ROBUST AND ADAPTIVE DEEP REINFORCEMENT LEARNING

by

Suzan Ece Ada

B.S., Industrial Engineering, Koç University, 2015

M.S., Software Engineering with Thesis, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2025

to Suzan Ada, Enise Ada, Mehmet Ada, Sevil Özsoy and Müçteba Özsoy.

ACKNOWLEDGEMENTS

I wish to express my deepest and most sincere gratitude to my advisors, Assoc. Prof. Emre Uğur and Prof. Erhan Öztop. Their mentorship has been invaluable, shaping my development as a researcher. I am truly grateful for their unwavering support and profound guidance. I am also indebted to the trust they placed in me, which sustained me through many critical turning points. Finally, I am thankful for the profoundly fruitful and supportive research environment they fostered and for their steadfast encouragement throughout my doctoral studies.

My heartfelt thanks also go to my master's advisor, Prof. H. Levent Akın, whose early mentorship and trust laid the foundation for my doctoral studies. I am also grateful to Prof. Georg Martius for the valuable opportunity and his hospitality during my research visit to the University of Tübingen. His mentorship and indispensable guidance have left a lasting impact on me.

I would like to thank my thesis committee and qualifying exam members, Assist. Prof. İnci Baytaş, Assist. Prof. Fatma Güney, Prof. Arzucan Özgür, Assist. Prof. Özgür Öğüz, and Assoc. Prof. Sinan Kalkan for their support, which has been instrumental in strengthening this work. I am also deeply grateful to Assist. Prof. Birkan Yılmaz, Assist. Prof. Berk Gökberk, Assoc. Prof. Atay Özgövde, Dr. Murat Özyurt, and Assoc. Prof. Özlem Durmaz İncel. Serving as a teaching assistant under their guidance was a formative experience and a constant source of inspiration.

I extend my thanks to my friends and lab members Deniz Baran Aksoy, Egemen İşgüder, Alper Ahmetoğlu, Muhammet Hatipoğlu, Yiğit Yıldırım, Yunus Şeker, Tuba Girgin, Irmak Güzey, , Mehmet Özer, Emre Bilgili, Hanne Say, Ahmet Tekden, Alara Dirik, İbrahim Özcan, Deniz Bilge Akkoç, Toprak Temir, Hakan Aktaş, Fatih Doğangün, Fırat Gamsız, Serdar Bahar, Gökçe Uluduğan, Barış Yamansavaşçılar, Özdeniz Dolu, Tomáš Daniš, René Geist, Ji Shi, Leonard Franz, Nico Gürtler, Anselm Paulus, Mikel

Zhobro, Boya Zhang, Cansu Sancaktar, Iris Andrussow, Anna Manasyan.

Finally, I could not have reached this milestone without the unconditional love and constant encouragement of my mother, Dr. Enise Özlem Ada, my father, Prof. Dr. Recep Mehmet Ada, and my brother, Turhan Can Ada. I dedicate this thesis in loving memory of my grandparents, Suzan Ada, Sevil Özsoy, and Müçteba Özsoy. I owe a profound debt of gratitude to my grandmother, Suzan Ada, who played a significant role in raising me. Loved and cherished by many, she was a woman of exceptional intellect who possessed the sharpest mind I have ever known. I miss her every day, but the bond we shared made me who I am today. To Can Mangır, I offer my deepest love and thanks for his enduring support and patience.

This thesis was partially supported by the INVERSE project (101136067) funded by the European Union, JSPS KAKENHI Grant Numbers JP23K24926, JP25H01236, JP90542217 Grant-in-Aid for Scientific Research – the project with number 22H03670, the project JPNP16007 commissioned by the New Energy and Industrial Technology Development Organization (NEDO), the Scientific and Technological Research Council of Turkey (TUBITAK, 118E923), German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A, and TÜBİTAK BİDEB 2224-A Grant Program for Participation in Scientific Meetings Abroad. The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

Reinforcement Learning," IEEE Robotics and Automation Letters, vol. 9, no. 4, pp. 3116–3123, April 2024, doi:10.1109/LRA.2024.3363530 (© 2024 IEEE [2]).

ABSTRACT

ROBUST AND ADAPTIVE DEEP REINFORCEMENT LEARNING

This thesis addresses the challenge of developing robust and adaptive reinforcement learning (RL) agents that operate in complex, non-Markovian, and non-stationary environments. Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP) addresses few-shot adaptation under unknown dynamics when reward signals are missing at test time. UMCNP learns a dynamics model to enable sample-efficient adaptation through self-generated trajectories. Episodic Return Progress with Bidirectional Progressive Neural Networks (ERP-BPNN) presents a human-inspired framework for multi-task learning by integrating a novel intrinsic motivation signal (ERP) for autonomous task switching with a bidirectional progressive neural network architecture, thereby facilitating effective skill transfer among morphologically different agents. State Reconstruction for Diffusion Policies (SRDP) confronts the challenge of generalization to out-of-distribution states in offline RL by incorporating a state reconstruction loss into the diffusion policy learning process. Forecasting in Non-stationary Offline RL (FORL) mitigates non-trivial non-stationarities by unifying conditional diffusion models with probabilistic zero-shot time-series foundation models. This framework proactively forecasts and corrects for abrupt, hidden observation offsets. Empirical evaluations across a range of continuous control, offline RL benchmarks, and robotics tasks confirm the efficacy of these methods. Our results demonstrate significant improvements in meta-testing sample efficiency, faster convergence via bidirectional skill transfer with return progress, superior generalization to out-of-distribution states, and robust performance against abrupt, non-Markovian shifts in the observation function.

ÖZET

GÜRBÜZ VE UYARLANABİLİR DERİN PEKİŞTİRMELİ ÖĞRENME

Bu tez, karmaşık, Markov olmayan ve durağan olmayan ortamlarda gürbüz ve uyarlanabilir pekiştirmeli öğrenme tabanlı etmenler geliştirme zorluğunu ele almaktadır. Koşullu Sinirsel Süreçler ile Gözetimsiz Meta-Sınama (UMCNP), sınama zamanında ödül sinyalleri eksik olduğunda bilinmeyen dinamikler altında az örnekli uyarlamayı hedeflemektedir. UMCNP, kendi ürettiği gezingeler aracılığıyla etkin örneklemli uyarlama sağlamak için bir model öğrenir. Çift yönlü İlerleyen Sinir Ağları ile Epizodik Getiri İlerlemesi (ERP-BPNN), otonom görev geçişi için yeni bir içsel motivasyon sinyali (ERP) ile morfolojik olarak farklı etmenler arasında yetenek aktarımını sağlayan çift yönlü ilerleyen sinir ağlarını birleştirerek insan esinli bir çoklu görev pekiştirmeli öğrenme çatısı sağlar. Yayınım Politikaları için Durum Geri Çatma (SRDP), bir durum geri çatma yitimini yayınım politikası öğrenme sürecine dahil ederek çevrimdışı pekiştirmeli öğrenmede dağılım dışı durumlara genelleme yapma zorluğunun üstesinden gelir. Durağan Olmayan Çevrimdışı Pekiştirmeli Öğrenmede Öngörü (FORL), koşullu yayınım modellerini olasılıksal örneksiz zaman dizisi temel modelleri ile birleştirerek zorlu durağan olmayan durumları ele alır. Bu çatı, ani, saklı gözlem sapmalarını proaktif olarak öngörür ve düzeltir. Sürekli kontrol, çevrimdışı pekiştirmeli öğrenme deneyleri ve robotik görevler üzerindeki deneysel değerlendirmeler, bu yöntemlerin etkinliğini doğrulamaktadır. Çıkarımlarımız, meta sınama örneklem etkinliğinde önemli iyileşmeler, getiri ilerlemesi kullanılarak eğitilen çift yönlü yetenek aktarımı yoluyla daha hızlı yakınsama, dağılım dışı durumlara üstün genelleme ve gözlem işlevindeki ani, Markov olmayan kaymalara karşı gürbüz başarı göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT vii
ÖZET viii
LIST OF FIGURES xiii
LIST OF TABLES xvi
LIST OF SYMBOLS xviii
LIST OF ACRONYMS/ABBREVIATIONS xxi
1. INTRODUCTION 1
2. RELATED WORK 8
2.1. Meta Reinforcement Learning 8
2.1.1. Parameterized Policy Gradient Based Few-Shot Meta-Reinforcement Learning 8
2.1.2. Black Box Few-Shot Meta-Reinforcement Learning 9
2.1.3. Few-Shot Meta Reinforcement Learning with Task Inference . . . 9
2.1.4. Model-based Meta Reinforcement Learning 10
2.2. Multi-task Reinforcement Learning 11
2.3. Curriculum Reinforcement Learning 11
2.4. Offline Reinforcement Learning 12
2.4.1. Dynamic Programming-based Offline Reinforcement Learning . . 13
2.4.2. Importance Sampling-based Offline Reinforcement Learning . . . 13
2.4.3. Model-based Offline Reinforcement Learning 14
2.4.4. Robust Offline Reinforcement Learning 15
2.4.5. Diffusion Models in Offline Reinforcement Learning 15
2.5. Reinforcement Learning in Non-Stationary Environments 16
2.6. Conditional Neural Processes 17
2.7. Human Learning 17
2.8. Intrinsic Motivation 18
3. BACKGROUND 20

3.1. Reinforcement Learning 20
3.2. Non-Stationary Environments 21
3.3. Diffusion Models 22
4. UNSUPERVISED META-TESTING WITH CONDITIONAL NEURAL PROCESSES FOR HYBRID META-REINFORCEMENT LEARNING 24
4.1. Introduction 24
4.2. Preliminaries 26
4.2.1. Model-Agnostic Meta-Learning 26
4.2.2. No-Reward Meta Learning 26
4.2.3. Conditional Neural Processes 27
4.3. Unsupervised Meta-Testing with Conditional Neural Processes 28
4.4. Experiments 32
4.4.1. 2D Point Agent with Unknown Artificial Force Field 33
4.4.2. Cartpole with Unknown Angle Sensor Bias 35
4.4.3. Walker-2D with Random Dynamics Parameters 37
4.4.4. Implementation Details 39
4.5. Conclusion 39
5. BIDIRECTIONAL PROGRESSIVE NEURAL NETWORKS WITH EPISODIC RETURN PROGRESS FOR EMERGENT TASK SEQUENCING AND ROBOTIC SKILL TRANSFER 41
5.1. Introduction 41
5.2. Method 43
5.2.1. Bidirectional Progressive Neural Networks 44
5.2.2. Task Switching by a Novel Intrinsic Motivation Signal: Episodic Return Progress 46
5.2.3. Multi-task Reinforcement Learning 48
5.2.4. Evaluation Metrics 49
5.2.5. Implementation Details 52
5.3. Experiments 52
5.3.1. Single Goal Multi-Task Learning Between Morphologically Different Robots 53

5.3.2. Results 53
5.3.3. ERP-based Dynamic Task Selection 57
5.4. Conclusion 58
6. DIFFUSION POLICIES FOR OUT-OF-DISTRIBUTION GENERALIZATION IN OFFLINE REINFORCEMENT LEARNING 60
6.1. Introduction 60
6.2. Preliminaries 61
6.2.1. Diffusion Q-Learning 61
6.3. Method 62
6.3.1. State Reconstruction for Diffusion Policies 62
6.3.2. Out-of-Distribution States in Offline RL 64
6.4. Experiments 66
6.4.1. 2D-Multimodal Contextual Bandit Experiments 66
6.4.2. Multimodal UR10 Experiments 70
6.4.3. Missing Data Maze Environment 72
6.4.4. D4RL Benchmark 75
6.5. Conclusion 76
7. FORECASTING IN OFFLINE REINFORCEMENT LEARNING FOR NON-STATIONARY ENVIRONMENTS 77
7.1. Introduction 77
7.2. Method 81
7.2.1. Problem Statement 81
7.2.2. FORL Diffusion Model 84
7.2.3. Forecasting with Zero-Shot Foundation Model 86
7.2.4. FORL State Estimation 87
7.3. Experimental Setup and Baselines 89
7.3.1. Baselines 90
7.3.1.1. Offline Reinforcement Algorithm Backbones 90
7.3.1.2. Extensions with Zero-Shot FM 92
7.3.1.3. DMBP+LAG 92
7.3.2. Experimental Setup 93

7.3.2.1. Zero-shot Foundation Model and Time-Series Datasets 93
7.3.2.2. Offline Reinforcement Learning Environments 94
7.3.3. Implementation Details 95
7.3.4. Results 98
7.3.4.1. No Access to Past Offsets 100
7.3.4.2. Dimension-wise Closest Match Ablations 104
7.3.4.3. Intra-episode Non-stationarity 107
7.3.4.4. Offset Scaling Results 108
7.3.4.5. Policy-Agnostic 109
7.3.4.6. Preliminary Results with Uniform Scaling and Bias . . 113
7.3.5. Sensitivity of FORL to Forecasting Errors 113
7.3.6. State Prediction Accuracy 114
7.3.7. Sensitivity to Diffusion-generated Sample Size 115
7.4. Conclusion 116
8. DISCUSSION 117
9. CONCLUSION 121
REFERENCES 122
APPENDIX A: HYPERPARAMETERS USED IN FORL 149

LIST OF FIGURES

Figure 4.1. Meta-training procedure of our proposed method UMCNP 29

Figure 4.2. Meta-testing procedure of our proposed method UMCNP 29

Figure 4.3. Illustrations in point agent environment. 31

Figure 4.4. Post-update cumulative reward in point agent experiments. 32

Figure 4.5. Cartpole environment with sensor bias and Walker-2D agent with random dynamics parameters. 35

Figure 4.6. Post-update cumulative reward in test tasks compared with baselines in cartpole with sensor bias environment. 37

Figure 4.7. Violin plots of post-update cumulative reward from unseen 10 meta-test tasks. 38

Figure 5.1. Reacher robot arm environments. 43

Figure 5.2. ERP-BPNN architecture. 45

Figure 5.3. Plots comparing ERP-BPNN with baselines during training across different performance metrics. 55

Figure 5.4. Illustrations of end-effector paths and final positions. 56

Figure 5.5. Task selection frequency plots. 57

Figure 6.1. Training dataset, ground truth dataset and in-distribution results. 65

Figure 6.2. SRDP and BC-Diffusion in restrictive 2D-Multimodal environment 66

Figure 6.3. SRDP and BC-Diffusion in 2D-Multimodal environment 67

Figure 6.4. Comparison of BC-CVAE, BC-MLE, and BC-MMD in 2D-Multimodal Contextual Bandit 68

Figure 6.5. SRDP and BC-Diffusion in UR10 real robot setup 71

Figure 6.6. Illustrations of missing data maze2d, and antmaze D4RL environments. 73

Figure 6.7. D4RL Gym-MuJoCo environments 73

Figure 7.1. Maze navigation setting with time-varying perceptual offsets. . . . 78

Figure 7.2. Overview of the proposed FORL framework at test-time. 80

Figure 7.3. Candidate states generated by FORL Diffusion Model. 84

Figure 7.4. Distribution of samples produced by DCM. 88

Figure 7.5. Zero-shot forecasting results for the first two time-series in univariate time-series datasets. 94

Figure 7.6. Kitchen complete environment in D4RL offline RL benchmark [21]. 94

Figure 7.7. Cube-single-play and antmaze-large-navigate environments in OG-Bench benchmark [32]. 95

Figure 7.8. Illustrations of the states predicted by FORL and baselines. 99

Figure 7.9. Scores of FORL diffusion model and the baselines when we do not have access to past offsets. 100

Figure 7.10. Comparison of average normalized scores and prediction errors across all navigation experiments FORL and baseline methods. . 101

Figure 7.11. Performance comparison without access to past offsets. 101

Figure 7.12. Candidate Selection Algorithms. 104

Figure 7.13. Intra-Episode Performance 107

Figure 7.14. Impact of offset scaling on average normalized scores. 109

Figure 7.15. Bar plots for offset scaling factors in navigation environments. . . 110

Figure 7.16. Scores for different diffusion generated sample sizes. 116

LIST OF TABLES

Table 4.1. UMCNP Results in the point environment 34

Table 4.2. Post-update cumulative Reward of UMCNP and the baselines. . . 36

Table 5.1. Results ERP-BPNN, RANDOM-BPNN, and RANDOM-MLP across the robotics performance metrics. 55

Table 6.1. Ablation Study for SRDP Scaling Parameters over training iterations at 4000 intervals with Chamfer Distance. 70

Table 6.2. Comparison of SRDP with DQL baseline in Missing Data and Standard Maze2D Environment. 72

Table 6.3. Comparison of SRDP with the DQL baseline in D4RL. 74

Table 7.1. Normalized scores for FORL framework and the baselines with DQL policy. 97

Table 7.2. Normalized scores for FORL and the baselines using FQL in OG-Bench benchmarks. 98

Table 7.3. Using a Zero-Shot FM in combination with FORL-DM in no access to past offsets setting. 102

Table 7.4. Additional results for no-access to past offsets setting. 103

Table 7.5. Normalized scores for FORL (DCM) and the baselines in Cube-Single-Play. 105

Table 7.6. Normalized scores for DCM candidate selection and the baselines. . 106

Table 7.7. Comparison of algorithm performance on error metrics. 107

Table 7.8. Intra-episode non-stationarity results with f=50. 108

Table 7.9. Normalized scores for FORL and baselines with TD3BC on maze environments. 111

Table 7.10. Normalized scores for FORL and baselines with RORL on maze environments. 111

Table 7.11. Normalized scores for FORL framework and the baselines with IQL policy. 112

Table 7.12. Sensitivity analysis of FORL to forecasting errors. 113

Table 7.13. Comparison of counterfactual prediction errors. 115

Table A.1. Hyperparameters for FORL. 149

Table A.2. Hyperparameters for DQL [29, 172, 200]. 150

Table A.3. Hyperparameters for Implicit Q-Learning (IQL) [183, 214, 187, 216]. 150

Table A.4. Hyperparameters for RORL [77, 213, 200]. 151

Table A.5. Hyperparameters for Flow Q-Learning (FQL) [188, 189]. 152

Table A.6. Hyperparameters for TD3+BC [182, 215, 200]. 152

LIST OF SYMBOLS

$\boldsymbol{a}$	action
$\mathcal{A}$	Action space
$A^{\pi}(s, a)$	Advantage function for a policy π
B_i	Set of transition tuples for task i
b^j	Offset for episode j
$b_l^{(m)}$	Bias vector for layer l of module m
$b_l^{(m:t)}$	Bias vector for lateral connections
C	Context length (number of past offsets)
$\boldsymbol{c}$	Offset scaling factor
c_1, c_2	Loss function coefficients
$\mathcal{D}$	Offline Reinforcement Learning dataset
$\mathcal{D}_{\text{diffusion}}$	Set of samples from the diffusion model
$\mathcal{D}_{\text{timeseries}}$	Set of samples from the time-series model
$\mathbf{e}$	Observation function
Δo_t	Change in observation at timestep t
f_θ	A function parameterized by θ, e.g., a neural network
f_ϕ	Shared feature extraction module
f_ψ	State reconstruction head
g	Goal position
H	Horizon length
$h_l^{(m)}$	Hidden activation of layer l for module m
$\hat{\mathcal{J}}$	Objective function
k	Optimization iteration counter
K_{init}	Number of initial (jumpstart) training iterations
$\mathcal{L}$	Loss function
L	Total number of layers in a network
m	Index for a task-specific network module
M	Set of network modules

M_{avg}	Averaging module
$\mathcal{M}_{\text{train}}$	Markov Decision Process (MDP) during training
$\{\hat{\mathcal{M}}_j\}$	Sequence of test Partially Observable MDPs (POMDPs)
n	Diffusion timestep
N	Final diffusion timestep
n_l^m	Number of neurons in layer l of module m
o	Observation
$\mathcal{O}_j$	Observation space for episode j
P	Transition kernel
$\boldsymbol{P}$	Number of parallel Reinforcement Learning environments
Q^π	State-action value function (Q-function) for a policy π
R	Reward function
$R_m(k)$	Average episodic return for module m at iteration k
$\boldsymbol{s}$	State
$\mathcal{S}$	State space
$S(\cdot)$	Entropy bonus
t	Reinforcement Learning timestep
$\mathcal{T}$	A task
$U_l^{(m:t)}$	Weight matrix for lateral connections from module t to m
$V^\pi(s)$	Value function for a policy π
w	Window size
$W_l^{(m)}$	Weight matrix for layer l of module m
$\boldsymbol{x}_0$	True data sample at diffusion timestep 0
$\boldsymbol{x}_n$	Noisy data sample at diffusion timestep n
$\boldsymbol{z}$	Latent representation
β_n	Per-step noise variance at diffusion timestep n
γ	Discount factor
$\boldsymbol{\epsilon}$	Noise vector sampled from a standard Gaussian distribution
ζ	Number of episodes completed in each environment
η	Scaling parameter for Q-function guidance

κ	Scope of non-stationarity
λ	Weighting hyperparameter for the state reconstruction loss
μ	Mean of a Gaussian distribution
π_θ	Policy parameterized by θ
π_β	Behavior policy
ρ_0	Initial state distribution
σ	Standard deviation
Σ	Covariance matrix
$\boldsymbol{\tau}_{(t,w)}$	Trajectory subsequence (history) over window w
θ	Parameters of a network
θ_D	Parameters of the decoder network
θ_E	Parameters of the encoder network
$\boldsymbol{\theta}_{\text{offset}}$	Offset parameter
ω	Rotation parameter for the artificial force field

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
AI	Artificial Intelligence
AntM-M-D	Antmaze-Medium-Diverse
AntM-U-D	Antmaze-Umaze-Diverse
AntM-L-D	Antmaze-Large-Diverse
AntM-L-Nav	Antmaze-Large-Navigate
Cube-S-P	Cube-Single-Play
BC	Behavior Cloning
BC-CVAE	Behavior Cloning with Conditional Variational Autoencoder
BC-Diffusion	Behavior Cloning in Diffusion Policies
BC-MLE	Behavior Cloning with Maximum Likelihood Estimation
BC-MMD	Behavior Cloning with Maximum Mean Discrepancy
BPNN	Bidirectional Progressive Neural Network
CaDM	Context-aware Dynamics Model
CI	Contextual-Interference
CloseMean	Closest Diffusion Sample to Forecasted Mean
CloseMedian	Closest Diffusion Sample to Forecasted Median
CNP	Conditional Neural Processes
CQL	Conservative Q-Learning
CVAE	Conditional Variational Autoencoder
DCM	Dimension-wise Closest Match
DDPM	Denoising Diffusion Probabilistic Models
DoF	Degrees-of-Freedom
DM-MAD	Diffusion Model with Median Absolute Deviation
DM-RANSAC	Diffusion Model with Random Sample Consensus
DM-RUNNING-μ	Diffusion Model with Running Mean
DM-RUNNING-μ-p	Diffusion Model with Running Mean with Episodic Reset
DMBP	Diffusion Model-Based Predictor

DMBP-Lag	Diffusion Model-Based Predictor with Mean Forecasted Offset from Lag-Llama
DPM	Diffusion Probabilistic Models
DQN	Deep Q-Network
DQL	Diffusion Q-Learning
DQL-MeanLag	Diffusion Q-Learning with Mean Forecasted Offset from Lag-Llama
DQL-MedLag	Diffusion Q-Learning with Median Forecasted Offset from Lag-Llama
ERP	Episodic Return Progress
FM	Foundaton Model
FORL	Forecasting in Non-stationary Offline Reinforcement Learning
FORL-DM	Forecasting in Non-stationary Offline Reinforcement Learning's Diffusion Model Component
FQL	Flow Q-learning
FQL-MeanLag	Flow Q-learning with Mean Forecasted Offset
GMM	Gaussian Mixture Model
HLag	Diffusion Model Predicted Offset History-based Lag-Llama
HLag-DCM	History-based Lag-Llama with Dimension-wise Closest Match
IM	Intrinsic Motivation
IQL	Implicit Q-Learning
KDE	Kernel Density Estimation
Kitchen	Kitchen-Complete
Lag	Zero-Shot Forecasting Foundation Model Lag-Llama Extension
LP	Learning Progress
MAML	Model-Agnostic Meta-Learning
MAD	Median Absolute Deviation
MaxLikelihood	Maximum likelihood candidate selection
Maze2d-M	Maze2d-Medium
Maze2D-L	Maze2D-Large
MDP	Markov Decision Process
MedDCM	Median-based Dimension-wise Closest Match

MedNoise	Median-based noise
Meta-RL	Meta-Reinforcement Learning
MLP	Multi-Layer Perceptron
NORML	No-Reward Meta-Learning
NSDP	Non-Stationary Decision Processes
OOD	Out-of-Distribution
PE-TS	Probabilistic Ensembles with Trajectory Sampling
PNN	Progressive Neural Network
POMDP	Partially Observable Markov Decision Process
PPG	Parameterized Policy Gradient
PPO	Proximal Policy Optimization
PPO-CaDM	Proximal Policy Optimization with Context-aware Dynamics Model
RANDOM-BPNN	Random task selection with Bidirectional Progressive Neural Network
RANDOM-MLP	Random task selection with Multi-Layer Perceptron
RANSAC	Random Sample Consensus
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
RORL	Robust Offline Reinforcement Learning
RORL-MeanLag	Robust Offline RL with Mean Forecasted Offset from Lag-Llama
SAC	Soft Actor-Critic
SDE	Stochastic Differential Equations
SRDP	State Reconstruction for Diffusion Policies
TD3+BC	Twin Delayed Deep Deterministic Policy Gradient with Behavior Cloning
TD3+BC-MeanLag	TD3+BC with Mean Forecasted Offset from Lag-Llama
UMCNP	Unsupervised Meta-Testing with Conditional Neural Processes
Vanilla DM	Vanilla Dynamics Model

1. INTRODUCTION

A central challenge in artificial intelligence (AI) is the development of systems capable of reasoning and decision-making that exceed human performance. This thesis confronts this challenge by focusing on a critical and pervasive bottleneck: the human mind's limitations in navigating a complex and uncertain future. The human mind, constrained by cognitive biases and computational limitations, cannot comprehend all possible evolving future scenarios to select the best possible action given the available data.

How can we formalize this problem and build AI agents that can anticipate and make good decisions for a realistic and potentially non-Markovian, uncertain future? The goal is not to pursue the full automation of human choice, but rather to push the boundaries of decision-making systems capable of mapping out robust and adaptive strategies. To realize this vision, such a system must be robust to unseen data, adaptive to new scenarios from few samples, capable of autonomous task switching while learning effectively from multiple tasks, and proactive in anticipating future events and adapting when its current environment deviates from predictions.

Addressing this challenge requires navigating several prominent fields within Reinforcement Learning (RL) [3]. Deep RL is the foundational area that leverages deep learning to make decisions by learning from real-time or recorded interactions with the environment. Despite the remarkable achievements of deep RL algorithms [4, 5], their naive integration reveals significant limitations, such as poor sample efficiency, limited generalization, and adaptation, which have led to a wide range of specialized subfields. Tackling these challenges thus necessitates identifying and engaging with these subfields, each of which addresses specific components of our problem.

The real-world is inherently non-stationary, demanding that the agent be robust to evolving conditions, such as failures or drifts in sensors and actuators [6] affecting the

observation functions. This challenge of adaptation under uncertainty is crucial across diverse application areas, including autonomous driving, disassembly with manipulation robots, medical diagnosis, power grid control, and decision systems that must adapt to evolving human behavior or physiology over time [7]. In these settings, relying on deep RL agents trained on a single task can be catastrophic. This problem is captured by the general non-stationary RL formulation [8] where each component of the MDP or POMDP can change over time. Future-oriented non-stationary reinforcement learning techniques, such as Proactively Synchronizing Tempo (ProST) [9] and Prognosticator [10], tackle the evolution of transition and reward functions over time. However, they presuppose that states can be fully observed and that online interaction with the environment is feasible during training, which is not always possible in practical applications.

Each component of the decision process can change in non-stationary environments; thus, we are faced with distributions of tasks, as tasks can be defined by these components. For instance, different tasks can mean different transition functions, observation functions, and state-action spaces (e.g., morphologically different agents). We need algorithms that can learn in-distribution tasks efficiently, generalize to out-of-distribution tasks, and anticipate forthcoming tasks for improved decision-making.

The few-shot meta-reinforcement learning (meta-RL) framework [11, 12] aims to learn a distribution of tasks during meta-training and adapt to new tasks from the same distribution using a few samples from the new task during meta-testing. Given the high cost associated with collecting data in the real-world at test-time, minimizing the number of online interactions for rapid adaptation is essential for real-world applications. Thus, meta-RL [12] emerges as a promising direction for fast adaptation to an unknown test task. However, since a meta-RL algorithm requires samples from the test task for adaptation, its performance is limited by the number of samples collected from the unknown test task. Improving sample efficiency alleviates dependence on the costly and potentially dangerous process of collecting real-world experiences in an unknown task. This challenge is compounded when the reward signal from the test task is either

unavailable or unreliable due to issues such as sensor malfunctions [13, 11]. To improve sample efficiency during meta-testing in settings with unknown rewards, we need to learn an expressive meta-model for cost-effective sample generation.

Learning a distribution of tasks often involves knowledge transfer among different tasks. Prior approaches [14] have focused on transferring skills between agents with identical action spaces [15, 16] and tasks with pixel-level state inputs [17, 18, 19]. In particular, achieving multi-task reinforcement learning between agents with different physical structures can provide insights for human-inspired RL and act as a critical driving force for machine learning research [20]. This is particularly significant when state and action spaces are different, which provides a unique perspective on knowledge generalization.

Another key challenge involves leveraging large datasets and generalizing to unseen situations, which are critical components of intelligent systems. The former is the focus of offline Reinforcement Learning (RL), which has garnered significant attention for learning from previously collected datasets without interacting with the real world [21]. Offline RL avoids costly or risky online interactions [22, 23]. The main challenges of offline RL are the distribution shift and uncertainty estimation [22]. Since state and action distributions in the offline dataset can differ from those encountered in the evaluation environment, dealing with Out-of-Distribution (OOD) state and action samples is a prominent topic in offline RL [24]. Similarly, OOD generalization is crucial for developing reliable systems that are robust to unexpected conditions. While offline RL aims to find better policies than the behavior policy that generated the trajectories in the dataset without making assumptions about the agents' expertise, it struggles when faced with states not present in the training set.

This thesis argues that achieving robust, anticipatory decision-making requires a fundamental shift from reactive policies to proactive systems capable of reasoning about an uncertain, non-stationary future. We believe that this proactive capability is not monolithic but is built upon distinct yet complementary algorithmic pillars, which this

thesis systematically constructs and validates: sample efficient adaptation in Chapter 4, interleaved multi-task learning in Chapter 5, generalization to OOD states in static experience datasets in Chapter 6, and finally, anticipatory decision making in Chapter 7 to be truly proactive.

First, an agent must adapt rapidly to new tasks with limited data, even when feedback, like rewards, is missing. We address this in Chapter 4 by introducing Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP), a novel hybrid few-shot meta-reinforcement learning (meta-RL) method that uniquely combines, yet distinctly separates, Parameterized Policy Gradient-based (PPG) and task inference-based few-shot meta-RL. Tailored for settings where the reward signal is missing during meta-testing, our method increases sample efficiency without requiring additional samples in meta-training. UMCNP leverages the efficiency and scalability of Conditional Neural Processes (CNPs) [25] to reduce the number of online interactions required in meta-testing. During meta-training, samples previously collected through PPG meta-RL are efficiently reused for learning tasks with hidden contexts in an offline manner. UMCNP infers the latent representation of the transition dynamics model with unknown parameters from a single test task rollout or a few test task transitions. This approach enables us to generate rollouts for self-adaptation by interacting with the learned dynamics model, rather than the test environment. We demonstrate that our method can adapt to an unseen test task using significantly fewer samples during meta-testing than the baselines in the 2D-Point Agent and continuous control meta-RL benchmarks, namely, the Cartpole with unknown angle sensor bias and the Walker agent with random dynamics parameters. This work [1] is published in IEEE Robotics and Automation Letters.

Drawing inspiration from efficient multi-task learning in humans, we investigate training across multiple tasks, particularly between agents with different morphologies. We introduce a novel multi-task reinforcement learning framework named Episodic Return Progress with Bidirectional Progressive Neural Networks (ERP-BPNN) in Chapter 5 for this setting. The proposed ERP-BPNN model learns in a human-

like interleaved manner by autonomously switching tasks based on a novel intrinsic motivation signal and allows bidirectional skill transfer among tasks. ERP-BPNN is a general architecture applicable to several multi-task learning settings. We present the details of its neural architecture and show its ability to enable effective learning and skill transfer among morphologically different robots. The developed Bidirectional Progressive Neural Network (BPNN) architecture enables bidirectional skill transfer without requiring incremental training and seamlessly integrates online task arbitration. The task arbitration mechanism developed is based on Episodic Return progress (ERP), a novel intrinsic motivation (IM) signal. To evaluate our method, we use quantifiable robotics metrics such as 'expected distance to goal' and 'path straightness', in addition to the usual reward-based measure of episodic return common in RL. With simulation experiments, we show that ERP-BPNN achieves faster cumulative convergence and improves performance in all metrics considered among morphologically different reacher robots [26, 27] compared to the baselines. Overall, our method provides a human-inspired and efficient multi-task RL approach with interleaved learning, making it highly suitable for lifelong learning applications. This work [28] is published in IEEE Access.

Recognizing the importance of leveraging large datasets while ensuring robustness to out-of-distribution states, we propose State Reconstruction for Diffusion Policies (SRDP) in Chapter 6. Offline RL methods leverage previous experiences to learn better policies than the behavior policy used for data collection. However, they face challenges in handling distribution shifts due to the lack of online interaction during training. To this end, we propose a novel method named State Reconstruction for Diffusion Policies (SRDP) that incorporates state reconstruction feature learning in the class of diffusion policies [29] to address the problem of out-of-distribution (OOD) generalization. Our method promotes the learning of generalizable state representations to alleviate the distribution shift caused by OOD states. To illustrate the OOD generalization and faster convergence of SRDP, we design a novel 2D Multimodal Contextual Bandit environment, realize it on a 6-DoF real-world UR10 robot, and compare its performance with prior algorithms. In addition, we show the importance of the proposed state reconstruction via ablation studies. We assess the performance of our model on standard offline RL

benchmarks (D4RL) [21], namely the navigation of an 8-DoF ant in a maze and the forward locomotion of half-cheetah, hopper, and walker2d, [26, 27] achieving good performance. Then, we demonstrate that our method can achieve a 167% improvement over the competing baseline on a sparse continuous control navigation task where various regions of the state space are removed from the offline RL dataset in [30], including the region encapsulating the goal. This work is published in IEEE Robotics and Automation Letters [2].

Finally, confronting the critical problem of anticipatory decision making after addressing its tangential components in our aforementioned works, we design the Forecasting in Non-stationary Offline RL (FORL) framework. Existing offline RL methods often assume stationarity or only consider synthetic perturbations at test time [31]. Such assumptions often fail in real-world scenarios characterized by abrupt, time-varying offsets. These offsets can lead to partial observability, causing agents to misperceive their true state and degrade performance. To overcome this challenge, FORL unifies zero-shot time-series foundation models and conditional diffusion-based candidate state generation, trained without presupposing any specific pattern of future non-stationarity. FORL targets environments prone to unexpected, potentially non-Markovian offsets, requiring robust agent performance from the onset of each episode. Empirical evaluations on offline RL benchmarks [21, 32] augmented with real-world time-series data from [33, 34] to simulate realistic patterns of non-stationarity demonstrate that FORL consistently improves performance compared to competitive baselines. By integrating zero-shot forecasting with the agent's experience, we aim to bridge the gap between offline RL and the complexities of real-world, non-stationary environments. This work is accepted to the Neural Information Processing Systems (NeurIPS) as a spotlight [35].

The organization of this thesis is presented as follows: Chapter 2 reviews related work spanning meta-RL, curriculum RL, offline RL, diffusion models in RL, non-stationary RL, multi-task learning, and lifelong learning. Chapter 3 presents a general overview of the background material. Chapter 4 through Chapter 7 present our core

published works detailing the methods and results for UMCNP, ERP-BPNN, SRDP, and FORL, respectively. Chapter 8 discusses the limitations and future work across these contributions. Chapter 9 concludes the thesis with key findings.

2. RELATED WORK

This chapter surveys the key research areas that provide the foundation for the novel methods developed in this thesis. We first examine the literature on meta-reinforcement learning, multi-task learning, and curriculum learning. We then connect these areas to the specific challenges of learning from static datasets and decision making in changing environments by reviewing Offline Reinforcement Learning and non-stationary RL, along with the relevant bayesian function approximation techniques and human-inspired concepts that our work builds upon.

2.1. Meta Reinforcement Learning

Meta-learning provides a framework for learning new tasks more efficiently from limited experience by using knowledge from previously learned tasks [36, 12]. Meta-learning has demonstrated promising results in regression, classification, and deep RL [12, 13, 37, 38, 11].

2.1.1. Parameterized Policy Gradient Based Few-Shot Meta-Reinforcement Learning

Parameterized policy gradient-based few-shot meta-RL methods [39, 40, 12, 11, 41] utilize policy gradient methods in the meta learning inner loop [11]. For instance, in Model-Agnostic Meta-Learning (MAML) [12], a parameterized policy gradient meta-learning algorithm is used, where training and testing tasks are sampled from the same distribution. No-reward Meta-Learning (NORML), proposed by Yang et al. [13], is an extension of the MAML-RL framework [12] for unsupervised meta-testing where the environment dynamics changes across tasks instead of the reward function. NORML aims to utilize past experience to quickly adapt to tasks with unknown parameters from a few samples without reward signals. While these methods have demonstrated promising results, they fall short in sample efficiency in meta-testing. We instead

leverage the conditional neural processes [25] in Chapter 4 to reduce the number of experiences an agent needs for adaptation to the test task.

