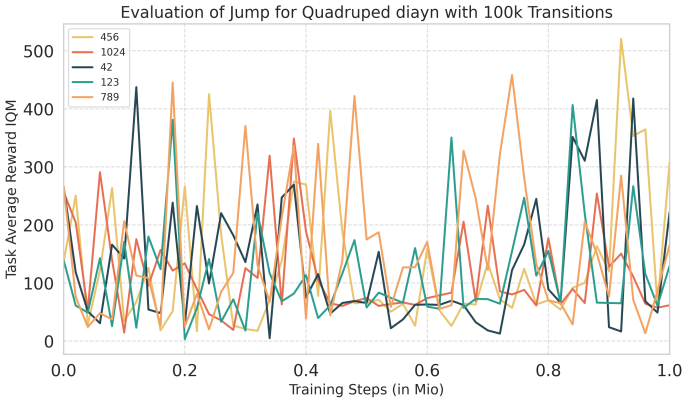

Evaluation of Roll for Quadruped random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
789
456
1024


Evaluation of Roll for Quadruped random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789

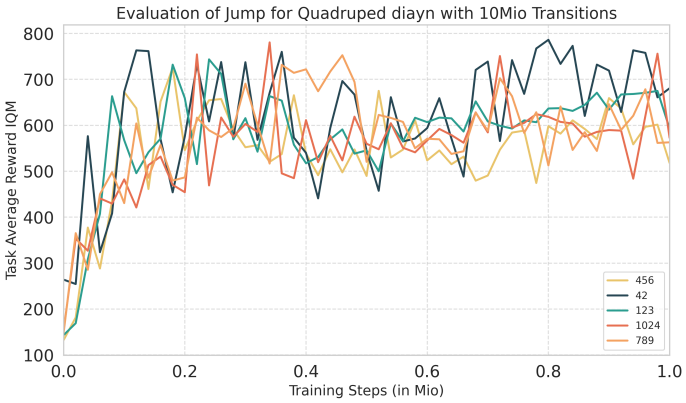

Evaluation of Jump for Quadruped random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
789
456
1024


Evaluation of Jump for Quadruped rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
456
1024
789
42


Evaluation of Jump for Quadruped rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
1024
789
42
123
456

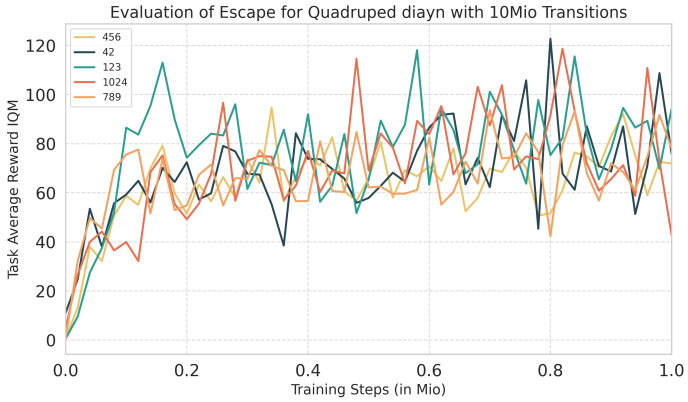

Evaluation of Jump for Quadruped diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
123
789


Evaluation of Jump for Quadruped random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789

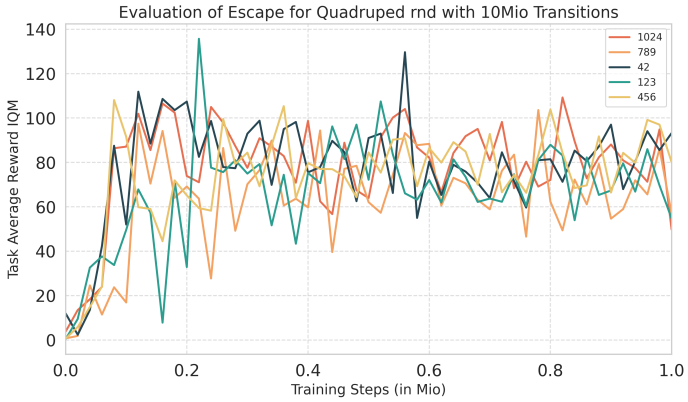

Evaluation of Jump for Quadruped diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
42
123
1024
789


Evaluation of Escape for Quadruped rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
456
1024
789
42

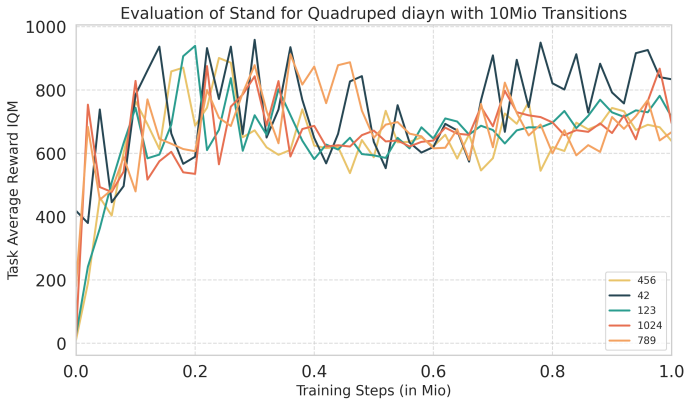

Evaluation of Escape for Quadruped random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789

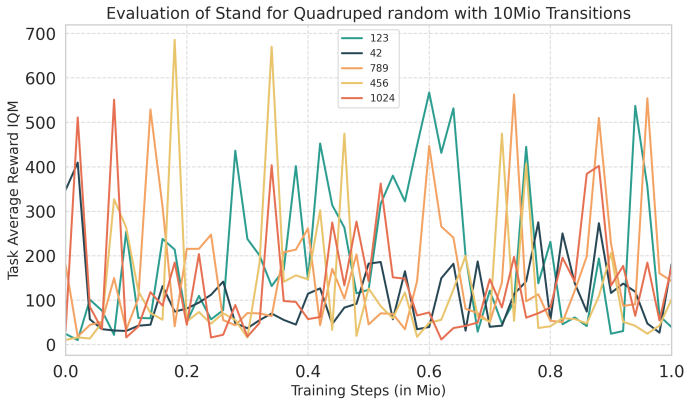

Evaluation of Escape for Quadruped diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
42
123
1024
789

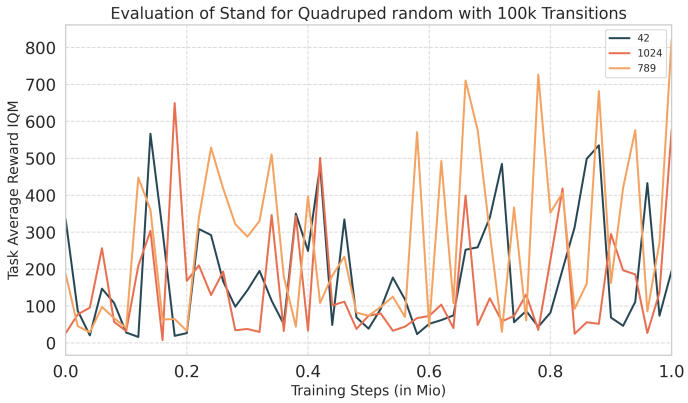

Evaluation of Escape for Quadruped random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
789
456
1024

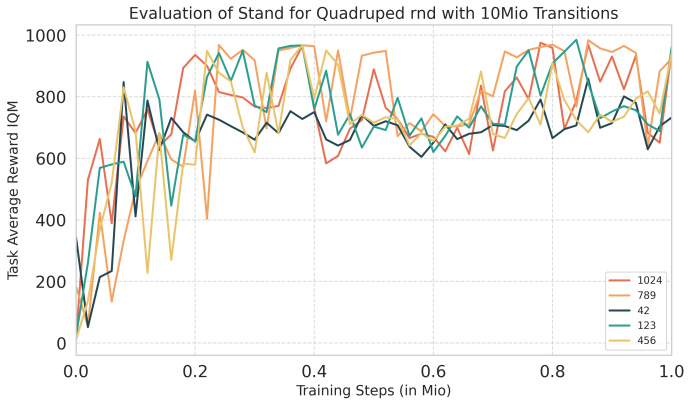

Evaluation of Escape for Quadruped rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
1024
789
42
123
456