2.1.2. Black Box Few-Shot Meta-Reinforcement Learning

Black box few-shot meta-RL methods assume less structure in the meta-learning algorithm compared to parameterized policy gradient few-shot meta-RL methods while sacrificing generalization [11]. Prior work on black box meta-RL learns the task-specific context variable from a set of experiences collected by a control policy. Notably, Mishra et al. [42], use memory-augmented models like temporal convolutions and soft attention to learn a contextual representation. Within the domain of meta-imitation learning, Duan et al. [43] encode expert demonstrations into a learned context embedding on which the policy is conditioned on.

2.1.3. Few-Shot Meta Reinforcement Learning with Task Inference

Few-shot meta-RL methods that use task-inference attempt to infer the underlying MDP of the unknown test task in the meta-RL inner loop [11]. Zintgraf et al. [44] use a recurrent neural network (RNN) architecture to encode the posterior over ordered transitions. The posterior belief is used to augment the state variable on which the policy is conditioned. Unlike FOCAL [45], which employs a deterministic encoder, Probabilistic Embeddings for Actor-Critic RL (PEARL) [46], and Variational Bayes-Adaptive Deep RL (VARIBAD) [44] use a stochastic encoder. Latter methods train the inference model by optimizing the Evidence Lower Bound (ELBO)[47] objective. Fully-Offline Context-based Actor-critic meta-reinforcement Learning (FOCAL), on the other hand, clusters transitions in the latent space using an inverse power loss distance metric learning objective under the assumption that tasks are deterministic and that there exists a one-to-one function from the task $\mathcal{T}$ to the cartesian product of the functional space of the transition kernel P and the functional space of the reward function r. In contrast to these approaches, while the goal of UMCNP (Chapter 4) is to identify the latent task parameter in the inner loop, we target few-shot meta-RL

with unsupervised meta-testing where the reward information is missing. Moreover, our method decouples task inference from PPG in meta-RL, allowing flexibility. This separation improves sample efficiency in meta-testing even after training has concluded, as long as the samples from meta-training are accessible.

Rapid Motor Adaptation (RMA) [48], categorized as a model-free task inference meta-RL approach [11], has shown remarkable performance in sim-to-real applications on an A1 robot. RMA initially trains a base policy that takes the latent extrinsics vector encoded by the environment factor encoder, the state, and the previous action. Then, it trains an adaptation module to predict latent extrinsics from a history of state-action tuples. During deployment, it uses the adaptation module and the base policy since the environment factors are hidden during testing. In contrast to previous methods, RMA can adapt to the test task within an episode. Although UMCNP can also adapt within an episode due to its flexibility regarding the number of samples used in training, it does not require environment parameters during training, which may not be available in a robotics dataset, and is only trained with transitions.

2.1.4. Model-based Meta Reinforcement Learning

Model-based meta-RL algorithms focus on learning the dynamics and the reward function of the environment. While they are sample efficient and enjoy the advantages of supervised learning, they can have lower asymptotic performance [11]. Context-aware Dynamics Model (CaDM) [49] encodes a fixed number of transitions and, similar to RMA [48], can adapt within an episode. It can be combined with model-free RL by using the latent vector concatenated with the states to train a model-free policy gradient algorithm. CaDM trains the dynamics model and the context encoder by predicting forward and backward dynamics from a fixed number of state and action transitions. Unlike CaDM, UMCNP handles backward and forward state prediction in a permutation-invariant manner while being agnostic to the number of observations, offering flexibility during testing without the need to train for varying future and history lengths.

2.2. Multi-task Reinforcement Learning

Multi-task learning involves sharing skills and knowledge between multiple tasks, where each task is identified as either a source or a target task during training. Typically, tasks share parts of the neural network model, and the model integrates a task conditioning parameter that defines the task during training. PathNet [50] is a technique that uses a tournament selection genetic algorithm to evolve pathways of a neural network for multi-task, lifelong, and forward transfer learning. However, PathNet [50] is trained consecutively for reinforcement learning tasks, meaning that the source task needs to be trained until convergence before moving on to the target task. This procedure does not allow backward transfer of tasks. Similarly, in Progressive Neural Networks (PNN) [51], training the source task until convergence is required to transfer skills to other tasks connected to the trained task.

Current state-of-the-art methods do not consider human-inspired interleaved learning as a viable approach for multi-task learning; yet, there are potential benefits to adopting a human-like learning strategy. In this vein, ERP-BPNN detailed in Chapter 5 supports bidirectional transfer and does not require convergence in one task to use previously learned representations in other tasks. Hence, ERP-BPNN fits better into the multi-task learning framework, where all tasks are both source and target tasks throughout training. This is essential because the BPNN architecture allows for the integration of learning progress and bidirectional transfer.

2.3. Curriculum Reinforcement Learning

Curriculum learning methods focus on discovering a goal or task sequencing procedure that can lead to faster convergence during training or improved performance compared to random sequencing [52]. Most curriculum learning techniques require a priori domain knowledge of tasks to distinguish task levels for training from easier tasks to more difficult tasks [53, 54, 55]. For example, when predicting the output of a short Python code with Long Short Term Memory (LSTM) networks [56], the task levels

can be identified by the number of nestings and the number of digits in the integers [57]. After manually identifying these task difficulty measures, it can be demonstrated that during training, a combination of a random curriculum and a naive curriculum, in which tasks are selected in ascending order of difficulty, performs better than using only a random or a naive curriculum [57]. However, the difficulty of each task might not be readily available a priori in the robotics domain; thus, automatic or emergent curriculum formation can be desirable. Deep Q-Networks (DQN) with prioritized experience replay [58] assign importance to transitions based on their associated temporal difference error, thereby selecting data for more frequent replay in single-task reinforcement learning. On the other hand, our method prioritizes which task network is allowed to learn in an online manner. Hence, it operates at a higher level than the prioritization scheme of DQN with prioritized experience replay and creates emergent interleaved task-switching patterns on the fly. As such, the learning schedule obtained is quite different from what can be obtained through a transition-level curriculum or a usual task curriculum, where each task is learned to completion.

2.4. Offline Reinforcement Learning

The goal of offline RL is to learn a policy from fixed datasets without interacting online with the environment [22]. A high-level conceptual overview of existing work in offline reinforcement learning (RL) identifies three predominant strategies: policy constraint methods [59, 60, 29], pessimistic value function methods, which assign low values to OOD actions [61], and model-based offline RL methods. Policy constraint methods actively avoid querying OOD actions during training by leveraging probabilistic metrics, which can be explicit [62, 63], implicit [64, 65], f-divergence [66], or integral probability metrics [67]. These metrics ensure that the learned policy π_θ remains close to the behavior policy π_β that generated the offline RL dataset [22]. Similarly, pessimistic value function approaches regularize the value function or the Q-function to avoid overestimations in OOD regions. Model-based offline RL methods [68], on the other hand, focus on learning the environment's dynamics, capitalizing on the strengths of supervised learning approaches. However, in the same vein, these methods

are susceptible to the distribution shift problem in offline RL [22]. Below, we provide the relevant taxonomy of standard offline RL approaches, then we present robust offline RL and the use of diffusion models in offline RL. We detail the offline RL algorithms used in our experiments in Section 7.3.1.1.

2.4.1. Dynamic Programming-based Offline Reinforcement Learning

In dynamic programming-based offline RL methods [22], the Q-function is approximated using the offline RL dataset, which comprises samples collected by the behavior policy. As the learned policy diverges from the behavior policy, Q-function estimates tend to exhibit overestimation in regions where uncertainty is high [67]. Prior works impose a constraint on the statistical distance between the learned policy and the behavior policy in the policy optimization objective [22] or the reward function [63]. However, these methods require the computation of the behavior policy through behavior cloning to enforce a constraint on the learned policy. Nair et al. [65] alleviated the need for computing the behavior policy by approximating the policy objective using an advantage-weighted maximum likelihood. Still, this method is prone to distribution shift as the policy can query the OOD actions during training. In Conservative Q-Learning (CQL) [69], the Q-function is approximated using a minimax objective where overestimated action values are minimized, and actions from the dataset are maximized. However, conservative estimation of the value function is prone to overfitting when data is scarce [22].

2.4.2. Importance Sampling-based Offline Reinforcement Learning

Importance sampling-based offline RL methods attempt to approximate the learned policy or expected return via off-policy RL techniques. However, off-policy RL allows environment interaction in the training loop, whereas offline RL learns from a static replay buffer constructed before training. The accuracy of the importance sampling estimator depends on the proximity of the learned policy to the policy used for data collection, the dimensionality of the state-action space, and the horizon of

the task. Hence, these algorithms have limited applicability to real tasks and impose constraints on the objective of obtaining the best policy supported by the data. Prior work on importance sampling-based offline RL addresses the bias-variance trade-off. The marginal importance ratio [70] can be used via dynamic programming to reduce bias. The doubly robust estimator [71] uses recursive regression-based value evaluation to reduce variance. However, these methods are still prone to distribution shifts, as they exploit OOD regions in the policy improvement step. Hence, in Chapter 6, we use an auxiliary state reconstruction signal in the policy improvement step that encourages learning more generalizable state features.

2.4.3. Model-based Offline Reinforcement Learning

Model-based offline RL methods focus on deriving the environment model by estimating the transition function, unlike model-free methods. Although, theoretically, their advantage over model-free methods has not yet been proven [22], they offer sample efficiency and quick adaptation. In the model-based offline RL setting, the model cannot correct OOD states in addition to the OOD actions. To avoid OOD states and actions, a scaled uncertainty function penalizes the reward function over state action pairs in [72]. However, estimating an uncertainty function that accurately quantifies uncertainty regions over the state action space is a challenging open problem [22]. Conservative Model-Based RL [68], on the other hand, follows a similar approach used in CQL [69] and penalizes overestimated Q-values of state action tuples sampled from the model distribution. Most model-based RL algorithms are myopic and fail to make accurate predictions in tasks where the dimensions of the state action space are high [22]. Other works view the problem through the lens of sequence modeling and use high-capacity transformers [73], [74]. However, these architectures are computationally expensive to train over long horizons and often require careful tuning of hyperparameters, as validation and hyperparameter optimization in offline RL are challenging.

2.4.4. Robust Offline Reinforcement Learning

Park et al. [75] emphasize the challenges of generalizing policies to test-time states that are not in the support of the offline RL dataset. While prior works have investigated the issue of generalization [2, 76], in testing-time robust RL methods [31], this challenge is exacerbated by the introduction of noise into the states by an unknown adversary. Testing-time robust offline RL methods such as DMBP [31] and RORL [77] address scenarios where a noise-free, stationary dataset is used for training, but corruption is introduced at test-time. This is distinct from training-time robust offline RL [78, 79], which assumes a corrupted training dataset. Both RORL [77] and DMBP [31] assume access only to a clean, uncorrupted offline RL dataset (similar to FORL in Chapter 7), and they are evaluated in a perturbed environment. To the best of our knowledge, FORL is the first work to extend this setting to a non-Markovian, time-evolving, non-stationary deployment environment. We focus on time-dependent exogenous factors from real-data which is aligned with the passive non-stationary environment definition [8].

2.4.5. Diffusion Models in Offline Reinforcement Learning

Diffusion models are probabilistic generative models widely used in computer vision [80, 81, 82, 83], natural language processing [84, 85], and RL [86, 29, 87, 88]. Diffusion probabilistic models (DPM) [80] formulate a forward diffusion process by adding a small amount of Gaussian noise to the data samples. By learning the reverse diffusion process, diffusion models can generate samples from a simple prior such as a standard Gaussian. Under a suitable parameterization, the weighted variational lower bound objective in denoising diffusion probabilistic models (DDPMs) [89] is close to a denoising score matching objective over multiple noise levels. The sampling procedure is closely related to annealed Langevin dynamics in image synthesis tasks. Score-based generative models (SGMs) use a noise-conditional score network to estimate the scores at different noise levels after perturbing the data with a variance schedule [90]. Although our approach, SRDP (Chapter 6), is developed for the offline RL framework, it can

be applied to conditional diffusion models that use classifier guidance. In our case, however, the guidance comes from the states.

In RL [91, 92], diffusion models [80] have seen widespread adoption due to their remarkable expressiveness, particularly in representing multimodal distributions, scalability, and stable training properties. In the context of offline RL, diffusion models have been used for representing policies [29, 87, 88, 2], planners [86, 93], data synthesis [94, 95], and removing noise [31]. Notably, Diffusion Q-learning [29] leverages conditional diffusion model policies to learn from offline RL datasets, maintaining proximity to behavior policy while using Q-value function guidance.

In SRDP (Chapter 6), we introduce an architectural change to the noise prediction model of diffusion policies and add a state reconstruction signal to the behavior cloning objective to generalize to out-of-distribution (OOD) states. Our method FORL (Chapter 7) harnesses diffusion models in another way: it learns from a sequence of action and effect tuples, leveraging the multimodal capabilities of diffusion models to identify diverse candidate locations of the hidden states.

2.5. Reinforcement Learning in Non-Stationary Environments

Existing works in non-stationary reinforcement learning predominantly focus on adapting to changing transition dynamics and reward functions. Ackermann et al. [96] propose an offline RL framework that incorporates structured non-stationarity in the reward and transition functions by learning hidden task representations and predicting them at test time. Although our work also investigates the intersection of non-stationary environments and offline RL, we assume stationarity during training. To learn adaptive policies online, meta-learning algorithms have also been used as a promising approach [97, 98, 1]. Al-shedivat et al. [98] explore a competitive multi-agent environment where transition dynamics change. While these approaches provide valuable insights, they often require samples from the current environment and struggle in non-trivial non-stationarity, highlighting the need for anticipatory methods [10, 9]. Examples of

such future oriented approaches include Proactively Synchronizing Tempo (ProST) [9] and Prognosticator [10], which address the evolution of transition and reward functions over time. ProST [9] leverages a forecaster, namely Auto-Regressive Integrated Moving Average (ARIMA), and a model predictor to optimize for future policies in environments to overcome the time-synchronization issue in time-elapsing MDPs. This approach aligns with our focus on time-varying environments and similarly utilizes a real-world finance dataset to induce non-stationarity. Both ProST and Prognosticator assume that states are fully observable during testing, and online interaction with the environment is possible during training, conditions that are not always feasible in the real world. Instead, our approach assumes that states are not fully observable and that direct interaction with the environment during training is not feasible, necessitating that the policy be learned exclusively from a pre-collected dataset.

2.6. Conditional Neural Processes

Conditional Neural Processes (CNPs) [25] predict the parameters of a probability distribution conditioned on a permutation invariant prior data and target input. CNPs aim to represent a family of functions using Bayesian Inference and the high representational capacity of neural networks [99, 100, 101, 102]. Prior approaches have leveraged the robust latent representations obtained by the CNP model in high-dimensional trajectory generation [103], long-horizon multimodal trajectory prediction [103, 104], and sketch generation tasks [105].

2.7. Human Learning

Numerous examples demonstrate how neuroscience and artificial intelligence have paved the way for each other [106, 107, 8]. In this vein, the ability of humans to acquire multiple skills with ease throughout their lives may guide machine learning research in multi-task learning and lifelong learning. Human learning, especially during infancy and childhood, is characterized by autonomous engagement in play, i.e., exploration, where no task is learned completely in one sitting. Besides being ecologically unreasonable to

focus on learning a single task until mastery, interleaved learning may allow positive skill transfer among tasks from partial learning if adequate mechanisms are engaged. This notion is supported by studies on the contextual-interference (CI) effect, which show that practicing tasks in an interleaved regime often results in improved learning compared to practicing in a blocked order [108]. Benefits of CI have been linked to increased brain activity during interleaved practice as opposed to repetitive practice [109, 110]. In addition, the CI effect not only improves information retention but also skill transfer between similar tasks [111].

On the other hand, machine learning settings usually prefer blocked learning to interleaved learning. Multi-task learning settings generally assume that either a random task (or a subset of tasks) is chosen for each training trial or that training proceeds to the next task after one task is mastered, with a few exceptions [112, 113]. In the case of continual learning, tasks are learned sequentially as each task arrives [114]. This is in contrast to how humans learn. It is clearly evident that the beneficial impacts of interleaved learning have not been thoroughly examined within the machine learning literature thus far. Investigating these effects has the potential to enhance the alignment of artificial intelligence with the underlying mechanisms of the human brain.

2.8. Intrinsic Motivation

Intrinsic motivation (IM) is a significant topic in infant cognitive development and learning, which refers to motivation originating from innate satisfaction instead of the extrinsic rewards gained from the environment [115, 116]. IM has been adopted to enable open-ended robot learning [117] and improve self-supervised exploration [118, 119]. Intrinsic rewards can arise from exploring novel states, satisfying an ingrained curiosity, or the rate of acquiring skills and knowledge in an environment. The inclusion of intrinsic rewards alongside the extrinsic rewards from the environment is one way to solve challenging exploration problems [120] and discover a diverse set of skills [121] in deep reinforcement learning. In our approach, unlike the existing uses of IM in reinforcement learning, we propose to utilize ERP in Chapter 5 as a higher-level IM

signal to dynamically switch tasks for learning in an online manner instead of modulating the reward signal guiding RL. Consequently, the dynamic task switching carried out by the task selection mechanism leads to an emergent interleaved multi-task learning regime.

3. BACKGROUND

In this chapter, we first formalize the sequential decision-making problem using the standard reinforcement learning and offline RL frameworks. We then define formalisms for non-stationary environments and Partially Observable MDPs (POMDPs). Finally, we review the principles of diffusion probabilistic models, the generative methodology we employ in Chapters 6 and 7.

3.1. Reinforcement Learning

Markov Decision Processes (MDPs) are often used to formalize RL. An MDP is defined by the tuple $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, P, R, \rho_0, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel ($s_{t+1} \sim P(\cdot|s_t, a_t)$), $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ_0 is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor [122, 3].

The objective of an RL agent is to maximize the expected cumulative discounted reward $\mathbb{E}_\pi \left[\sum_t \gamma^t R\left(s_t, a_t\right)\right]$ by learning a policy $\pi_\theta(a_t|s_t)$, parameterized by θ. The advantage function for the state and action tuple (s, a) can be formulated as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, where the value function is

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right], \tag{3.1}$$

for the policy starting from the state s [123]. The Q-function is defined as

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a \right], \tag{3.2}$$

following the policy π conditioned on the state s and action a. The advantage function estimates the relative advantage of taking an action in that state with respect to the estimated value of that state [124].

In offline RL, the agent is tasked with learning the best policy supported by the dataset of transitions denoted by $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}$. The dataset is constructed

from the rollouts obtained by a behavior policy $\pi_\beta(a|s)$ [22]. For clarity, we use s' as s_{t+1} interchangeably to denote the next state.

3.2. Non-Stationary Environments

A Partially Observable Markov Decision Process (POMDP) is defined by the tuple [125, 7, 8]

$$\hat{\mathcal{M}} = \big(\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, \mathbf{e}, \rho_0, \gamma\big), \tag{3.3}$$

where $\mathcal{O}$ is the observation space and $\mathbf{e}$ is the observation function. The observation function, $\mathbf{e}$, can be deterministic ($\mathbf{e} : \mathcal{S} \to \mathcal{O}$) [126, 8] or stochastic ($\mathbf{e}(o \mid s)$ for $o \in \mathcal{O}, s \in \mathcal{S}$) [7].

To capture a broader range of non-stationary RL scenarios, Khetarpal et al. [8] present a general non-stationary RL formulation, which allows each component of the underlying MDP or POMDP to evolve over time. Concretely, this time-varying tuple is denoted as $\big(\mathcal{S}(t), \mathcal{A}(t), P(t), R(t), \mathbf{e}(t), \mathcal{O}(t)\big)$ where each element is represented by a function $\upsilon(i, t)$, indicating its variation with time t and input i. Scope [8], denoted by a set κ, specifies which of these components vary, and a driver determines how they evolve.

Definition 3.2.1 (Scope of Non-Stationarity [8])**.** Given the general non-stationary RL framework, the scope of non-stationarity [8] is the subset

$$\kappa \subseteq \{\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, \mathbf{e}\},$$

indicating which components of the environment evolve over time.

Following Khetarpal et al. [8], we distinguish different drivers of non-stationarity as passive, active, and hybrid. Passive drivers of non-stationarity imply that exogenous factors alone govern the evolution of the environment, independent of the agent's actions [8]. When the driver of non-stationarity is active, the agent's actions affect how the environment evolves over time. In the hybrid setting, the non-stationarity is driven by both the agent's actions and exogenous factors [8].

Notably, Chandak et al. [7] defines a Non-Stationary Decision Process (NSDP) as a sequence of POMDPs, with non-stationarity confined to the subset $\kappa \subseteq \{P, R, \mathbf{e}\}$, where the initial state distribution $\rho_{0,j}$ varies between POMDPs. Existing research in non-stationary RL largely focuses on the episodic evolution of transition dynamics and reward functions [96]. In contrast, the evolution of observation functions remains underexplored, despite its potential for real-world applicability and thus demands further investigation [8]. Handling inaccurate state information is crucial in non-stationary RL since an agent may encounter non-stationarity due to its own imperfect perception of the state, while the underlying physics of the environment remains unchanged [8].

3.3. Diffusion Models

Diffusion probabilistic models [80, 127, 89], commonly referred to as diffusion models for brevity, are a class of probabilistic generative models that seek to generate new samples by learning the underlying probability distribution of the data. The forward diffusion process follows a Markov chain to gradually destroy the structure of an original data sample, $\mathbf{x}_0$, by adding noise to obtain a sequence of noisy samples $\mathbf{x}_1 ... \mathbf{x}_N$. Here, the Gaussian noise added to the data depends on a variance schedule $\{\beta_n \in (0,1)\}_{n=1}^{N}$, where β_n is the per-step variance at diffusion timestep n. Since the sequences of noisy samples are available during training, we can train a neural network to predict the noise $\boldsymbol{\epsilon}$ added to the sample at a given timestep. At a high level, we can generate samples from Gaussian noise through an iterative denoising procedure in the reverse diffusion process by using the predicted noise added at each diffusion timestep.

In diffusion models, we can directly compute the noisy sample at any timestep n in closed form. By defining the variance β_n with $\alpha_n = 1 - \beta_n$, we obtain the distribution $q(\mathbf{x}_n \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_n; \sqrt{\bar{\alpha}_n}\mathbf{x}_0, (1 - \bar{\alpha}_n)\mathbf{I})$ using recursion and reparameterization, where $\bar{\alpha}_n$ is defined by the cumulative product $\bar{\alpha}_n = \prod_{i=1}^{n} \alpha_i$. To estimate the reverse process, we learn the parameters of $p_\theta(\mathbf{x}_{n-1} \mid \mathbf{x}_n) = \mathcal{N}(\mathbf{x}_{n-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_n, n), \boldsymbol{\Sigma}_\theta(\mathbf{x}_n, n))$. The joint

distribution of the reverse diffusion process is denoted by

$$p_\theta\left(\mathbf{x}_{0:N}\right)=p\left(\mathbf{x}_N\right)\prod_{n=1}^{N}p_\theta\left(\mathbf{x}_{n-1}\mid\mathbf{x}_n\right), \tag{3.4}$$

where $\mathbf{x}_N$ is sampled from the isotropic Gaussian distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$. Using a pre-specified time-dependent schedule, Ho et al. [89] formulated the simplified loss function in DDPMs for diffusion timestep n with

$$L_n=\mathbb{E}_{n,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_n}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon},n\right)\right\|^2\right], \tag{3.5}$$

where $\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_n}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon},n\right)$ is the noise predicted by the neural network. Here, n is sampled from the uniform distribution $n\sim U(\{1,\ldots,N\})$, and $\boldsymbol{\epsilon}$ is the true noise sampled from the Gaussian distribution $\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$.

4. UNSUPERVISED META-TESTING WITH CONDITIONAL NEURAL PROCESSES FOR HYBRID META-REINFORCEMENT LEARNING

4.1. Introduction

This chapter investigates adaptive meta-RL settings, where the transition function varies across tasks drawn from a distribution, and the reward signal is missing during meta-testing. Our work builds on the meta-RL framework [128, 12, 11], where a distribution of tasks shares a common structure but differs in the transition dynamics model. This assumption allows unsupervised meta-testing [13, 11] in the new task with unknown transition dynamics model parameters.

Imagine a setting where an agent is tasked with moving to a goal location in different environments parameterized by different force fields. Different force fields push the agent in different directions; requiring the agent to move with few trial-and-error attempts. Similarly, in instances of sensor malfunction, an unknown bias can result in inaccurate signals, necessitating the robot to adapt to a new environment with minimal damage [129, 13].

In these settings, each transition function depends on a hidden dynamics parameter during meta-training and meta-testing. Sample efficiency and not relying on reward signals are crucial, especially when engaging with a novel scenario referred to as the "test task," where the number of real-world interactions is limited. No-reward Meta Learning [13] is an unsupervised meta-testing algorithm that adapts without reward signals using a learned advantage function. Similarly, Context-aware Dynamics Model (CaDM) [49] and Rapid Motor Adaptation (RMA) [48] adapt within an episode without relying on rewards during meta-testing, though they use a fixed length of transitions during training and testing.

We introduce a novel hybrid meta-RL formulation that decouples task inference from parameterized policy gradient-based (PPG) meta-RL [12, 11]. Our approach diverges from prior works [46, 44] by targeting meta-testing sample efficiency through a modular approach while not requiring reward information during meta-testing. This strategy substantially improves sample efficiency in meta-testing by predicting next states given target state-action queries, thereby generating cost-effective samples from the meta-model rather than relying on costly online environment interactions. Furthermore, Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP) allows reusing samples collected from PPG meta-RL meta-training for task inference in an offline manner, without compromising sample efficiency during meta-training. This decoupling enhances the flexibility and applicability of our approach in scenarios where retraining the meta-policy with online interactions to improve meta-testing sample efficiency is not feasible.

The key contributions of our method, UMCNP, include (1) improving sample efficiency via fast adaptation to the unseen test task (which requires only a single rollout or a few transitions from the unseen test task compared to the 25 rollouts used in prior works [13]), (2) generating and learning meta-dynamics models with no access to test task parameters and rewards while being permutation invariant and agnostic to the number of samples, (3) utilizing offline datasets for task inference in meta-RL with unsupervised meta-testing, and (4) decoupling PPG-based and task inference-based meta-RL, offering flexibility of application and enhancing informative trajectory generation. We evaluate our method using a point agent, a cartpole with sensor bias, and the widely used Walker Agent Environment featuring variable dynamics [13]. Our results demonstrate a remarkable improvement in sample efficiency over the unsupervised meta-testing baselines.

4.2. Preliminaries

4.2.1. Model-Agnostic Meta-Learning

In meta-RL, we learn to adapt to a distribution of tasks represented by $p(T)$, where each task is distinguished by a unique Markov Decision Process (MDP) despite sharing a common structure. Typically, the tasks share identical state and action spaces, yet they vary in terms of their reward functions and transition dynamics. A prominent approach for gradient-based meta-RL is the Model-Agnostic Meta-Learning (MAML) approach [12]. During meta-training, MAML trains a meta-policy π_θ, also known as an initial or base policy [11], parameterized by θ. To adapt quickly to a new task from this distribution, the meta-policy requires only a small number of samples, $\mathcal{D}_i^{train}$, following the equation

$$\phi_i = \theta + \alpha_{in} \nabla_\theta \hat{\mathcal{J}}(\theta, \mathcal{D}_i^{train}), \tag{4.1}$$

where α_{in} is the adaptation learning rate, ϕ_i is the adapted policy parameter, and $\hat{\mathcal{J}}(\theta, \mathcal{D}_i^{train})$ is the estimated RL objective for task T_i [12]. After applying this adaptation step separately to a batch of tasks in the inner loop, we obtain $\mathcal{D}_i^{test}$ from each task with the adapted parameters. Subsequently, we compute the meta-objective $\max_\theta \sum_{T_i \sim p(\mathcal{T})} \hat{\mathcal{J}}(\phi_i, \mathcal{D}_i^{test})$ [12] in the outer loop using

$$\max_\theta \sum_{T_i \sim p(\mathcal{T})} \hat{\mathcal{J}}(\theta + \alpha_{in} \nabla_\theta \hat{\mathcal{J}}(\theta, \mathcal{D}_i^{train}), \mathcal{D}_i^{test}) \tag{4.2}$$

with respect to the meta policy parameters θ [12, 11]. During evaluation, the agent collects a few samples in the new test task and adapts its meta policy using Equation (4.1).

4.2.2. No-Reward Meta Learning

Provided that the change in dynamics can be represented as a set of state ($\boldsymbol{s}$)-action ($\boldsymbol{a}$)-next-state ($\boldsymbol{s'}$) tuples from the same task $\{(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'})\}$, NORML [13] learns a pseudo-advantage function $A_\psi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'})$ [13]. It is important to note that the aim of A_ψ is to guide the meta-policy adaptation in Equation (4.1) instead of fitting to the advantage function. Thus, we refer to the learned advantage function

as the pseudo-advantage function. Task-specific gradient update in NORML, $\boldsymbol{\phi}_i = \boldsymbol{\theta} + \boldsymbol{\theta}_{\text{offset}} + \alpha_{in} \nabla_{\boldsymbol{\theta}} \hat{\mathcal{J}}_{\mathcal{T}_i}^{\text{NoRML}} \left(\boldsymbol{\theta}, D_i^{\text{train}} \right)$, is computed [13] with

$$\boldsymbol{\theta} + \boldsymbol{\theta}_{\text{offset}} + \alpha_{in} \sum_{D_i^{\text{train}}} A_{\boldsymbol{\psi}} \left(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}' \right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} \left(\boldsymbol{a} \mid \boldsymbol{s} \right), \tag{4.3}$$

where A_ψ is used to compute task-specific parameters in the meta-learning inner loop from a set of state, action, and next-state transitions of task i denoted as D_i^{train} that do not contain reward signals. In this adaptation phase, the pseudo-advantage function guides the policy gradient update. To encourage decoupling between the meta policy and the task-specific policy in the inner loop, we learn an offset parameter $\boldsymbol{\theta}_{\text{offset}}$ end-to-end [13]. This offset parameter, added in the inner loop adaptation formulated in Equation (4.3), is identical for all tasks.

The learned pseudo-advantage function A_ψ is optimized in the meta-learning outer loop. Using D_i^{train} and D_i^{test}, we not only update $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$ in the outer loop, but we also update the offset parameters $\boldsymbol{\theta}_{\text{offset}}$ [13]

$$\begin{bmatrix} \theta \\ \theta_{\text{offset}} \\ \psi \end{bmatrix} \leftarrow \begin{bmatrix} \theta \\ \theta_{\text{offset}} \\ \psi \end{bmatrix} + \alpha_{out} \sum_{T_i \sim \mathcal{T}} \nabla_{\left[\theta^T \theta_{\text{offset}}^T \psi^T\right]^T} \hat{\mathcal{J}}_{T_i}(\phi_i, D_i^{\text{test}}), \tag{4.4}$$

where $\nabla \hat{\mathcal{J}}_{T_i}(\phi_i, D_i^{\text{test}})$ is the gradient of the meta objective formulated as

$$\sum_{D_i^{\text{test}}} A^\pi(s,a) \nabla \log \pi_{\phi_i}(a|s), \tag{4.5}$$

$A^\pi(s,a)$ is the RL advantage function, and α_{out} is the learning rate.

4.2.3. Conditional Neural Processes

The architecture of Conditional Neural Processes (CNPs) [25] consists of parameter-sharing encoders and a decoder named the query network [25]. Initially, a set of random input and output pairs $(x_{f^i}, y_{f^i}^{true})$ is sampled from a function $f^i \in F$. These pairs are then encoded into latent representations through a parameter-sharing encoder network. Subsequently, the average of these representations is computed, further ensuring that the model is invariant to permutations and the number of inputs. The averaged

representation is then concatenated with the target input query and passed to the query network. Finally, the query network generates the predicted mean and standard deviation for the queried input (x^q_{fi}).

4.3. Unsupervised Meta-Testing with Conditional Neural Processes

In this section, we present our method, UMCNP, for learning dynamics models from trajectories collected by PPG meta-RL. By sampling actions from the policy, the UMCNP model facilitates the autoregressive generation of complete trajectories. This process involves querying the model with the action and the state, receiving a prediction of the next state, and then feeding this state back into the model alongside new actions. A learned pseudo-advantage function takes state, action, and next state as input and guides the policy gradient during adaptation. As a result, UMCNP effectively creates models of unknown environments from fewer samples without accessing environment parameters or rewards, addressing the challenge of unsupervised meta-testing.

Gradient-Based Meta Reinforcement Learning with Unsupervised Meta-Testing. During meta-training, we store transitions for each $task_i$ as a set of transition tuples $B_i = \{(s, a, s')\}_{t=0}^{H-1} \subset S \times A \times S$ without the task parameter and the rewards. We then use these batches to create our offline dataset illustrated in Figure 4.1.

Policy parameters in the inner loop adaptation in Equation (4.3) and the outer loop meta-optimization in Equation (4.4) are updated using vanilla PG [130] and actor-critic Proximal Policy Optimization (PPO) [4] methods, respectively. We use the Adam optimizer [131] for both the inner loop and the [13] outer loop updates. After we learn the meta-policy, the offset parameter $\boldsymbol{\theta}_{offset}$ and the pseudo-advantage function $A_\psi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'})$ using NORML, we train our UMCNP model using the offline dataset of transitions $D = \{B_i\}$.

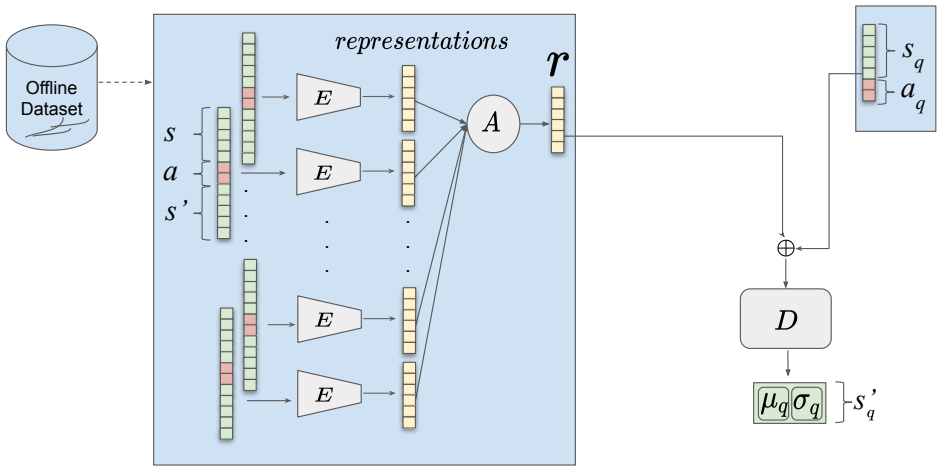


Figure 4.1. Meta-training procedure of our proposed method UMCNP (© 2024 IEEE [1]).

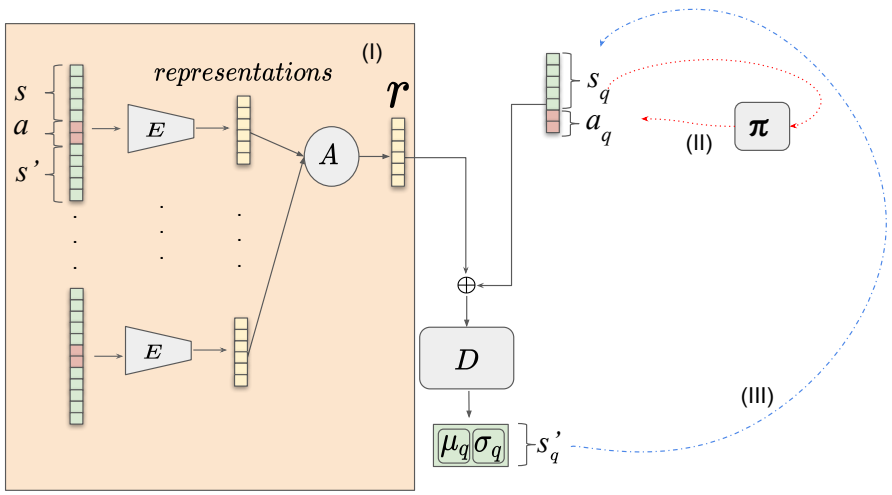


Figure 4.2. Meta-testing procedure of our proposed method UMCNP (© 2024 IEEE [1]).

Unsupervised Meta-Testing with Conditional Neural Processes (UMCNP). We train the UMCNP offline using unlabeled batches of transition tuples that we previously collected during the first phase of online PPG meta-training. Figure 4.1 illustrates the

second phase of the training procedure for UMCNP directed towards task inference. Notably, the transition dynamics parameter is hidden, and the data does not contain reward information during training and testing. In each meta-training iteration, we randomly select an unlabeled batch $B_i = \{(s, a, s')\}_{t=0}^{H-1}$ from the offline dataset D. From this batch, we randomly select a set of task-specific transition tuples $\{(s_k, a_k, s'_k)\}$ and a target query tuple (s_q, a_q, s'_q) without replacement. We utilize (s_q, a_q, s'_q) for the target state-action query $[s_q, a_q]$ and the true target next state label $[s'_q]$.

Next, we encode each tuple (s_k, a_k, s'_k) into a fixed size representation using a parameter sharing encoder network. These representations are then passed through an averaging module M_{avg} to obtain a learned latent vector $\boldsymbol{r}$ that represents the hidden environment transition function parameter used in batch B_i. We then concatenate the resulting latent representation $\boldsymbol{r}$ with the $[s_q, a_q]$ to predict the distribution parameters μ_q, σ_q of the next state s'_q conditioned on $\boldsymbol{r}$ and the target query $[s_q, a_q]$, using the formula

$$\mu_q, \sigma_q = f_{\theta_D}\left([s_q, a_q] \oplus \frac{1}{k}\sum_k g_{\theta_E}\left(s_k, a_k, s'_k\right)\right), \tag{4.6}$$

where f_{θ_D}, g_{θ_E} are the decoder and encoder networks, (s_k, a_k, s'_k) is a randomly sampled transition from the set B_i. The loss function of UMCNP can be expressed as

$$\mathcal{L}(\theta_E, \theta_D) = -\log P\left(s_q^{(\prime)true} \mid \mu_q, \mathit{softplus}(\sigma_q)\right). \tag{4.7}$$

The softplus activation function [132] is applied to σ_q, and the resulting output, along with μ_q, is a parameter of a multivariate normal distribution with a diagonal covariance. A high variance in this distribution can indicate high uncertainty in the predicted next state value.