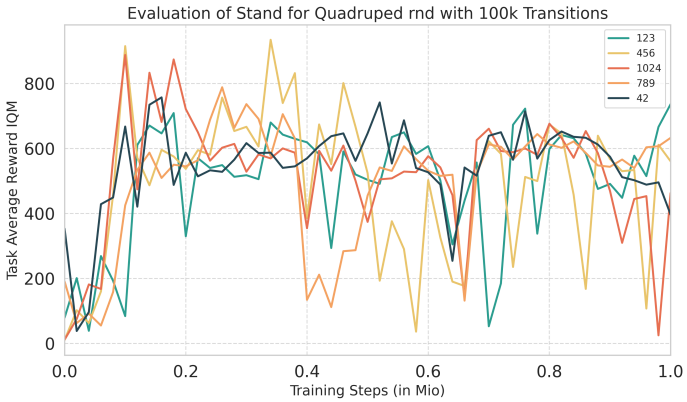

Evaluation of Escape for Quadruped diayn with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
1024
42
123
789

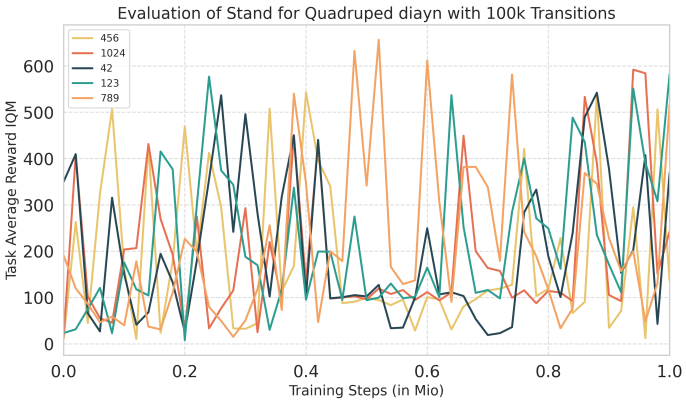

Evaluation of Stand for Quadruped diayn with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
456
42
123
1024
789


Evaluation of Stand for Quadruped random with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
123
42
789
456
1024

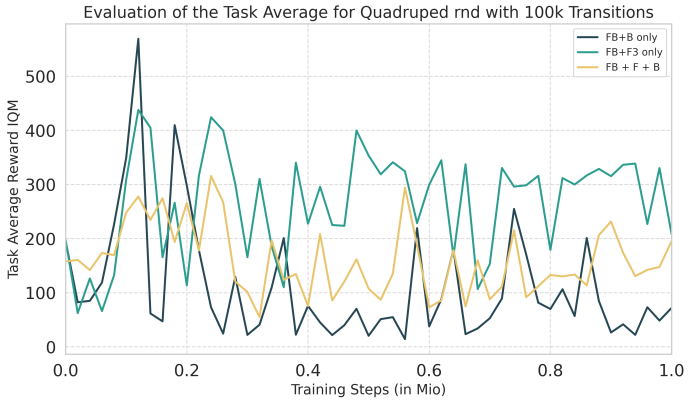

Evaluation of Stand for Quadruped random with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
42
1024
789

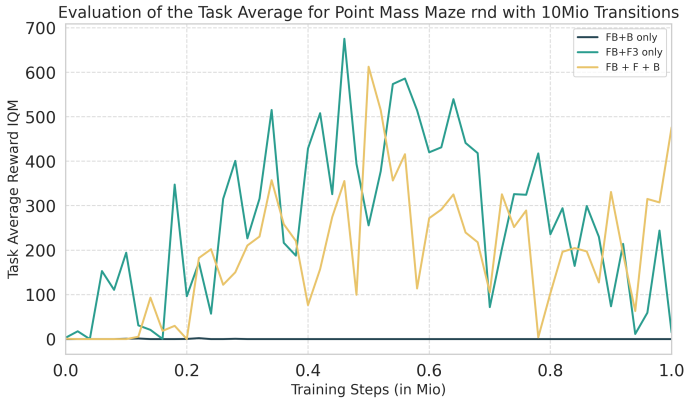

Evaluation of Stand for Quadruped rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
1024
789
42
123
456

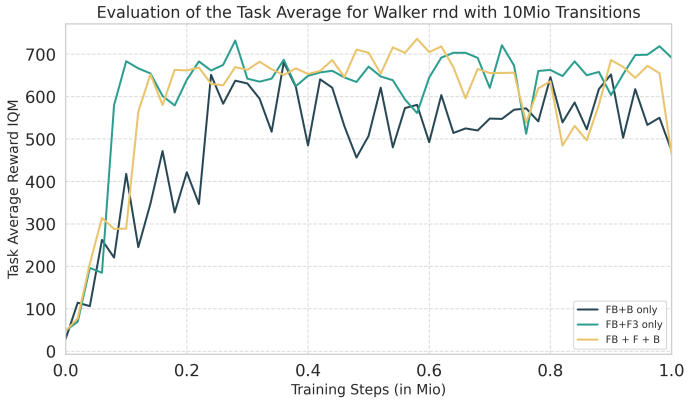


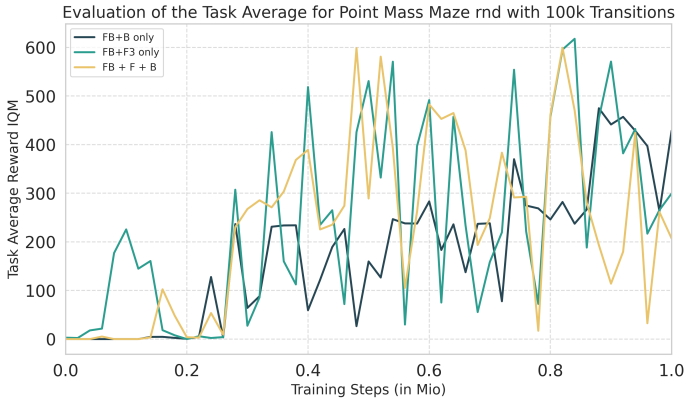


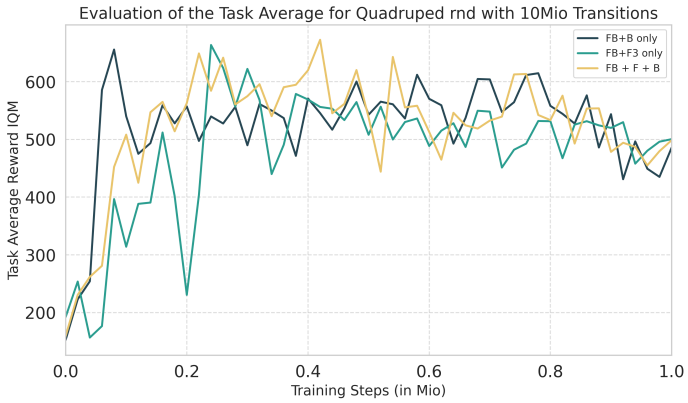
B.2 Ablations Design Neural Network Architecture

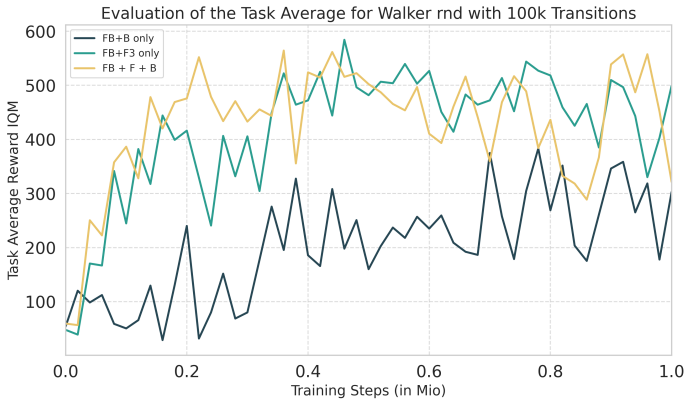

Evaluation of the Task Average for Quadruped rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

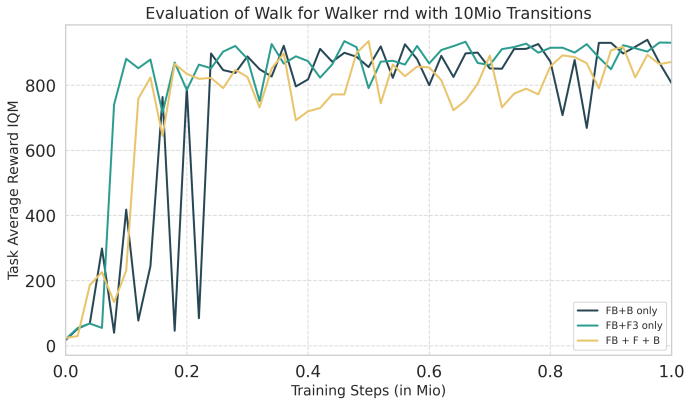

Evaluation of the Task Average for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

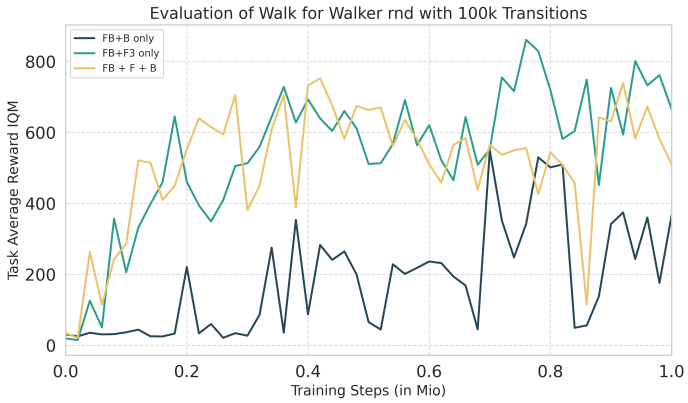

Evaluation of the Task Average for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

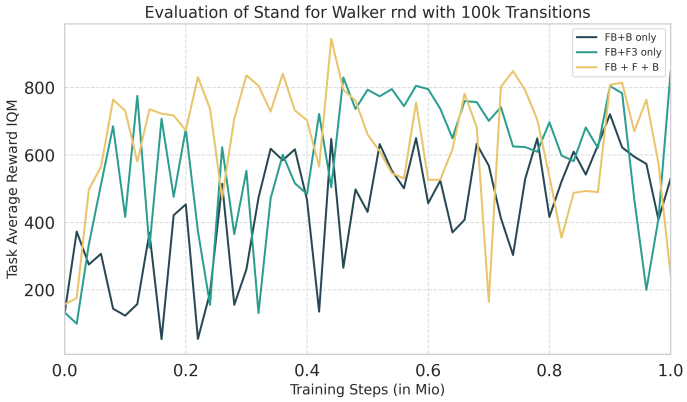

Evaluation of the Task Average for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

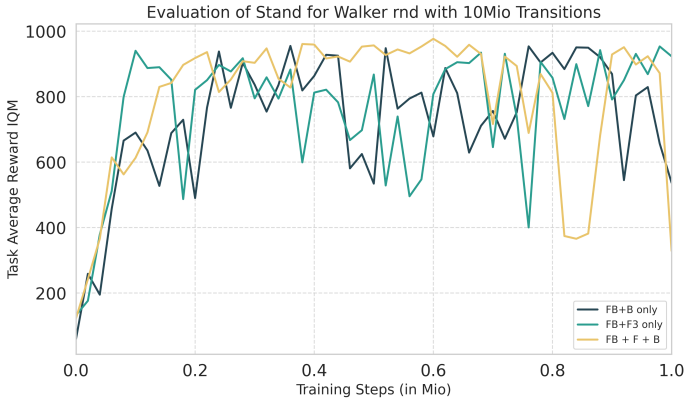

Evaluation of the Task Average for Quadruped rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

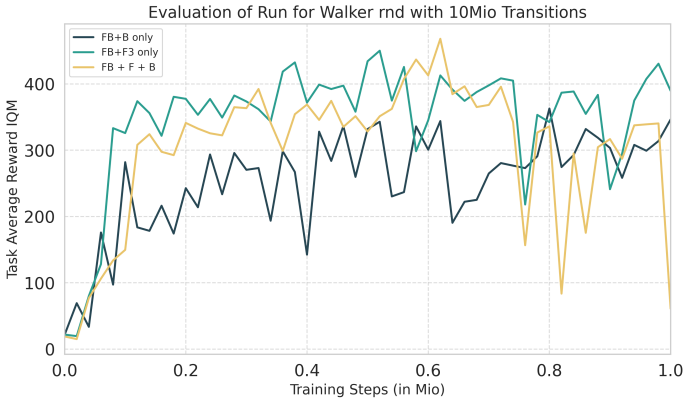

Evaluation of the Task Average for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