During meta-testing in Figure 4.2, the agent collects a few samples $\{(s_k, a_k, s'_k)\}$ from the unseen test task without rewards. A trained encoder network with shared weights encodes each sample into a fixed-size representation. Figure 4.2 shows (I) how an averaging module (M_{avg}) forms a learned shared representation of the test task dynamics. Using this shared latent vector representing the hidden task parameter, we

can generate cost-effective rollouts using our meta-policy. This representation helps in predicting the parameters of the next-state distribution with the state-action query $[s_q, a_q]$.

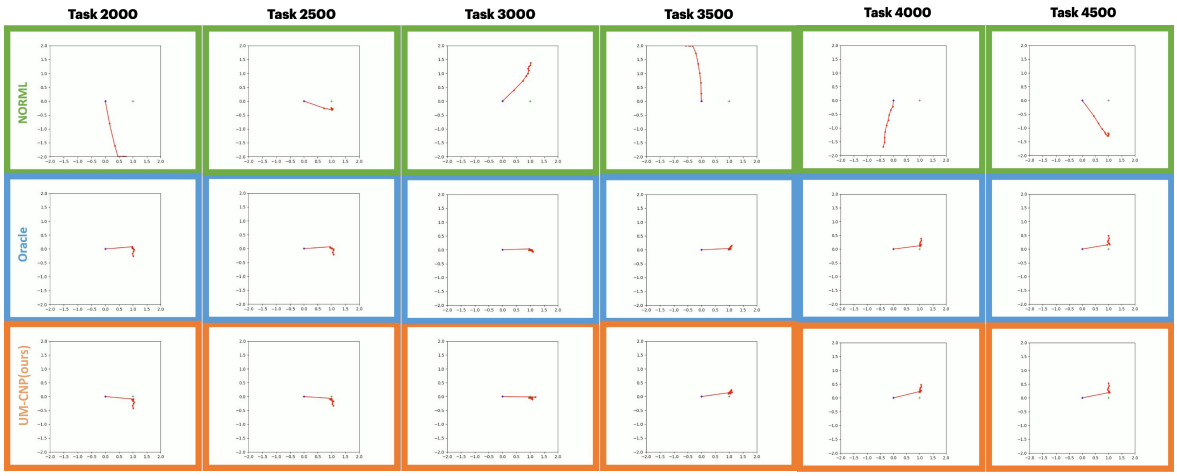


Figure 4.3. Test task indices 2000, 2500, 3000, 3500, 4000, and 4500 correspond to task parameter ω with values $-2\pi/10$, 0, $2\pi/10$, $4\pi/10$, $6\pi/10$, $8\pi/10$ respectively (© 2024 IEEE [1]).

For every rollout generated by UMCNP, we use the same true initial state sampled from the test task. (II) We sample the action query a_q from the stochastic meta-policy conditioned on this state, depicted in Figure 4.2. Due to the stochasticity of the meta-policy trained with PPO [4], we generate a diverse set of rollouts from our UMCNP model which can facilitate informative exploration for our meta-policy. (III) We then use the predicted next state to sample the subsequent action query from the policy (see Figure 4.2), repeating the steps (II-III) and querying the decoder until the episode concludes. We combine these rollouts generated from UMCNP with the ground-truth transitions or rollout sampled from the test task.

The pseudo-advantage network $A_\psi(s, a, s')$ computes advantage estimates for each tuple in the combined set of generated rollouts and a ground-truth rollout. Finally,

we perform a gradient update on the meta-policy for fast adaptation to the test task. This update uses the estimated advantage values, the learned parameter offset, and the rollouts generated by UMCNP alongside a single rollout or a few transitions from the test task. Although the reward information is completely missing in meta-testing, and only available during the first phase of meta-training, our model is capable of a remarkably sample-efficient unsupervised adaptation.

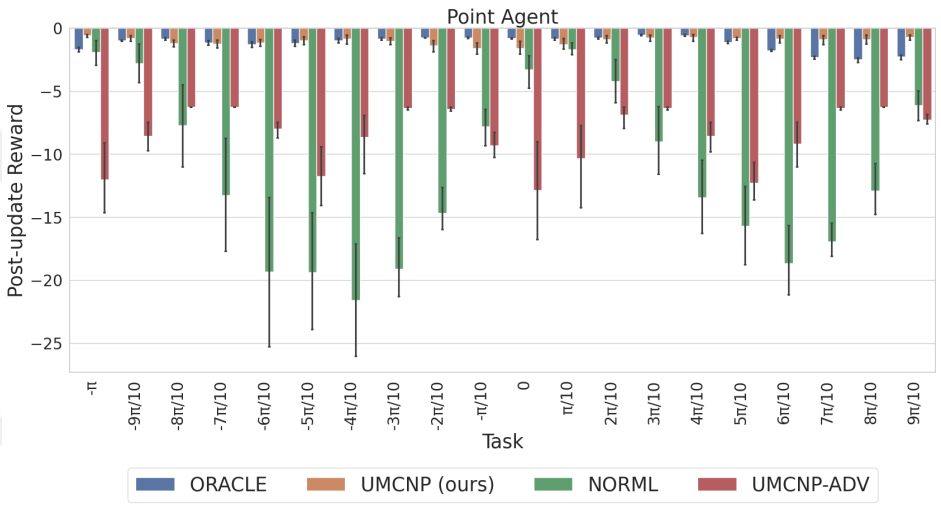


Figure 4.4. Point agent plot shows the post-update cumulative reward (sum of rewards) (y-axis) in test tasks, computed after finetuning with 24 generated rollouts and 1 actual rollout for UMCNP (ours) (orange), 1 actual rollout for NORML [13] (green), and 25 actual rollouts for ORACLE/Ground-truth (blue) and UMCNP-ADV (red) with 95% confidence intervals (© 2024 IEEE [1]).

4.4. Experiments

In this section, we analyze the performance of our method and compare it with meta-RL baselines in 2D point agent and continuous control tasks, including cartpole with sensor bias and bipedal locomotion of a walker agent with random dynamics parameters. The ORACLE agent, trained using NORML, serves as a baseline to demonstrate the performance when extensive sampling from the test task is allowed.

UMCNP, NORML [13], Probabilistic Ensembles with Trajectory Sampling CaDM (PE-TS CaDM) [49, 133], Vanilla CaDM [49], PE-TS [133], Proximal Policy Optimization-CaDM (PPO-CaDM) [49, 4], and Vanilla Dynamics Module [49] have limited access to the test task during meta-testing, unlike ORACLE [13]. Despite this limitation, UMCNP achieves fast adaptation to the test task with significantly less interaction with the test task than the baselines, matching ORACLE's performance. In all our experiments, we report results with 95% confidence intervals computed over 5 random seeds.

4.4.1. 2D Point Agent with Unknown Artificial Force Field

The goal of the point agent, initialized at $(x = 0, y = 0)$, is to move to the position $(x = 1, y = 0)$, where $(x, y) \in [-2, 2] \times [-2, 2]$ are the positions on the 2D plane. We are interested in a meta-RL setting used in [13] where the reward function is identical across multiple tasks. Different tasks are created by generating different artificial force fields that push the agent in different directions using the parameter (ω). Dynamics of the 2D point task in [13] follow

$$\begin{bmatrix} s_{t+1}^{(1)} \\ s_{t+1}^{(2)} \end{bmatrix} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} a_t^{(1)} \\ a_t^{(2)} \end{bmatrix} + \begin{bmatrix} s_t^{(1)} \\ s_t^{(2)} \end{bmatrix}, \tag{4.8}$$

where state vector denoted by $\left[s_t^{(1)}, s_t^{(2)}\right]^T$ corresponds to the x, y-coordinates on the 2D plane, rotation parameter is defined by ω, and the action vector at time t, $\left[a_t^{(1)}, a_t^{(2)}\right]^T$, refer to the movement in the x,y-directions with $a_t^{(1)}, a_t^{(2)} \in [-1, 1]$ [13]. The fixed episode length in this environment is 10. We use the same reward function, the negative Euclidean distance from the goal position, and hyperparameters in NORML for comparative analysis.

Table 4.1. 2D point agent environment post-update reward with 95% confidence intervals (© 2024 IEEE [1]).

Point Agent	**mean**	**median**
UMCNP(ours)	**-1.02** ± 0.09	**-0.91**
NORML	-11.49 ± 1.44	-11.01
UMCNP-ADV	-8.49 ± 0.58	-7.46
ORACLE	-1.20 ± 0.12	-0.94

In the Point Agent environment, 5000 tasks are defined over the $[-\pi, \pi]$ interval. The agent is initially trained across a distribution of 5000 tasks, i.e. in environments with 5000 different force fields. Then, it is evaluated in unseen test tasks. The rollouts obtained from direct interactions with the test task will be referred to as actual rollouts. We have also trained our UMCNP model with the advantage values corresponding to the state, action, next state tuples recorded during meta-training and we name this variation UMCNP-ADV. UMCNP-ADV takes in state, action, next state, and advantage tuples and predicts the mean and standard deviation of the next state and the advantage for the state-action target query. At meta-test time, the ORACLE agent is allowed to collect 25 actual rollouts from the unseen test task. Conversely, NORML agent, UMCNP and UMCNP-ADV are allowed to use a single actual rollout for fine-tuning the meta-policy. For a reliable evaluation, we use the same actual rollout data for UMCNP and UMCNP-ADV models that is used for the single rollout NORML fine-tuning.

Figure 4.3 illustrates the trajectories followed by the adapted policies, where the trajectories of UMCNP and ORACLE agents are similar. Figure 4.4 shows the post-update cumulative reward obtained in the test tasks for ORACLE, UMCNP, NORML, and UMCNP-ADV with blue, orange, green, and red respectively. The results in Table 4.1 show the mean with 95% confidence intervals and median post-update cumulative rewards collected in the test tasks. Adaptation using 24 generated rollouts from the UMCNP model achieves -1.02 ± 0.09, similar to the ground truth ORACLE with (25 actual rollouts) -1.20 ± 0.12, whereas NORML underperforms with

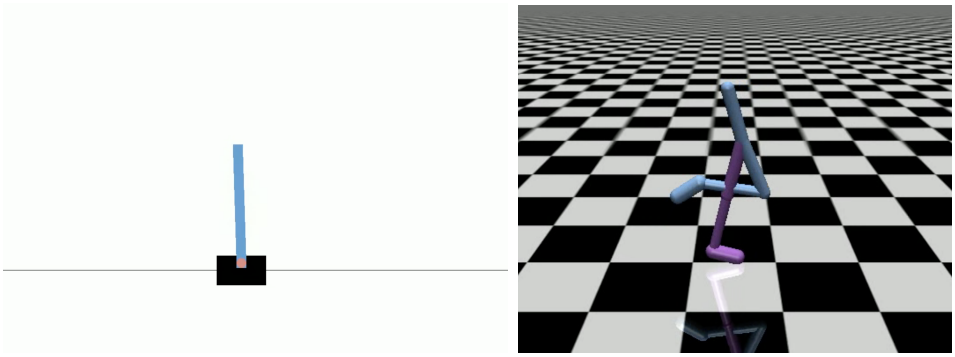


Figure 4.5. Cartpole environment with sensor bias and Walker-2D agent with random dynamics parameters (© 2024 IEEE [1]).

-11.49 ± 1.44. Although UMCNP-ADV performs better than NORML, we observe that UMCNP is significantly better than UMCNP-ADV. Future work will explore architectural improvements, such as separate decoders, to enhance UMCNP-ADV's performance.

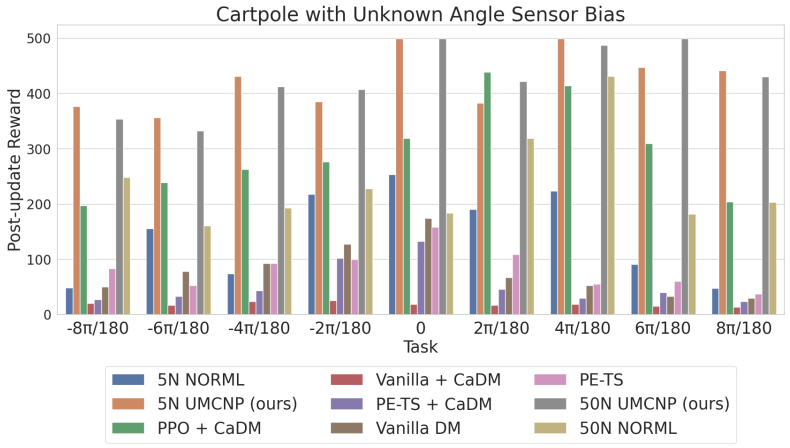
4.4.2. Cartpole with Unknown Angle Sensor Bias

We evaluate our method on the Cartpole with Angle Sensor Bias environment (Figure 4.5) [13], an extension of the Cartpole environment simulated in MuJoCo, Gym [26, 27]. In this experiment, tasks differ by a hidden position sensor drift parameter in the range $[-8°, 8°]$. In meta-training, tasks are sampled from a uniform distribution where the hidden variable parameterizes the dynamics function. The state space consists of the position/angle and velocity/angular velocity of the cart and the pole. To evaluate the adaptation performance where a limited number of transition tuples are available when the maximum episode length of the environment is 500, we compared the performance of UMCNP to NORML with only 5 and 50 transitions, PE-TS CaDM [49], Vanilla CaDM [49], PPO-CaDM [49], trained with history-length and future transition length of 5 and Random Shooting with 1000 candidates for planning following the Cartpole implementation details in [49], PE-TS [133], Vanilla DM [49] by using the code available in [134]. Importantly, CaDM requires access to the reward function during training for Model Predictive Control [135] and we have provided the reward function

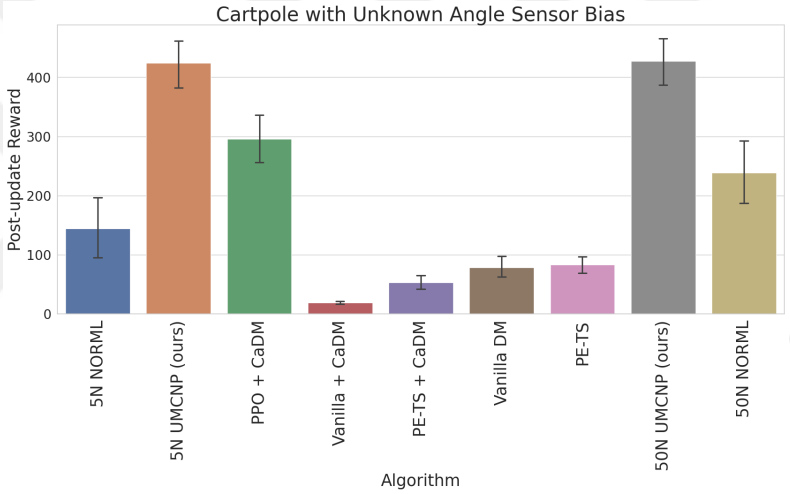
Table 4.2. Post-update cumulative Reward of UMCNP and the baselines (© 2024 IEEE [1]).

Cartpole Agent	**Mean**	**Median**
5N NORML	144.56 ± 48.72	45
5N UMCNP (ours)	424.13± 39.79	**499**
PPO + CaDM	295.40 ± 39.07	280
Vanilla + CaDM	18.62 ± 2.32	16
PE-TS + CaDM	52.82 ± 12.36	35
Vanilla DM	78.22 ± 16.87	57
PE-TS	82.87 ± 14.15	69
50N UMCNP (ours)	**427.02** ± 42.76	**499**
50N NORML	238.51 ± 55.59	174
Walker-2D Agent	**Mean**	**Median**
ORACLE	1755.07 ± 370.89	1303.97
UMCNP (ours)	**1844.62** ± 338.80	**1602.97**
NORML	1608.97 ± 316.83	1135.64
10N UMCNP (ours)	1684.56 ± 355.71	1072.68
10N NORML	1566.38 ± 308.84	1203.82
PPO + (Vanilla + CaDM)	171.18 ± 57.00	199.05
Vanilla DM + CaDM	22.31 ± 17.79	-7.28
PE-TS + CaDM	-1.20 ± 1.86	-2.47
Vanilla DM	-4.34 ± 0.37	-4.47
PE-TS	-0.48 ± 1.45	-2.42

definitions during training. Consistent with other experiments, results in Figure 4.6 and Table 4.2 indicate that UMCNP outperforms the baselines and CaDM performs better when integrated in model-free RL. Figure 4.6 shows (a) post-update cumulative Reward (y-axis) in test tasks compared with baselines and (b) overall performance with 95% Confidence Intervals. 5N UMCNP (ours), 5N NORML and 50N UMCNP (ours), 50N NORML uses 5 and 50 test task transitions respectively. X-axis represents the test task parameter.



(a)



(b)

Figure 4.6. (a) Post-update cumulative Reward (y-axis) in test tasks compared with baselines. (b) Overall performance compared with baselines with 95% Confidence Intervals. 5N UMCNP (ours), 5N NORML and 50N UMCNP (ours), 50N NORML uses 5 and 50 test task transitions respectively (© 2024 IEEE [1]).

4.4.3. Walker-2D with Random Dynamics Parameters

In order to examine the sample efficiency of UMCNP in complex, higher dimensional control tasks, we evaluate our method in the Walker-2D Agent Environment with random dynamics parameters [37] with a maximum episode length of 1000 in an unsupervised meta-testing setup where the agent receives no reward signal from the test task. Different locomotion tasks are created by unknown body mass, body inertia,

damping on different joints, and friction parameters. The scaling parameters for body mass, body inertia, and friction parameters are sampled from a range determined by raising 1.5 to powers uniformly distributed between -3 and 3, in the interval $[1.5^{-3}, 1.5^{3}]$ [37]. Likewise, the scaling parameters for damping across different joints follow the same methodology but use a base of 1.3 covering the range $[1.3^{-3}, 1.3^{3}]$ [37].

Walker environments with slimmer distributions, for instance only parameterizing the body mass or reducing the interval of the uniform distributions does not show a significant increase in performance for NORML compared to domain randomization. These scenarios do not require sophisticated adaptation methods; hence, we use more established meta-RL benchmarks [37, 13].

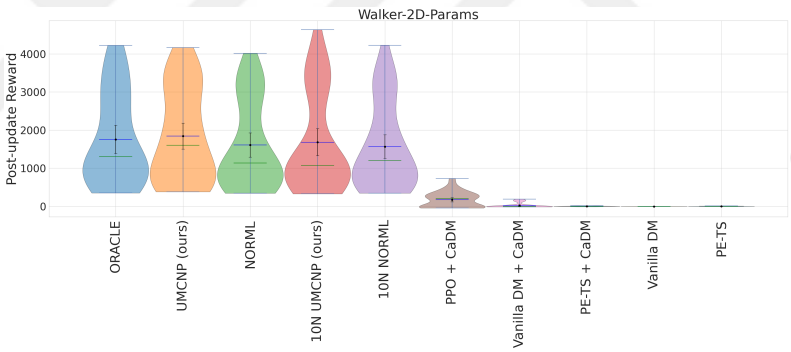


Figure 4.7. Violin plots of post-update cumulative reward from unseen 10 meta-test tasks (© 2024 IEEE [1]).

We use only 40 tasks for meta-training sampled from a uniform distribution and compare our method with the baselines on 10 test tasks similar to the meta-testing experiments with rewards [46]. Figure 4.7 show the post-update cumulative reward in meta-testing, where NORML, UMCNP use a single rollout, 10N NORML, 10N UMCNP use 10 transitions, and PE-TS CaDM [49], Vanilla CaDM [49], PPO-CaDM [49] with normalization and Cross Entropy Method [136] for planning following [49], use a history and future transition length of 10, and the ORACLE uses 25 rollouts for adaptation. We report these results with 95% Confidence Intervals (black). These plots combine

vertically aligned kernel density plots that represent the data distribution, with markers clearly indicating the minimum, maximum, mean (blue), and median (green) values.

Results in Table 4.2 indicate that UMCNP, not only improved sample efficiency but has the potential to improve meta-test adaptation performance for a larger range of unseen tasks. One reason for that can be the bias-variance trade-off in noisy real samples and generated samples. UMCNP network is trained on the meta-train dataset, hence it can generate less noisy and potentially more robust samples that are seen during training for the meta-policy which can make the adaptation more robust.

4.4.4. Implementation Details

Our model features a 128-dimensional representation and two hidden layers, each with 128 units, in both the encoder and the decoder. It is trained with a batch size of 32 with 50 input tuples and 50 target query tuples for 500000 iterations using Adam optimizer [131] with a learning rate of 0.0001. Additionally, the model utilizes the last 10% of trajectories from D^{train} which are gathered by the meta-policy throughout the meta-training phase.

4.5. Conclusion

We propose Unsupervised Meta-Testing with Conditional Neural Processes, a novel meta-RL with unsupervised meta-testing method that can generate useful rollouts for sample-efficient meta-policy adaptation. The agent can adapt without requiring rewards during meta-testing and learns without explicit task parameters, only using the interaction data, during both meta-training and meta-testing. Our results indicate that the meta-test performance of our method, when fine-tuned with generated data, shows significantly better performance in point and CartPole environments, as well as improved performance in Walker environments. Crucially, its performance is similar to the ground-truth method, which is allowed to interact with the environment significantly more. This sample efficiency across all experiments is achieved through the generation

of samples from fewer number of transition tuples.

In conclusion, we demonstrate that leveraging generated data for meta-updates not only improves sample efficiency but also enhances post-update performance. Additionally, the successful application of CNPs to high-dimensional inputs, such as image data [103], suggests promising avenues for future work. Specifically, applying our findings to high-dimensional sensorimotor spaces, such as manipulator robots equipped with RGB-D cameras, is an exciting direction for further exploration.

5. BIDIRECTIONAL PROGRESSIVE NEURAL NETWORKS WITH EPISODIC RETURN PROGRESS FOR EMERGENT TASK SEQUENCING AND ROBOTIC SKILL TRANSFER

5.1. Introduction

Human brain and behavior provide a rich venue that can inspire novel control and learning methods for robotics. In an attempt to exemplify such a development, drawing inspiration from how humans acquire knowledge and transfer skills among tasks, we introduce a novel multi-task reinforcement learning framework named Episodic Return Progress with Bidirectional Progressive Neural Networks (ERP-BPNN).

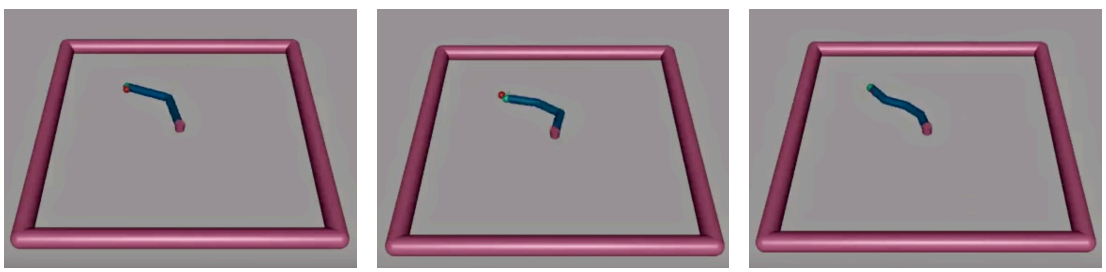
Developing robots capable of autonomous and continual learning effectively requires the exploitation of acquired knowledge without human intervention, which could be described as the main goal of lifelong robot learning [137, 138, 139]. A key feature of human learning is autonomous task switching and interleaved learning, which is not addressed in mainstream machine learning and robotics research [140]. Traditional deep RL methods often do not include mechanisms to exploit the potential benefits of interleaved learning and bidirectional skill transfer based on partial learning. In contrast to prior works, our proposed model integrates human-like interleaved learning, leverages intrinsic motivation for autonomous task-switching, and includes a novel bidirectional progressive architecture tailored for deep multitask RL. Thus, we aim to fill this gap by developing a framework that can sustain, and importantly, benefit from interleaved task learning while allowing bidirectional skill transfer among tasks.

In humans, it is shown that interleaved learning yields improved recall of information and better memory retention in the long run rather than blocked learning [141, 142, 143]. Supporting this behavioral data, the human brain is endowed with

mechanisms against task interference and forgetting, such as internal rehearsal of experiences and memory consolidation [144, 145]. To enable interleaved learning, the question of when and which task to engage during multi-task learning must be answered. Developmental learning offers some inspiration: during learning, an infant autonomously decides what to do or play without external directions dictating what task (s)he needs to engage in. This behavior of infants is usually associated with the notion of Intrinsic Motivation (IM), which guides behavior through a putative internal reward system [146]. IM can be based on curiosity, novelty, learning progress (LP), or even challenge; as such, it is also used in robotics as LP [147], curiosity [148], and in machine learning as novelty [149], or surprise [150]. In the current study, we adopt an IM approach and propose a novel learning progress (LP) signal for RL tasks that guides task switching. Overall, inspired by the above discussion, we aim to develop a multi-task reinforcement learning (RL) framework with autonomous task switching, which can benefit from interleaved task learning without suffering from task interference. We argue that such a learning system may benefit a wide range of robot learning scenarios ranging from human-like learning for a social robot to skill transfer applications among morphologically different robots.

One of the key challenges in lifelong learning is catastrophic interference/forgetting, which needs to be considered if a robot is to learn continually. When novel instances to be learned diverge greatly from previously observed ones, new information may overwrite the already acquired knowledge by modifying the representations shared among multiple tasks, leading to catastrophic forgetting. To minimize the interference while learning a novel task, in the literature, several techniques have been proposed, such as restricting the probable update(s) on the network parameters, dynamic resource allocation, or rehearsing the old task samples while learning the new ones [137]. In this study, similar to the Progressive Neural Networks (PNN) [51], we utilize task-specific resources while learning to prevent possible task interference but also allow bidirectional inter-task connectivity to support positive skill transfer.

In sum, to enable human-like interleaved multi-task learning while avoiding task interference, we develop a multi-task reinforcement learning system composed of (1) a novel architecture called BPNN that improves the PNN [51, 151] and (2) a novel Intrinsic Motivation signal, Episodic Return Progress (ERP), for task-switching. Unlike the PNN architecture, which restricts skill transfer to the forward direction and requires learning previous tasks until convergence before transferring to the next task, our BPNN method enables bidirectional skill transfer during training. This means that skill transfer can happen in midway along multi-task learning among all tasks without the necessity of one task waiting for another to finish. The ERP signal evaluates task progress based on episodic return values, detecting the task that significantly contributes to enhancing overall performance across multiple tasks. The efficacy of the proposed multi-task learning framework is shown by its application to the learning of reaching skill by morphologically different robots, namely two degrees of freedom (2-DoF), 3-DoF, and 4-DoF manipulators (Figure 5.1). The conducted systemic experiments show that synergistic multi-task learning is possible due to bidirectional inter-task skill transfer provided by the proposed BPNN architecture and the ERP-based task switching.



(a) (b) (c)

Figure 5.1. (a) 2-DoF, (b) 3-DoF, (c) 4-DoF Reacher Robot Arm Environments.

5.2. Method

We propose a novel framework that integrates a bidirectional progressive neural network (BPNN) and soft task-switching mechanism crafted for RL, inspired by [140]. The BPNN architecture consists of bidirectional lateral connections among the hidden

layers of each fully connected task network to allow skill transfer. By allocating a separate network module for each task, the model avoids negative task transfer at the core level but allows potential positive transfer to take place due to the bidirectional lateral connections among networks.

A high-level overview of the ERP-BPNN architecture in Figure 5.2 (a), (b) demonstrates ERP selecting Task 1 (a) then Task 2 (b), showcasing bidirectional flow for skill transfer in a many-to-many fashion among three tasks. Graphical representation of ERP-BPNN framework with ERP task switching in Figure 5.2 (c) illustrates the gradient flow when Task 1 is selected to learn. Weight updates are denoted by red (Task 1) arrows. Dashed gray arrows indicate no gradient flow during the learning update. Tasks 1, 2, and 3 refer to RL tasks for 2-Dof, 3-DoF, and 4-DoF Reacher Robot arms as illustrated in Figure 5.1 (a), (b), (c) respectively.

5.2.1. Bidirectional Progressive Neural Networks

We initialize a fully connected neural network module $m \in M$ with task parameters θ_m for each task $\mathcal{T}_i$ with index i. As each task corresponds to a module, we will use "network module" and "task" terms interchangeably. Since each robot controls different numbers of joints, each network module has task-specific output layers with a different number of neurons $n^m_{\{l=L\}}$ where $l \in \{1..L\}$ refers to the layer index.

During training, each network module receives the input of the task selected for training. In this way, the lateral activations can receive representations of other tasks for skill transfer. We compute the hidden activations $h_l^{(m)} \in \mathbb{R}^{n_l^m}$

$$h_l^{(m)} = f\left(W_l^{(m)} h_{(l-1)}^{(m)} + b_l^{(m)} + \sum_{t \neq m} U_l^{(m:t)} h_{l-1}^{(t)} + b_l^{(m:t)}\right), \tag{5.1}$$

where $U_l^{(m:t)}$, $b_l^{(m:t)}$ denote the weight matrix and the bias vector corresponding to the lateral connections of the ordered pair of modules (m, t) from the previous layer $h_{l-1}^{(t)}$ of the module t to the l^{th} layer of module m. $W_l^{(m)} \in \mathbb{R}^{n_l^m \times n_{l-1}^m}$ and $b_l^{(m)}$ are the weight matrix and bias vector, respectively, for layer l of module m, and f is the element-wise

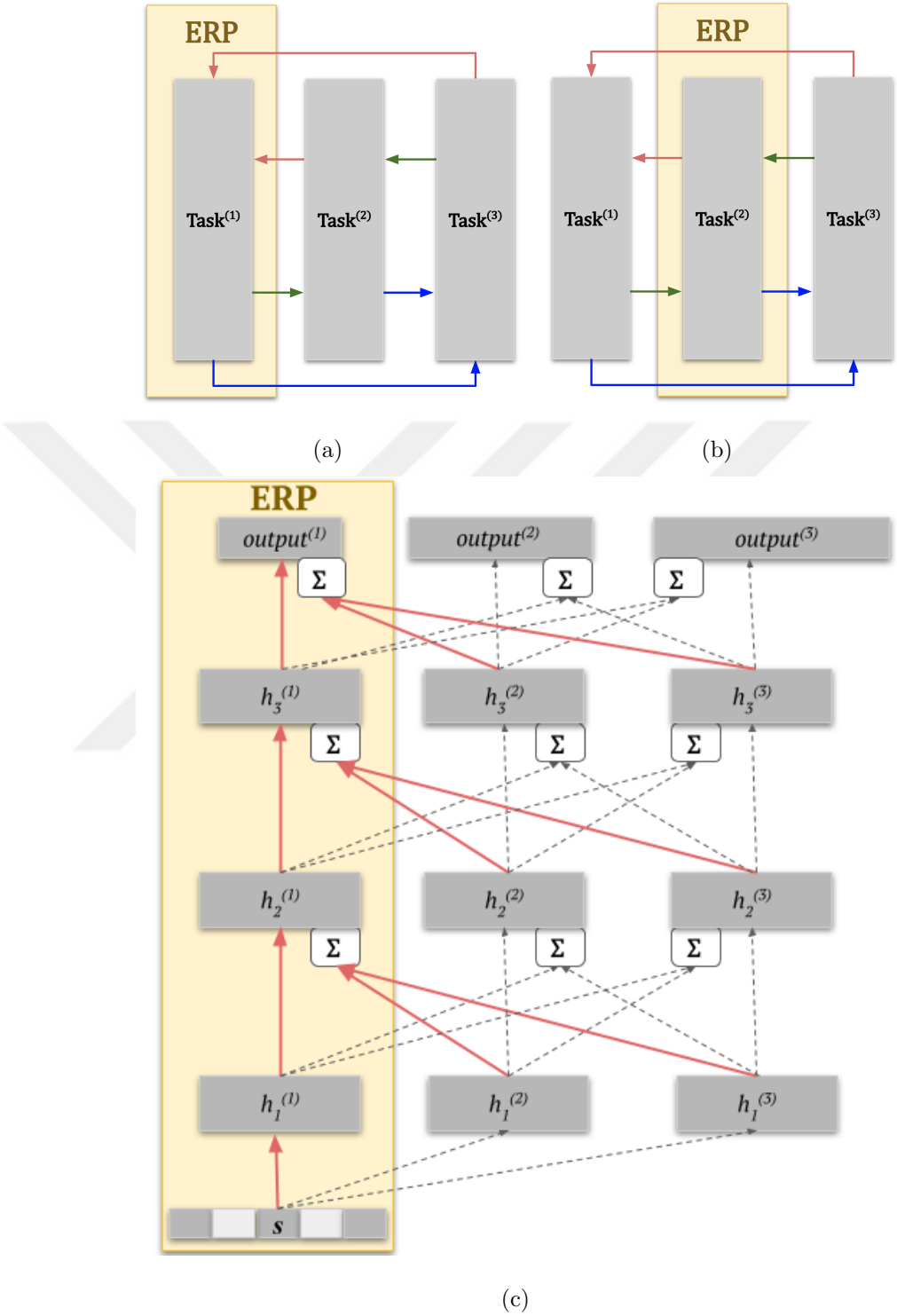


Figure 5.2. A high-level (a,b) ERP-BPNN architecture demonstrates ERP selecting Task 1 (a) then Task 2 (b), showcasing bidirectional flow for skill transfer in a many-to-many fashion among three tasks. Graphical representation of ERP-BPNN framework with ERP task switching, wherein (c) Task 1 is selected to learn.

activation function.

In the BPNN architecture, linear layers are utilized to transmit information from other tasks, represented by $\boldsymbol{\Sigma}$ in Figure 5.2(c), functioning as a module for summation. The linear transformations applied to the previous layers of other tasks allow the tuning of the incoming lateral signals. In particular, a negative transfer can be suppressed, and a positive transfer can be enhanced by the tuning of lateral weights through gradient descent-based learning. In this work, we use the activation function $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ following the suggested hyperparameters for Proximal Policy Optimization (PPO) [4] for continuous control. In the initial phase, the lateral connections of each module are frozen, and all tasks are individually trained to jumpstart task-specific learning for a predetermined number of K_{init} training iterations. In the experiments reported in Section 4, we set $K_{init} = 20$. In the subsequent learning steps, the parameters of the task modules that are not selected for training are frozen to avoid negative transfer between tasks. This prevents the gradient flow into the networks associated with the remaining tasks. On the other hand, the parameters of and the lateral connections incoming to the task module selected for training are unfrozen to facilitate task-specific learning as well as the skill transfer from other tasks.

At a high level, this training mechanism and architecture enables bidirectional information transfer by allowing gradient flow through a subset of neural network parameters associated with the task and using information from other tasks. How the dynamic task selection takes place is described next.

5.2.2. Task Switching by a Novel Intrinsic Motivation Signal: Episodic Return Progress

We propose a task-switching mechanism for multi-task reinforcement learning based on a novel IM signal, namely average Episodic Return Progress (ERP), that captures the learning progress of an RL agent. At each optimization iteration k, for each task network module m, we record the expected discounted cumulative reward,

denoted by $G_m(k)$. To compute $G_m(k)$, we first normalize the immediate rewards to ensure the exponential moving average of the rewards has a constant variance and clip them between (-10,10) to stabilize training [152]. Then we compute the mean of the returns over $\boldsymbol{P} * \zeta$ trajectories where $\boldsymbol{P} = 8$ is the number of parallel RL environments, and $\zeta = 2$ is the number of episodes completed in each RL environment. Then, we take ERP for task m at step k, $ERP_m(k)$, as the slope of the line that is fitted to $G_m(k-w+1), G_m(k-w+2), \ldots, G_m(k)$ using least squares, where w is a predetermined window size ($w = 5$ in the experiments reported in this chapter). The least squares solution is

$$ERP_m(k) = \frac{w\left(\sum_{i=0}^{w-1} X_k[i]Y_m(k)[i]\right) - \sum_{i=0}^{w-1} X_k[i] \sum_{i=0}^{w-1} Y_m(k)[i]}{w\left(\sum_{i=0}^{w-1} \left(X_k[i]\right)^2\right) - \left(\sum_{i=0}^{w-1} X_k[i]\right)^2}, \tag{5.2}$$

where $X_k = [k-w+1, k-w+2, ..., k]$ and $Y_m(k) = [G_m(k-w+1), G_m(k-w+2), ..., G_m(k)]$.

For bootstrapping ERP computation for each task, at the beginning of a multi-task learning session, each task is given an initial ERP bootstrapping run of $K_{init} > w$ iterations. Then, we compute the ERP for each module individually for the initial run and subsequently at each iteration to select the task that has made the highest recent progress. The objective of the ERP procedure is to select tasks dynamically by identifying the task with the steepest recent increase in return, so that the selected task can continue its rapid progress.

Selecting the task with the highest progress for training is particularly important as it can facilitate skill transfer to the other tasks, utilizing the bidirectional lateral connections of BPNN. For instance, the tasks that have made less progress can benefit from the steeper improvement of the selected task. Since ERP is computed for all tasks at each iteration, other tasks can benefit from the top-performing task and increase their chances of getting selected. The window size should be chosen to strike a balance between reducing noise and tracking recent updates. A large window size, w, can lead to selecting the initially top-performing task for an extended number of iterations. Correspondingly, a small window size can introduce noise during training. Thus, to

monitor the recent changes in the progress of all tasks, we empirically select the window size as five based on a grid search. The consecutive selection of the same task will eventually reach a plateau during training, either because the task is learned or a local minimum is encountered, where learning in other tasks may help the plateaued task to jumpstart, as the reduction in progress of the current task allows other tasks with more progress to be selected. This dynamic task selection regime can be considered analogous to the flow state theory [153], which strives to maintain a balance between challenging and effortless tasks.