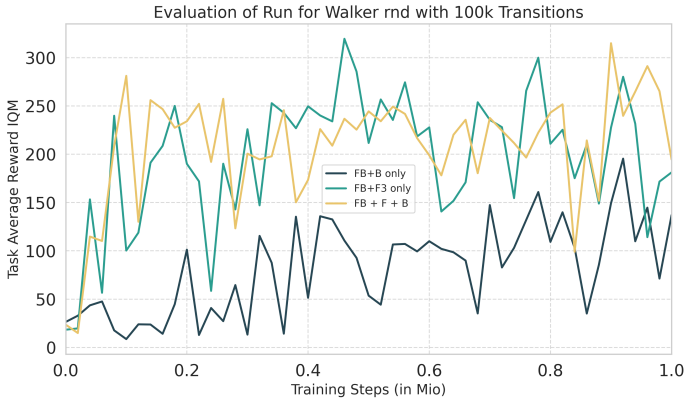

Evaluation of Walk for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

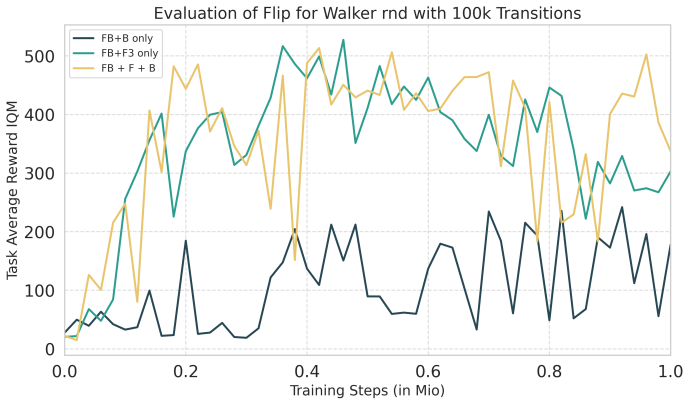

Evaluation of Walk for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

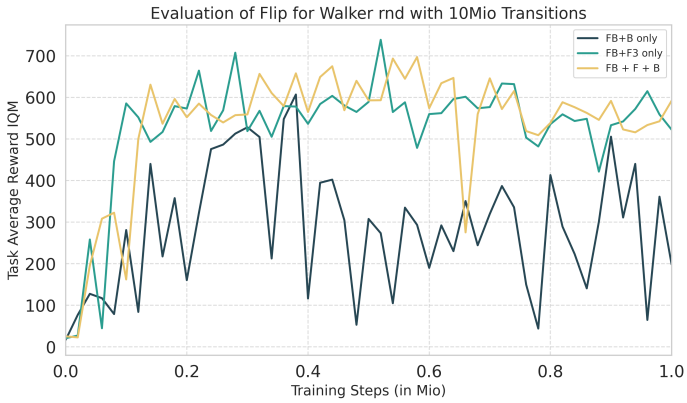

Evaluation of Stand for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

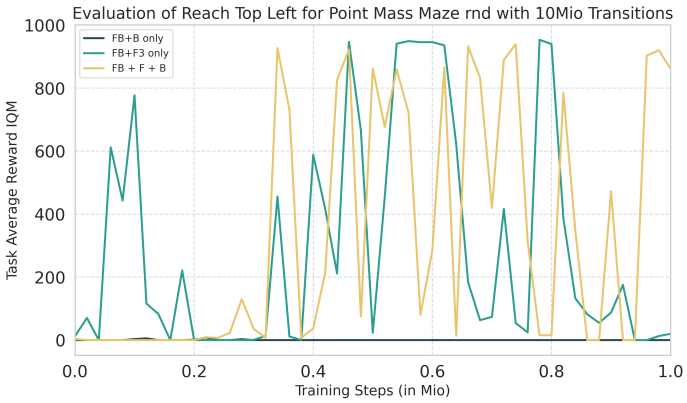

Evaluation of Stand for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

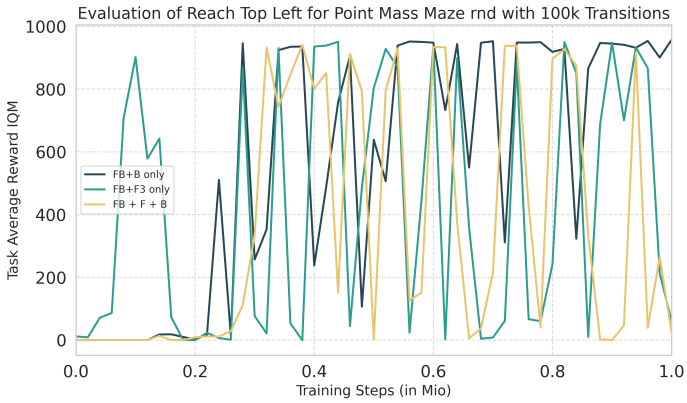

Evaluation of Run for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

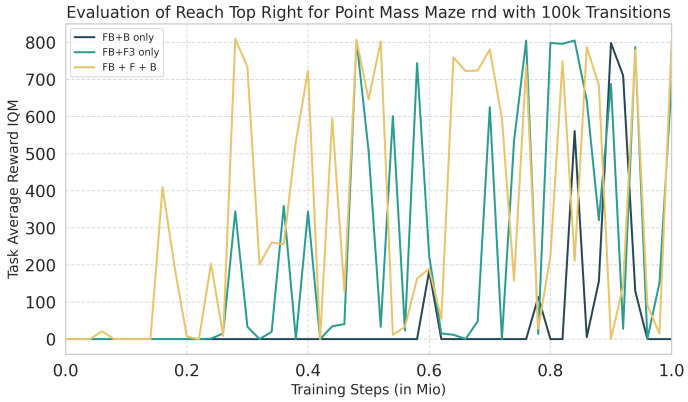

Evaluation of Run for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

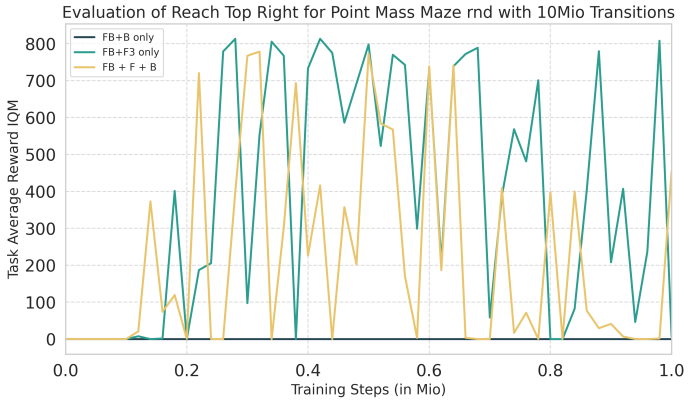

Evaluation of Flip for Walker rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

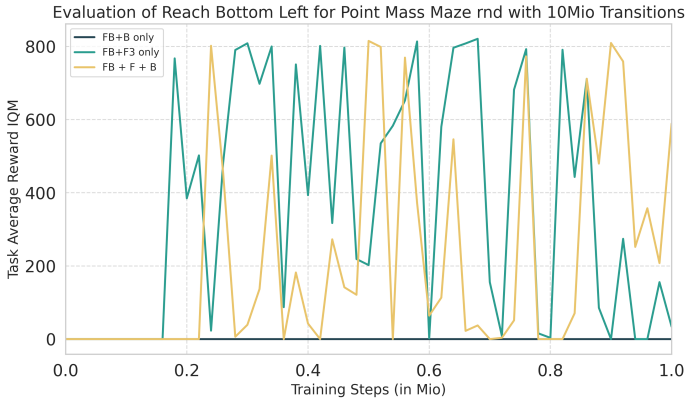

Evaluation of Flip for Walker rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

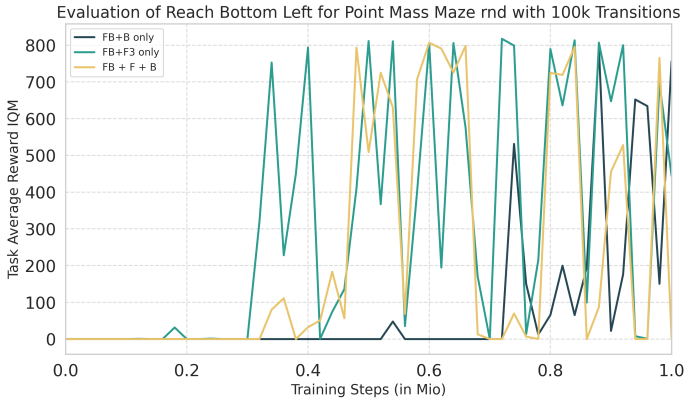

Evaluation of Reach Top Left for Point Mass Maze rnd with 10Mio Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

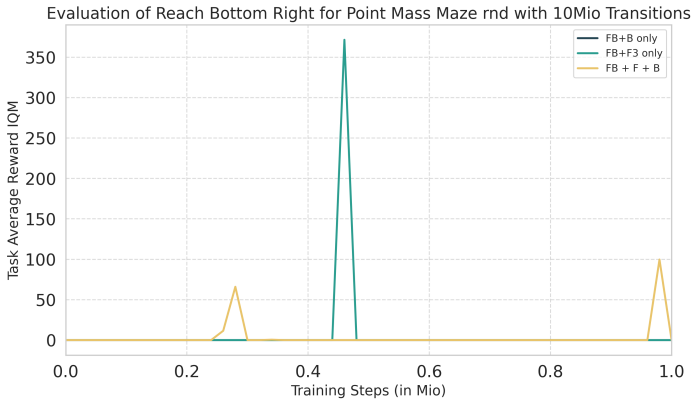

Evaluation of Reach Top Left for Point Mass Maze rnd with 100k Transitions
Task Average Reward IQM
Training Steps (in Mio)
FB+B only
FB+F3 only
FB + F + B

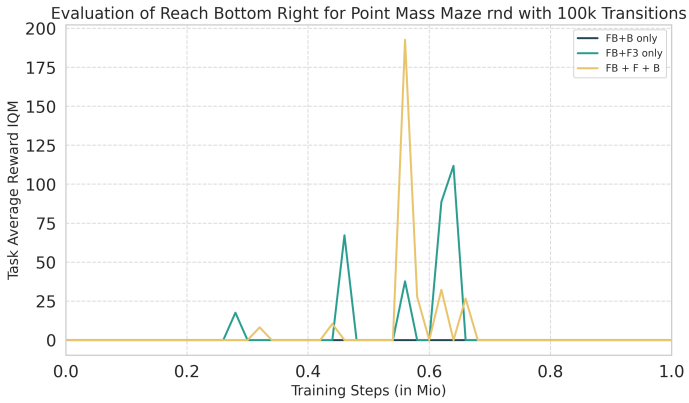

Evaluation of Reach Top Right for Point Mass Maze rnd with 100k Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

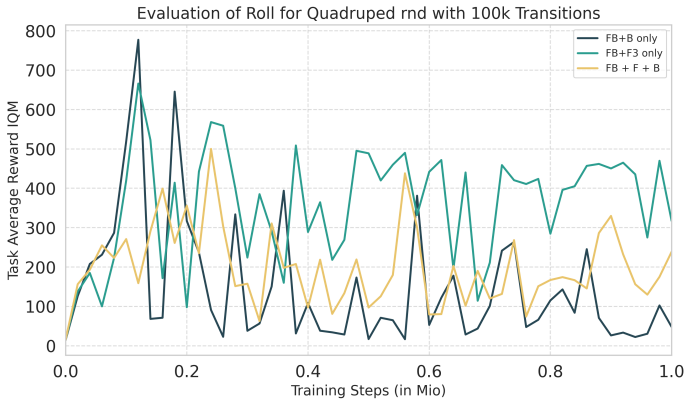

Evaluation of Reach Top Right for Point Mass Maze rnd with 10Mio Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

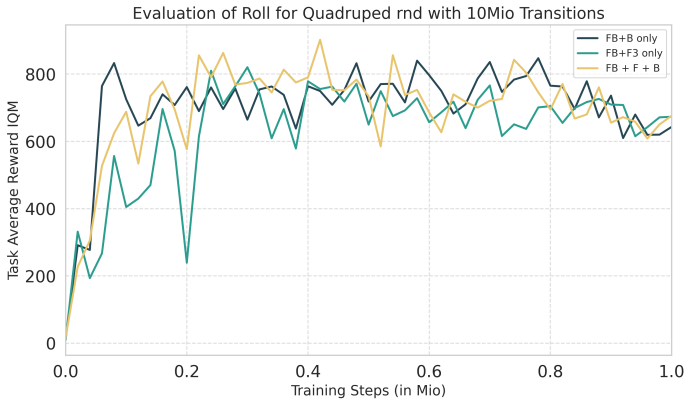

Evaluation of Reach Bottom Left for Point Mass Maze rnd with 10Mio Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

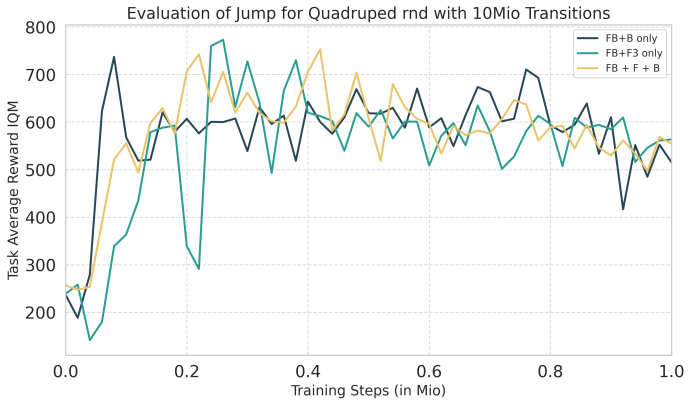

Evaluation of Reach Bottom Left for Point Mass Maze rnd with 100k Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

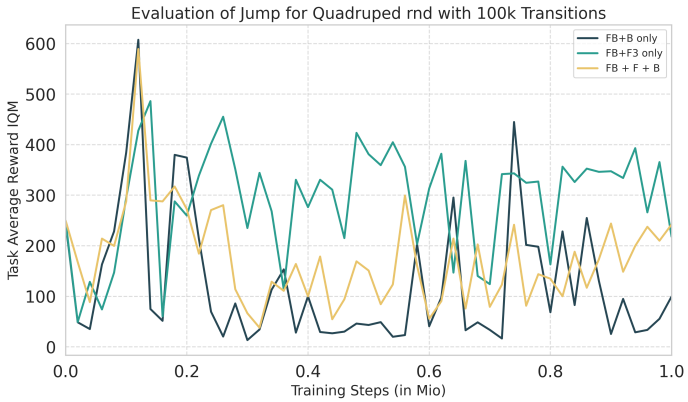

Evaluation of Reach Bottom Right for Point Mass Maze rnd with 10Mio Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

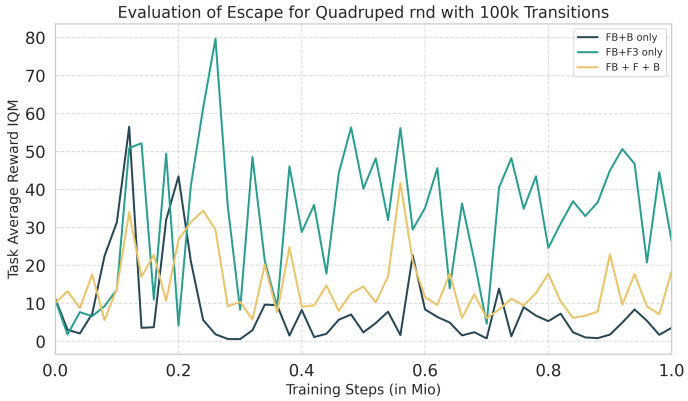

Evaluation of Reach Bottom Right for Point Mass Maze rnd with 100k Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

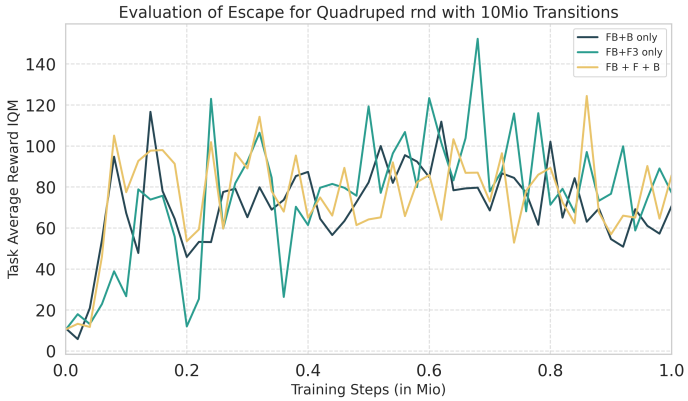

Evaluation of Roll for Quadruped rnd with 100k Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

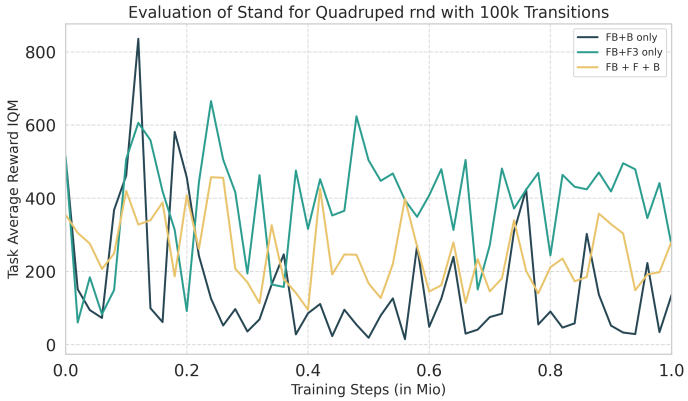

Evaluation of Roll for Quadruped rnd with 10Mio Transitions
FB+B only
FB+F3 only
FB + F + B
Task Average Reward IQM
Training Steps (in Mio)

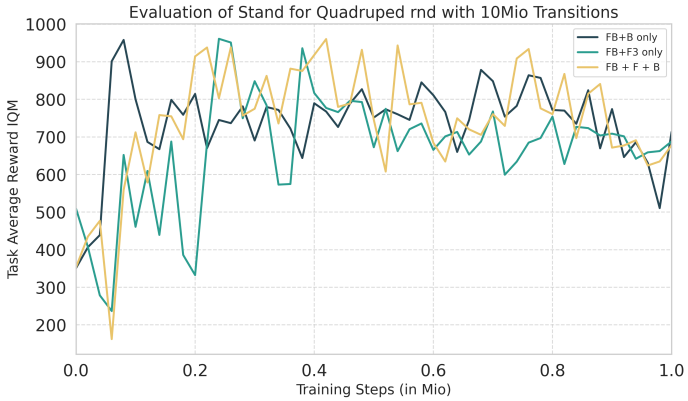












B.3 Ablation Designs: Full Conservatism Results

For each dataset-domain pair, we report the score at the step for which the all-task IQM is maximized when averaging across five seeds, and the constituent task scores at that step. Bracketed numbers represent the 95% confidence interval obtained by a stratified bootstrap. The results on all the datasets, across all tasks for 10M transitions for our method can be found in Table B.1. The results on all the datasets, across all tasks for 100k transitions for our method can be found in Table B.2.