5.2.3. Multi-task Reinforcement Learning

We initialize two separate BPNN architectures for critic and policy networks to integrate our method into the actor-critic reinforcement learning framework. In this manner, the policy network receives the state for the corresponding task as input and outputs the mean of a diagonal multivariate Gaussian distribution with a learned log standard deviation parameter independent of the state. Correspondingly, the critic network receives the state and learns the value function. Hence, only the parameters of the training task's actor-critic modules and their corresponding lateral connections are updated using Adam optimizer [131] during task learning. We use the tuned hyperparameters in [154] for the reacher task as there are multiple extensions of PPO [155]. This extended PPO loss $L_{PPO}^{\mathcal{T}_i}$ with a clipped value function loss and a value function entropy bonus can be formulated [4] as

$$L_{PPO}^{\mathcal{T}_i} = \mathbb{E}_t[L_t^{CLIP_{\mathcal{T}_i}}(\theta) - c_1 L_t^{VF_{\mathcal{T}_i}}(\theta) + c_2 S^{\mathcal{T}_i}(\theta(\pi)(\boldsymbol{s}_t)], \tag{5.3}$$

where $L_t^{CLIP_{\mathcal{T}_i}}(\theta)$ is the clipped PPO surrogate objective, $c_1 L^{VF_{\mathcal{T}_i}}(\theta)$ is the clipped value loss with coefficient c_1, $c_2 S^{\mathcal{T}_i}(\theta(\pi))(\boldsymbol{s}_t)$ is the entropy bonus for the actor network with coefficient c_2. Although the actor and critic neural networks are separate, we can denote them collectively as θ as in previous works [4] for brevity. For instance, $\theta(\pi)$ and $\theta(\phi)$ refer to the actor-network and critic-networks parameters, respectively.

Algorithm 1 ERP-BPNN Warm-up

Require: $\mathcal{T}_i$: Task with index i, $\mathcal{T}$: set of tasks, $m_{\mathcal{T}_i}$: network module of $\mathcal{T}_i$

Constants: $\zeta = 2$ (#episodes), $P = 8$ (#parallel tasks), $K_{init} = 20$ (#jumpstart training iterations)
for all $\mathcal{T}_i$ **do**
  Unfreeze module $m_{\mathcal{T}_i}$
  for $k \in \{1, ..., K_{init}\}$ **do**
    for $p \in \{1, ..., P\}$ **in parallel do**
      Sample ζ episodes by running policy $\pi_{\theta^{old}_{\mathcal{T}_i}}$
      $\forall t$, Compute Advantage estimates $\hat{A}_t$
    end for
    Record average episodic return $G_{m_{\mathcal{T}_i}}(k)$
    Optimize $L^{\mathcal{T}_i}_{PPO}$ w.r.t. $\theta_{\mathcal{T}_i}$ via Equation (5.3)
    $\theta^{old}_{\mathcal{T}_i} \leftarrow \theta_{\mathcal{T}_i}$
  end for
  Freeze module $m_{\mathcal{T}_i}$
end for

5.2.4. Evaluation Metrics

Designing a reward function is crucial and challenging in deep reinforcement learning, primarily because the reward function may not fully capture all attributes expected from a learning agent. One example is the reacher environment, which can require tuning of the reward function coefficients or introducing additional parameters. However, this requires significant time and resources to adhere to the predetermined metrics and careful tuning of the parameters. In this sense, evaluating how a straightforward, uninformed reward function performs in terms of metrics meaningful for the task domain at hand is important. Therefore, in this section, in addition to the classical RL metric of episodic return, we also present domain-specific metrics used to evaluate the collective performance of morphologically different agents. To account for early

Algorithm 2 ERP-BPNN

Require: $\mathcal{T}_i$: Task with index i, $\mathcal{T}$: set of tasks, $m_{\mathcal{T}_i}$: network module of $\mathcal{T}_i$

Constants: $\zeta = 2$ (#episodes), $P = 8$ (#parallel tasks), $K_{init} = 20$ (#jumpstart training iterations)
while not done **do**
  Calculate progress for all tasks using ERP
  Choose task $\mathcal{T}_i$ with maximum ERP
  Unfreeze module $m_{\mathcal{T}_i}$
  for $p \in \{1, ..., P\}$ **in parallel do**
    Sample ζ episodes by running policy $\pi_{\theta^{old}_{\mathcal{T}_i}}$
    $\forall t$, Compute Advantage estimates $\hat{A}_t$
  end for
  Record average episodic return $G_{m_{\mathcal{T}_i}}(k)$
  Optimize $L^{\mathcal{T}_i}_{PPO}$ w.r.t. $\theta_{\mathcal{T}_i}$ via Equation (5.3)
  $\theta^{old}_{\mathcal{T}_i} \leftarrow \theta_{\mathcal{T}_i}$
  Freeze module $m_{\mathcal{T}_i}$
  for all $\mathcal{T}_j \in T - \{\mathcal{T}_i\}$ **do**
    for $p \in \{1, ..., P\}$ **in parallel do**
      Sample ζ episodes by running policy $\pi_{\theta^{old}_{\mathcal{T}_j}}$
    end for
    Record average episodic return $G_{m_{\mathcal{T}_j}}(k)$
  end for
end while

stopping in RL, which is introduced as a regularization technique in [124], we evaluate all tasks after each iteration, save the best policies obtained up to that iteration, and use them in our evaluations. Notably, saving the best policy based on the cumulative performance of all tasks for all methods ensures an accurate comparison of the proposed method against baselines and enhances performance across all methods after training is completed.

Episodic Return. Maximum episodic return tracks the best expected discounted cumulative reward achieved across all tasks in an iteration. We report the best expected discounted cumulative reward after each training iteration to ensure a fair comparison with baselines. Crucially, the ERP task-switching procedure provides an inherent early-stopping procedure, provided that we allow a fixed number of cumulative training iterations under resource constraints. Since the task is to reach a given point in space with minimum effort, we follow the reward function definition in [156]

$$R = -\|p_{\text{ee}} - p_{\text{target}}\|_2 - \|a_t\|_2^2, \tag{5.4}$$

where a is the action, $\|a_t\|_2^2$ is the control cost, p_{ee}, and p_{target} are the positions of the end-effector (fingertip) and the target, respectively.

Distance To the Goal. Distance to the goal is the minimum expected ℓ_2-norm distance between the final end-effector position and the goal position obtained over all tasks. The reacher environment only terminates after 50 timesteps, which is equal to the episode length. Hence, we expect the end-effector of the reacher to stay in the immediate vicinity of the goal until episode termination.

Deviation from Shortest Path to the Goal. At a high level, we expect an efficient learning agent to follow the shortest path to the goal, which is also desirable for many robotic applications. Thus, we introduce a straightness metric that measures the expected deviation from the shortest path to the goal taken by the manipulator's end-effector over all tasks. We define this deviation metric, across tasks as

$$\mathbb{E}_{\mathcal{T}}\left[\left(\sum_{t=1}^{H}\left\|p_{\text{ee}}^{\mathcal{T}}(t) - p_{\text{ee}}^{\mathcal{T}}(t-1)\right\|_2\right) - \left\|g - p_{\text{ee}}^{\mathcal{T}}(0)\right\|_2\right], \tag{5.5}$$

where $\mathcal{T}$ is a task, $p_{\text{ee}}^{\mathcal{T}}(t)$ is the position of the end-effector for task $\mathcal{T}$ at timestep t, g is the goal, and $p_{\text{ee}}^{\mathcal{T}}(0)$ is the initial end-effector position. Here, we first compute the length of the spatial trajectory by summing the ℓ_2-norm of the distance between the current and the previous end-effector position. Subsequently, we subtract the ℓ_2-norm of the distance between the goal position and the initial end-effector position to obtain

the deviation from the shortest path to the goal for task $\mathcal{T}$. After each iteration, we calculate the path length traveled by all agents for every task and record the smallest expected deviation from the shortest path to the goal.

5.2.5. Implementation Details

Given the morphological diversity among robots, which results in different state space dimensions, and our BPNN architecture having a fixed input layer, we have standardized the state input dimensions for robots with 2-DoF and 3-DoF to match the state space dimension of the 4-DoF robot. Hence, we apply zero-padding to the dimensions corresponding to the angular velocity and the angle of each missing link. To encourage skill transfer among networks, we limit the available computational resources to the task network modules by setting the number of hidden layers to three and the hidden layer size to two.

For the reacher tasks, we use PPO [4] as the RL algorithm and adopt the hyperparameter set from [154] for the remaining hyperparameters. After each training iteration, we sample trajectories from each task, excluding the task trained most recently, using the latest BPNN actor and critic parameters. Then, we compute the ERP, maximum episodic return, minimum expected distance to the goal, and minimum expected deviation from the shortest path to the goal for each task. Additionally, we run eight parallel environments per task to reduce simulation time. Each environment runs two episodes, with each episode lasting 50 steps, thereby accumulating a total of 800 timesteps per task.

5.3. Experiments

In this section, we first present the environment used for benchmarking multi-task learning for morphologically different robots, illustrated in Figure 5.1. Then, we elaborate on ERP-based task switching and present results obtained with the evaluation metrics introduced in Section 5.2.4.

5.3.1. Single Goal Multi-Task Learning Between Morphologically Different Robots

To demonstrate skill transfer between morphologically different robots, we modified the Reacher-v2 environment, simulated in MuJoCo [157] using Gymnasium [156]. The learning experiments have been conducted on a computer equipped with an NVIDIA 3090 GPU and 10-core Intel i9 CPU at 3.7GHz. The experiment setup consists of three different reacher environments, each having distinct action spaces, specifically featuring 2-DoF, 3-DoF, and 4-DoF robot arms. Although each environment has unique dynamics due to its distinct morphologies, it shares the same reward functions. The reward function defined in Equation (5.4) comprises a control cost, the negative vector norm of the end-effector of the reacher, and a predetermined goal. We consider the baselines RANDOM-BPNN and random Multi-Layer Perceptron (RANDOM-MLP) to evaluate the performance of our ERP-BPNN approach across various metrics. RANDOM-BPNN abides by random task selection while featuring the same underlying BPNN architecture as ERP-BPNN. Similarly, RANDOM-MLP follows the same random task selection procedure as RANDOM-BPNN but utilizes a separate actor-critic network pair for each task with no lateral connections among different tasks. Accordingly, RANDOM-MLP can be regarded as training a PPO algorithm for each task separately.

5.3.2. Results

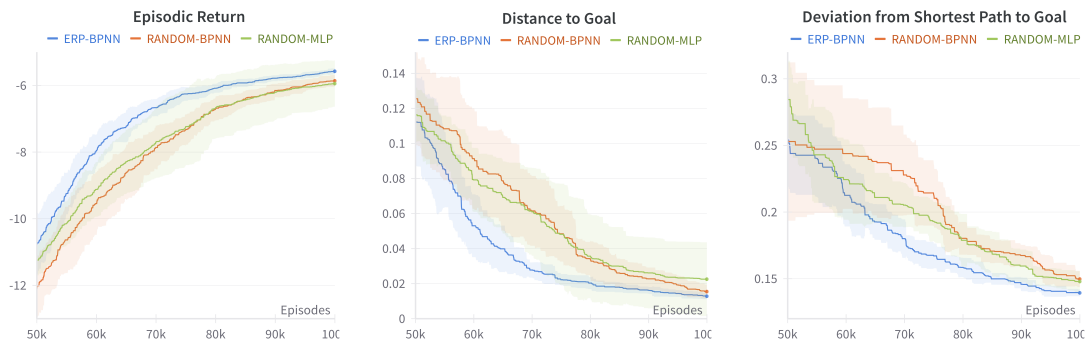
In this section, we report the results of our method, ERP-BPNN, compared to the baselines of RANDOM-BPNN and RANDOM-MLP, based on the evaluation metrics presented in Section 5.2.4. Table 5.1 shows each method's mean and standard deviation results obtained during 100K episodes across five random seeds for each metric. The results indicate that ERP-BPNN has superior performance compared to the baselines across all metrics while having the lowest standard deviation. More importantly, as seen from the Episodic Return plot in Figure 5.3 (a), the proposed model, ERP-BPNN, achieves faster convergence than the baselines.

Since the combination of BPNN architecture with ERP-based task selection (ERP-BPNN) yields the best results, surpassing the random task selection strategy (RANDOM-BPNN), it can be argued that the ERP task selection procedure is essential for successful multi-task learning with positive transfer among tasks. Notably, the random task selection strategy (RANDOM-BPNN), although inferior to ERP-BPNN, performs better than the RANDOM-MLP baseline in terms of Episodic Return and yields better endpoint accuracy measured as Distance to Goal at the end of the training (Table 5.1). This indicates the lateral connections among task networks may facilitate limited positive skill transfer even with random task selection. However, this picture is not reflected in the Deviation from the Shortest Path to the Goal measure as random task selection leads to negative transfer for the Deviation from the Shortest Path to the Goal measure (see Figure 5.3 (c), episodes $>$ 55K). In the early stages of learning, the performance of ERP-BPNN and RANDOM-BPNN are better than RANDOM-MLP (episodes $< 55K$); however, while the proposed ERP-BPNN enjoys positive skill transfer and thus continues to improve its progress, RANDOM-BPNN suffers from negative interference degrading to a performance worse than RANDOM-MLP in this metric (Figure 5.3 (c), episodes $>$ 55K).

In order to get an intuitive understanding of the learned policy performances, we illustrate typical trajectories followed by the end-effector of the reacher robot in Figure 5.4 (a) when controlled by policies acquired by the 1750^{th} learning iteration, corresponding to approximately 80K episodes. Observe that at this training stage, the proposed ERP-BPNN has already acquired the skill to reach the target accurately through a less curved path compared to the baselines. In particular, the RANDOM-BPNN diverges from the shortest path to the goal for the 4-DoF robot arm, whereas the RANDOM-MLP diverges from the shortest path to the goal for both 3-DoF and 4-DoF Reacher Robot arms.

Table 5.1. Episodic Return, Distance to Goal, and Deviation from Shortest Path to Goal results across 5 random seeds using ERP-BPNN, RANDOM-BPNN, and RANDOM-MLP.

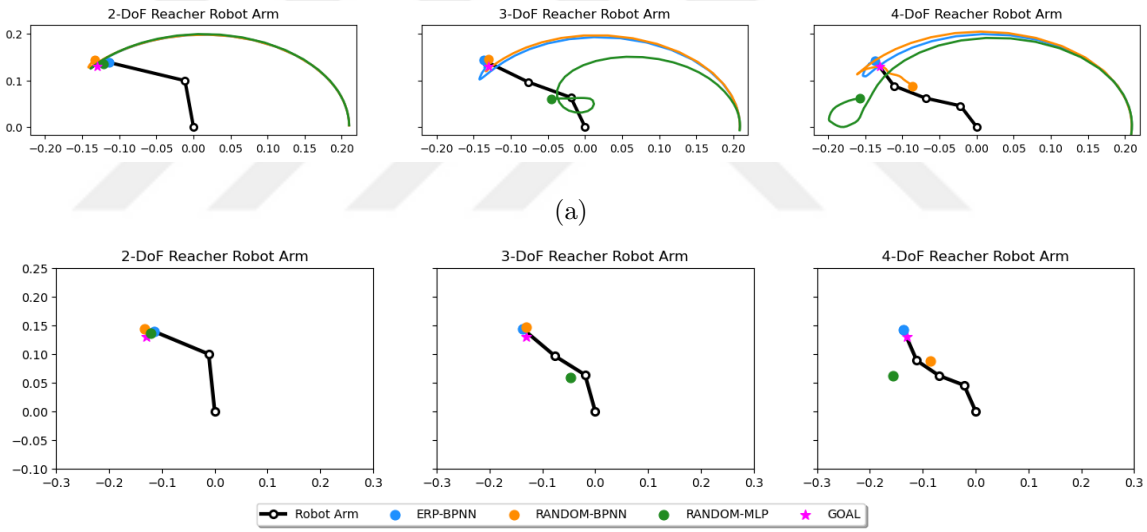
Episodic Return	**ERP-BPNN**	**RANDOM-BPNN**	**RANDOM-MLP**
60,000 episodes	$\mathbf{-7.95 \pm 0.48}$	-9.55 ± 0.94	-9.12 ± 0.82
75,000 episodes	$\mathbf{-6.26 \pm 0.20}$	-7.32 ± 0.45	-7.24 ± 1.15
80,000 episodes	$\mathbf{-6.09 \pm 0.13}$	-6.72 ± 0.34	-6.69 ± 0.91
100,000 episodes	$\mathbf{-5.58 \pm 0.08}$	-5.86 ± 0.17	-5.95 ± 0.69
Distance to Goal	**ERP-BPNN**	**RANDOM-BPNN**	**RANDOM-MLP**
60,000 episodes	$\mathbf{5.3 \pm 1.79}$	9.13 ± 2.86	7.92 ± 1.82
75,000 episodes	$\mathbf{2.25 \pm 0.33}$	4.85 ± 1.58	4.82 ± 2.89
80,000 episodes	$\mathbf{2.03 \pm 0.34}$	3.42 ± 0.97	3.57 ± 2.39
100,000 episodes	$\mathbf{1.29 \pm 0.18}$	1.55 ± 0.4	2.25 ± 2.1
Deviation from Shortest Path	**ERP-BPNN**	**RANDOM-BPNN**	**RANDOM-MLP**
60,000 episodes	$\mathbf{21.25 \pm 2.33}$	24.39 ± 4.15	22.44 ± 3.01
75,000 episodes	$\mathbf{16.71 \pm 0.93}$	21.29 ± 2.32	19.3 ± 1.48
80,000 episodes	$\mathbf{15.85 \pm 0.73}$	18.03 ± 1.23	17.88 ± 1.97
100,000 episodes	$\mathbf{13.94 \pm 0.27}$	14.97 ± 0.6	14.79 ± 0.8



(a) (b) (c)

Figure 5.3. Performances of the proposed model, ERP-BPNN, and the two baselines of RANDOM-BPNN and RANDOM-MLP across five random seeds are shown in terms of (a) maximum episodic return, (b) minimum expected final end-effector distance to goal, and (c) minimum expected deviation from the shortest path to the goal.

Figure 5.4 (b) shows the reaching ability, a typical goal position and the final end-effector positions reached for each robot, obtained on morphologically different robots by the proposed model and the baselines. Final end-effector positions reached using each method indicate that the agent trained using ERP-BPNN is consistently closest to the goal across all environments compared to the baselines. Likewise, the results for Deviation from Shortest Path to Goal in Figure 5.4 (a) show that the agent trained with ERP-BPNN can reach the goal by taking a shorter path than the baselines in the 4-DoF reacher robot arm environment. Importantly, accurate reaching using a straighter path can be learned significantly faster by our proposed method ERP-BPNN compared to the baselines as evidenced by the distance plots given in Figure 5.3 (b).



(a)

(b)

Figure 5.4. Typical end-effector paths (a) and final end-effector positions (b) with the policies learned by ERP-BPNN (ours), and the baselines of RANDOM-BPNN and RANDOM-MLP (at 1750^{th} policy update) are demonstrated.

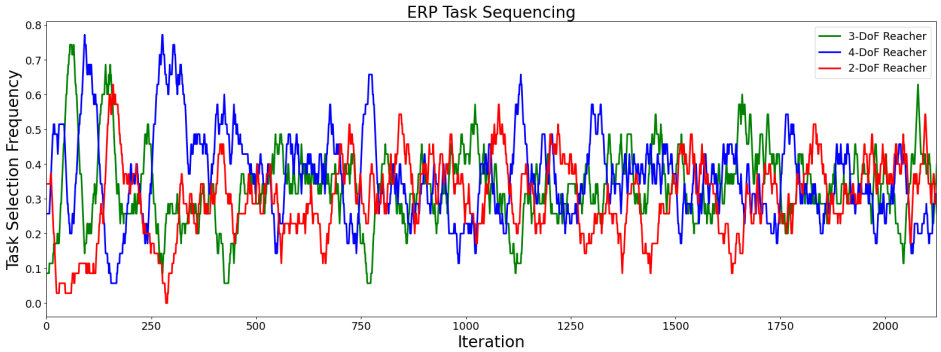


Figure 5.5. Episodic return progress based task sequencing with an iteration window size of $\nu = 35$ is illustrated.

5.3.3. ERP-based Dynamic Task Selection

To examine the task engagement patterns that emerge with the ERP-based task selection mechanism, we record the selection frequency of each task over a moving learning window, i.e., a fixed number of learning iterations. By using this scheme, the selection frequencies of the tasks are plotted against training iteration counts in Figure 5.5. As can be seen, although ERP primarily selects the more straightforward 2-DoF Reacher Robot Arm as the one to learn initially (Figure 5.5, time range: 0-10), it gradually starts selecting more complex tasks with 3-DoF and 4-DoF Reacher Robots (Figure 5.5 time range: 10-150). Then, we observe that ERP continues selecting the 4-DoF Reacher Robot for skill transfer more frequently (Figure 5.5, time range: 250-500). This task selection behavior suggests that the 4-DoF Reacher Robot has benefited from the skill transfer of the other tasks and started to learn more rapidly later in the training.

Upon further elaboration of the episodic return plots for all tasks in Figure 5.3 (a) and standard deviations across iterations in Table 5.1, ERP-BPNN exhibits a consistent overall improvement across training iterations, achieving a faster convergence with a lower standard deviation compared to the baselines. In line with this, faster convergence of ERP-BPNN indicates that consecutive selections of the 4-DoF Reacher task have

collectively improved performance.

5.4. Conclusion

This chapter introduces Episodic Return Progress with Bidirectional Progressive Neural Networks (ERP-BPNN), a novel multi-task reinforcement learning approach incorporating a human-like interleaved learning mechanism, and shows its application to skill transfer across morphologically different robots. ERP-BPNN comprises a Bidirectional Progressive Neural Network (BPNN) that enables efficient, bidirectional skill transfer and an Episodic Return Progress (ERP) mechanism for dynamic task selection. The BPNN architecture is specifically designed to enable skill transfer through bidirectional lateral connections, drawing inspiration from the human brain's adeptness at retaining existing knowledge while acquiring new skills. ERP-based task selection complements BPNN by autonomously selecting the tasks to engage in learning during training, eliminating the need for prior domain knowledge to evaluate difficulty levels of tasks. We demonstrate that ERP-based task selection, with BPNN, achieves higher performance and faster convergence than the baselines across all the tested metrics of episodic return, distance to the goal, and deviation from shortest path to goal.

The insights presented in this work are closely related to the field of human brain-inspired reinforcement learning research. By implementing an intrinsic motivation signal and leveraging curiosity-driven exploration within the context of multi-task RL, ERP-BPNN framework closely aligns with biological processes observed in the human brain. Specifically, the behavior of dopaminergic neurons, known for their critical role in reward-based learning, mirrors the temporal difference (TD) prediction errors utilized in RL algorithms [158, 159, 106]. BPNN provides a mechanism for a shared multi-task RL architecture that prevents task interference during skill transfer by adjusting knowledge transfer from other tasks during training. In addition, the ERP-guided freezing of lateral connections among task modules prevents interference in the previously acquired knowledge. Furthermore, the bidirectional lateral connections augment noise in the task-specific computational flow akin to noisy computations in the brain [8], alleviating

catastrophic forgetting [160] while amplifying plasticity [161]. Analogous to the modular control architecture in cerebellum [162], as corroborated by Functional Magnetic Resonance Imaging studies [163, 164], BPNN adopts a modular architecture. This architecture choice is instrumental in decreasing catastrophic forgetting via instantiating a module for each task.

6. DIFFUSION POLICIES FOR OUT-OF-DISTRIBUTION GENERALIZATION IN OFFLINE REINFORCEMENT LEARNING

6.1. Introduction

An agent must make robust decisions when it encounters states that are out-of-distribution with respect to the state distribution of the training dataset. In this chapter, we develop a generalizable generative offline reinforcement learning model, State Reconstruction for Diffusion Policies (SRDP), to learn robust skills from offline datasets by extrapolating sequential decision-making to OOD states. Our model can be categorized as a policy-regularized offline RL algorithm where the divergence from the behavior policy that collected the dataset is discouraged.

There is a growing interest in applying diffusion models in robotics. Notably, Chi et al. [165] have demonstrated the effectiveness of convolutional and transformer-based diffusion networks in representing multimodal action distributions with visuomotor policy learning for behavior cloning. Likewise, other behavior cloning [166, 167] methods have also learned multimodal expert behavior, though not specifically targeting offline RL and diffusion models. Despite these achievements, our current focus is on OOD generalization in offline RL without a visual component while believing in the potential applicability of our findings in visuomotor policy learning. Diffusion Q-Learning (DQL) [29] and Diffuser [86] are RL algorithms that utilize diffusion models [80, 127, 89] by guiding the diffusion process toward regions that can yield a high reward. Diffuser [86] use diffusion models in the planning procedure to generate trajectories from the diffusion model, while DQL [29] generates actions based on a state-conditional diffusion model and uses Q-function maximization with behavior cloning loss. Even though DQL [29] can represent multimodal actions, it is often unstable in OOD state regions.

Autoencoders can reconstruct a subset of OOD samples with low error by learning representations in the bottleneck layer [168]. Denouden et al. [168] highlight that OOD samples close to a linear or nonlinear latent dimension manifold of the training data can have a low reconstruction loss. Hence, guidance from reconstruction loss and similar architectural designs can benefit OOD generalization. Mutlu et al. [169] use reconstruction loss to provide hints to the generator. Although we employ separate output heads to integrate the state reconstruction loss, we use a shared representation layer to generalize to OOD states close to the latent dimension manifold.

Our key contributions include SRDP, a novel offline RL method that alleviates the distribution shift incurred by OOD states using representation learning. SRDP is tailored to guide diffusion policies using a state reconstruction signal. Through extensive experiments, ablation studies, as well as real robot experiments, we demonstrate that incorporating state reconstruction signal at each diffusion timestep not only enhances the performance of foundational diffusion models in severe OOD settings where a large portion of data is missing during training, but also in widely used offline RL tasks.

6.2. Preliminaries

To address the OOD states in Offline RL, we propose incorporating an auxiliary state reconstruction loss in diffusion policies. We first describe the details of Diffusion Q-Learning (DQL) [29] and its implementation. Subsequently, we derive a robust diffusion policy that leverages representation learning. Finally, we discuss our approach using a 2D multimodal contextual bandit environment.

6.2.1. Diffusion Q-Learning

DQL uses a conditional diffusion model to generate actions conditioned on states [29]. The reverse diffusion chain conditioned on state s is defined by $\pi_\theta\left(\boldsymbol{a}_{0:N} \mid \boldsymbol{s}\right)$ where N denotes the final diffusion timestep. The action obtained by the reverse diffusion process is then used in Q-learning and policy learning in an iterative procedure. Initially,

a minibatch of MDP tuples $\{(s, a, r, s')\}$ are sampled from the offline RL dataset. The next action a' is generated by the target diffusion policy $\pi_{\theta'}$ conditioned on the next state s'. Using double Q-learning trick by [170] and Bellman operator minimization by [59, 171], DQL minimizes

$$\mathbb{E}_{\substack{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')\sim\mathcal{D} \\ \boldsymbol{a}'_0\sim\pi_{\theta'}}}\left[\left\|\left(r\left(\boldsymbol{s},\boldsymbol{a}\right)+\gamma\min_{i=1,2}Q_{\phi'_i}\left(\boldsymbol{s}',\boldsymbol{a}'_0\right)\right)-Q_{\phi_i}\left(\boldsymbol{s},\boldsymbol{a}\right)\right\|^2\right], \tag{6.1}$$

where $\mathcal{D}$ is the offline RL dataset, $Q_{\phi'_i}$ are the target critic networks and Q_{ϕ_i} are the critic networks. Diffusion policies are optimized by policy improvement using Q-function and behavior cloning loss minimization. We update the critic networks similarly to evaluate the improvement from the state reconstruction loss.

6.3. Method

6.3.1. State Reconstruction for Diffusion Policies

Diffusion policies learn to generate actions with the guidance of states. Since state information is available in the training and evaluation phase, the diffusion policy can be conditioned on the state to generate actions. The diffusion model learns to predict the noise ϵ added to the input at each diffusion timestep n following the simplified loss in DDPM. Diffusion policies extend this idea to the RL framework by concatenating the noisy input (noisy action) vector and the embedded diffusion timestep with the state vector during training [29]. In contrast, using an auxiliary head, SRDP learns to extract the state representation in a shared representation layer. In brief, SRDP operates as follows. The shared SRDP feature extraction module f_ϕ maps the time embedding, state, and noisy action to a latent representation $\boldsymbol{z}$. This representation is then directed into distinct task-specific heads: the diffusion head f_θ predicts the added noise at that diffusion timestep for the given noisy action and state, and the auxiliary head f_ψ reconstructs the input state. Importantly, the state representation signal is deeply embedded for all diffusion timesteps sampled during training through the state reconstruction loss.

We use a shared fully connected module f_ϕ to map the noisy action $\sqrt{\bar{\alpha}_n}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}$, time-embedding and the state $\boldsymbol{s}$ into a shared representation

$$\boldsymbol{z} = f_\phi\left(\sqrt{\bar{\alpha}_n}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, \boldsymbol{s}, n\right). \tag{6.2}$$

The diffusion head $\boldsymbol{f_\theta}$ uses this representation to predict the noise added at timestep n, while state head $\boldsymbol{f_\psi}$ learns to reconstruct the state. Thus, diffusion policy loss $\boldsymbol{\mathcal{L}_{DP}}$ is defined by

$$\mathbb{E}_{\substack{n\sim U(\{1,\ldots,N\}) \\ \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}}\left[\left\|\boldsymbol{\epsilon} - f_\theta\left(f_\phi\left(\sqrt{\bar{\alpha}_n}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, \boldsymbol{s}, n\right)\right)\right\|^2\right], \tag{6.3}$$

where diffusion timesteps are sampled from the uniform distribution U over the set $\{1,\ldots,N\}$.

State reconstruction guidance is also propagated through the network during policy learning. The state reconstruction loss $\mathcal{L}_\Re$ minimizes

$$\mathbb{E}_{\substack{t\sim U(\{1,\ldots,N\}) \\ \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}}\left[\left\|\boldsymbol{s} - f_\psi\left(f_\phi\left(\sqrt{\bar{\alpha}_n}\boldsymbol{a} + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, \boldsymbol{s}, n\right)\right)\right\|^2\right], \tag{6.4}$$

where f_ψ takes the shared representation as input to predict the state. Behavior cloning loss $\boldsymbol{\mathcal{L}_{BC}}$ in SRDP, jointly learns the contextual representation of the state and the noise added to the action by minimizing

$$\boldsymbol{\mathcal{L}_{BC}} = \boldsymbol{\mathcal{L}_{DP}} + \lambda\mathcal{L}_\Re, \tag{6.5}$$

where λ is a hyperparameter that controls the weight of the state reconstruction loss. A key point to note is that the state reconstruction loss in this step is propagated through the network for each randomly selected diffusion timestep.

At each training iteration, we first sample a mini-batch of transitions $\{(s, a, r, s')\}$ from the offline RL dataset. Then, we sample a mini-batch of next actions $\{(a'_0)\}$ from the target diffusion policy using $\{(s')\}$ from the offline RL dataset[29]. The mini-batch of transitions and the sampled $\{(a'_0)\}$ are used to update the critic network using Equation (6.1) following the implementation in [29, 172]. After the critic update, we sample $\{(a_0)\}$ from our policy π_θ through an iterative denoising procedure. To guide the action generation procedure to high reward regions, we subtract the scaled [23] expectation of Q-function, $\frac{\eta\mathbb{E}_{\boldsymbol{s}\sim\mathcal{D},\boldsymbol{a}_0\sim\pi_\theta}\left[Q_\phi(\boldsymbol{s},\boldsymbol{a}_0)\right]}{\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\left|Q_\phi(\boldsymbol{s},\boldsymbol{a})\right|\right]}$, from SRDP behavior cloning loss

$\mathcal{L}_{\boldsymbol{BC}}$ in Equation (6.5) to obtain $\mathcal{L}_{\boldsymbol{SRDP}}$. Crucially at this minimization step, the state reconstruction loss is propagated through the network for each diffusion timestep, promoting learning more descriptive features from the OOD-state samples. Then, we update the target policy network and the target critic networks.

In the evaluation phase, we sample actions from our policy through an iterative denoising procedure. First, we sample $a_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ at diffusion timestep N. Then, we concatenate this sampled action a_N with our current state from the environment and the embedding of the diffusion timestep to obtain the input vector for our diffusion policy. Subsequently, our diffusion model predicts the noise $\epsilon_\theta = f_\theta\left(f_\phi\left(\boldsymbol{a}_n, \boldsymbol{s}, n\right)\right)$ added to the action at that diffusion timestep given the state while the reconstructed states from f_ψ are not used. To obtain the action a_0 generated by our model, we run the reverse diffusion chain by computing $\boldsymbol{a_{n-1}}|\boldsymbol{a_n} = \frac{1}{\sqrt{\alpha_n}}\left(\boldsymbol{a}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}} f_\theta\left(f_\phi\left(\boldsymbol{a}_n, \boldsymbol{s}, t\right)\right)\right) + \sqrt{\beta_n}\epsilon$. As suggested in [89] to enhance the sampling quality, we sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $n = N, \ldots, 2$ and assign $\epsilon = 0$ for $n = 1$.

6.3.2. Out-of-Distribution States in Offline RL

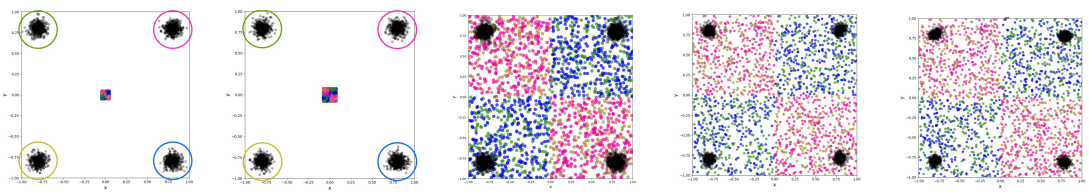
We aim to learn a robust policy from offline RL datasets consisting of expert and novice demonstrations with different modalities. Therefore, we designed an environment named 2D multimodal contextual bandit to evaluate the agent's performance in OOD states. Previous work [29] used a 2D-bandit environment to illustrate the advantage of using a highly expressive model to represent policies. Here, we extend this environment to a contextual bandit setting to assess the trained policy in OOD states. The primary objective in this task is to learn multimodal expert behavior, particularly in challenging OOD states. Thus, reward guidance from Q-function is not used in this context to isolate the regularization induced by SRDP behavior cloning loss.

We consider a two-dimensional continuous state and action space where the states and actions are characterized as real-valued x- and y-coordinates. To illustrate multimodal expert behaviors, the states in each pair of quadrants map to actions

generated from a mixture of two Gaussian distributions. Given a state (x,y), the agent needs to generate an action $[a_1(x,y), a_2(x,y)]$ that has a high probability in the matching mixture of two Gaussians. Subsequently, we construct a training dataset by sampling states from two uniform distributions and actions from two Gaussian mixture models (GMM), where each Gaussian mixture comprises two Gaussian distributions. For a given state s, an action a is sampled from a GMM with two Gaussians as $a \sim 0.5 \cdot \mathcal{N}(\boldsymbol{\mu}_1(s), \boldsymbol{\Sigma}) + 0.5 \cdot \mathcal{N}(\boldsymbol{\mu}_2(s), \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma}$ is the diagonal covariance matrix with $\sigma = [0.05, 0.05]$, $\boldsymbol{\mu}_1$, and $\boldsymbol{\mu}_2$ are the mean vectors of Gaussian distributions. We use the following procedure to determine the values of the mean vectors $\boldsymbol{\mu}_1(s)$, and $\boldsymbol{\mu}_2(s)$

$$\mu_1(s), \mu_2(s) = \begin{cases} (-0.8, -0.8), (0.8, 0.8) & \text{if } s \in [-1.0, 0.0] \times [0.0, 1.0] \\ (-0.8, -0.8), (0.8, 0.8) & \text{if } s \in [0.0, 1.0] \times [-1.0, 0.0] \\ (-0.8, 0.8), (0.8, -0.8) & \text{if } s \in [-1.0, 0.0] \times [-1.0, 0.0] \\ (-0.8, 0.8), (0.8, -0.8) & \text{if } s \in [0.0, 1.0] \times [0.0, 1.0] \end{cases}, \tag{6.6}$$

where a state (s) is sampled from the uniform distributions ($s_{train} \sim \mathcal{U}([-0.05, 0.05] \times [-0.05, 0.05])$ and $s_{train} \sim \mathcal{U}([-0.08, 0.08] \times [-0.08, 0.08])$) when constructing the training data. During testing, we sample states from subregions within the Euclidean plane larger than those in the training dataset. In particular, we sample states from the uniform distribution $\mathcal{U}([-1.0, 1.0] \times [-1.0, 1.0])$ to evaluate the policies in OOD states.



(a) (b) (c) (d) (e)

Figure 6.1. The training dataset (s_{train}) in (a) is constructed with $\mathcal{U}([-0.05, 0.05] \times [-0.05, 0.05])$ and used for experiments in Figure 6.2 while s_{train} in (b) uses $\mathcal{U}([-0.08, 0.08] \times [-0.08, 0.08])$ with results in Figure 6.3 and Figure 6.4. State samples for ground truth is shown in (c). Results for training with in-distribution data are illustrated for BC-Diffusion (d) and SRDP(ours) (e) ([2]).

6.4. Experiments

This section empirically evaluates our proposed approach, i.e. state reconstruction guidance on the 2D-Multimodal Contextual Bandit environment, multimodal real-robot setup, sparse reward continuous control maze navigation dataset with missing data generated in [30] for offline RL pretraining, and D4RL benchmarks from [21].