Table B.1 Comparison of FB, VC-FB, and MC-FB and our method across different datasets and tasks for 10M transition dataset

Dataset	Domain	Task	FB	VC-FB	MC-FB	FB-PERL
RND	walker	walk	821 (758–883)	864 (850–879)	792 (728–857)	890 (816-945)
		stand	928 (925–930)	878 (854–903)	873 (812–934)	953 (905-977)
		run	281 (242–320)	351 (328–374)	343 (320–366)	395 (333-467)
		flip	525 (452–598)	542 (513–571)	598 (538–657)	604 (557-642)
		all tasks	639 (616–661)	659 (647–670)	651 (632–671)	710 (690-756)
	quadruped	stand	957 (952–963)	863 (777–950)	949 (939–958)	897 (752-965)
		roll	920 (895–944)	831 (741–921)	890 (874–906)	793 (701-856)
		jump	736 (721–751)	630 (570–690)	705 (703–707)	668 (560-775)
		escape	94 (63–125)	59 (50–68)	66 (47–86)	84 (61-105)
		all tasks	656 (638–674)	579 (522–635)	635 (628–642)	610 (527-655)
	Maze	top right	0 (0–0)	425 (153–698)	270 (7–533)	687 (362-810)
		top left	612 (313–911)	454 (138–769)	773 (611–934)	347 (0-902)
		bottom right	0 (0–0)	0 (0–0)	0 (0–0)	121 (0-609)
		bottom left	268 (0–536)	270 (2–539)	1 (0–2)	565 (65-797)
		all tasks	219 (86–353)	287 (117–457)	261 (159–363)	430 (270-529)
	all domains	all tasks	504	508	516	583
DIAYN	walker	walk	459 (266–652)	536 (305–766)	519 (315–722)	727 (468-882)
		stand	478 (463–494)	447 (422–472)	517 (433–602)	860 (621-973)
		run	87 (81–93)	84 (78–89)	87 (65–110)	203 (162-262)
		flip	235 (151–319)	251 (151–352)	301 (213–388)	492 (438-523)
		all tasks	315 (247–383)	329 (251–408)	356 (273–439)	570 (491-632)
	quadruped	stand	763 (725–801)	785 (739–810)	804 (756–851)	775 (690-926)
		roll	767 (726–808)	785 (740–801)	761 (736–786)	761 (692-898)
		jump	628 (587–669)	620 (600–641)	608 (594–622)	665 (597-758)
		escape	65 (62–69)	69 (59–74)	67 (55–79)	81 (59-111)
		all tasks	544 (517–572)	550 (536–580)	547 (535–559)	570 (509-663)
	Maze	top right	0 (0–0)	0 (0–0)	0 (0–0)	479 (0-705)
		top left	654 (565–742)	928 (907–950)	814 (725–903)	429 (0-756)
		bottom right	0 (0–0)	0 (0–0)	8 (0–16)	0 (0-0)

Dataset	Domain	Task	FB	VC-FB	MC-FB	FB-PERL
		bottom left	169 (0–337)	7 (0–14)	49 (0–98)	1 (0-3)
		all tasks	205 (171-240)	233 (227-240)	217 (180-254)	177 (152-200)
	all domains	all tasks	355	371	373	439
RANDOM	walker	walk	148 (70-225)	170 (139-231)	174 (142-243)	145 (69-304)
		stand	318 (281-355)	343 (296 371)	355 (298-393)	376 (270-429)
		run	51 (45-58)	101 (74-121)	100 (67-112)	60 (42-84)
		flip	57 (47-67)	106 (49-117)	103 (65-140)	126 (64-229)
		all tasks	143 (116-171)	180 (129-205)	183 (136-219)	177 (152-200)
	quadruped	stand	417 (381-453)	450 (390 492)	431 (371-464)	203 (60-444)
		roll	231 (116-346)	292 (255-388)	303 (160-445)	160 (43-397)
		jump	287 (123-450)	311(261-354)	340 (275-393)	159 (31-357)
		escape	10 (6-14)	10 (5-12)	12 (9-16)	7 (1-14)
		all tasks	211 (148-274)	253 (180-315)	260 (196-327)	139 (34-303)
	Maze	top right	0 (0–0)	0 (0–0)	0 (0–0)	0 (0–0)
		top left	309 (4-615)	317 (5-629)	307 (0-614)	177 (4–851)
		bottom right	0 (0–0)	0 (0–0)	0 (0–0)	0 (0–0)
		bottom left	0 (0–0)	0 (0–0)	0 (0–0)	0 (0–0)
		all tasks	77 (0-153)	79 (1-157)	76 (0-153)	44 (1-213)
	all domains	all tasks	144	171	173	120
ALL	all domains	all tasks	334	350	354	381

Table B.2 Comparison of FB, VC-FB, and MC-FB and our method across different datasets and tasks for 100k transition dataset

Dataset	Domain	Task	FB	VC-FB	MC-FB	FB-PERL
RND	walker	walk	184 (108-274)	446 (435-460)	247 (137-318)	416 (381-438)
		stand	558 (500-637)	624 (603-639)	480 (401-517)	761 (725-796)
		run	101 (88-144)	179 (159-194)	106 (72-145)	167 (147-202)
		flip	163 (90-203)	325 (294-350)	164 (120-198)	427 (415-441)
		all tasks	266 (233-283)	396 (381-407)	252 (188-288)	443 (429-460)
	quadruped	stand	134 (91-188)	331 (199-405)	171 (71-369)	646 (540-740)

Dataset	Domain	Task	FB	VC-FB	MC-FB	FB-PERL
		roll	139 (68-234)	141 (98-212)	132 (40-251)	650 (572-730)
		jump	121 (79-193)	159 (105 212)	97 (37-191)	529 (471-606)
		escape	7 (3-12)	8 (3-15)	5 (1-12)	104 (85-136)
		all tasks	93 (69-137)	168 (104-201)	104 (38 212)	482 (436-538)
	Maze	top right	0 (0-26)	0 (0-203)	99 (9-377)	0 (0–0)
		top left	384 (0-724)	662 (218 903)	723 (363-895)	0 (0–0)
		bottom right	0 (0–0)	0 (0–0)	0 (0–0)	0 (0–0)
		bottom left	0 (0-0)	479 (70-748)	384 (0-776)	36 (0–179)
		all tasks	102 (0-181)	323 (177-412)	270 (154-459)	9 (0-45)
	all domains	all tasks	154	295	208	311
DIAYN	walker	walk	93 (58-113)	262 (141-370)	261 (175-351)	182 (105-225)
		stand	498 (381-652)	455 (401-492)	423 (375-595)	395 (350-418)
		run	98 (79-114)	83 (75-94)	81 (71-108)	92 (77-102)
		flip	193 (136-212)	229 (195-249)	183 (150-239)	190 (110-223)
		all tasks	274 (214-301)	252 (195-291)	265 (195-322))	215 (181-235)
	quadruped	stand	459 (397-530)	430 (394-482)	458 (396-513)	389 (168-540)
		roll	460 (409-492)	415 (392-439)	456 (407-494)	203 (77-314)
		jump	363 (318-419)	358 (324-400)	373 (341-403)	262 (82-349)
		escape	45 (35-58)	32 (27-43)	42 (37-50)	10 (5-14)
		all tasks	322 (285-364)	296 (272-327)	331 (302-342)	216 (158-265)
	Maze	top right	0 (0–0)	0 (0–0)	27 (0-86)	0 (0–0)
		top left	576 (76-876)	911 (615-927)	853 (572-932)	148 (0–741)
		bottom right	0 (0–0)	0 (0–0)	0 (0–0)	0 (0–0)
		bottom left	0 (0–1)	0 (0–0)	0 (0–0)	0 (0–0)
		all tasks	144 (19-219)	227 (153-231)	213 (153-241)	37 (0–185)
	all domains	all tasks	247	183	270	156
RANDOM	walker	walk	76 (50-121)	123 (84-140)	119 (59-211)	45 (35-49)
		stand	238 (201-279)	223 (206-244)	210 (187-239)	215 (179-249)
		run	38 (32-48)	40 (37-46)	32 (27-38)	35 (29-46)
		flip	47 (40-60)	63 (41-99)	44 (38-55)	44 (42-49)

Dataset	Domain	Task	FB	VC-FB	MC-FB	FB-PERL
		all tasks	102 (91-119)	113 (98-125)	108 (78-127)	85 (76-94)
	quadruped	stand	278 (154-493)	269 (48-618)	172 (68-284)	417 (52-682)
		roll	105 (53-188)	130 (74-185)	178 (101-402)	241 (18-460)
		jump	75 (30-155)	78 (23-226)	147 (44-261)	178 (53-358)
		escape	5 (2-9)	2 (1-11)	6 (1-14)	8 (1-12)
		all tasks	149 (93-159)	125 (49-218)	126 (106-165)	211 (33-307)
	Maze	top right	0 (0-0)	0 (0-0)	0 (0-0)	0 (0-0)
		top left	18 (0-55)	26 (5-106)	10 (0-33)	88 (0-437)
		bottom right	0 (0–0)	0 (0–0)	0 (0–0)	0 (0-0)
		bottom left	0 (0–0)	0 (0–0)	0 (0–0)	0 (0-0)
		all tasks	4 (0-13)	6 (1-26)	2 (0-8)	22 (0-109)
	all domains	all tasks	85	81	79	106
ALL	all domains	all tasks	162	186	186	191