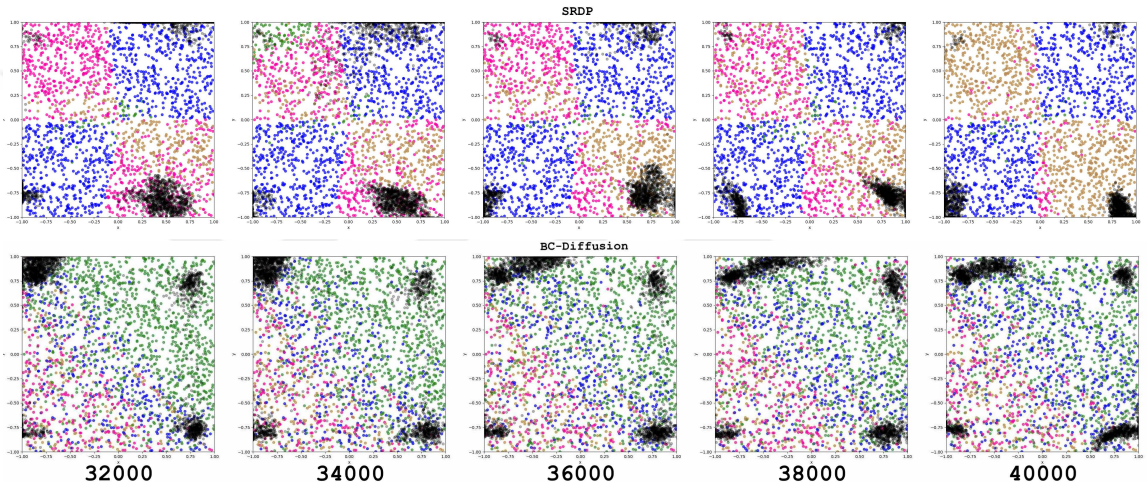


Figure 6.2. Top to bottom: SRDP ($\lambda = 1.25$) and BC-Diffusion [29]. Dataset for training is constructed from Figure 6.1 (a) ($s_{train} \sim \mathcal{U}([-0.05, 0.05] \times [-0.05, 0.05])$) and ground truth is illustrated in Figure 6.1 (c). The number of training iterations in each column increases incrementally from left to right by 2000, starting from 32000, as the convergence for the more restricted dataset takes longer (© 2024 IEEE [2]).

6.4.1. 2D-Multimodal Contextual Bandit Experiments

The training dataset in 2D-Multimodal Contextual Bandit $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{k}$, consists of $k = 10000$ state-action tuples with horizon $H = 1$. To examine the policy in OOD states, the states generated for training are sampled from the uniform distribution $s_{train} \sim \mathcal{U}([-0.05, 0.05] \times [-0.05, 0.05])$ in Figure 6.1 (a), and $s_{train} \sim \mathcal{U}([-0.08, 0.08] \times [-0.08, 0.08])$ in Figure 6.1(b). The states used for the evaluation are

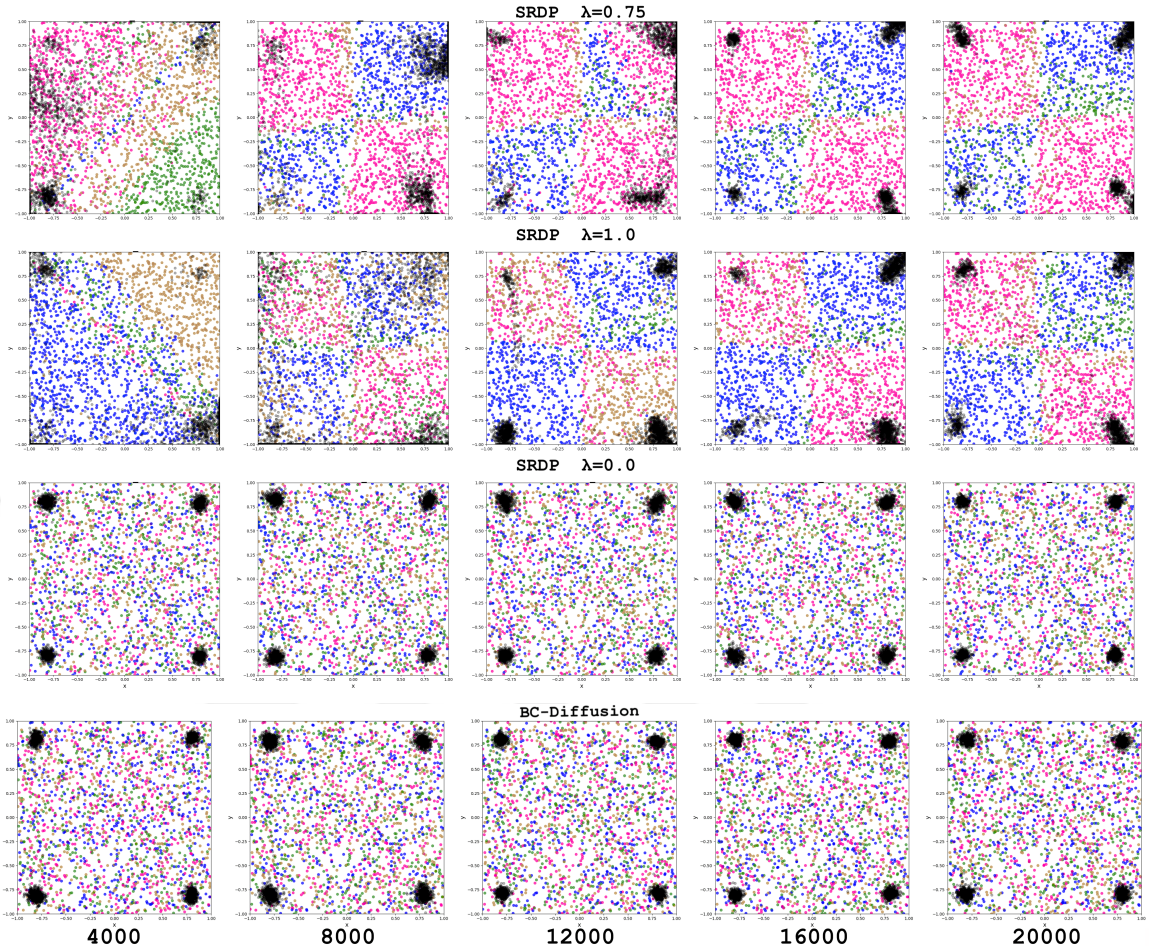


Figure 6.3. Top to bottom rows show the results from SRDP (proposed model) with the scaling parameter $\lambda = 0.75$, $\lambda = 1.0$, $\lambda = 0.0$ (meaning no state reconstruction loss) and BC-Diffusion [29]. The training dataset is constructed from $s_{train} \sim \mathcal{U}([-0.08, 0.08] \times [-0.08, 0.08])$. The number of training iterations increases incrementally by 4000 from left to right (© 2024 IEEE [2]).

sampled from the uniform distribution of $s_{test} \sim \mathcal{U}([-1, 1] \times [-1, 1])$. Ground truth for OOD state generalization corresponding to the training datasets in Figure 6.1 (a) and (b) is illustrated in Figure 6.1 (c) where the black points show the actions in the dataset following Equation (6.6). Similarly, the black points in Figure 6.2 (a) and (b) show the actions generated by the SRDP and BC-Diffusion policies, respectively. If a state generates an action in the first, second, third, and fourth quadrants, the state is colored magenta, green, light brown, and blue, respectively. Chamfer distance between the ground truth contextual action distribution and the actions generated using the

in-distribution dataset (Figure 6.1) for BC-Diffusion is 0.047, and SRDP(ours) is 0.037.

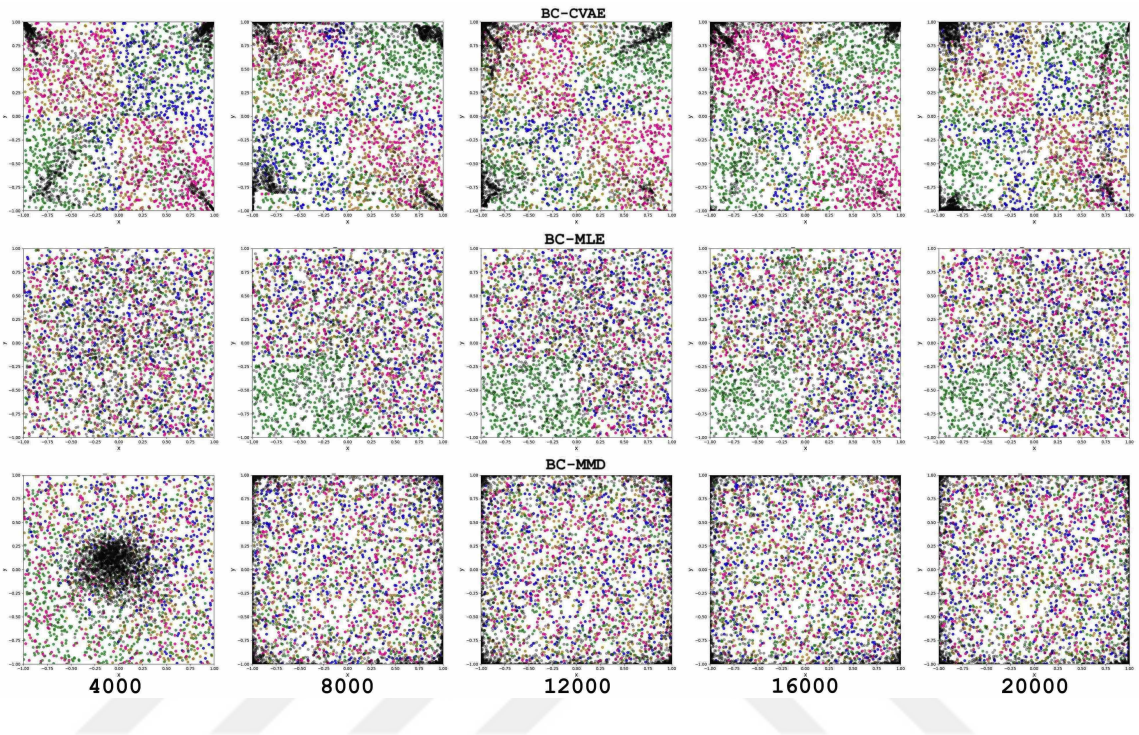


Figure 6.4. BC-CVAE represents the policy by a conditional variational autoencoder (CVAE) [59], BC-MLE[23] uses a Multivariate Gaussian to model the policy with a diagonal covariance matrix, BC-MMD [67] uses a CVAE that learns the behavior policy to constrain a Gaussian policy. The training dataset is constructed from $s_{train} \sim \mathcal{U}([-0.08, 0.08] \times [-0.08, 0.08])$ (© 2024 IEEE [2]).

Visually distinguishable quadrants with respective coloring in Figure 6.2 (a) (SRDP) indicate that the reverse diffusion process can generate actions accurately for OOD states. In contrast, Figure 6.2 (b) shows that BC-Diffusion memorizes actions without using state information. Subsequently, results in Figure 6.2 show that the representation learning with state reconstruction loss promotes finding expert skills in multimodal data, achieving faster convergence. Although a larger portion of the state space is not included in the training dataset in Figure 6.2, SRDP can generalize well to OOD states and learn at least one of the two expert behaviors. Conversely, BC-Diffusion learns the action distribution of the dataset, yet it cannot assign the correct action distribution for the given state. Accurate partitioning of the state space,

in terms of the actions taken, is particularly important in real-world robotics tasks where naive memorization of the action distribution in the data without state dependence can have severe consequences. Although SRDP can represent a single mode in this challenging OOD task where only 0.25% of the state space is present in the data, it can correctly learn an expert policy. All hidden layers have the same size of 16 for SRDP and BC-Diffusion.

In the next set of experiments, the training dataset includes states ($s_{train} \sim \mathcal{U}([-0.08, 0.08] \times [-0.08, 0.08])$) and actions a_{train} generated by Equation (6.6). Figure 6.3 and Figure 6.4 show that only SRDP can represent multimodal expert distributions and partition the state space into visible quadrants at earlier iterations, improving training stability and reducing computation costs. Consistent with the experiments in Figure 6.2, BC-Diffusion only learns the action distributions in the dataset, omitting the state information in OOD states.

As an ablation study, we provide illustrations with different scaling parameters that control the weight of state reconstruction loss in Figure 6.3. We use a dual head architecture with hidden layer sizes $(16^{shared}, 16^{shared}, 16)$ for SRDP and three hidden layers with size 16 for BC-Diffusion. Notice that SRDP with scaling parameter $\lambda = 0$ performs similarly to the BC-Diffusion baseline in Figure 6.3 as this reduces SRDP to BC-Diffusion. We compute Chamfer distances between the ground truth actions and the actions generated by SRDP with varying scaling parameters (λ) and BC-Diffusion for each group of states. Table 6.1, shows the sum of Chamfer distances over each group of states across five random seeds, at 4000 intervals. Chamfer distance is particularly suitable for our case because it measures the distance between two sets of points. BC-Diffusion is prone to overfitting since the chamfer distance increases as the training progresses. The results show that SRDP performs significantly better and converges faster than BC-Diffusion with appropriate scaling parameter selection.

Table 6.1. Ablation Study for SRDP Scaling Parameters over training iterations at 4000 intervals with Chamfer Distance (© 2024 IEEE [2]).

Method	4000	8000	12000	16000	20000
$SRDP(\lambda = 0.0)$	1.62 ± 0.31	1.51 ± 0.06	1.63 ± 0.26	1.61 ± 0.34	1.63 ± 0.32
$SRDP(\lambda = 0.25)$	1.49 ± 0.25	1.03 ± 0.59	0.91 ± 0.62	0.82 ± 0.64	0.81 ± 0.67
$SRDP(\lambda = 0.5)$	$\mathbf{1.1} \pm 0.3$	1.07 ± 0.55	1.01 ± 0.72	0.91 ± 0.68	0.79 ± 0.55
$SRDP(\lambda = 0.75)$	1.62 ± 0.24	$\mathbf{0.88} \pm 0.51$	0.83 ± 0.43	0.54 ± 0.28	$\mathbf{0.39} \pm 0.16$
$SRDP(\lambda = 1.0)$	1.24 ± 0.48	0.95 ± 0.35	$\mathbf{0.68} \pm 0.55$	$\mathbf{0.47} \pm 0.43$	0.44 ± 0.41
$BC\text{-}Diffusion$	1.41 ± 0.01	1.51 ± 0.16	1.54 ± 0.18	1.6 ± 0.14	1.62 ± 0.15

6.4.2. Multimodal UR10 Experiments

We design a multimodal UR10 robot experiment to illustrate the application of our method in the real world and highlight the importance of OOD generalization. Our hardware setup consists of a 6-Dof UR10 manipulator robot mounted with a Robotiq gripper illustrated in Figure 6.5. Similar to the experiments in [165] designed for a visuomotor policy learning task, we define the actions for our policy as the space positional commands of the end-effector. In the setting visualized in Figure 6.5, the robot learns to reach the vegetables or the coffee maker from a random initial state. If the gripper is opened in states in region A (red/magenta) or D (yellow/light brown) (enabling the use of the pink coffee stirrer), it should reach the coffee maker for stirring. Conversely, if the gripper is enclosed (allowing the use of the green knife) in region B (blue) or C (green), it should reach the vegetables for chopping.

The training dataset is constructed from the true space positional commands of the end-effector, $s_{train} \sim \mathcal{U}([-1.0, -0.9] \times [0.03, 0.17]$. We sample states from the uniform distribution $\mathcal{U}([-1.2, -0.7] \times [-0.25, 0.45])$ to evaluate the policies in OOD states. The action distributions are generated using the same procedure outlined in Equation (6.6), though the origin and GMM distributions are shifted for the real UR10 space. Specifically, for states with respect to the center position at $(-0.95, 0.1)$, action distributions for the right upper, left upper, left bottom, and right bottom regions

are produced with corresponding means $\mu_1(s) = (-0.8, 0.35)$, $\mu_2(s) = (-1.1, 0.35)$, $\mu_3(s) = (-1.1, -0.15)$, $\mu_4(s) = (-0.8, -0.15)$ and diagonal covariance matrix $\boldsymbol{\Sigma}$ with $\sigma = 0.015$.

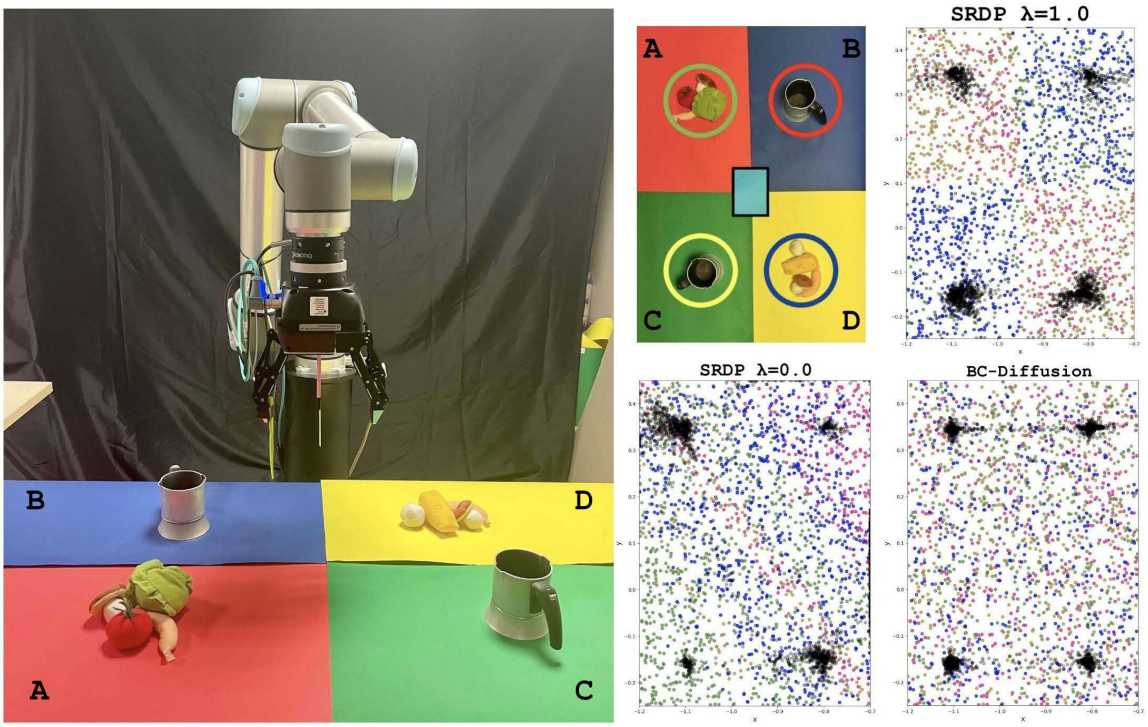


Figure 6.5. SRDP($\lambda = 1.0$) with BC-Diffusion and $SRDP(\lambda = 0.0)$. State samples in the training dataset are within the light blue region. The color-coded regions are labeled as follows: A (red/magenta), B (blue), C (green), and D (yellow/light brown) (© 2024 IEEE [2]).

We evaluate SRDP and baselines trained with 75 diffusion timesteps across five random seeds. Chamfer distance results of SRDP ($\lambda = 1.0$) compared to the baseline with respect to the ground truth contextual action distribution in the dataset is (0.071 ± 0.02). For DQL and SRDP($\lambda = 0.0$) the chamfer distances are (0.27 ± 0.01), (0.28 ± 0.02) respectively. We use a dual head architecture with hidden layer sizes $(32^{shared}, 16^{shared}, 32)$ for SRDP($\lambda = 0.0$) and SRDP($\lambda = 1.0$) and $(16, 16, 16)$ for BC-Diffusion. SRDP($\lambda = 0.0$) is BC-Diffusion with a bottleneck layer in the middle, without

a state reconstruction loss. The action generation time for SRDP across 10 trials is 8.34 ± 0.28 ms on an NVIDIA 4090 GPU. Results indicated that the architectural change did not yield better OOD generalization; hence, the state reconstruction signal is essential in learning multimodal expert behavior from OOD data.

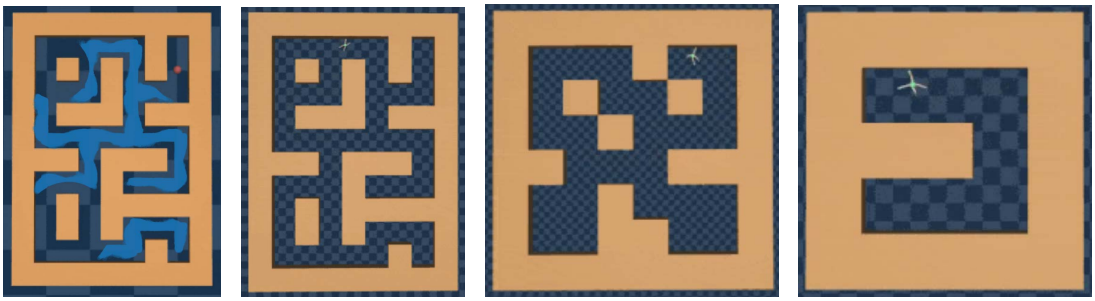
6.4.3. Missing Data Maze Environment

Maze2D-v1 navigation environments were introduced as benchmarks for offline RL in D4RL [21]. In this environment, a 2D force-actuated ball robot must navigate through a closed maze to reach a fixed goal location during evaluation. The Maze2D dataset consists of trajectories collected by a planner agent, which uses a PD controller to reach a random goal from a random initial state location. The actions are defined as the linear forces applied in the x-y directions, whereas the observations are defined as the concatenation of the position and linear velocity of the ball in the x-y direction.

Table 6.2. Comparison of SRDP with DQL baseline in Missing Data and Standard Maze2D Environment (© 2024 IEEE [2]).

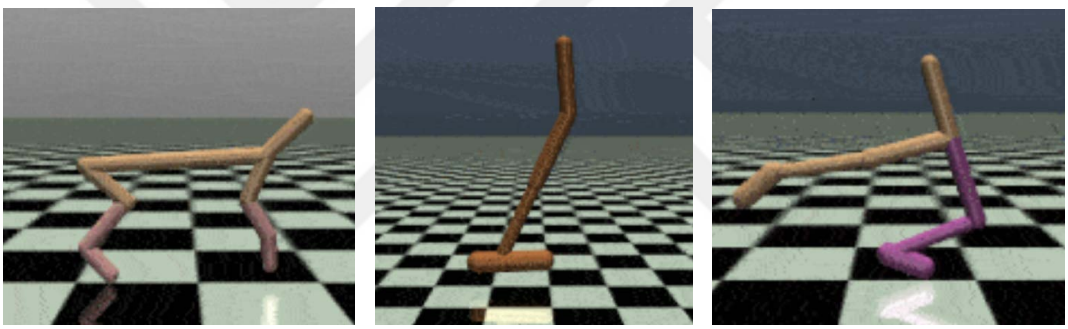
Maze2D-Missing-Data-Large	**Normalized Score**
$SRDP(\lambda = 1.0)$	$\mathbf{35.0} \pm 28.2$
$SRDP(\lambda = 0.75)$	23.9 ± 20.4
$SRDP(\lambda = 0.25)$	20.0 ± 37.9
$SRDP(\lambda = 0.0)$	1.7 ± 2.3
DQL	13.1 ± 15.3
Maze2D-Large	**Normalized Score**
$SRDP(\lambda = 0.75)$	$\mathbf{203.6} \pm 19.7$
DQL	189.1 ± 15.3

To assess the performance of SRDP in OOD states, we use the sparse reward Maze2D offline RL dataset "maze2d-missing-data-large-v1" by Mark et al. [30], illustrated in Figure 6.6 (a), where the data collected in the vicinity of three circles, one



(a) (b) (c) (d)

Figure 6.6. (a) Maze2d environment with missing data "maze2d-missing-data-large-v1" from [30]. D4RL [21, 26] AntMaze Environments with three layouts: (b) Antmaze-large-v0, (c) Antmaze-medium-v0, (d) Antmaze-umaze-v0 (© 2024 IEEE [2]).



(a) (b) (c) Walker2d

Figure 6.7. D4RL (a) Half-cheetah (b) Hopper (c) Walker2d Gym-MuJoCo environments [21, 27, 156, 26] (© 2024 IEEE [2]).

encapsulating the goal, is missing. This task is used to evaluate the performance of online and offline finetuning after offline pretraining in [30]. In our context, we focused on results without online finetuning, aligning with the more challenging experimental framework adopted throughout this study. Notably, the agent starts from a random initial state during evaluation, and a significant portion of state data is absent near the goal and random initial state locations.

An ablation study on this environment demonstrating the advantage of SRDP in Table 6.2 shows the mean and standard deviation of normalized scores obtained

Table 6.3. Comparison of SRDP with the DQL baseline in D4RL (© 2024 IEEE [2]).

AntMaze	**DQL**	**SRDP(ours)**
AntMaze-Umaze	96.0	**96.4** $\pm$ 1.5
AntMaze-Umaze-Diverse	84.0	**89.8** $\pm$ 5.1
AntMaze-Medium-Play	**79.8**	78.4 $\pm$ 8.1
AntMaze-Medium-Diverse	82.0	**90.6** $\pm$ 6.0
AntMaze-Large-Play	49.0	**63.0** $\pm$ 8.0
AntMaze-Large-Diverse	61.7	**62.6** $\pm$ 3.5
Average	75.4	**80.1**
Gym	**Diffusion-QL**	**SRDP(ours)**
Halfcheetah-Medium	51.5	**51.9** $\pm$ 1.5
Hopper-Medium	96.6	**96.8** $\pm$ 3.1
Walker2d-Medium	87.3	**88.1** $\pm$ 0.5
Halfcheetah-Medium-Replay	**48.3**	48.1 $\pm$ 0.3
Hopper-Medium-Replay	102.0	**102.1** $\pm$ 0.1
Walker2d-Medium-Replay	98.0	**98.1** $\pm$ 1.2
Halfcheetah-Medium-Expert	97.2	**97.5** $\pm$ 0.1
Hopper-Medium-Expert	112.3	**112.6** $\pm$ 0.4
Walker2d-Medium-Expert	**111.2**	111.1 $\pm$ 0.4
Average	89.3	**89.6**

across five random seeds. More specifically, the results indicate the superiority of SRDP ($\lambda = 1.0$) compared to DQL [29] with $(256, 256, 256)$. It is important to note that SRDP with a dual head architecture with hidden layer sizes $(512^{shared}, 256^{shared}, 512)$ and scaling parameter $\lambda = 0$ becomes DQL without a state reconstruction decoder. In addition, SRDP performs superior to the baseline for "maze2d-large-v1" sparse environment in D4RL without missing data.

6.4.4. D4RL Benchmark

D4RL [21], comprising extensive robotics datasets, has been widely used as a benchmark for offline RL algorithms. In AntMaze environments (with suffix '-v0' in the benchmarks), the objective of an 8-DoF quadruped ant robot is to navigate to a 2D goal location in various mazes. Correspondingly, in Gym-MuJoCo ([27, 26]) half-cheetah, hopper, and walker2d environments (suffix '-v2'), the objective is to achieve forward locomotion. Since offline RL differs from imitation learning, each dataset in Table 6.3 includes various amounts of suboptimal data. For these experiments, we follow the details of the online implementation used in DQL, where they assume that the policies can be evaluated at fixed intervals with few online interactions. This is a form of early stopping in RL introduced in [124] where the hyperparameter, the number of training epochs, is tuned for DQL and the proposed model, SRDP. We report the average normalized scores of undiscounted returns for SRDP and DQL in Table 6.3, where 100 corresponds to expert-level behavior compared to the normalized scores for DQL from [29].

Antmaze datasets are generated following non-Markovian and suboptimal policies, sparse rewards, and multitask data design procedures [21]. Figure Figure 6.6 (b), (c), (d) shows the AntMaze environments where each maze layout, large, medium, umaze, has different levels, such as play and diverse. The performance scores across five random seeds in Table 6.3 demonstrate that the proposed SRDP model is superior for all cases except "AntMaze-Medium-Play". OOD generalization is beneficial in multitask settings; hence, SRDP can enhance performance significantly.

Gym-MuJoCo environments, comprising continuous control tasks in MuJoCo [26], have been commonly used in deep RL [21, 74, 23, 173, 174]. In D4RL, datasets are collected using the online RL interaction data of a Soft Actor-Critic (SAC) agent [173]. To evaluate an offline RL algorithm in narrow data distribution and suboptimal behavior policy settings, "medium", "medium-replay", and "medium-expert" datasets are generated. Medium datasets are generated by collecting the rollouts from a medium-

performing policy, whereas medium-replay datasets are generated by keeping a replay buffer of rollouts until the RL policy reaches a medium-level performance. Medium-expert datasets include expert trajectories by 50% in addition to medium-level rollouts. Results presented in Table 6.3 show that our proposed model, SRDP, performs better than DQL [29] across five random seeds.

6.5. Conclusion

In this work, we propose SRDP for OOD generalization in offline RL, a representation learning-based method built on top of the recent class of diffusion policies introduced by [29]. To show the multiple skills learned by the model when evaluated in OOD states, we designed a 2D Multimodal Contextual Bandit environment and applied it to a real robotic scenario. Compared to prior work, our results indicate that SRDP can represent multimodal policies, partition the state space more accurately, and converge faster in real-robot and simulation environments. In addition, results show that SRDP can learn superior models compared to previous work in Antmaze and Gym-MuJoCo [26, 27] environments in D4RL benchmarks [21] with various levels and agents. Finally, on a sparse continuous control navigation task where critical regions of the state space are completely removed from the offline RL dataset, our method performs significantly better than the standard diffusion policy-based RL.

7. FORECASTING IN OFFLINE REINFORCEMENT LEARNING FOR NON-STATIONARY ENVIRONMENTS

7.1. Introduction

Building on our work in Chapter 6, which addressed generalization to OOD states in offline RL, we now confront a key limitation of that approach: the assumption of fully observable states. Agents trained on fully observable states often fail when deployed in environments characterized by noisy or corrupted observations.

While robust offline RL methods address test-time perturbations, such as sensor noise or adversarial attacks [31], a critical gap persists in addressing non-stationarity within the observation function, a challenge that fundamentally alters the agent's perception of the environment over time.

Prior online algorithms that consider the scope of non-stationarity as the observation function focus on learning agent morphologies [175] and generalization in Block MDPs [176]. While this scope of non-stationarity holds significant potential for real-world applications [8], it remains underexplored.

We focus on the episodic evolution of the observation function at test-time in offline RL. In our setup, each dimension of an agent's state is influenced by an unknown, time-dependent constant additive offset that remains fixed within a single operational interval (an "episode"). This leads to a stream of evolving observation functions [7], extending across multiple future episodes, where the offsets remain hidden throughout the prediction window. For instance, industrial robots might apply a daily calibration offset to each joint, while sensors can exhibit a deviation until the next scheduled recalibration. Similarly, in healthcare or finance, data may be partially aggregated or withheld to comply with regulations, effectively obscuring finer-grained variations and leaving a single offset as the dominant factor per episode. By only storing these

representative offsets, we circumvent the challenges of continuous interaction buffers in bandwidth-constrained or privacy-sensitive environments. Because the offset can differ across state dimensions (e.g., different sensor or actuation channels), each state dimension can be affected by a different unknown bias that stays constant for that episode but evolves differently across episodes.

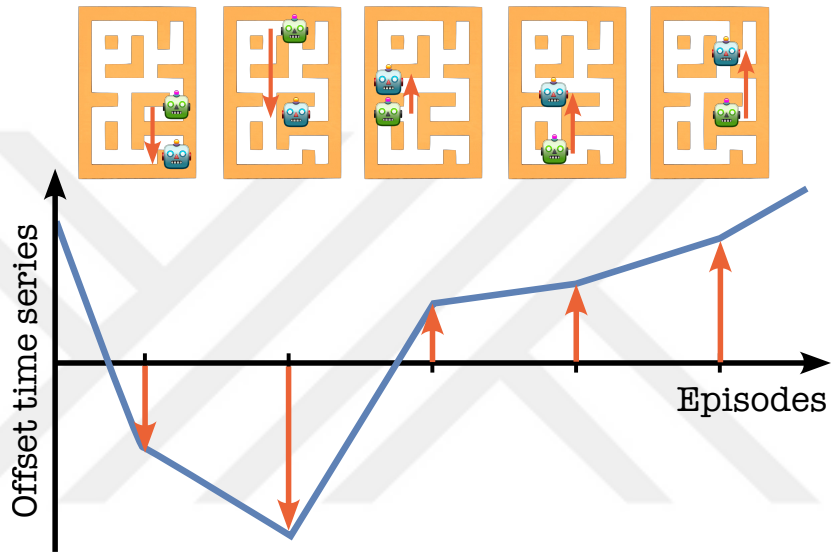


Figure 7.1. The agent does not know its location in the environment because its perception is offset every episode j by an unknown offset b^j (only vertical offsets are illustrated). FORL leverages historical offset data and offline RL data (from a stationary phase) to forecast and correct for new offsets at test time. Ground-truth offsets are hidden throughout the evaluation episodes.

Approaches that assume predefined perturbations can struggle with these abrupt, episodic shifts, because such offsets violate the typical assumption of smoothly varying observation functions. Frequent retraining, hyperparameter optimization, or extensive online adaptation to new observation function evolution patterns is costly and risky due to the trial-and-error approach in safety-critical settings. It may be infeasible if these patterns no longer reflect the restrictive assumptions made during training.

By separating offset data (episodic calibration values) from the massive replay buffers, a zero-shot forecasting-based approach can anticipate each new offset from the beginning of the episode without requiring policy updates or making assumptions about task evolution at test time [177]. Modeling these multidimensional additive offsets as stable, per-episode constants presents a robust and efficient way to handle time-varying conditions in non-stationary environments, where the evolution of tasks follows a non-Markovian, time-series pattern [9], mitigating the risks of online exploration.

We consider an offline RL setting during training where we only have access to a standard offline RL dataset collected from a stationary environment [31] with fully observable states. Initial data may be collected under near-ideal conditions and then gradually affected by wear, tear, or other natural shifts, even as the underlying physical laws (dynamics) remain unchanged. At test time, however, we evaluate in a non-stationary environment where both the observation function and the observation space change due to time-dependent external factors. This setup can be interpreted as environments shifting along observation space dimensions while the initial state of the agent is sampled from a uniform distribution over the state space.

A simplified version of this setup for an offset affecting only one dimension of the state is illustrated in Figure 7.1, which depicts only vertical offsets for clarity. Here, the agent knows it is in a maze but does not know where it is within the maze. Furthermore, it will remain uncertain of its location across all episodes at test-time, as in every episode a new offset leads to a systematic misalignment between perceived and actual positions. Importantly, these offsets may not conform to Gaussian or Markovian assumptions; instead, they may stem directly from complex, real-world time-series data [9] and remain constant throughout each episode. As a result, standard noise-driven or parametric state-estimation techniques, which typically rely on smoothly varying or randomly perturbed functions, cannot reliably identify these persistent, episode-wide offsets that are not available after the episode terminates. While zero-shot forecasting can adjust observation offsets, its performance depends on the forecaster's accuracy. Similarly, integrating zero-shot forecasting into a model-based offline RL approach [31]

can underperform when real-world offsets deviate from predefined assumptions about future observation functions. Our approach uses the insight that the belief of the true states can be refined from a sequence of actions and effects. For instance, in maze navigation, if an agent misjudges its location and hits a wall, analyzing its actions and delta observations leading to the collision can provide evidence for likely locations within the maze.

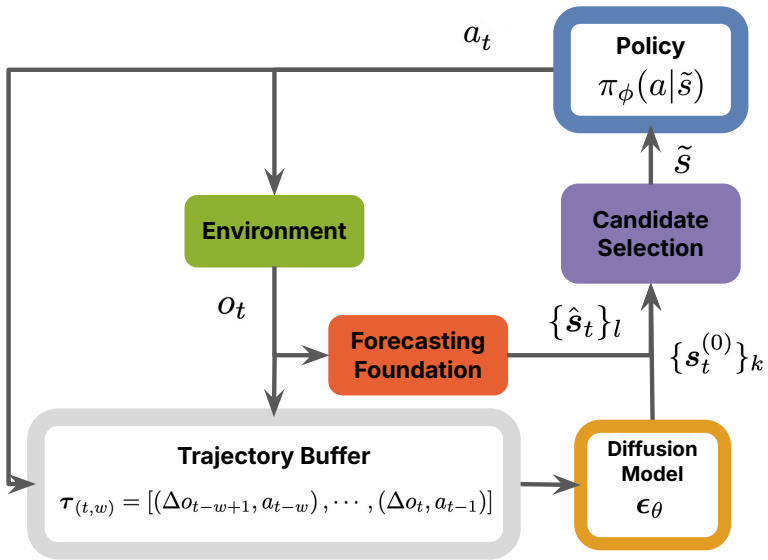


Figure 7.2. Overview of the proposed FORL framework at test-time.

We propose the Forecasting in Non-stationary Offline RL (FORL) framework (Figure 7.2) for test-time adaptation in non-stationary environments where the observation function is perturbed by an arbitrary time-series. Our framework has two main ingredients: forecasting offsets with a zero-shot time-series forecasting model [178] from past episode offsets (ground truth offsets after the episode terminates are not accessible at test-time) and a within-episode update of the state estimation using a conditional diffusion model [89] trained on offline stationary data. In Figure 7.2, the observations are processed by both the trajectory buffer and the time-series forecasting foundation module [178]. Observation changes and actions sampled from the policy, $(\Delta o, a)$ are stored in the trajectory buffer. The diffusion model generates candidate states $\{s_t^{(0)}\}_k$

conditioned on $\boldsymbol{\tau}_{(t,w)}$. The candidate selection module then generates the estimated $\tilde{s}_t$.

We unify the strengths of probabilistic forecasting and decision-making under uncertainty, to enable continuous adaptation when the environment diverges from predictions. Consequently, our framework: accommodates future offsets without assuming specific non-stationarity patterns during training, eliminating the need for retraining and hyperparameter tuning when the agent encounters new, unseen non-stationary patterns at test time, and targets non-trivial non-stationarities at test time without requiring environment interaction or knowledge of POMDPs during training. FORL introduces a novel, modular framework combining a conditional diffusion model (FORL-DM) for multimodal belief generation with a lightweight Dimension-wise Closest Match (DCM) fusion strategy, validated by extensive ablations on no-access to past offsets, policy-agnostic plug-and-play, offset magnitude and inter/intra-episode drifts. We propose a novel benchmark that integrates offsets from real-world time-series datasets with standard offline RL benchmarks, and demonstrate that FORL consistently outperforms baseline methods.

7.2. Method

In this section, we formulate our problem statement and describe our FORL diffusion model trained on the offline RL dataset to predict plausible states. Then, we introduce our online state estimation method, Dimension-wise Closest Match that uses plausible states predicted by the multimodal FORL diffusion model (DM) and the states predicted from past episodes by a probabilistic unimodal zero-shot time-series foundation model.

7.2.1. Problem Statement

Training (Offline Stationary MDP). We begin with an episodic, stationary Markov Decision Process (MDP) $\mathcal{M}_{\text{train}} = (\mathcal{S}, \mathcal{A}, P, R, \rho_0)$, where the initial state distribution ρ_0 is a uniform distribution over the state space $\mathcal{S}$. We only have access to an offline RL

dataset $\mathcal{D} = \{(\boldsymbol{s}_t^k, \boldsymbol{a}_t^k, \boldsymbol{s}_{t+1}^k, r_t^k)\}$ with k transitions collected from this MDP. Crucially, our FORL diffusion model and a diffusion policy [29] are trained offline using this dataset, such as the standard D4RL benchmark [21], without making any assumptions about how the environment might become non-stationary at test time.

Test Environment (Sequence of POMDPs). At test time, the agent faces an infinite sequence of POMDPs $\{\hat{\mathcal{M}}_j\}_{j=1}^{\infty}$. Each POMDP $\hat{\mathcal{M}}_j$ is described by a 7-tuple [125] $\hat{\mathcal{M}}_j = \big(\mathcal{S}, \mathcal{A}, \mathcal{O}_j, P, R, \rho_0, \mathbf{e}_j\big)$, where the state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel P and the reward function R remain identical to the training MDP. $\mathbf{e}$ is the observation function, where we restrict ourselves to deterministic versions ($\mathbf{e} : \mathcal{S} \to \mathcal{O}$) [126, 8].

Non-stationary environments can be formulated in different ways. In Khetarpal et al. [8], a general non-stationary RL formulation is put forward, which allows each component of the underlying MDP or POMDP to evolve over time

$$\big(\mathcal{S}(t), \mathcal{A}(t), P(t), R(t), \mathbf{e}(t), \mathcal{O}(t)\big). \tag{7.1}$$

A set κ determines which components vary, while a driver dictates their evolution. Specifically, passive drivers of non-stationarity suggest that external factors alone influence the environment's evolution In this work, we consider the scope of non-stationarity [8] (detailed in Section 3.2) that encompasses only the observation function and thus also the observation space, i.e. $\kappa = \{\mathbf{e}, \mathcal{O}\}$.

We consider the case where the non-stationarity is unfolding over episodes and where the observation function $\mathbf{e}_j$ is different at each episode j. The change in the observation function is assumed to have an additive structure and is independent of the agent's actions (passive non-stationarity [8]). Concretely, the function $\mathbf{e}_j$ offsets states s_t by a fixed offset $b^j \in \mathbb{R}^n$:

$$\mathcal{O}_j \;=\; \{\, s + b^j \,:\, s \in \mathcal{S}\}, \quad \mathbf{x}_j(s) \;=\; s + b^j.$$

Importantly, the sequence (b^j) can evolve under arbitrary real-world time-series data, and the agent does not have access to the ground-truth information throughout the evaluation, similar to scenarios where observations are only available periodically and shifts occur between these intervals. Thus, the episodes have a temporal order, relating to Non-Stationary Decision Processes (NSDP) that define a sequence of POMDPs [7] (Section 3.2).

Partial Observability and Historical Context. Since b^j is never directly observed for P episodes into the future, each $\hat{\mathcal{M}}_j$ is a POMDP. The agent receives only the offset-shifted observations $\{o_t\}$, where $o_t = s_t + b^j$ holds without any noise. Moreover, the agent may have access to a limited historical context of previous offsets $(b^{j-C}, \ldots, b^{j-1})$ at discrete intervals P, but no direct information about future offsets $(b^j, b^{j+1}, \ldots b^{j+P-1})$. Hence, the agent must forecast and/or adapt to unknown future offsets without prior non-stationary training.

Partial Identifiability. Despite observing $o_t = s_t + b^j$, the agent cannot generally disentangle s_t from b^j. For any single observation, there are infinite possible pairs of state and offset that yield $o_t = s' + b'$. Additionally, the initial state distribution ρ is uniform and does not provide information about b. Thus, we can only form a belief over s_t and refine that belief based on two sources of information: (a) the sequence of actions and effects observed within an episode and (b) the sequence of past identified offsets. To exploit source (a), we use a predictive model of commonly expected outcomes using a diffusion model, which will be explained next. To make use of source (b), we use a zero-shot forecasting model (see Section 7.2.3 for details). Afterwards, both pieces of information are fused (Section 7.2.4).

In our setting, we assume that the offsets added to the states are unobservable at test time, while the transition dynamics of the evaluation environment remain unchanged. To eventually reduce the uncertainty about the underlying state, we perform filtering or belief updates using the sequence of past interactions. To understand why the history

of interactions is indicative of a particular ground truth state, consider the following example in a maze environment illustrated in Figure 7.3. When the agent is moving north for three steps and then bumps into a wall, possible ground truth states can only be those three steps south of any wall following that trajectory pattern. The agent cannot observe the hidden green trajectory of ground-truth states; it only has access to the sequence of observation changes (Δo) and action vectors, which narrows down the possible positions to four candidate regions, exactly those identified by our model. Clearly, the distribution of possible states is highly multimodal, such that we propose using a diffusion model as a flexible predictive model of plausible states given the observed actions and outcomes. Diffusion models excel at capturing multimodal distributions [29], making them well-suited for our task. We train the diffusion model on offline data without offset ($b_j = 0$) which we detail below.

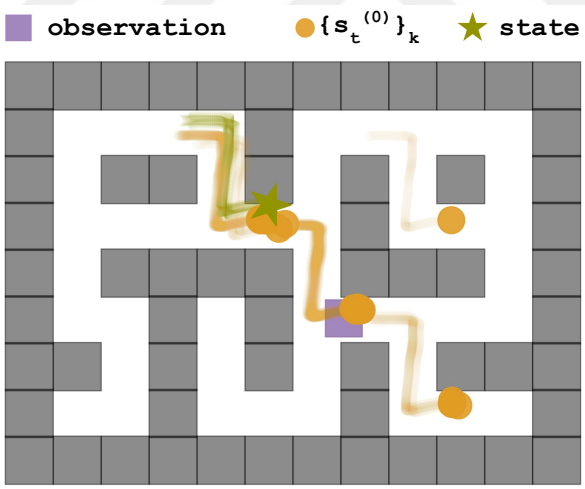


Figure 7.3. Candidate states generated by FORL Diffusion Model.

7.2.2. FORL Diffusion Model

To distinguish between the trajectory timesteps in reinforcement learning and the timesteps in the diffusion process, we use the subscript $t \in \{0, \ldots, T\}$ to refer to RL timesteps, and $n \in \{0, \ldots, N\}$ for diffusion timesteps. We first begin by defining a

subsequence of a trajectory

$$\boldsymbol{\tau}_{(t,w)} = \left[(\Delta o_{t-w+1}, a_{t-w}), \cdots, (\Delta o_t, a_{t-1})\right], \tag{7.2}$$

where delta observations $\Delta o_t = o_t - o_{t-1} = s_t - s_{t-1}$ denote the state changes (effects), w is the window size.

Using a conditional diffusion model, we aim to model the distribution $p\left(\boldsymbol{s}_t \mid \boldsymbol{\tau}_{(t,w)}\right)$. For that, we define the reverse process (denoising) as

$$p\big(s_t^{(N)}\big) \prod_{n=1}^{N} p_\theta \left(\boldsymbol{s}_t^{(n-1)} \mid \boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}\right), \quad p\big(s_t^{(N)}\big) = \mathcal{N}(\boldsymbol{s}_t^{(N)}; \mathbf{0}, \mathbf{I}), \tag{7.3}$$

and p_θ is modeled as the distribution $\mathcal{N}(\boldsymbol{s}_t^{(n-1)}; \mu_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n), \Sigma_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n))$ with learnable mean and variance. We approximate this mean $\mu_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n)$ using a conditional noise model ϵ_θ with

$$\frac{\boldsymbol{s}_t^{(n)}}{\sqrt{\alpha_n}} - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_{(n)}}\sqrt{\alpha_n}} \epsilon_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n), \tag{7.4}$$

and fix the covariance $\Sigma_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n)$ with $(1-\alpha_n)\mathbf{I}$ [89].

We can train the noise prediction model by sampling $\boldsymbol{s}_t^{(n)}$ for any diffusion timestep in forward diffusion process. Following Song et al. [179] and Ho et al. [89], we compute a noisy sample $s_t^{(n)}$ based on the true sample $s_t = s_t^{(0)}$:

$$s_t^{(n)} = \sqrt{\bar{\alpha}_n} s_t + \sqrt{1-\bar{\alpha}_n}\epsilon, \tag{7.5}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is the noise, $\bar{\alpha}_n = \prod_{i=1}^{n} \alpha_{(i)}$.

Selecting a large number of diffusion timesteps can significantly increase the computational complexity of our algorithm. Hence, we use variance preserving stochastic differential equations (SDE) [179] following the formulation in [180]

$$\alpha_n = e^{-\left(\beta_{\min}\left(\frac{1}{N}\right) + (\beta_{\max} - \beta_{\min})\frac{2n-1}{2N^2}\right)}, \tag{7.6}$$

where $\beta_{\max} = 10$ and $\beta_{\min} = 0.1$ for empirical reasons.

We can equally learn to predict the true samples by learning a noise model [181]. Hence, we train a noise model $\epsilon_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n)$ that learns to predict the noise vector

ϵ. By using the conditional version of the simplified surrogate objective from [89], we minimize

$$\mathcal{L}_p(\theta) = \mathop{\mathbb{E}}_{n,\boldsymbol{\tau},\boldsymbol{s_t},\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n\right)\right\|^2\right], \tag{7.7}$$

where $\boldsymbol{s}_t$ is the state sampled from the dataset D for $t \sim U_T(\{w, \ldots, T-1\})$, $\boldsymbol{s}_t^{(n)}$ is computed according to Equation (7.5), ϵ is the noise, $n \sim U_D(\{1, \ldots, N\})$ is the uniform distribution used for sampling the diffusion timestep.

We use the true data sample $\boldsymbol{s_t}$ from the offline RL dataset to obtain the noisy sample in Equation (7.5). Leveraging our model's capacity to learn multimodal distributions, we generate a set of k samples $\{\boldsymbol{s}_t^{(0)}\}$ as our predicted state candidates in parallel from the reverse diffusion chain. We use the noise prediction model [89] with reverse diffusion chain $\boldsymbol{s}_t^{(n-1)} \mid \boldsymbol{s}_t^{(n)}$ formulated as

$$\frac{\boldsymbol{s}_t^{(n)}}{\sqrt{\alpha_n}} - \frac{1-\alpha_n}{\sqrt{\alpha_n(1-\bar{\alpha}_{(n)})}}\epsilon_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n) + \sqrt{1-\alpha_n}\epsilon, \tag{7.8}$$

where noise is sampled with $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ for $n = N, \ldots, 1$, and $\epsilon = 0$ for $n = 1$ [89]. Below we detail how the set of generated state candidates are used during an episode.

7.2.3. Forecasting with Zero-Shot Foundation Model

Because we assume that the offsets b^j originate from a time series, we propose using a probabilistic zero-shot forecasting foundation model (Zero-Shot FM), such as Lag-Llama [178], to forecast future offsets from past ones. We assume that after P episodes the true offsets get revealed, and we predict the offsets for the following P episodes. Using the probabilistic Zero-Shot FM we generate $(\hat{b}_l^j, \ldots, \hat{b}_l^{j+P-1})$, where (l) denotes the number of samples generated for each episode (timestamp). Since Lag-Llama is a probabilistic model, it can generate multiple samples per timestamp, conditioned on C number of past contexts $(b^{j-C}, \ldots, b^{j-1})$. In practice, we forecast every dimension of b independently, as the Zero-Shot FM, (Lag-Llama [178]) is a univariate probabilistic model.

Algorithm 3 Candidate Selection

Sample $\{\{\hat{b}\}_l^1, \ldots, \{\hat{b}\}_l^P\} \sim Zero-ShotFM$
for each episode $p = 1, \cdots, P$ **do**
  $t = 0$
  Reset environment $o_0 \sim \mathcal{E}$
  $\tilde{s} \leftarrow o_0 - \overline{\{\hat{b}\}_l^p}$
  Initialize $\boldsymbol{\tau}_{(t,w)}$
  while not *done* **do**
    Sample $a^{(0)} \sim \pi_\phi(a|\tilde{s})$
    Take action $a^{(0)}$ in $\mathcal{E}$, observe o_{t+1}
    $\{\hat{s}_{t+1}^b\}_l \leftarrow o_{t+1} - \{\hat{b}\}_l^p$
    $\boldsymbol{\tau}_{(t+1,w)} = \text{PUSH}\left(\boldsymbol{\tau}_{(t,w)}, \left(\Delta o_{t+1}, a^{(0)}\right)\right)$
    $\boldsymbol{\tau}_{(t+1,w)} = \text{POP}\left(\boldsymbol{\tau}_{(t+1,w)}, \left(\Delta o_{t-w+1}, a_{t-w}\right)\right)$
    if $t > w$ **then**
      Sample $\{\boldsymbol{s}_{t+1}^{(0)}\}_k$ from FORL by Equation (7.8)
      $\tilde{s} \leftarrow \textbf{DCM}(\{\boldsymbol{s}_{t+1}^{(0)}\}_k, \{\hat{s}_{t+1}^b\}_l)$
    else
      $\tilde{s} \leftarrow o_{t+1} - \overline{\{\hat{b}\}_l^p}$
    end if
    $t \leftarrow t + 1$
  end while
end for

7.2.4. FORL State Estimation

The next step in our method is to fuse the information from the forecaster and the diffusion model into a state estimate used for control at test time.

At the beginning of an episode, no information can be obtained from the diffusion model, so for the first w steps we only rely on the forecaster's mean prediction, i.e.

$\tilde{s}_t = o_t - \overline{\hat{b}^j}$ where the mean is taken over the l samples.

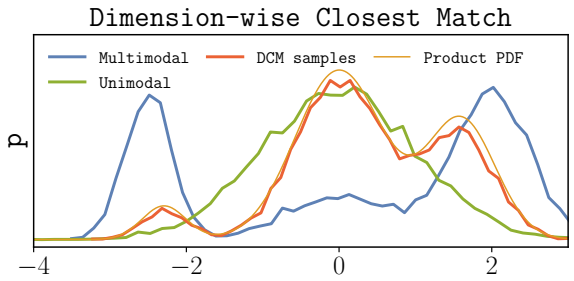


Figure 7.4. Distribution of samples produced by DCM.

As soon as a window size w of steps are taken, our FORL state estimation improves on the inferred state as detailed below. Figure 7.2 offers an overview of the entire system and Algorithm 3 provides a detailed pseudocode. To recap, the diffusion model generates samples $\{\boldsymbol{s}_t^{(0)}\}_k$ from the in-episode history τ defined in Equation (7.2). These samples represent a multimodal distribution of plausible state regions. The Zero-Shot FM generates l samples of offsets $\{\hat{b}\}_l$ from which we compute forecasted states using $\{\hat{\boldsymbol{s}}_t\}_l = o_t - \{\hat{b}\}_l$.

FORL: Dimension-wise Closest Match (DCM). We propose a lightweight approach to sample a good estimate based on the samples from the multimodal (diffusion model $\{\boldsymbol{s}_t^{(0)}\}_k$) and unimodal (Zero-Shot FM $\{\hat{s}_t^b\}_l$) distributions. Let $\mathcal{D}_{\text{diffusion}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, $\mathcal{D}_{\text{timeseries}} = \{\mathbf{y}_1, \ldots, \mathbf{y}_l\}$, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^n$. Then, Dimension-wise Closest Match (DCM) constructs $\mathbf{z} \in \mathbb{R}^n$ by

$$z_d = y_{j^*(d),d} \quad \text{where} \quad j^*(d) = \arg\min_j \Big(\min_i \big| x_{i,d} - y_{j,d} \big| \Big), \tag{7.9}$$

where $d = 1 \ldots n$. In other words, for each dimension d, we choose the sample from $\mathcal{D}_{\text{timeseries}}$ that has the closest sample in $\mathcal{D}_{\text{diffusion}}$. The process is straightforward yet effective, and under ideal sampling conditions for a toy dataset in Figure 7.4, where histograms for 10k samples are used for illustration, DCM approximately samples

from the product distribution. DCM uses a non-parametric search to find the forecast sample with the highest score, which corresponds to the minimum dimension-wise distance. DCM's prediction error is governed by the accuracy of the forecast samples in the unimodal $\mathcal{D}_{\text{timeseries}}$ that achieves this best score. As we will demonstrate in the experiments, this approach empirically yields lower maximum errors and is more stable compared to other methods.

Algorithm 3 summarizes the entire inference process at test time. We begin the episode by relying on the forecasted states $\tilde{s}_0$. As more transitions $(\Delta o_t, a_{t-1})$ become available, the FORL diffusion model proposes candidate states $\{\boldsymbol{s}_t^{(0)}\}_k$ through retrospection, reasoning over past in-episode experiences to adapt state estimation on the fly when they begin to diverge from predictions. We then invoke DCM to blend the diffusion model's candidates with the foundation model's unimodal forecasts and obtain the final state estimate $\tilde{s}_t$. We use an off-the-shelf offline RL policy such as Diffusion Q-Learning (DQL) [29] to select the agent's action a_t.

By combining a powerful zero-shot forecasting model with a conditional diffusion mechanism, FORL addresses partial observability in continuous state and action space when ground-truth offsets are unavailable. This procedure is performed in the absence of ground-truth offsets for past, current, and future episodes over the interval $j : j + P$ at test time. DCM provides a computationally inexpensive yet effective way of using the multimodal diffusion candidates and unimodal time-series forecasts. This robust adaptation approach yields a state estimate $\tilde{s}_t$, aligned with the agent's retrospective experience in the stationary offline RL dataset with the prospective external offset forecast.

7.3. Experimental Setup and Baselines

We evaluate FORL across navigation, and manipulation tasks in D4RL [21] and OGBench [32] offline RL environments each augmented with five real-world non-stationarity domains sourced from [33]. Our experiments primarily address the following

questions.

- Does FORL maintain state-of-the-art performance when confronted with unseen non-stationary offsets?
- How can we use FORL when we have no access to delayed past ground truth offsets?
- How does DCM compare to other fusion approaches?
- Can FORL handle intra-episode non-stationarity?
- How gracefully does performance degrade as offset magnitude $\boldsymbol{c}$ is scaled from zero (no offset) to one (our evaluation setup)?
- Can FORL serve as a plug-and-play module for different offline RL algorithms without retraining?

To answer the research questions posed and rigorously evaluate our algorithm, we compare its performance against three main categories of baselines.

7.3.1. Baselines

First, we introduce the set of recent and widely used offline RL algorithm backbones used throughout our experiments (Section 7.3.1.1). Second, we detail the naive application of these backbones as well as their extensions using the Zero-Shot FM forecasting module. Finally, we introduce DMBP, a testing-time model-based robust offline RL method, along with its own forecasting-augmented variant, DMBP+LAG.

7.3.1.1. Offline Reinforcement Algorithm Backbones. DQL [29] is an offline-RL algorithm that uses policy regularization via a conditional diffusion model [89]. The method and loss functions are detailed in Section 6.2.1.

Twin Delayed Deep Deterministic Behavior Cloning (TD3BC) [23] extends TD3 [182] to the offline RL setup. TD3+BC incorporates a behavior cloning regularization term, normalizes state features within the offline RL dataset, and scales the Q-function

using a hyperparameter with an added normalization term. TD3BC is a straightforward yet effective method while also being computationally efficient.

Robust Offline Reinforcement Learning (RORL) algorithm [77] addresses adversarial perturbations of the observation function by learning a conservative policy that aims to be robust to out-of-distribution (OOD) state and action pairs. To achieve this, it introduces a conservative smoothing mechanism that balances mitigating abrupt changes in the value function for proximate states and avoiding value overestimation in risky regions absent from the dataset. Concretely, RORL regularizes both the policy and the value function, leveraging bootstrapping Q-functions and conservative smoothing of the perturbed states. This formulation yields robust training under adversarial perturbations in the observation function while preserving strong performance even in unperturbed environments.

Implicit Q-Leaning (IQL) [183] first learns a value function by expectile loss and Q-function by Mean Squared Error (MSE) Loss without using the policy and instead using actions from the dataset. In doing so, they avoid approximating the values of unseen actions. Then, it learns the policy using advantage weighted regression [184, 185, 64, 65] using the learned Q-funcion and value function. We use the open-source implementation of IQL from [186], which references the source [187].

Flow Q-learning (FQL) [188] is a recent offline RL policy that shows strong performance on the OGBench [32]. We use FQL for the offline RL environments in OGBench and adopt the hyperparameters from the open-source implementation [189]. Similar to DQL, which uses diffusion models, FQL can learn an expressive policy. FQL trains two policies: (i) a flow policy trained with flow matching on the offline RL dataset for behavior cloning conditioned on the state, and (ii) a one-step policy trained with a distillation loss using the flow policy [190, 191, 192, 193, 194] and a critic loss. This approach avoids expensive backpropagation through time; thus, it is fast during inference and training.

7.3.1.2. Extensions with Zero-Shot FM. For our baselines, we extend the offline RL backbones $\pi \in \{$DQL [29], FQL [188], IQL [183],RORL [77],TD3+BC [23]$\}$ with three variations. The underlying policies throughout our experiments are identical policy checkpoints for both our method and the baselines. When not explicitly stated with a suffix -T, -I, -R, -F we use DQL policy [29].

- π: The policy directly generates actions conditioned on the observations. We include it to demonstrate how large episodic offsets degrade policy performance, and its consistently poor performance underscores the difficulty introduced by our offset settings.
- $\pi + LAG - \bar{s}$: We use Zero-Shot FM (Lag-Llama [178]) to forecast episodic offsets over a horizon of P episodes using pre-deployment offset history, then subtract the mean predicted offset corresponding to that episode from the observations. DQL then generates actions conditioned on the sample mean of the forecasted states $\{\hat{s}_t^b\}_l$. Although $\pi + LAG - \bar{s}$ outperforms π, straightforward application of forecasting and decision making reveals performance degradation when offsets lack adaptive correction.
- $\pi + LAG - \tilde{s}$: We use the sample median instead of the sample mean in $\pi + LAG - \bar{s}$, to mitigate outlier effects. As shown in Figure 7.10, median-based bias removal yields no significant improvement over the mean-based approach.

7.3.1.3. DMBP+LAG. We extend the Diffusion Model–Based Predictor (DMBP) [31], a model-based robust offline RL method trained on unperturbed D4RL benchmarks, to correct test-time state observation perturbations. DMBP has proven robust under Gaussian noise (varying σ), uniform noise, and adversarial perturbations, including maximum action-difference (MAD) and minimum Q-value (Min-Q) attacks. Similar to our framework, perturbed states are passed to DMBP, which removes the perturbations; the generated state is then fed to the policy. For fair comparison, we adopt the policies also used in their work for our experiments, RORL [77], TD3+BC [23], DQL [29].

DMBP+LAG extends DMBP with the forecasting module in our experiments. At each timestep, we remove offset biases using the sample mean of forecasted states $\{\hat{s}_t^b\}_l$ (as in $\pi + LAG - \bar{s}$) before DMBP correction. DMBP+LAG thus sequentially applies forecast-based offset compensation, followed by model based perturbation removal, and then queries the policy.

7.3.2. Experimental Setup

7.3.2.1. Zero-shot Foundation Model and Time-Series Datasets. We extract the first two univariate series from five time-series datasets: Real-Data-A (Australian-Electricity-Demand) [195], Real-Data-B (Electricity) [196], Real-Data-C (Electricity-Hourly) [195, 196], Real-Data-D (Electricity-Nips) [196, 197], and Real-Data-E (Exchange-Rate [198], [195] all accessed via GluonTS [33, 34]. Forecasts for the first two series of each domain are provided in Figure 7.5.

We aim to capture a broad spectrum of scenarios for a comprehensive evaluation. For instance, the Electricity-Hourly dataset provides hourly electricity usage data from various consumers, while the Australian-Electricity-Demand dataset offers 30-minute interval records of electricity demand across different Australian states. The Exchange-Rate dataset, on the other hand, includes daily exchange rates of multiple currencies, including those of Australia, the United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore.

To effectively represent diverse offset patterns in multiple directions, we apply feature scaling to the time-series data using a normalization $x'_c = \frac{x-\bar{x}}{\max(x)-\min(x)}$ where sample mean $\bar{x}$, $\min(x)$ and $\max(x)$ are computed from the available data up to the context length. Furthermore, we scale these values using the minimum and maximum state values observed in the offline RL dataset [21] for navigation and the minimum and maximum state values of the initial state distribution at test time for manipulation [21], ensuring diverse observation spaces that can accurately represent a wide range of scenarios.

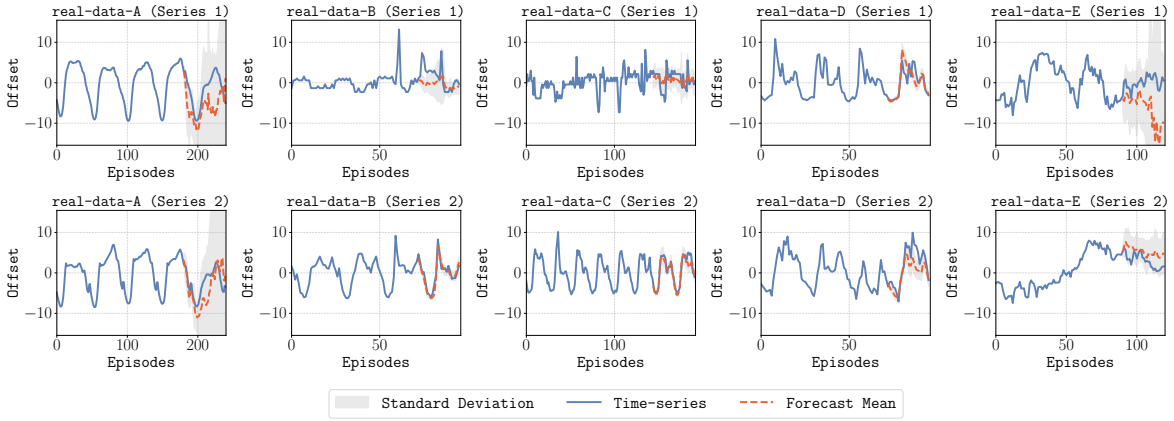


Figure 7.5. Zero-shot forecasting results for the first two time-series in univariate time-series datasets.

7.3.2.2. Offline Reinforcement Learning Environments. We use the standard D4RL [21] offline RL environments [21] with no modifications during training, namely Antmaze-Medium-Diverse (AntM-M-D), Maze2d-Medium (Maze2d-M), Antmaze-Large-Diverse (AntM-L-D), Maze2D-Large (Maze2D-L), Maze2D-Medium Maze2d-M, and Antmaze-Umaze-Diverse (AntM-U-D), where initial states are randomized in both the evaluation environment and the offline dataset. Figure 6.6 (b), (c), (d), Figure 7.6 illustrate the maze and Kitchen-Complete (Kitchen) environments, respectively. The Maze2D-L and Maze2d-M environments share the same maze configurations as AntM-L-D, AntM-M-D respectively. For manipulation tasks, we train on the standard Kitchen environment. The offsets affect the state dimensions associated with the base and shoulder joint angles.

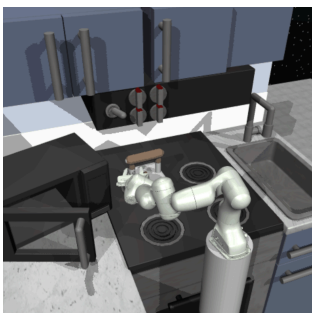


Figure 7.6. Kitchen complete environment in D4RL offline RL benchmark [21].

OGBench benchmark [32] contains both standard and goal-conditioned offline reinforcement learning tasks. To induce non-stationarity at test time, we follow the procedure from our D4RL experiments and use time-series data from GluonTS [34, 33]. For all tasks in Figure 7.7, we use the default 'singletask-v0' variant. We report results using the FQL algorithm [188] with its officially recommended hyperparameters. For the Antmaze-Large-Navigate (AntM-L-Nav) environment, we use the first two time series from the GluonTS Real-Data-A,B,C,D,E datasets and apply an offset scaling factor of $\boldsymbol{c} = 0.5$. For Cube-Single-Play (Cube-S-P), we apply offsets to the first 17 observation dimensions, which include all joint positions, joint velocities, and end effector variables (position and yaw), using the first 17 time series from each of the GluonTS Real-Data-A,B,C,D,E datasets. Because the Real-Data-A dataset only has five time series, we cycle through them repeatedly until all 17 dimensions are covered. Across all Cube-S-P experiments, we use an offset scaling factor of $\boldsymbol{c} = 0.25$.

Figure 7.7. Cube-single-play and antmaze-large-navigate environments in OGBench benchmark [32].

7.3.3. Implementation Details

During training, we use the original offline RL dataset without offsets. At test-time, the offsets affect the first two state dimensions, where each offset sequence is drawn from the first two univariate time–series from diverse datasets. The agent's policy receives only the offset-corrupted observations, with no direct access to the true underlying

states throughout P episodes. The time–series forecasting model, given the past C ground-truth offsets $(b^{j-C}, \cdots, b^{j-1})$, predicts the future offsets $(b^{j}, \cdots, b^{j+P})$ during testing. FORL leverages these predictions and in-episode experience to dynamically adapt to unknown external perturbations. All experiments use 5 random seeds, except for the focused error analysis across all evaluation episodes for the task shown in our illustrative maze environment, with results reported in Table 7.7, and preliminary results for affine transformations, where we use one seed.

The architecture of our FORL Model is a noise prediction conditional TemporalUnet diffusion model [199, 86, 31]. Different from the architecture used in [31], we concatenate each element in $\boldsymbol{\tau}_{(t,w)}$ with $\boldsymbol{s}_t^{(n)}$ and feed it to our model without additional encoders, using the diffusion timestep embedding in the Residual Temporal Blocks. For the TemporalUnet architecture, we concatenate the Unet model output with the time-embedding before feeding it to fully connected layers, particularly in the antmaze environments due to the large input size. The set of hyperparameters is provided in Table A.1. Although the FORL conditional diffusion model is specifically utilized for time-dependent offsets in the first two dimensions of the state vector, it is trained for general-purpose state prediction, enabling it to predict all dimensions of the state in Maze2d, Antmaze, and OGBench environments. This approach is taken because we do not assume prior knowledge of the evaluation environment.

The method for setting seeds involves a function that initializes the seed across all relevant libraries (PyTorch, CUDA, NumPy, Gym Environment, and Python's random module) to ensure the replicability of results. We use the open source implementation of DMBP with the suggested hyperparameters [200], and the pre-trained Lag-Llama model[178].

Table 7.1. Normalized scores for FORL framework and the baselines with DQL policy.

Maze2d-M	DQL	DQL+LAG-$\bar{s}$	DMBP+LAG	FORL (ours)
Real-Data-A	30.2 ± 6.5	30.2 ± 8.6	25.1 ± 9.8	**63.3 ± 6.7**
Real-Data-B	14.1 ± 12.1	**53.4 ± 14.6**	**41.2 ± 21.1**	**66.5 ± 18.2**
Real-Data-C	-2.3 ± 3.3	56.7 ± 18.5	56.9 ± 18.4	**86.3 ± 15.7**
Real-Data-D	4.7 ± 5.0	36.9 ± 16.3	38.5 ± 14.2	**103.4 ± 11.9**
Real-Data-E	3.5 ± 8.8	8.7 ± 6.0	11.4 ± 2.8	**51.2 ± 13.7**
Average	10.0	37.2	34.6	**74.1**
Maze2D-L	DQL	DQL+LAG-$\bar{s}$	DMBP+LAG	FORL (ours)
Real-Data-A	16.2 ± 5.5	2.4 ± 1.1	4.2 ± 5.8	**42.9 ± 4.1**
Real-Data-B	-0.5 ± 2.9	5.5 ± 9.0	15.0 ± 14.6	**34.9 ± 9.2**
Real-Data-C	0.9 ± 1.7	16.6 ± 7.5	26.8 ± 8.4	**45.6 ± 4.1**
Real-Data-D	3.0 ± 6.6	8.6 ± 3.2	13.4 ± 4.1	**58.4 ± 6.5**
Real-Data-E	-2.1 ± 0.4	**2.6 ± 3.4**	**0.9 ± 3.7**	**12.0 ± 9.9**
Average	3.5	7.1	12.1	**38.8**
AntM-U-D	DQL	DQL+LAG-$\bar{s}$	DMBP+LAG	FORL (ours)
Real-Data-A	22.7 ± 3.0	41.0 ± 5.2	45.7 ± 4.8	**65.3 ± 8.7**
Real-Data-B	24.2 ± 3.5	48.3 ± 7.0	**62.5 ± 13.2**	**74.2 ± 10.8**
Real-Data-C	21.7 ± 3.5	50.4 ± 8.3	60.4 ± 3.9	**78.8 ± 8.5**
Real-Data-D	5.8 ± 2.3	26.7 ± 6.3	29.2 ± 5.9	**75.8 ± 8.0**
Real-Data-E	6.0 ± 6.8	58.0 ± 16.6	59.3 ± 7.6	**81.3 ± 6.9**
Average	16.1	44.9	51.4	**75.1**
AntM-M-D	DQL	DQL+LAG-$\bar{s}$	DMBP+LAG	FORL (ours)
Real-Data-A	31.0 ± 6.5	**40.0 ± 5.7**	**39.7 ± 4.0**	**44.0 ± 7.9**
Real-Data-B	23.3 ± 4.8	**48.3 ± 4.8**	**43.3 ± 16.0**	**55.8 ± 7.0**
Real-Data-C	10.0 ± 2.3	**48.3 ± 3.4**	**49.6 ± 3.7**	**52.9 ± 9.5**
Real-Data-D	11.7 ± 5.4	46.7 ± 7.5	41.7 ± 6.6	**64.2 ± 8.6**
Real-Data-E	**18.7 ± 4.5**	**27.3 ± 8.6**	**26.0 ± 5.5**	**26.7 ± 4.7**
Average	18.9	42.1	40.1	**48.7**
AntM-L-D	DQL	DQL+LAG-$\bar{s}$	DMBP+LAG	FORL (ours)
Real-Data-A	11.0 ± 1.9	11.3 ± 4.9	9.0 ± 4.5	**34.3 ± 5.7**
Real-Data-B	5.8 ± 4.8	9.2 ± 4.6	8.3 ± 2.9	**46.7 ± 11.9**
Real-Data-C	5.4 ± 2.4	22.1 ± 5.6	17.9 ± 3.8	**33.8 ± 6.8**
Real-Data-D	2.5 ± 2.3	14.2 ± 3.7	14.2 ± 6.3	**46.7 ± 12.6**
Real-Data-E	**5.3 ± 3.8**	**3.3 ± 2.4**	**3.3 ± 0.0**	**11.3 ± 7.3**
Average	6.0	12.0	10.5	**34.6**

Table 7.2. Normalized scores for FORL and the baselines using FQL in OGBench benchmarks.

Cube-S-P	FQL	FQL+LAG-$\bar{s}$	FORL-F
Real-Data-A	0.0 ± 0.0	0.0 ± 0.0	**23.7 ± 3.6**
Real-Data-B	0.0 ± 0.0	15.0 ± 7.0	**60.0 ± 7.0**
Real-Data-C	0.4 ± 0.9	10.0 ± 1.7	**42.1 ± 5.6**
Real-Data-D	0.0 ± 0.0	0.8 ± 1.9	**70.0 ± 13.0**
Real-Data-E	0.0 ± 0.0	0.0 ± 0.0	**32.7 ± 9.5**
Average	0.1	5.2	**45.7**
AntM-L-Nav	FQL	FQL+LAG-$\bar{s}$	FORL-F
Real-Data-A	**22.7 ± 2.2**	1.3 ± 0.7	**24.3 ± 4.3**
Real-Data-B	21.7 ± 5.4	**29.2 ± 8.8**	**40.0 ± 7.6**
Real-Data-C	5.0 ± 1.1	34.6 ± 6.7	**55.8 ± 3.7**
Real-Data-D	0.8 ± 1.9	37.5 ± 5.1	**75.8 ± 5.4**
Real-Data-E	**10.0 ± 4.1**	3.3 ± 0.0	**15.3 ± 8.7**
Average	12.0	21.2	**42.2**

7.3.4. Results

Figure 7.8 illustrates an agent navigating the Maze2D-L environment where the true position is labeled as "state". The agent receives an observation indicating where it believes it is located due to unknown time-dependent factors. The candidate states predicted by the FORL diffusion model are shown as circles. Importantly, the agent's $(\Delta o, a)$-trajectory can reveal possible states for the agent. FORL's diffusion model (DM) component predicts these candidate states by using observation changes (Δo) and corresponding actions (a). The possible candidate regions where the agent can be are limited, and our model successfully captures these locations. FORL's candidate selection module (DCM) using the samples from the forecaster, and the diffusion model recovers a close estimate for the state. In contrast, the baseline DQL+LAG-$\bar{s}$ relies on the forecaster [178] for state predictions that are significantly farther from the actual state. Consistent with the results in Figure 7.10, FORL reduces prediction errors at test-time, thereby improving performance.