APPENDIX C IMPLEMENTATION DETAILS

C.1 Operator Architectures

Listing C.1 First working operator

```
class BidirectionalAttentionMixnet(nn.Module):
    """BidirectionalAttentionMixnet layer."""

    def __init__(
        self,
        z_dim,
        attention_config: dict,
        device,
    ):
        super(BidirectionalAttentionMixnet, self).__init__()
        self.config = attention_config
        self.device = device
        self.build_layers(z_dim)

    def build_layers(self, z_dim: int) -> None:
        """Build the neural network architecture from
           Attention and linear layers."""
        self.attention_layers = nn.ModuleList()
        self.linear_layers = nn.ModuleList()

        # Attention layers
        for _ in range(self.config["n_attention_layers"]):
            self.attention_layers.append(CrossAttention(z_dim,
                device=self.device))
            self.attention_layers.append(nn.LayerNorm(z_dim))
            self.attention_layers.append(nn.Dropout(self.
               config["dropout_rate"]))

        # Linear layers
        self.linear_layers.append(
            nn.Linear(trans_input_dim, self.config["
               trans_hidden_dimension"])
        )

        for _ in range(self.config["n_linear_layers"] - 2):
            self.linear_layers.append(
                nn.Linear(
                    self.config["trans_hidden_dimension"],
                    self.config["trans_hidden_dimension"],
                )
            )

        self.output = nn.Linear(self.config["
           trans_hidden_dimension"], 1)
```

```
        self.apply(weight_init)
        self.to(self.device)

    def compute_norms(
        self,
        forward_representation: torch.Tensor,
        backward_representation: torch.Tensor,
    ) -> Tuple[float, float]:
        """Compute the norms of the forward and backward
           representations."""
        return torch.norm(forward_representation, p=2, dim=1),
            torch.norm(
            backward_representation, p=2, dim=1
        )

    def forward(
        self,
        forward_representation: torch.Tensor,
        backward_representation: torch.Tensor,
        compute_z=False,
    )-> torch.Tensor:
        """
        Compute the FoB results from the forward and backward
           representations.
        input: F and B of dimension [batch_size, z_dimension]
        output: FoB, of dimension [batch_size, 1]
        """

        f_len, b_len = self.compute_norms(
            forward_representation, backward_representation
        )

        # Attention layers
        forward_representation = backward_representation = nn.
           functional.normalize(
            forward_representation, p=2, dim=1
        ) + nn.functional.normalize(
            backward_representation, p=2, dim=1
        )  # why do we merge forward and backward
           representations ?

        # first layer
        combined = self.attention_layers[0](
            backward_representation.unsqueeze(1),
            forward_representation.unsqueeze(1),
        )
        combined = combined + backward_representation.
           unsqueeze(1)

        for layer in self.attention_layers[1:]:
            if isinstance(layer, CrossAttention):
                combined = layer(combined,
                   forward_representation.unsqueeze(1))
```

```
                combined += backward_representation.unsqueeze
                    (1)
            else:
                combined = layer(combined)

        combined_output = combined.squeeze(1)

        # Linear layers
        # Initial linear transformation if needed based on
            model configuration
        x = self.linear_layers[0](combined_output)

        for layer in self.linear_layers[1:]:
            if self.config["use_linear_res"]:
                x = x + layer(x)
            else:
                x = layer(x)

        return (
            self.output(x).squeeze() * f_len * b_len
        )
```

Listing C.2 Proposed operator

```
    class TransformerDecoder(nn.Module):
    def __init__(
        self,
        z_dimension: int,
        config: Dict[str, Any],
        device: torch.device,
        dual_rep: bool = False,
        feature_reg: bool = False,
        dropout: float = 0.1,
        **kwargs,
    ):
        super().__init__()

        self.dual_rep = dual_rep
        self.z_dimension = z_dimension
        self.device = device
        self.feature_reg = feature_reg
        self.config = config

        # Neural Network Architecture
        config_block = {
            "n_embd": config["hidden_dimension"],
            "n_head": 1,
            "attn_pdrop": dropout,
            "resid_pdrop": dropout,
        }
        self.affine_emb = nn.Parameter(torch.randn(1, config["
            hidden_dimension"]))
        self.affine_emb2 = nn.Parameter(torch.randn(1, config[
```

```
            "hidden_dimension"]))

        self.transformer = nn.ModuleDict(
            dict(
                wte=nn.Linear(z_dimension, config["
                    hidden_dimension"]),
                drop=nn.Dropout(dropout),
                h=nn.ModuleList(
                    [Block(config_block) for _ in range(config
                        ["hidden_layers"])]
                ),
                ln_f=nn.LayerNorm(config["hidden_dimension"]),
            )
        )
        self.lm_head = nn.Linear(config["hidden_dimension"],
            1, bias=False)

    def forward(
        self, F: torch.Tensor, B: torch.Tensor, compute_z:
            bool = False
    ) -> Tuple[torch.Tensor, torch.Tensor]:
        x = F * B

        x = self.transformer.wte(
            x.unsqueeze(1)
        )  # token embeddings of shape (batch, 1 , emb_dim)
        x = x + self.affine_emb  # automatic broadcasting of
            affine_emb along batch_size

        # pass through the NN architecture
        for block in self.transformer.h:
            x = block(x)
        x = self.transformer.ln_f(x)  # shape (batch, 1,
            emb_dim)

        x = x.squeeze(1) + self.affine_emb2
        x = self.lm_head(x)  # shape (batch, 1)

        return x
```

C.2 B Architecture

As per the original authors, we pre-normalize for all architecture $z : z \leftarrow \sqrt{d}\frac{z}{\|z\|^2}$. B is batch-normalized. The non-gaussian MLP actor uses layer normalization.

Listing C.3 Proposed B structure

```
class BackwardDecoder(nn.Module):
    """Backwards model implemented with GPT Transformer
        Decoder architecture."""
```

```
    def __init__(
        self,
        observation_length: int,
        z_dimension: int,
        hidden_dimension: int,
        hidden_layers: int,
        device: torch.device,
        nhead: int = 8,
        dropout: float = 0.1,
        **kwargs,
    ):

        super().__init__()
        self.z_dimension = z_dimension
        self.device = device
        config = {
            "n_embd": hidden_dimension,
            "n_head": nhead,
            "attn_pdrop": dropout,
            "resid_pdrop": dropout,
        }
        self.transformer = nn.ModuleDict(
            dict(
                wte=nn.Linear(observation_length,
                   hidden_dimension),
                wpe=nn.Linear(1, hidden_dimension),
                drop=nn.Dropout(dropout),
                h=nn.ModuleList([Block(config) for _ in range(
                   hidden_layers)]),
                ln_f=nn.LayerNorm(hidden_dimension),
            )
        )
        self.lm_head = nn.Linear(
            hidden_dimension, z_dimension, bias=False
        )

    def forward(
        self, observation: torch.Tensor, position_encoding:
           bool = True
    ) -> torch.Tensor:
        """
        Takes observation and processes it through Transformer
            architecture.
        Args:
            observation: state tensor of shape [batch_dim,
               observation_length]
        Returns:
            z: embedded tensor of shape [batch_dim,
               z_dimension]
        """
        pos = torch.arange(
            0, 1, dtype=torch.float, device=self.device
        )  # shape (t).unsqueeze(0) # shape (1, t)
```

```
        pos =  torch.ones((1), dtype=torch.float32, device =
           self.device)

        # forward the GPT model itself
        tok_emb = self.transformer.wte(
            observation.unsqueeze(1)
        )  # token embeddings of shape (batch_size, t, n_embd)
        if position_encoding:
            pos_emb = self.transformer.wpe(
                pos
            )  # position embeddings of shape (1, t, n_embd)
        else:
            pos_emb = torch.zeros_like(tok_emb)

        x = tok_emb + pos_emb

        for block in self.transformer.h:
            x = block(x)
        x = self.transformer.ln_f(x)

        output = self.lm_head(x)  # [batch_size, 1, z]
        output = output.squeeze(1)  # [batch_size, z]

        # L2 normalize then scale to radius sqrt(z_dimension)
        output = torch.sqrt(
            torch.tensor(self.z_dimension, dtype=torch.float32
               , device=self.device)
        ) * torch.nn.functional.normalize(output, dim=1)

        return output
```