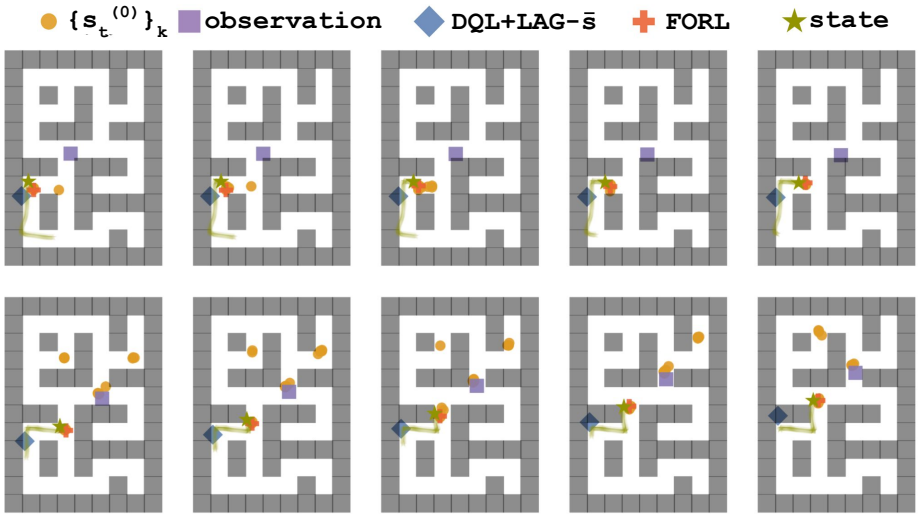


Figure 7.8. Illustrations of the states predicted by FORL and baselines.

FORL outperforms both pure forecasting (DQL+LAG-$\bar{s}$) and the two-stage strategy that first predicts offsets with a time-series model and then applies a noise-robust offline RL algorithm (DMBP+LAG). Its advantage is consistent across previously unseen non-stationary perturbations from five domains, each introducing a distinct univariate series into a separate state dimension at test time. We present the average normalized scores for expected cumulative rewards over the prediction length P across multiple episodes run in the D4RL [21] and OGBench [32] for each time-series in Tables 7.1 and 7.2. In this chapter, bold values in all tables indicate the best performance and those not significantly different from the best based on the pairwise Welch's t-test ($p > 0.05$). Figure 7.10 plots the ℓ_2 norm between the ground-truth states s_t and those predicted by FORL and the baselines in the Antmaze and Maze2d environments. Consistent with the average scores, FORL achieves the lowest prediction error on average.

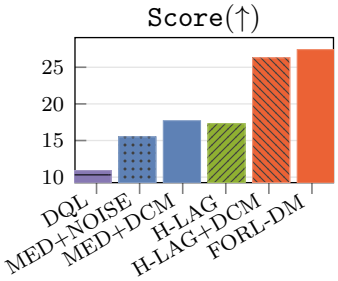


Figure 7.9. Scores of FORL diffusion model and the baselines when we do not have access to past offsets.

7.3.4.1. No Access to Past Offsets. We evaluate different variants of using FORL's diffusion model (DM) and Zero-Shot FM when we do not have any access to past offsets. FORL-DM, which we also refer to as DM for brevity, utilizes the candidate states generated by the FORL's diffusion model component (Section 7.2.2), which can be a multimodal distribution (Figure 7.3). Compared to DM in Figure 7.9, the full FORL framework yields a 97.8% relative performance improvement. Notably, DM performs on par with our extended baselines that incorporate historical offsets and forecasting, namely DMBP+LAG, DQL+LAG-$\bar{s}$, and DQL+LAG-$\tilde{s}$. Moreover, without access to historical offset information before evaluation, DM achieves a 151.4% improvement over DQL, demonstrating its efficacy as a standalone module trained solely on a standard, stationary offline RL dataset without offset labels.

H-LAG maintains a history of offsets generated by the diffusion model over the most recent C episodes excluding the evaluation interval P, since offsets are not revealed after episode termination at test-time. We then feed this history into the Zero-Shot FM to generate offset samples for the next P evaluation episodes. These samples are applied directly at test time.

H-LAG+DCM initially follows the same procedure in H-LAG to obtain predictions from Zero-Shot FM. Then, we apply DCM to these predicted offsets and the candidate states generated by FORL's diffusion model. Empirically, H-LAG+DCM outperforms

H-LAG, demonstrating that DCM with FORL's diffusion model can improve robustness. Overall, scores and prediction errors indicate that just using the samples from DM has better scores on average, while H-LAG+DCM is more stable in Figure 7.11.

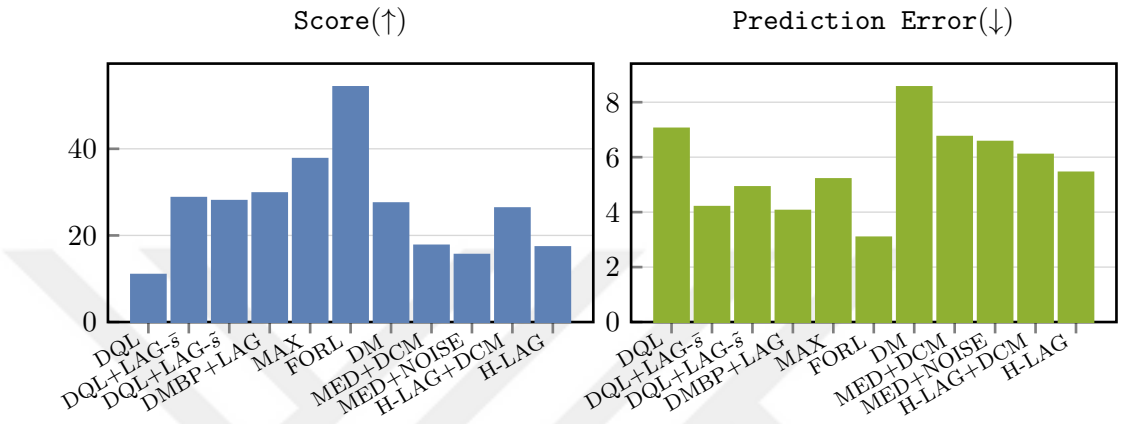


Figure 7.10. Comparison of average normalized scores and prediction errors across all navigation experiments FORL and baseline methods.

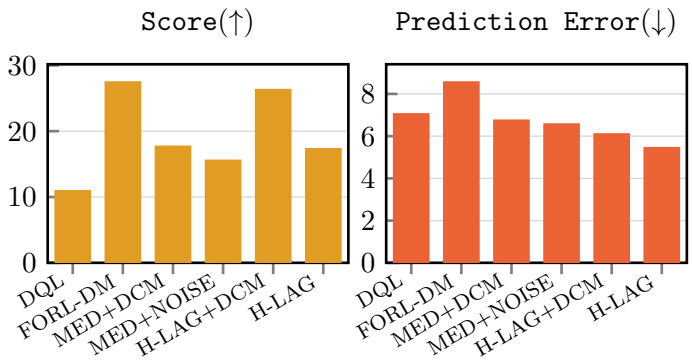


Figure 7.11. Performance comparison without access to past offsets.

FORL-DM, directly uses the state predicted by the diffusion model component. Given the multimodal nature of these candidate states, this selection does not fully leverage our framework. Yet, FORL-DM outperforms the baselines when no past offsets are used. Additional comparisons in Table 7.4 with other standard statistical methods like DM-MAD, DM-RANSAC, DM-RUNNING-μ, DM-RUNNING-μ-p indicate that only using the sample predicted by the diffusion model yields higher scores on average.

Table 7.3. Using a Zero-Shot FM in combination with FORL-DM in no access to past offsets setting.

Maze2d-M	DQL	MED+NOISE	MED+DCM	H-LAG	H-LAG+DCM	FORL-DM
Real-Data-A	30.2 ± 6.5	27.4 ± 14.5	27.4 ± 12.2	29.7 ± 11.3	**48.8 ± 10.0**	**55.2 ± 10.6**
Real-Data-B	14.1 ± 12.1	23.9 ± 14.3	**33.1 ± 19.5**	4.5 ± 12.3	**50.3 ± 12.9**	**56.8 ± 24.7**
Real-Data-C	-2.3 ± 3.3	17.6 ± 8.1	-2.6 ± 2.5	26.5 ± 5.4	30.3 ± 5.6	**52.8 ± 10.3**
Real-Data-D	4.7 ± 5.0	18.9 ± 9.1	**64.1 ± 12.7**	18.2 ± 13.7	1.8 ± 3.8	**60.1 ± 20.2**
Real-Data-E	3.5 ± 8.8	21.8 ± 15.6	-2.1 ± 1.6	**56.7 ± 11.6**	**45.7 ± 12.4**	**60.5 ± 18.2**
Average	10.0	21.9	24.0	27.1	35.4	**57.1**
Maze2D-L	DQL	MED+NOISE	MED+DCM	H-LAG	H-LAG+DCM	FORL-DM
Real-Data-A	**16.2 ± 5.5**	6.4 ± 2.9	-1.8 ± 0.5	7.1 ± 3.9	7.3 ± 2.0	**11.1 ± 4.3**
Real-Data-B	-0.5 ± 2.9	-0.1 ± 1.7	-1.4 ± 1.7	-1.4 ± 2.3	**0.9 ± 4.5**	**13.4 ± 10.3**
Real-Data-C	0.9 ± 1.7	3.2 ± 4.7	-2.0 ± 0.6	3.4 ± 2.0	0.5 ± 1.7	**9.4 ± 2.5**
Real-Data-D	3.0 ± 6.6	3.7 ± 6.8	**47.6 ± 16.4**	1.0 ± 4.5	2.9 ± 4.0	7.4 ± 7.2
Real-Data-E	-2.1 ± 0.4	**3.8 ± 4.7**	**-0.5 ± 3.7**	**1.8 ± 2.7**	**7.2 ± 2.3**	**7.7 ± 7.6**
Average	3.5	3.4	8.4	2.4	3.8	**9.8**
AntM-U-D	DQL	MED+NOISE	MED+DCM	H-LAG	H-LAG+DCM	FORL-DM
Real-Data-A	22.7 ± 3.0	8.3 ± 2.4	8.3 ± 4.7	32.3 ± 6.3	**62.7 ± 11.2**	**48.7 ± 9.1**
Real-Data-B	24.2 ± 3.5	9.2 ± 5.4	25.0 ± 13.2	41.7 ± 5.1	33.3 ± 2.9	**56.7 ± 8.1**
Real-Data-C	21.7 ± 3.5	10.8 ± 7.7	**75.8 ± 6.4**	36.7 ± 3.2	59.2 ± 11.5	55.0 ± 8.7
Real-Data-D	5.8 ± 2.3	17.5 ± 11.6	6.7 ± 6.3	10.8 ± 2.3	42.5 ± 7.5	**54.2 ± 7.2**
Real-Data-E	6.0 ± 6.8	**50.0 ± 13.5**	2.0 ± 3.0	14.0 ± 8.3	**42.7 ± 11.9**	**53.3 ± 7.8**
Average	16.1	19.2	23.6	27.1	48.1	**53.6**
AntM-M-D	DQL	MED+NOISE	MED+DCM	H-LAG	H-LAG+DCM	FORL-DM
Real-Data-A	31.0 ± 6.5	34.3 ± 10.4	**55.0 ± 6.8**	15.0 ± 4.9	17.3 ± 6.3	4.3 ± 0.9
Real-Data-B	23.3 ± 4.8	15.8 ± 9.9	**42.5 ± 4.6**	33.3 ± 4.2	28.3 ± 3.5	6.7 ± 4.8
Real-Data-C	10.0 ± 2.3	23.3 ± 5.4	10.0 ± 4.3	31.2 ± 4.9	**40.4 ± 6.7**	4.6 ± 1.7
Real-Data-D	11.7 ± 5.4	26.7 ± 6.3	**33.3 ± 15.9**	9.2 ± 3.5	**36.7 ± 6.8**	3.3 ± 3.5
Real-Data-E	18.7 ± 4.5	**26.7 ± 18.1**	0.0 ± 0.0	14.7 ± 5.6	**46.0 ± 16.2**	6.0 ± 4.3
Average	18.9	25.4	28.2	20.7	**33.7**	5.0
AntM-L-D	DQL	MED+NOISE	MED+DCM	H-LAG	H-LAG+DCM	FORL-DM
Real-Data-A	**11.0 ± 1.9**	5.0 ± 3.9	1.3 ± 1.4	**9.0 ± 1.9**	**11.3 ± 4.1**	**11.0 ± 3.0**
Real-Data-B	**5.8 ± 4.8**	**7.5 ± 3.5**	**5.0 ± 5.4**	**9.2 ± 1.9**	**7.5 ± 1.9**	**11.7 ± 4.6**
Real-Data-C	5.4 ± 2.4	5.4 ± 2.4	1.7 ± 0.9	**10.8 ± 2.7**	**12.9 ± 5.8**	**12.1 ± 2.7**
Real-Data-D	2.5 ± 2.3	**15.8 ± 6.2**	**11.7 ± 4.6**	**8.3 ± 6.6**	**14.2 ± 6.3**	**15.0 ± 9.6**
Real-Data-E	**5.3 ± 3.8**	**5.3 ± 3.0**	1.3 ± 1.8	**8.7 ± 3.0**	**6.0 ± 2.8**	**8.7 ± 5.1**
Average	6.0	7.8	4.2	9.2	10.4	**11.7**

In MED+DCM, we first compute the median of the predicted offsets from the previous episode, beginning with the first episode predicted by FORL's diffusion model. We then fit a Gaussian distribution centered at this median and sample l offsets from it, matching the sample count of Zero-Shot FM. Next, we apply DCM to these samples

with the candidates generated by the diffusion model.

In MED+NOISE, we begin by computing the median offset produced by the diffusion model during the initial evaluation episode similar to MED+DCM. Thereafter, we treat these offsets as evolving according to a random walk where each offset is predicted as the previous value with white-noise increments.

DM-MAD follows a state estimation based on robust statistics [201]. DM-MAD computes the coordinate-wise median of the differences between the observation and each denoiser prediction and discards any sample whose absolute deviation from this median exceeds ϵ x Median Absolute Deviation (MAD). Then it takes the median of the remaining inliers to obtain a robust offset estimate, and subtracts that offset from the observation to obtain the state $\tilde{s}_t$.

Table 7.4. Additional results for no-access to past offsets setting.

Maze2d-M	DM-MAD	DM-RANSAC	DM-RUNNING-μ	DM-RUNNING-μ-p	FORL-DM
Real-Data-A	**44.3 ± 10.6**	**46.8 ± 12.4**	**37.8 ± 13.6**	37.4 ± 8.7	**55.2 ± 10.6**
Real-Data-B	**46.8 ± 19.0**	**44.6 ± 17.1**	**42.4 ± 24.5**	**51.6 ± 11.7**	**56.8 ± 24.7**
Real-Data-C	**46.4 ± 10.9**	**44.5 ± 10.1**	**41.8 ± 16.0**	**61.7 ± 14.1**	**52.8 ± 10.3**
Real-Data-D	**47.7 ± 22.9**	**49.9 ± 23.2**	**44.4 ± 24.1**	**52.5 ± 13.3**	**60.1 ± 20.2**
Real-Data-E	**42.9 ± 10.6**	**52.4 ± 15.7**	**48.3 ± 18.8**	24.2 ± 12.2	**60.5 ± 18.2**
Average	45.6	47.6	42.9	45.5	**57.1**

DM-RANSAC is a RANdom SAmple Consensus (RANSAC) [202] based state estimation using the samples from our DM. DM-RANSAC calculates the offsets using the states generated by our DM and the observation, setting an adaptive per-dimension threshold as ϵ x MAD. Then, it repeatedly samples a random offset candidate and chooses the candidate whose inlier set (differences within that threshold) is the largest. Then, it takes the average of those inliers to estimate the offset.

DM-RUNNING-μ computes a numerically stable, global running mean [203, 204] of offsets from DM aggregated across all timesteps and episodes.

DM-RUNNING-μ-p computes an online average of offsets [203, 204] from DM (Running-μ) per episode p. Unlike DM-RUNNING-μ, in DM-RUNNING-μ-p the statistics are reset to zero at the beginning of each episode.

7.3.4.2. Dimension-wise Closest Match Ablations. We compare FORL (DCM) against four alternative fusion strategies in Figure 7.12. In FORL(KDE), for each dimension, we fit a kernel density estimator (KDE) on $\mathcal{D}_{\text{diffusion}} = \{\boldsymbol{s}_t^{(0)}\}_k$ and then we evaluate that probability density function for each point in $\mathcal{D}_{\text{timeseries}}$. Then, we take the product of these densities in each dimension to obtain the weight for each sample $\hat{s}_t^b$. We obtain a single representative sample by taking the weighted average of samples in $\mathcal{D}_{\text{timeseries}}$. To ensure stability, when the sum of the weights is near zero, we use the mean of the $\mathcal{D}_{\text{timeseries}}$ as the states. We use Scott's rule [205] to compute the bandwidth. DM-FS-$\bar{s}$, DM-FS-$\tilde{s}$ select the closest prediction from DM to the mean and median of the Zero-Shot FM's predictions, respectively. FORL (MAX) constructs a diagonal multivariate distribution from the dimension-wise mean and standard deviation of the forecasted states, then selects the sample predicted by our diffusion model with the highest likelihood under that distribution.

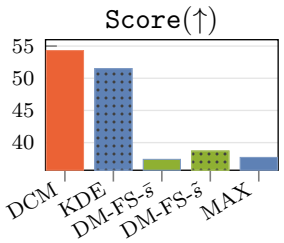


Figure 7.12. Candidate Selection Algorithms.

Although all baselines fuse information using the same two sets generated by the diffusion model and Zero-Shot FM, DCM has higher performance. In Table 7.7 we compute the maximum, minimum, and mean prediction error values over the test episodes used in Figure 7.8. The results in Table 7.6, Table 7.7, and Table 7.5 show

that FORL (DCM) more consistently performs better than other methods. Although the KDE-based FORL(KDE) method performs well in AntM-M-D, it significantly underperforms in Cube-S-P (Table 7.5). Moreover, it requires bandwidth selection and a fallback mechanism to handle numerical instability, which highlights the practical advantage of DCM.

Table 7.5. Normalized scores for FORL (DCM) and the baselines in Cube-Single-Play.

Cube-S-P	FQL	FORL-DM-F	FQL+LAG-$\bar{s}$	FORL (MAX)-F	FORL(KDE)-F	FORL-F
Real-Data-A	0.0 ± 0.0	3.0 ± 1.4	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**23.7 ± 3.6**
Real-Data-B	0.0 ± 0.0	6.7 ± 5.6	15.0 ± 7.0	43.3 ± 4.8	2.5 ± 2.3	**60.0 ± 7.0**
Real-Data-C	0.4 ± 0.9	4.6 ± 1.7	10.0 ± 1.7	**39.6 ± 4.4**	**38.3 ± 3.8**	**42.1 ± 5.6**
Real-Data-D	0.0 ± 0.0	2.5 ± 2.3	0.8 ± 1.9	7.5 ± 1.9	0.0 ± 0.0	**70.0 ± 13.0**
Real-Data-E	0.0 ± 0.0	8.0 ± 3.0	0.0 ± 0.0	**21.3 ± 8.0**	0.0 ± 0.0	**32.7 ± 9.5**
Average	0.1	5.0	5.2	22.3	8.2	**45.7**

FORL (MAX), also referred to as MAX for brevity, can fail when the forecast mean of $D_{timeseries}$ is biased, misleading it to select a candidate from a geometrically distant mode that appears more likely under an inaccurate forecast. In contrast DCM, succeeds because its state estimation is not dependent on the forecast's mean, but on a non-parametric search for the forecast sample with the highest score (minimum dimension-wise distance). Hence, DCM's prediction error is governed by the accuracy of the forecast sample in $D_{timeseries}$ with the best score. Empirically this yields lower maximum and mean errors compared to MAX. The timeseries forecaster can generate a large set of samples that can be systematically biased, which is why we observe that the DQL+LAG-$\bar{s}$ and DQL+LAG-$\tilde{s}$ also have high maximum prediction error in Table 7.7. FORL (DCM) outperforms FORL (MAX) and FORL-DM in terms of maximum and mean error, demonstrating stability.

Table 7.6. Normalized scores for DCM candidate selection and the baselines.

Maze2d-M	DM-FS-$\bar{s}$	DM-FS-$\tilde{s}$	FORL (MAX)	FORL(KDE)	FORL(DCM)
Real-Data-A	40.6 ± 9.6	**73.7 ± 8.4**	41.2 ± 8.2	49.7 ± 5.8	**63.3 ± 6.7**
Real-Data-B	59.8 ± 16.9	59.3 ± 20.0	58.9 ± 14.1	**96.2 ± 14.0**	66.5 ± 18.2
Real-Data-C	64.9 ± 17.0	67.8 ± 17.7	66.1 ± 16.4	**94.9 ± 13.8**	**86.3 ± 15.7**
Real-Data-D	45.1 ± 21.6	45.0 ± 19.5	44.4 ± 21.6	**86.9 ± 14.5**	**103.4 ± 11.9**
Real-Data-E	12.5 ± 6.3	20.8 ± 7.2	11.8 ± 5.5	**49.3 ± 16.5**	**51.2 ± 13.7**
Average	44.6	53.3	44.5	**75.4**	74.1
Maze2D-L	DM-FS-$\bar{s}$	DM-FS-$\tilde{s}$	FORL (MAX)	FORL(KDE)	FORL(DCM)
Real-Data-A	11.9 ± 5.5	20.8 ± 6.2	11.1 ± 2.3	6.6 ± 2.2	**42.9 ± 4.1**
Real-Data-B	**27.9 ± 14.7**	**25.1 ± 11.7**	**28.3 ± 7.1**	20.9 ± 7.2	**34.9 ± 9.2**
Real-Data-C	34.6 ± 6.8	32.4 ± 5.8	**34.6 ± 13.6**	36.0 ± 7.3	**45.6 ± 4.1**
Real-Data-D	16.4 ± 12.0	15.5 ± 9.0	18.4 ± 9.9	11.8 ± 3.2	**58.4 ± 6.5**
Real-Data-E	**8.4 ± 5.0**	**7.9 ± 3.9**	**9.2 ± 6.1**	**5.7 ± 5.1**	**12.0 ± 9.9**
Average	19.8	20.3	20.3	16.2	**38.8**
AntM-U-D	DM-FS-$\bar{s}$	DM-FS-$\tilde{s}$	FORL (MAX)	FORL(KDE)	FORL(DCM)
Real-Data-A	56.0 ± 13.3	58.7 ± 7.4	59.7 ± 9.7	**80.3 ± 4.1**	65.3 ± 8.7
Real-Data-B	**69.2 ± 11.3**	**76.7 ± 11.3**	65.0 ± 10.5	**82.5 ± 8.0**	**74.2 ± 10.8**
Real-Data-C	**72.1 ± 5.4**	**75.8 ± 4.8**	**76.2 ± 5.4**	**67.9 ± 7.9**	**78.8 ± 8.5**
Real-Data-D	65.0 ± 11.3	61.7 ± 15.1	63.3 ± 6.8	**85.0 ± 4.8**	**75.8 ± 8.0**
Real-Data-E	**76.7 ± 16.2**	**74.7 ± 18.3**	**72.0 ± 15.0**	**75.3 ± 8.0**	**81.3 ± 6.9**
Average	67.8	69.5	67.2	**78.2**	75.1
AntM-M-D	DM-FS-$\bar{s}$	DM-FS-$\tilde{s}$	FORL (MAX)	FORL(KDE)	FORL(DCM)
Real-Data-A	39.0 ± 9.5	27.3 ± 7.5	36.0 ± 4.8	**62.7 ± 6.3**	44.0 ± 7.9
Real-Data-B	34.2 ± 9.0	35.8 ± 8.1	36.7 ± 7.5	**59.2 ± 8.0**	**55.8 ± 7.0**
Real-Data-C	36.2 ± 2.8	34.6 ± 2.4	37.1 ± 3.7	**49.2 ± 10.1**	**52.9 ± 9.5**
Real-Data-D	25.0 ± 6.6	17.5 ± 3.5	37.5 ± 6.6	**77.5 ± 8.6**	64.2 ± 8.6
Real-Data-E	28.7 ± 3.0	25.3 ± 5.1	26.7 ± 7.1	**36.7 ± 5.8**	26.7 ± 4.7
Average	32.6	28.1	34.8	**57.1**	48.7
AntM-L-D	DM-FS-$\bar{s}$	DM-FS-$\tilde{s}$	FORL (MAX)	FORL(KDE)	FORL(DCM)
Real-Data-A	**25.0 ± 7.9**	21.0 ± 3.5	21.7 ± 7.9	21.7 ± 8.3	**34.3 ± 5.7**
Real-Data-B	25.0 ± 5.1	**30.0 ± 7.5**	20.0 ± 6.8	**47.5 ± 15.2**	**46.7 ± 11.9**
Real-Data-C	25.8 ± 4.6	21.7 ± 2.4	23.8 ± 6.4	**40.0 ± 7.6**	**33.8 ± 6.8**
Real-Data-D	14.2 ± 7.0	15.8 ± 5.4	21.7 ± 7.5	**35.0 ± 14.3**	**46.7 ± 12.6**
Real-Data-E	**21.3 ± 8.4**	**22.0 ± 7.7**	**20.7 ± 4.9**	8.0 ± 3.0	**11.3 ± 7.3**
Average	22.3	22.1	21.6	30.4	**34.6**

Table 7.7. Comparison of algorithm performance on error metrics.

Algorithm	**Minimum Error**	**Maximum Error**	**Mean Error**
FORL-DCM	0.02	**2.40**	**0.87 ± 0.60**
FORL-MAX	**0.01**	9.33	2.05 ± 1.74
DQL	2.26	11.30	5.51 ± 2.13
FORL-DM *(no past offsets)*	**0.01**	9.94	3.68 ± 2.25
DQL+LAG-$\bar{s}$	0.04	6.28	1.76 ± 1.13
DQL+LAG-$\tilde{s}$	0.05	6.56	1.87 ± 1.19
DMBP+LAG	0.03	6.28	1.69 ± 1.11

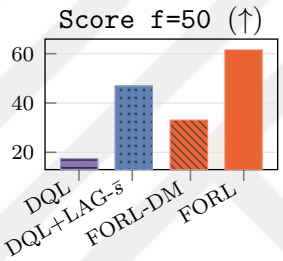


Figure 7.13. Intra-Episode Performance

7.3.4.3. Intra-episode Non-stationarity. In this setting, the agent is subject to an unknown, time-dependent offset. This is modeled as an intra-episode non-stationarity, where the offset value dynamically changes every $f = 50$ timesteps. The agent must operate under this changing offset, as the true values are only revealed after the episode terminates.

The Zero-shot forecasting foundation module can generate samples before the episode begins. Our diffusion model (FORL-DM) itself does not rely on the forecasts of the foundation module and only tracks observation changes and actions which are invariant to the offsets. The DCM can adaptively fuse information from both models at each timestep without requiring any hyperparameters. Table 7.8 and Fig. 7.13 show the average scores for DQL vs. FORL-DM and DQL+LAG-$\bar{s}$ vs. FORL. Among the algorithms that do not use any past ground truth offsets DQL and FORL-DM, only

using the diffusion model of FORL significantly increases performance. When we have access to past offsets, FORL obtains a superior performance. This shows that our method covers both cases, when information is available and not available, even when offsets are not constant throughout the episode.

Table 7.8. Intra-episode non-stationarity results with f=50.

Maze2d-M	DQL	FORL-DM	DQL+LAG-$\bar{s}$	FORL
Real-Data-A	31.6 ± 1.6	**55.7 ± 8.7**	29.7 ± 9.2	17.8 ± 4.4
Real-Data-B	-4.7 ± 0.2	42.2 ± 9.3	63.0 ± 9.2	**100.1 ± 8.5**
Real-Data-C	-4.7 ± 0.2	39.2 ± 7.2	**88.4 ± 9.4**	**81.0 ± 8.3**
Real-Data-D	25.9 ± 5.3	37.0 ± 8.4	43.1 ± 8.1	**101.0 ± 7.1**
Real-Data-E	-4.7 ± 0.2	**57.0 ± 10.2**	5.8 ± 2.2	**66.7 ± 8.3**
Average	8.7	46.2	46.0	**73.3**
AntM-U-D	DQL	FORL-DM	DQL+LAG-$\bar{s}$	FORL
Real-Data-A	51.0 ± 7.4	47.2 ± 10.6	**70.6 ± 14.5**	**78.2 ± 8.0**
Real-Data-B	**63.8 ± 10.2**	**51.2 ± 10.3**	26.0 ± 16.8	**61.4 ± 9.7**
Real-Data-C	61.0 ± 9.3	53.0 ± 6.6	**91.2 ± 2.5**	**89.0 ± 5.1**
Real-Data-D	17.6 ± 5.9	53.8 ± 5.7	78.0 ± 4.4	**90.0 ± 2.9**
Real-Data-E	0.2 ± 0.4	51.0 ± 11.6	**80.6 ± 9.7**	**85.6 ± 5.5**
Average	38.7	51.2	69.3	**80.8**
AntM-M-D	DQL	FORL-DM	DQL+LAG-$\bar{s}$	FORL
Real-Data-A	42.8 ± 6.9	7.2 ± 1.8	**50.4 ± 2.9**	**54.8 ± 5.2**
Real-Data-B	5.0 ± 3.1	37.4 ± 3.8	**41.2 ± 4.8**	**47.2 ± 4.6**
Real-Data-C	12.0 ± 5.1	26.8 ± 2.4	**71.2 ± 2.9**	61.4 ± 4.5
Real-Data-D	24.0 ± 4.7	30.4 ± 2.9	**61.6 ± 5.0**	**70.2 ± 6.6**
Real-Data-E	2.2 ± 2.3	21.0 ± 3.3	**33.0 ± 5.0**	**37.0 ± 3.4**
Average	17.2	24.6	51.5	**54.1**
AntM-L-D	DQL	FORL-DM	DQL+LAG-$\bar{s}$	FORL
Real-Data-A	13.0 ± 5.5	5.8 ± 2.6	16.0 ± 3.8	**27.4 ± 3.6**
Real-Data-B	1.2 ± 0.8	11.4 ± 4.9	19.4 ± 4.6	**50.2 ± 9.7**
Real-Data-C	4.8 ± 2.9	7.2 ± 0.8	31.8 ± 4.4	**48.6 ± 4.3**
Real-Data-D	4.6 ± 1.3	11.0 ± 2.5	25.2 ± 2.9	**48.2 ± 6.2**
Real-Data-E	2.4 ± 1.5	**13.6 ± 4.8**	**12.8 ± 3.4**	**13.4 ± 2.2**
Average	5.2	9.8	21.0	**37.6**

7.3.4.4. Offset Scaling Results. We conduct an analysis to quantify FORL's sensitivity to different levels of non-stationarity. We scale the offset magnitude from 0 (standard D4RL offline RL environment [21] used in training) to 1 (our evaluation environment) using scaling factors $\boldsymbol{c} \in \{0, 0.25, 0.5, 0.75, 1.0\}$, and report performance over five random

seeds across five environments in Maze2d and Antmaze in Figure 7.15 each with five time-series datasets. This analysis delivers two key insights: first, FORL matches baseline performance on the stationary offline-RL dataset on which it was trained; second, it maintains superior results throughout the full range of non-stationarity magnitudes. Crucially, FORL's performance degrades gracefully as the magnitude of offsets increases in Figure 7.14, whereas DQL (without forecasting) suffers steep drops. Furthermore, both DMBP+LAG and DQL+LAG-$\bar{s}$ decline similarly. DMBP+LAG degrades slightly more gracefully than DQL+LAG-$\bar{s}$ in Maze2D-L and AntM-U-D in offline RL environment specific plots (Figure 7.15).

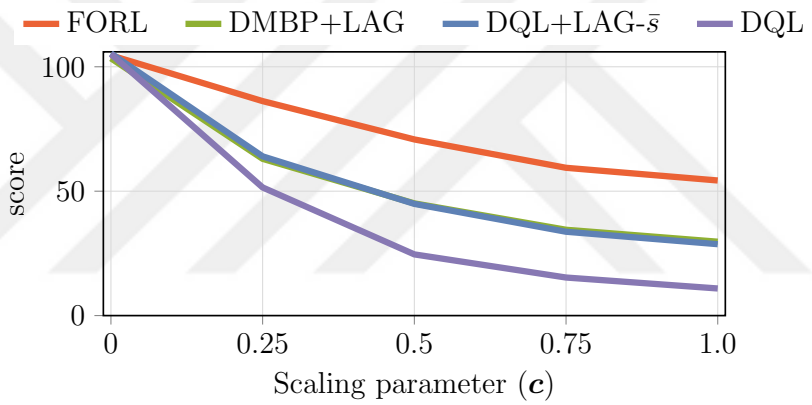


Figure 7.14. Impact of offset scaling on average normalized scores.

7.3.4.5. Policy-Agnostic. In the Maze2D-L and Maze2d-M experiments (see Table 7.10 and Table 7.9), we use Robust Offline RL (RORL) [77], and TD3+BC [23] offline RL algorithms instead of DQL [29], to analyze the effect of offline RL policy choice during evaluation. More specifically, Table 7.9 and Table 7.10 present the mean and standard deviation of normalized scores for the proposed FORL framework and baseline algorithms evaluated on various time-series and D4RL datasets. Algorithms are grouped by their underlying policies, TD3 with Behavior Cloning (TD3+BC) [23] and Robust Offline Reinforcement Learning (RORL) [77], to highlight that performance variations stem from the algorithms themselves rather than the policies employed. Suffixes -T and -R denote the use of TD3+BC and RORL policies, respectively.

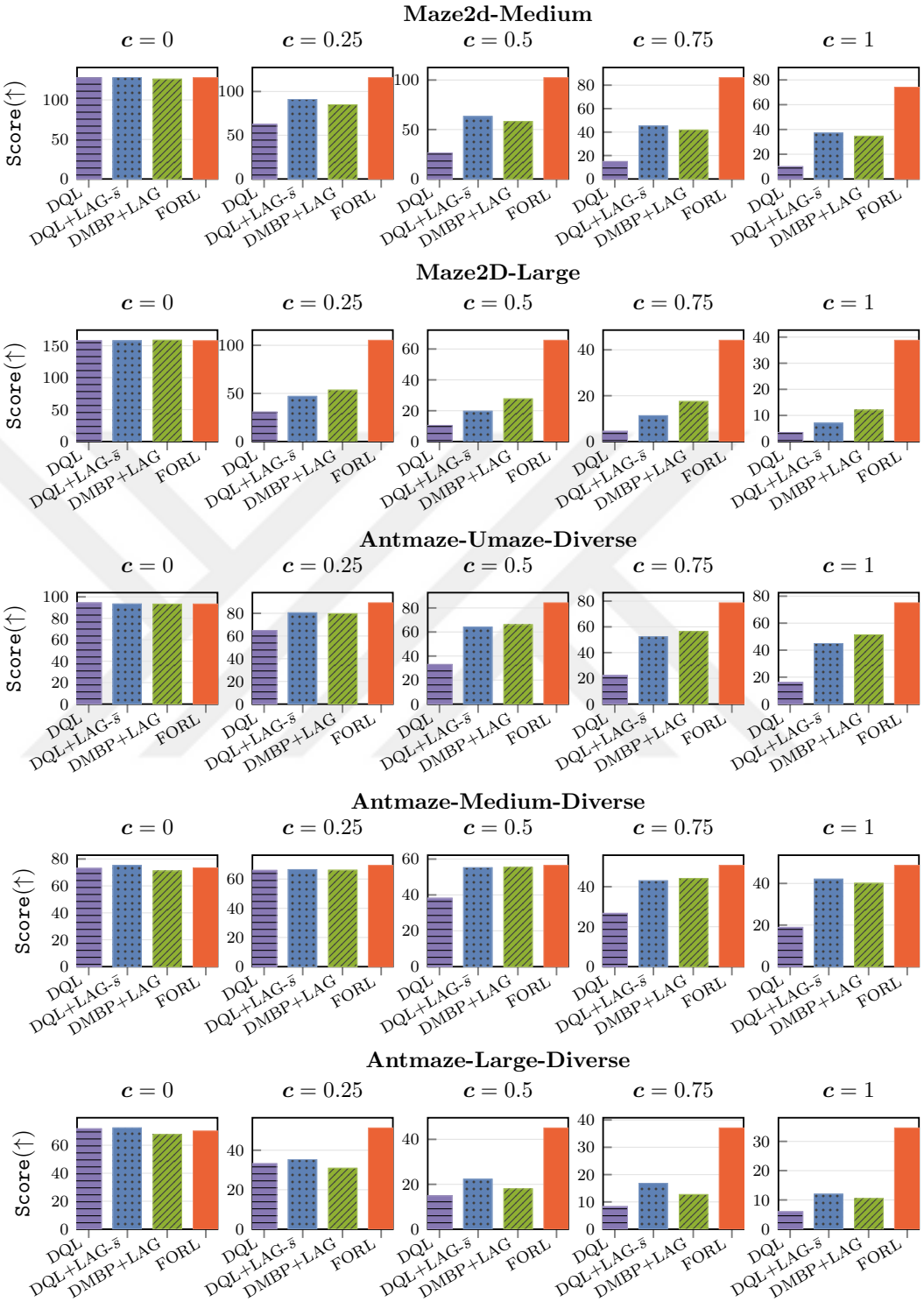


Figure 7.15. Bar plots for offset scaling factors in navigation environments.

Table 7.9. Normalized scores for FORL and baselines with TD3BC on maze environments.