C.3 F Architecture

Listing C.4 Proposed F structure

```
class ForwardRepresentationTransformer(nn.Module):

    def __init__(
        self,
        observation_length: int,
        action_length: int,
        preprocessor_hidden_dimension: int,
        preprocessor_feature_space_dimension: int,
        preprocessor_hidden_layers: int,
        preprocessor_activation: str,
        z_dimension: int,
        forward_hidden_dimension: int,
        forward_hidden_layers: int,
        device: torch.device,
        dual_rep: bool = False,
        feature_reg: bool = False,
```

```
        dropout: float = 0.1,
        **kwargs,
    ):
        super().__init__()

        self.dual_rep = dual_rep
        self.z_dimension = z_dimension
        self.device = device
        self.feature_reg = feature_reg

        # pre-processors
        self.obs_action_preprocessor = AbstractPreprocessor(
            observation_length=observation_length,
            concatenated_variable_length=action_length,
            hidden_dimension=preprocessor_hidden_dimension,
            feature_space_dimension=
               preprocessor_feature_space_dimension,
            hidden_layers=preprocessor_hidden_layers,
            device=device,
            activation=preprocessor_activation,
        )

        self.obs_z_preprocessor = AbstractPreprocessor(
            observation_length=observation_length,
            concatenated_variable_length=z_dimension,
            hidden_dimension=preprocessor_hidden_dimension,
            feature_space_dimension=
               preprocessor_feature_space_dimension,
            hidden_layers=preprocessor_hidden_layers,
            device=device,
            activation=preprocessor_activation,
        )

        # Neural Network Architecture
        config: dict[str, int | float] = {
            "n_embd": forward_hidden_dimension,
            "n_head": 4,
            "attn_pdrop": dropout,
            "resid_pdrop": dropout,
        }
        self.transformer = nn.ModuleDict(
            dict(
                wte=nn.Linear(
                    2* preprocessor_feature_space_dimension,
                       forward_hidden_dimension
                ),
                drop=nn.Dropout(dropout),
                h=nn.ModuleList([Block(config) for _ in range(
                   forward_hidden_layers)]),
                ln_f=nn.LayerNorm(forward_hidden_dimension),
            )
        )
        self.linear = nn.Linear(
```

```
        forward_hidden_dimension, z_dimension
        )

    def forward(
        self,
        observation: torch.Tensor,
        action: Optional[torch.Tensor] = None,
        z: Optional[torch.Tensor] = None,
    ) -> Tuple[torch.Tensor, torch.Tensor]:

        # pass through the embeddings
        obs_action_embedding = self.obs_action_preprocessor(
            torch.cat([observation, action], dim=-1)
        )
        obs_z_embedding = self.obs_z_preprocessor(torch.cat([
           observation, z], dim=-1))
        combined = torch.cat([obs_z_embedding,
           obs_action_embedding], dim=1)
        combined = combined.unsqueeze(1)

        # pass through the NN architecture
        x = self.transformer.wte(
            combined
        )  # token embeddings of shape (batch, 1 or 2, n_embd)
        x = self.transformer.drop(x)

        for block in self.transformer.h:
            x = block(x)
        x = x.flatten(start_dim=1)

        x1 = self.linear(x)

        return x1
```

C.4 Taylor Expansion

Listing C.5 Taylor2ndOrder structure

```
class TaylorSecondOrder(nn.Module):
def __init__(self, z_dim, device):
    super().__init__()
    self.z_dim = z_dim

    # First order terms - matrices of size [z_dim, z_dim]
    self.dx = nn.Parameter(torch.randn(z_dim, z_dim))
    self.dy = nn.Parameter(torch.randn(z_dim, z_dim))

    # Second order terms - matrices of size [z_dim, z_dim]
    self.dxx = nn.Parameter(torch.randn(z_dim, z_dim))
    self.dxy = nn.Parameter(torch.randn(z_dim, z_dim))
    self.dyy = nn.Parameter(torch.randn(z_dim, z_dim))
```

```
    # Function value and expansion point - vectors of size [
       z_dim]
    self.f_ab = nn.Parameter(torch.randn(z_dim))
    self.a = nn.Parameter(torch.randn(z_dim))
    self.b = nn.Parameter(torch.randn(z_dim))

def forward(self, x, y, compute_z=False):
    dx = (x - self.a)  # [batch_size, z_dimension]
    dy = (y - self.b)  # [batch_size, z_dimension]
    first_order = torch.matmul(dx, self.dx) + \
                  torch.matmul(dy, self.dy)
    second_order = (
        0.5 * torch.matmul(dx * dx , self.dxx) +
        torch.matmul(dx * dy, self.dxy) +
        0.5 * torch.matmul(dy * dy, self.dyy)
    )
    return torch.mean(self.f_ab + first_order + second_order,
       dim=-1)  # [batch_size]
```

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Professor Zhan for his invaluable guidance and support throughout this thesis. His insightful advice, weekly meetings, and encouragement were instrumental in shaping this work.

I would also like to thank my boyfriend, Jop, for his unwavering support throughout this journey. His patience, thoughtful advice, and emotional encouragement were invaluable during challenging moments.

Lastly, I extend my heartfelt thanks to Kexin, who collaborated with me on this project. Having started the work before I joined, her insights, ideas, and ongoing support were incredibly valuable. Our discussions greatly enriched my understanding.

I am truly grateful to each of them for their contributions, guidance, and support, which have made this thesis possible.

This thesis was supported by Carbon Neutrality and Energy System Transformation (CNEST) Program.

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：叶洛莲 日 期：2025.05.20

RESUME

个人简历

2001 年 04 月 30 日出生于法国。

2018 年 7 月考入 Lycée Pasteur (PCSI/PC)

2020 年 7 月考入巴黎萨克雷大学 (2024 世界大学学术排名全球第 12 名)

2023 年 7 月考入清华大学

学术论文

[1] ZHENG K., Teyssier L. and al. Towards Robust Zero-Shot Reinforcement Learning (submitted to NeurIPS 2025).

Personal Resume

Born on April 30, 2001, in France.

Admitted to Lycée Pasteur, (PCSI/PC) in July 2018.

Admitted to CentraleSupélec in July 2020 (ranked top 3% in all elite entrance exams).

Admitted to Tsinghua University in July 2023.

Publications

[1] ZHENG K., Teyssier L. and al. Towards Robust Zero-Shot Reinforcement Learning (submitted to NeurIPS 2025).

COMMENTS FROM THESIS SUPERVISOR

该论文聚焦零样本强化学习 (zero-shot RL) 这一前沿研究领域，允许强化学习策略在整个任务空间进行多任务学习，并在部署阶段给定任务具体奖励函数后，zero-shot 泛化至最优策略。论文针对现有 zero-shot RL 主流 Forward-Backward representation 方法中存在的模型表达能力不足，未充分策略学习在数据分布外（out-of-distribution,OOD）进行评估时所产生的反事实推理误差等技术问题，提出了全新的神经网络模型架构以增强算法模型的表达能力，并引入了基于样本内学习的 OOD 约束机制，实现了更好的策略学习性能与训练鲁棒性。整体研究调研充分，方法新颖，并在基准测试任务上进行了有效验证，对相关领域研究有一定的参考价值。该学生遵纪守法、品行端正、基础扎实、学术作风良好。

RESOLUTION OF THESIS DEFENSE COMMITTEE

This thesis presents a novel and impactful contribution to the field of zero-shot reinforcement learning (ZSRL). Addressing key challenges such as limited model expressiveness and out-of-distribution generalization, the proposed FB-PERL framework introduces a Transformer-based neural architecture to better capture complex temporal dependencies and improve policy generalization. Additionally, the work incorporates a quantile-based OOD regularization strategy that mitigates distributional shift during offline training, enhancing robustness in unseen tasks. The algorithm is thoroughly validated through extensive experiments on the ExoRL benchmark and detailed ablation studies, demonstrating superior performance and stability over existing approaches. The thesis is well-structured, technically rigorous, and clearly written, offering both theoretical insight and practical relevance to the field of reinforcement learning.

In the research work of this thesis, 叶洛涟 shows her solid knowledge on theoretical fundamentals, her professional knowledge, and her ability to conduct research independently. The thesis is well organized and written. During the oral defense, she presented her work clearly and answered questions correctly. The defense committee voted unanimously in favor of 叶洛涟's thesis, and recommended that the degree of Master of Science be conferred on her.