Maze2d-M	TD3+BC	TD3BC+LAG-$\bar{s}$	DMBP+LAG-T	FORL-T
Real-Data-A	**37.4 ± 9.5**	16.2 ± 5.1	16.2 ± 7.3	22.1 ± 6.6
Real-Data-B	3.6 ± 2.3	**6.1 ± 4.4**	**14.4 ± 8.1**	**28.6 ± 19.0**
Real-Data-C	-2.3 ± 1.3	**30.0 ± 10.0**	**19.3 ± 6.2**	**24.5 ± 10.2**
Real-Data-D	6.3 ± 3.4	15.5 ± 3.8	12.2 ± 2.6	**38.7 ± 13.4**
Real-Data-E	-3.7 ± 1.5	**9.7 ± 5.2**	**11.5 ± 6.6**	**15.1 ± 8.8**
Average	8.3	15.5	14.7	**25.8**
Maze2D-L	TD3+BC	TD3BC+LAG-$\bar{s}$	DMBP+LAG-T	FORL (ours)-T
Real-Data-A	**14.7 ± 5.7**	2.5 ± 2.7	4.8 ± 3.9	**20.7 ± 3.5**
Real-Data-B	-0.9 ± 2.0	4.6 ± 8.9	11.7 ± 12.6	**56.8 ± 14.4**
Real-Data-C	0.8 ± 1.9	21.6 ± 8.4	29.5 ± 13.7	**56.9 ± 14.6**
Real-Data-D	2.5 ± 4.4	14.9 ± 4.3	14.4 ± 6.8	**29.5 ± 10.3**
Real-Data-E	-2.3 ± 0.2	1.0 ± 4.2	2.0 ± 3.9	**8.0 ± 4.2**
Average	3.0	8.9	12.5	**34.4**

Table 7.10. Normalized scores for FORL and baselines with RORL on maze environments.

Maze2d-M	RORL	RORL+LAG-$\bar{s}$	DMBP+LAG-R	FORL-R
Real-Data-A	**80.7 ± 14.2**	57.9 ± 8.6	47.8 ± 10.2	52.7 ± 5.0
Real-Data-B	37.8 ± 7.5	**85.9 ± 19.8**	**91.0 ± 21.6**	**109.6 ± 19.5**
Real-Data-C	33.7 ± 4.6	89.2 ± 15.8	93.4 ± 16.3	**125.4 ± 14.5**
Real-Data-D	37.0 ± 17.0	71.2 ± 27.2	77.7 ± 26.2	**136.1 ± 11.9**
Real-Data-E	**60.9 ± 13.5**	10.0 ± 7.3	14.2 ± 8.6	**61.2 ± 14.6**
Average	50.0	62.8	64.8	**97.0**
Maze2D-L	RORL	RORL+LAG-$\bar{s}$	DMBP+LAG-R	FORL-R
Real-Data-A	12.2 ± 2.3	13.0 ± 2.0	4.3 ± 5.0	**56.9 ± 3.0**
Real-Data-B	1.2 ± 5.5	13.1 ± 14.7	28.5 ± 11.7	**98.5 ± 19.0**
Real-Data-C	3.1 ± 0.9	60.6 ± 8.5	39.4 ± 6.1	**139.0 ± 15.1**
Real-Data-D	-1.6 ± 0.7	17.9 ± 6.8	17.5 ± 6.0	**33.1 ± 2.3**
Real-Data-E	-0.9 ± 2.0	3.3 ± 4.4	2.2 ± 4.5	**32.2 ± 15.3**
Average	2.8	21.6	18.4	**71.9**

As outlined in Section 7.3.1.2, RORL+LAG-$\bar{s}$, and TD3BC+LAG-$\bar{s}$ extend RORL and TD3+BC by using the sample mean of the forecasted states $\{\hat{s}_t\}_l$ at each time step with the constant per-episode predicted b^j. Results indicate that using a robust offline RL algorithm, RORL, during training significantly increases performance (71.9) compared to IQL (29.1), DQL (38.8) and TD3+BC (34.4) at test time when used with FORL, no increase when used alone, and a marginal increase with Lag-Llama

and DMBP+LAG. Moreover, while extending TD3+BC with the forecasting module (TD3BC+LAG-$\bar{s}$), and a combination of forecasting module and diffusion model-based predictor (DMBP+LAG-T) improves performance, FORL-T achieves more robust performance across a diverse range of non-stationarities.

Table 7.11. Normalized scores for FORL framework and the baselines with IQL policy.

Maze2d-M	IQL	IQL+LAG-$\bar{s}$	DMBP+LAG-I	FORL (ours)-I
Real-Data-A	**39.5 ± 7.8**	16.0 ± 4.9	12.2 ± 7.7	19.5 ± 2.9
Real-Data-B	3.7 ± 7.3	13.1 ± 8.7	14.0 ± 9.5	**32.3 ± 13.1**
Real-Data-C	-1.1 ± 2.5	**33.1 ± 11.9**	**28.0 ± 10.9**	**31.8 ± 9.9**
Real-Data-D	10.1 ± 2.6	12.4 ± 5.6	12.7 ± 9.0	**40.0 ± 10.1**
Real-Data-E	-4.5 ± 0.2	12.4 ± 5.2	9.9 ± 2.9	**24.0 ± 8.4**
Average	9.6	17.4	15.4	**29.5**
Maze2D-L	IQL	IQL+LAG-$\bar{s}$	DMBP+LAG-I	FORL (ours)-I
Real-Data-A	**16.2 ± 6.2**	12.5 ± 5.5	6.0 ± 4.7	**24.3 ± 9.4**
Real-Data-B	-0.6 ± 2.6	3.3 ± 7.7	13.5 ± 10.5	**42.8 ± 9.8**
Real-Data-C	0.1 ± 1.5	27.8 ± 13.1	27.9 ± 5.5	**46.7 ± 9.2**
Real-Data-D	1.2 ± 3.7	11.2 ± 7.4	8.0 ± 7.5	**23.9 ± 8.4**
Real-Data-E	-2.3 ± 0.2	-1.5 ± 1.1	**1.5 ± 4.1**	**7.9 ± 7.2**
Average	2.9	10.7	11.4	**29.1**
AntM-U-D	IQL	IQL+LAG-$\bar{s}$	DMBP+LAG-I	FORL (ours)-I
Real-Data-A	23.0 ± 1.4	**43.7 ± 6.1**	**44.0 ± 9.6**	**50.3 ± 11.0**
Real-Data-B	21.7 ± 5.4	**55.0 ± 6.2**	**61.7 ± 9.5**	**70.8 ± 13.2**
Real-Data-C	20.0 ± 3.2	45.4 ± 4.0	58.3 ± 5.1	**73.8 ± 5.4**
Real-Data-D	6.7 ± 5.6	26.7 ± 8.6	29.2 ± 6.6	**77.5 ± 4.8**
Real-Data-E	8.0 ± 8.7	**60.0 ± 11.8**	**56.0 ± 9.8**	**68.0 ± 13.0**
Average	15.9	46.1	49.8	**68.1**
AntM-M-D	IQL	IQL+LAG-$\bar{s}$	DMBP+LAG-I	FORL (ours)-I
Real-Data-A	**18.3 ± 5.5**	**21.0 ± 5.8**	**24.7 ± 4.0**	17.7 ± 3.5
Real-Data-B	**7.5 ± 3.5**	**7.5 ± 5.4**	**8.3 ± 7.8**	**11.7 ± 7.5**
Real-Data-C	2.5 ± 3.7	**18.8 ± 6.1**	**15.8 ± 4.8**	**16.2 ± 7.4**
Real-Data-D	8.3 ± 4.2	**12.5 ± 5.9**	**12.5 ± 7.8**	**16.7 ± 2.9**
Real-Data-E	3.3 ± 4.1	**22.7 ± 8.0**	**22.7 ± 12.3**	**22.7 ± 6.4**
Average	8.0	16.5	16.8	**17.0**

Although using RORL as our base policy improved performance in Maze2d environments, FORL significantly outperforms both its naive usage, the extension with our forecasting module RORL+LAG-$\bar{s}$, and DMBP+LAG-R in Table 7.10. These results demonstrate that policies designed to be robust to sensor noise or adversarial

attacks fail to cope with evolving observation functions that introduce non-stationarity into the environment. Table 7.11 further shows that FORL-I outperforms the baselines IQL, IQL+LAG-$\bar{s}$, and DMBP+LAG-I across all environments. Similarly, Tables 7.2 and 7.5 show FORL (also referred to as FORL (DCM)) outperforms the baselines, when the baselines use FQL policy.

7.3.4.6. Preliminary Results with Uniform Scaling and Bias. We use the fourth series in each time-series domain to do isotropic scaling for the dimensions affected by a non-stationarity using a scaling factor of $\boldsymbol{d} = 0.5$ with bias coming from the first two series. We apply the feature scaling to time-series data with $x' = 1 - \boldsymbol{d} + \boldsymbol{d} \cdot \exp\left(\frac{x - \bar{x}}{2 \cdot (\max(x) - \min(x))}\right)$. The offset scaling for the bias uses $\boldsymbol{c} = 1$, which is the standard value in our experiments. As in the other ablations with scaling offsets, we use the DQL policy. In the Maze2D-L environments across all non-stationarities, FORL achieved a score of 34.0, and outperformed the baselines DQL+LAG-$\bar{s}$ (7.7) and DQL (1.8) on the same random seed. A large-scale analysis of more general transformations is left for future work.

7.3.5. Sensitivity of FORL to Forecasting Errors

To analyze the sensitivity of FORL to forecasting errors, we compare its performance against DQL+LAG-$\bar{s}$, which only uses the forecaster's predictions. Table 7.12 presents the average prediction error across all datasets in Antmaze and Maze2d environments with 5 seeds.

Table 7.12. Sensitivity analysis of FORL to forecasting errors.

Dataset	DQL+LAG-$\bar{s}$	FORL	Error Reduction
Real-Data-A	4.56	**3.32**	27.0%
Real-Data-B	3.66	**2.29**	37.4%
Real-Data-C	3.0	**2.69**	10.2%
Real-Data-D	4.29	**1.87**	**56.5%**
Real-Data-E	5.45	**5.21**	4.3%

In all datasets, FORL outperforms the DQL+MeanLag baseline. FORL achieves its greatest impact on moderately challenging forecasts (a 56.5% error reduction on Real-Data-D)). Its behavior at the extremes further demonstrates its robustness:

- FORL still refines the best forecast by 10.2% (Real-Data-C)
- FORL improves the worst forecast by 4.3% (Real-Data-E).

7.3.6. State Prediction Accuracy

To evaluate the state-prediction accuracy of our FORL framework, we compare it against DQL+LAG-$\bar{s}$. For each method, we report the mean ℓ_2 error between the true state s_t and the predicted state $\tilde{s}_t$ obtained during evaluation for diffusion generated sample size of 32.

We present the average prediction errors in Table 7.13; the table is organized as follows:

- Each row corresponds to the state estimation algorithm used at test time to generate states $\tilde{s}_t$, which are then provided to the policy to select actions.
- Each column corresponds to a method whose state estimates are evaluated on that same rollout.

The entry at row i, column j is the mean ℓ_2 error when method m_i is used in the environment, but predictions are produced by method m_j. When $i = j$, this entry measures the self-prediction error of each method; when $i \neq j$, it measures the error under an alternate method.

Across all method pairs, FORL achieves lower mean ℓ_2 errors, even in off-diagonal evaluations, demonstrating its superior state-prediction performance compared to DQL+LAG-$\bar{s}$. These findings are consistent with the normalized environment scores in Table 7.1.

Table 7.13. Comparison of counterfactual prediction errors.

Maze2d-M	DQL+LAG-$\bar{s}$	**FORL**
DQL+LAG-$\bar{s}$	1.68 ± 0.46	$\mathbf{1.35 \pm 0.48}$
FORL	1.39 ± 0.49	$\mathbf{1.25 \pm 0.43}$
Maze2D-L	DQL+LAG-$\bar{s}$	**FORL**
DQL+LAG-$\bar{s}$	2.37 ± 0.5	$\mathbf{1.63 \pm 0.57}$
FORL	1.91 ± 0.44	$\mathbf{1.39 \pm 0.62}$
AntM-L-D	DQL+LAG-$\bar{s}$	**FORL**
DQL+LAG-$\bar{s}$	8.07 ± 1.83	$\mathbf{5.33 \pm 2.17}$
FORL	7.16 ± 1.71	$\mathbf{5.89 \pm 2.96}$
AntM-M-D	DQL+LAG-$\bar{s}$	**FORL**
DQL+LAG-$\bar{s}$	5.33 ± 1.15	$\mathbf{4.75 \pm 1.98}$
FORL	4.94 ± 1.17	$\mathbf{4.74 \pm 1.63}$
AntM-U-D	DQL+LAG-$\bar{s}$	**FORL**
DQL+LAG-$\bar{s}$	3.51 ± 0.53	$\mathbf{2.04 \pm 0.47}$
FORL	3.27 ± 0.61	$\mathbf{2.31 \pm 0.51}$

7.3.7. Sensitivity to Diffusion-generated Sample Size

FORL (MAX) uses candidate states predicted by the diffusion model in FORL framework and Lag-Llama but uses maximum likelihood instead of DCM. Given the multimodal nature of the candidate state distributions, we conduct a sensitivity analysis on the number of denoiser samples, a shared hyperparameter for both FORL (MAX) and FORL. We report results averaged over 5 random seeds across 25 tasks (Antmaze and Maze2d with `real-data-A,B,C,D,E`). The results in Figure 7.16 indicate that the diffusion model's performance remains consistent across varying sample sizes, demonstrating robustness to the number of candidate states generated. For reference, the DMBP algorithm uses 50 denoiser samples.

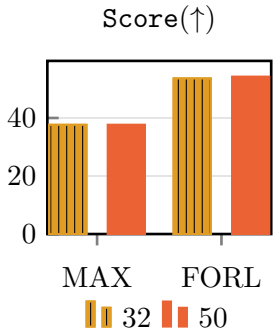


Figure 7.16. Scores for different diffusion generated sample sizes.

7.4. Conclusion

We introduce Forecasting in Non-stationary Offline RL (FORL), a novel framework designed to be robust to passive non-stationarities that arise at test time. This is crucial when an agent, trained on an offline RL dataset, is deployed in a non-stationary environment or when the environment begins to exhibit partial observability due to unknown, time-varying factors. FORL leverages diffusion probabilistic models and zero-shot time series foundation models to correct unknown offsets in observations thereby enhancing the adaptability of learned policies. Our empirical results across diverse time-series datasets, OGBench [32] and D4RL [21] benchmarks demonstrate that FORL not only bridges the gap between time-series forecasting and non-stationary offline RL but also consistently outperforms the baselines. Our approach is currently limited by assuming additive perturbations. For future work, we plan to extend our work to more general observation transformations.

8. DISCUSSION

This thesis addresses the challenge of developing agents capable of navigating complex futures, with the ambitious aim of operating beyond the limitations of human cognition. Our approach involved identifying and providing solutions for fundamental gaps that prevent agents from being robust and adaptive when the evolution of tasks is non-trivial. The contributions of this thesis provided a set of solutions to four key pieces of this puzzle:

- sample efficient adaptation from few samples when reward information is missing (UMCNP) (Chapter 4, [1]);
- skill transfer across morphologically different agents with episodic return progress leveraging a bidirectional progressive neural network (ERP-BPNN) (Chapter 5, [28]);
- generalization to out-of-distribution states in offline RL datasets (SRDP) (Chapter 6, [2]);
- anticipative decision making with time-varying, non-stationary observation functions (FORL) (Chapter 7, [35]).

UMCNP utilizes a data buffer collected during meta-RL training. A key advantage of this framework is its flexibility; it allows for the refinement of the environment's world model using previously collected data, bypassing the need to retrain the computationally expensive meta-RL agent. This circumvents the instability of online meta-RL training and tedious hyperparameter tuning. Hence, if a deployed system requires fewer samples than initially planned, our framework offers a practical approach for few-shot learning scenarios benefiting from a more stable training in a supervised manner.

Another key area of flexibility is that the trained UMCNP model can generate synthetic trajectories for the meta-RL policy adaptation from a variable number of transitions or a single rollout. In the walker environments we outperform the baselines

for ten transitions and one trajectory. Similarly, in the cartpole experiments, we surpassed the baselines using five and fifty transitions, and we did not observe a significant performance difference between these two cases.

Although UMCNP currently learns the transition dynamics for this setting, it is also possible to learn to infer the task's context when the reward function changes. As future work, our framework could be extended to predict both the next state and the reward, provided the reward signals are available at test time.

With ERP-BPNN, we introduce a novel intrinsic motivation signal, episodic return progress, and a bidirectional progressive neural network for reinforcement learning. This architecture performs effectively in multitask reinforcement learning, tailored for tasks with different action spaces. An important future work that remains is to extend our architecture to support heterogeneous learning, where each task may require a different type of learning mechanism. For instance, one module may support supervised learning while others may handle reinforcement and unsupervised learning tasks. Another direction involves enhancing the framework to accommodate more complex lifelong RL scenarios, including real-world applications. This can be achieved by increasing the number of tasks and improving the bidirectional skill transfer mechanisms to enhance learning capacity and generalization ability.

Given SRDP's foundation in representation learning, integrating a vision module appears feasible. To reduce inference time in diffusion policies, incorporating recent flow matching policies such as Flow Q-learning [188] presents a viable option. Extending SRDP to operate with trajectory-level diffusion probabilistic models also remains an avenue for future investigation.

We believe the general non-stationary RL formulation [8] to be crucial, yet progress requires combining RL with different paradigms such as forecasting and generative modeling. FORL aims to establish a foundation for systems that integrate the predictive capabilities of zero-shot probabilistic time-series foundation models to guide decision-

making. This work establishes a strong foundation, highlighting several clear and promising directions for future research. These directions can be broadly categorized by expanding the framework's conceptual scope and implementing targeted component enhancements.

On a conceptual level, FORL's scope can be significantly widened. We currently address passive drivers of non-stationarity, handling hybrid and active drivers are also natural next steps. Moreover enabling agents to continue learning post-deployment is essential, when the scope of non-stationarity includes other MDP components. Such capabilities would foster truly proactive and adaptive agents by continually leveraging new information streams.

At the component level, several targeted enhancements, such as adaptively managing and tracking forecasting errors could improve performance and robustness. Anticipating these errors additionally calls for OOD-detection mechanisms; when a high uncertainty is detected, our state predictions can become more conservative or auxiliary models can be dynamically activated to fuse additional sources of information. Furthermore, the additive observation function assumption can be relaxed. While this assumption enabled clear problem formulation and rigorous analysis in our initial setting, more sophisticated systems will need to accommodate more complex observation transformations by progressing from linear to nonlinear transformations. Finally, FORL's diffusion model component (FORL-DM), while performant, can exhibit high prediction error because the candidate states can be multimodal. The current fixed window size exacerbates this issue. A future direction is to use dynamic-length observation change and action trajectories rather than a fixed window, which could improve the accuracy of the belief states. This could also be beneficial for long-horizon tasks, as the agent refines its belief more over time.

Modeling the world as changing calls for benchmarking against human performance in non-stationary real-world settings and game environments that explicitly test anticipation of change. In finance, for example, an agent's performance could be

measured against that of humans on matched information sets and time periods, navigating the same volatile, non-stationary market dynamics. We can also design games reflecting passive, hybrid, or active drivers of non-stationarity, where dynamics are partially predictable. Large-scale human studies using these games or financial testbeds can quantify what people can anticipate and how they update their decisions under change. Together, these benchmarks can score foresight in sequential decision-making, setting the stage for the shift toward anticipatory AI agents.

Broadly, this thesis advocates for a paradigm shift in how we build sequential decision-making systems. While many traditional RL methods assume the environment to be stationary or adopt reactive strategies to handle non-stationarity, the contributions presented here represent a move toward an anticipatory and proactive AI. By explicitly targeting adaptation (UMCNP), transfer (ERP-BPNN), robustness (SRDP), and forecasting (FORL), we aim to lay the algorithmic foundations for agents that do not merely react to the present or expect repetitive and trivial task evolutions, but instead actively model, anticipate, and shape an uncertain future.

9. CONCLUSION

The central aim of this thesis is to develop artificial intelligence systems capable of anticipative reasoning and decision-making in complex, uncertain, and non-stationary environments. A key barrier to making robust decisions for realistic futures, particularly when leveraging offline RL datasets, stems from an unrealistic treatment of the evolution of non-stationarity. Current research often confines this phenomenon to predefined sinusoidal functions, simple Gaussian noise, or constrained adversarial attacks.

Recognizing the inherently temporal nature of the non-stationarity patterns studied in this thesis, we adopt a data-driven approach that models these changes using real-world time-series data, in line with the data-centric, large-scale training paradigms underlying recent successes in Large Language Models [206, 207, 208, 209, 210, 211]. Our approach focuses on adaptive learning from large-scale interaction datasets, reflecting the high-stakes and data-rich nature of many real-world dynamic systems.

Our contributions progressively built towards a set of capabilities required for an anticipatory agent. In Chapter 4, we developed a hybrid meta-RL framework that enables sample-efficient adaptation to new tasks when reward signals are unavailable, learning meta-world models from a static dataset of experiences collected by a meta-RL agent. In Chapter 5 (ERP-BPNN), we introduced a human-inspired multi-task learning architecture that, allows for autonomous task switching and bidirectional skill transfer among morphologically different agents. In Chapter 6 (SRDP), we tackled the problem of out-of-distribution generalization in offline RL, demonstrating that incorporating state reconstruction into diffusion policies leads to more robust policies. Finally, in Chapter 7 (FORL), we leveraged conditional diffusion models to unify offline RL with zero-shot probabilistic forecasting foundation models, creating a framework that can handle environments prone to realistic and time-varying sources of non-stationarities.

REFERENCES

1. Ada, S. E. and E. Ugur, "Unsupervised Meta-Testing With Conditional Neural Processes for Hybrid Meta-Reinforcement Learning", *IEEE Robotics and Automation Letters*, Vol. 9, No. 10, pp. 8427–8434, 2024.

2. Ada, S. E., E. Oztop and E. Ugur, "Diffusion Policies for Out-of-Distribution Generalization in Offline Reinforcement Learning", *IEEE Robotics and Automation Letters*, Vol. 9, No. 4, pp. 3116–3123, 2024.

3. Sutton, R. S. and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, Massachusetts, USA, 2018.

4. Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms", arXiv:1707.06347, 2017.

5. Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level Control Through Deep Reinforcement Learning", *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

6. Prudencio, R. F., M. R. Maximo and E. L. Colombini, "A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 35, No. 8, pp. 10237–10257, 2023.

7. Chandak, Y., *Reinforcement Learning for Non-stationary Problems*, Phd thesis, University of Massachusetts Amherst, 2022.

8. Khetarpal, K., M. Riemer, I. Rish and D. Precup, "Towards Continual Reinforcement Learning: A Review and Perspectives", *Journal of Artificial Intelligence Research*, Vol. 75, pp. 1401–1476, 2022.

9. Lee, H., Y. Ding, J. Lee, M. Jin, J. Lavaei and S. Sojoudi, "Tempo Adaptation in Non-stationary Reinforcement Learning", *Thirty-seventh Conference on Neural Information Processing Systems*, pp. 8253–8265, New Orleans, USA, 2023.

10. Chandak, Y., G. Theocharous, S. Shankar, M. White, S. Mahadevan and P. Thomas, "Optimizing for the Future in Non-stationary MDPs", *International Conference on Machine Learning*, pp. 1414–1425, PMLR, (Online), 2020.

11. Beck, J., R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn and S. Whiteson, "A Survey of Meta-Reinforcement Learning", arXiv:2301.08028, 2023.

12. Finn, C., P. Abbeel and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", *International Conference on Machine Learning*, pp. 1126–1135, PMLR, Sydney, Australia, 2017.

13. Yang, Y., K. Caluwaerts, A. Iscen, J. Tan and C. Finn, "NoRML: No-Reward Meta Learning", *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, Montreal QC, Canada, AAMAS '19, p. 323–331, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2019.

14. Vithayathil Varghese, N. and Q. H. Mahmoud, "A Survey of Multi-Task Deep Reinforcement Learning", *Electronics*, Vol. 9, No. 9, p. 1363, 2020.

15. Xu, Z., K. Wu, Z. Che, J. Tang and J. Ye, "Knowledge Transfer in Multi-Task Deep Reinforcement Learning for Continuous Control", H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin (Editors), *Advances in Neural Information Processing Systems*, Vol. 33, pp. 15146–15155, Curran Associates, Inc., (Online), 2020.

16. Teh, Y., V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess and R. Pascanu, "Distral: Robust Multitask Reinforcement Learning", *Advances*

in Neural Information Processing Systems, Vol. 30, pp. 4496–4506, Long Beach, USA, 2017.

17. Zhang, J., J. T. Springenberg, J. Boedecker and W. Burgard, "Deep Reinforcement Learning with Successor Features for Navigation Across Similar Environments", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2371–2378, Vancouver, Canada, 2017.

18. Yin, H. and S. J. Pan, "Knowledge Transfer for Deep Reinforcement Learning with Hierarchical Experience Replay", *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, USA, AAAI'17, p. 1640–1646, AAAI Press, 2017.

19. Liu, L. T., U. Dogan and K. Hofmann, "Decoding Multitask DQN in the World of Minecraft", *The 13th European Workshop on Reinforcement Learning (EWRL) 2016*, Barcelona, Spain, December 2016.

20. Roy, N., I. Posner, T. Barfoot, P. Beaudoin, Y. Bengio, J. Bohg, O. Brock, I. Depatie, D. Fox, D. Koditschek, T. Lozano-Perez, V. Mansinghka, C. Pal, B. Richards, D. Sadigh, S. Schaal, G. Sukhatme, D. Therien, M. Toussaint and M. Van de Panne, "From Machine Learning to Robotics: Challenges and Opportunities for Embodied Intelligence", arXiv:2110.15245, 2021.

21. Fu, J., A. Kumar, O. Nachum, G. Tucker and S. Levine, "D4RL: Datasets for Deep Data-driven Reinforcement Learning", arXiv:2004.07219, 2020.

22. Levine, S., A. Kumar, G. Tucker and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems", arXiv:2005.01643, 2020.

23. Fujimoto, S. and S. Gu, "A Minimalist Approach to Offline Reinforcement Learning", A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan (Editors), *Advances in Neural Information Processing Systems*, pp. 20132–20145, (Online), 2021.

24. Li, J., C. Tang, M. Tomizuka and W. Zhan, "Dealing with the Unknown: Pessimistic Offline Reinforcement Learning", *5th Annual Conference on Robot Learning*, pp. 1455–1464, London, UK, 2021.

25. Garnelo, M., D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende and S. A. Eslami, "Conditional Neural Processes", *International Conference on Machine Learning*, pp. 1704–1713, Stockholm, Sweden, 2018.

26. Todorov, E., T. Erez and Y. Tassa, "Mujoco: A Physics Engine for Model-based Control", *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, Algarve, Portugal, 2012.

27. Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, "OpenAI Gym", arXiv:1606.01540, 2016.

28. Ada, S. E., H. Say, E. Ugur and E. Oztop, "Bidirectional Progressive Neural Networks With Episodic Return Progress for Emergent Task Sequencing and Robotic Skill Transfer", *IEEE Access*, Vol. 12, pp. 69690–69699, 2024.

29. Wang, Z., J. J. Hunt and M. Zhou, "Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning", *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda, 2023.

30. Mark, M. S., A. Sharma, F. Tajwar, R. Rafailov, S. Levine and C. Finn, "Offline Retraining for Online RL: Decoupled Policy Learning to Mitigate Exploration Bias", arXiv:2310.08558, 2023.

31. Zhihe, Y. and Y. Xu, "DMBP: Diffusion Model-based Predictor for Robust Offline Reinforcement Learning Against State Observation Perturbations", *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024.

32. Park, S., K. Frans, B. Eysenbach and S. Levine, "OGBench: Benchmarking Offline Goal-Conditioned RL", *International Conference on Learning Representations*, Singapore, 2025.

33. Alexandrov, A., K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen and Y. Wang, "GluonTS: Probabilistic and Neural Time Series Modeling in Python", *Journal of Machine Learning Research*, Vol. 21, No. 116, pp. 1–6, 2020.

34. Alexandrov, A., K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen and Y. Wang, "GluonTS: Probabilistic Time Series Modeling in Python", arXiv:1906.05264, 2019.

35. Ada, S. E., G. Martius, E. Ugur and E. Oztop, "Forecasting in Offline Reinforcement Learning for Non-stationary Environments", *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, San Diego, USA, 2025.

36. Schmidhuber, J., *Evolutionary Principles in Self-Referential Learning, or on Learning How to Learn: The Meta-Meta-... Hook.*, Diploma thesis, Technische Universität München, 1987.

37. Rothfuss, J., D. Lee, I. Clavera, T. Asfour and P. Abbeel, "ProMP: Proximal Meta-Policy Search", *International Conference on Learning Representations*, Vancouver, Canada, 2018.

38. Liu, B., X. Feng, J. Ren, L. Mai, R. Zhu, H. Zhang, J. Wang and Y. Yang, "A Theoretical Understanding of Gradient Bias in Meta-Reinforcement Learning", A. H. Oh, A. Agarwal, D. Belgrave and K. Cho (Editors), *Advances in Neural Information Processing Systems*, Vol. 35, pp. 31059–31072, New Orleans,USA, 2022.

39. Gupta, A., R. Mendonca, Y. Liu, P. Abbeel and S. Levine, "Meta-Reinforcement Learning of Structured Exploration Strategies", *Advances in Neural Information Processing Systems*, Vol. 31, pp. 5302–5311, Montreal, Canada, 2018.

40. Li, Z., F. Zhou, F. Chen and H. Li, "Meta-SGD: Learning to Learn Quickly for Few-shot Learning", arXiv:1707.09835, 2017.

41. Vuorio, R., S.-H. Sun, H. Hu and J. J. Lim, "Multimodal Model-agnostic Meta-Learning via Task-aware Modulation", *Advances in Neural Information Processing Systems*, Vol. 32, pp. 1–12, Vancouver, Canada, 2019.

42. Mishra, N., M. Rohaninejad, X. Chen and P. Abbeel, "A Simple Neural Attentive Meta-Learner.", *International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.

43. Duan, Y., M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel and W. Zaremba, "One-Shot Imitation Learning.", I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett (Editors), *NIPS*, pp. 1087–1098, Long Beach, USA, 2017.

44. Zintgraf, L., K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann and S. Whiteson, "VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning", *International Conference on Learning Representations*, (Online), 2020.

45. Li, L., R. Yang and D. Luo, "FOCAL: Efficient Fully-Offline Meta-Reinforcement Learning via Distance Metric Learning and Behavior Regularization", *International Conference on Learning Representations*, 2020.

46. Rakelly, K., A. Zhou, C. Finn, S. Levine and D. Quillen, "Efficient Off-policy Meta-Reinforcement Learning via Probabilistic Context Variables", *International Conference on Machine Learning*, pp. 5331–5340, PMLR, Long Beach, USA, 2019.

47. Kingma, D. P. and M. Welling, "Auto-Encoding Variational Bayes", *International*

Conference on Learning Representations, Banff, Canada, 2014.

48. Kumar, A., Z. Fu, D. Pathak and J. Malik, "RMA: Rapid Motor Adaptation for Legged Robots", *RSS*, (Online), 2021.

49. Lee, K., Y. Seo, S. Lee, H. Lee and J. Shin, "Context-aware Dynamics Model for Generalization in Model-based Reinforcement Learning", *International Conference on Machine Learning*, pp. 5757–5766, (Online), 2020.

50. Fernando, C., D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel and D. Wierstra, "PathNet: Evolution Channels Gradient Descent in Super Neural Networks", arXiv:1701.08734, 2017.

51. Rusu, A. A., N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu and R. Hadsell, "Progressive Neural Networks", arXiv:1606.04671, 2016.

52. Bengio, Y., J. Louradour, R. Collobert and J. Weston, "Curriculum Learning", *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, Montreal, Quebec, Canada, 2009.

53. Narvekar, S., B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor and P. Stone, "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey", *Journal of Machine Learning Research*, Vol. 21, No. 181, pp. 1–50, 2020.

54. Weinshall, D., G. Cohen and D. Amir, "Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks", *International Conference on Machine Learning*, pp. 5238–5246, PMLR, Stockholm, Sweden, 2018.

55. Ugur, E. and J. H. Piater, "Emergent Structuring of Interdependent Affordance Learning Tasks", *4th International Conference on Development and Learning and on Epigenetic Robotics, ICDL-EPIROB 2014, October 13-16, 2014*, pp. 489–494, IEEE, Genoa, Italy,, 2014.

56. Hochreiter, S. and J. Schmidhuber, "Long Short-term Memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997, publisher: MIT Press.

57. Zaremba, W. and I. Sutskever, "Learning to Execute", arXiv:1410.4615, 2014.

58. Schaul, T., J. Quan, I. Antonoglou and D. Silver, "Prioritized Experience Replay", *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

59. Fujimoto, S., D. Meger and D. Precup, "Off-Policy Deep Reinforcement Learning without Exploration", *International Conference on Machine Learning*, pp. 2052–2062, Stockholm, Sweden, 2018.

60. Siegel, N., J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess and M. Riedmiller, "Keep Doing What Worked: Behavior Modelling Priors for Offline Reinforcement Learning", *International Conference on Learning Representations*, (Online), 2020.

61. Ma, Y. J., D. Jayaraman and O. Bastani, "Conservative Offline Distributional Reinforcement Learning", A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan (Editors), *Advances in Neural Information Processing Systems*, pp. 19235–19247, (Online), 2021.

62. Jaques, N., A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu and R. Picard, "Way Off-policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialog", arXiv:1907.00456, 2019.

63. Wu, Y., G. Tucker and O. Nachum, "Behavior Regularized Offline Reinforcement Learning", arXiv:1911.11361, 2019.

64. Peng, X. B., A. Kumar, G. Zhang and S. Levine, "Advantage-weighted Regression: Simple and Scalable Off-policy Reinforcement Learning", arXiv:1910.00177, 2019.

65. Nair, A., M. Dalal, A. Gupta and S. Levine, "Accelerating Online Reinforcement Learning with Offline Datasets", arxiv:2006.09359, 2020.

66. Nowozin, S., B. Cseke and R. Tomioka, "F-GAN: Training Generative Neural Samplers using Variational Divergence Minimization", *Advances in Neural Information Processing Systems*, Vol. 29, pp. 271–279, Barcelona, Spain, 2016.

67. Kumar, A., J. Fu, G. Tucker and S. Levine, "Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction", *Advances in Neural Information Processing Systems*, Vol. 32, Vancouver, Canada, 2019.

68. Yu, T., A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine and C. Finn, "COMBO: Conservative Offline Model-Based Policy Optimization", arXiv:2102.08363, 2021.

69. Kumar, A., A. Zhou, G. Tucker and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1179–1191, 2020.

70. Zhang, S., B. Liu and S. Whiteson, "GradientDICE: Rethinking Generalized Offline Estimation of Stationary Values", H. D. III and A. Singh (Editors), *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 11194–11203, PMLR, (Online), 2020.

71. Jiang, N. and L. Li, "Doubly Robust Off-policy Evaluation for Reinforcement Learning", arXiv:1511.03722, 2015.

72. Yu, T., G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn and T. Ma, "MOPO: Model-based Offline Policy Optimization", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 14129–14142, (Online), 2020.

73. Janner, M., Q. Li and S. Levine, "Offline Reinforcement Learning as One Big Sequence Modeling Problem", M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan (Editors), *Advances in Neural Information Processing Systems*,

Vol. 34, pp. 1273–1286, Curran Associates, Inc., Online, 2021.

74. Chen, L., K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas and I. Mordatch, "Decision Transformer: Reinforcement Learning via Sequence Modeling", *Advances in Neural Information Processing Systems*, pp. 15084–15097, Curran Associates, Inc., (Online), 2021.

75. Park, S., K. Frans, S. Levine and A. Kumar, "Is Value Learning Really the Main Bottleneck in Offline RL?", arXiv:2406.09329, 2024.

76. Mazoure, B., I. Kostrikov, O. Nachum and J. J. Tompson, "Improving Zero-Shot Generalization in Offline Reinforcement Learning using Generalized Similarity Functions", S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh (Editors), *Advances in Neural Information Processing Systems*, Vol. 35, pp. 25088–25101, Curran Associates, Inc., New Orleans, USA, 2022.

77. Yang, R., C. Bai, X. Ma, Z. Wang, C. Zhang and L. Han, "RORL: Robust Offline Reinforcement Learning via Conservative Smoothing", *Advances in Neural Information Processing Systems*, Vol. 35, pp. 23851–23866, New Orleans, USA, 2022.

78. Ye, C., R. Yang, Q. Gu and T. Zhang, "Corruption-robust Offline Reinforcement Learning with General Function Approximation", *Advances in Neural Information Processing Systems*, Vol. 36, pp. 36208–36221, New Orleans, USA, 2023.

79. Zhang, X., Y. Chen, X. Zhu and W. Sun, "Corruption-robust Offline Reinforcement Learning", *International Conference on Artificial Intelligence and Statistics*, pp. 5757–5773, PMLR, (Online), 2022.

80. Sohl-Dickstein, J., E. A. Weiss, N. Maheswaranathan and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics", *International Conference on Machine Learning*, pp. 2256–2265, Lille, France, 2015.

81. Yue, J., L. Fang, S. Xia, Y. Deng and J. Ma, "Dif-Fusion: Towards High Color Fidelity in Infrared and Visible Image Fusion with Diffusion Models", arXiv:2301.08072, 2023.

82. Wei, M., Y. Shen, Y. Wang, H. Xie and F. L. Wang, "RainDiffusion: When Unsupervised Learning Meets Diffusion Models for Real-world Image Deraining", arXiv:2301.09430, 2023.

83. Eschweiler, D. and J. Stegmaier, "Denoising Diffusion Probabilistic Models for Generation of Realistic Fully-Annotated Microscopy Image Data Sets", arXiv:2301.10227, 2023.

84. Bao, F., C. Li, Y. Cao and J. Zhu, "All are Worth Words: a ViT Backbone for Score-based Diffusion Models", arXiv:2209.12152, 2022.

85. White, J. C. and R. Cotterell, "Schrödinger's Bat: Diffusion Models Sometimes Generate Polysemous Words in Superposition", arXiv:2211.13095, 2022.

86. Janner, M., Y. Du, J. Tenenbaum and S. Levine, "Planning with Diffusion for Flexible Behavior Synthesis", *Proceedings of the 39th International Conference on Machine Learning*, pp. 9902–9915, Baltimore, USA, 2022.

87. He, L., L. Zhang, J. Tan and X. Wang, "Diffcps: Diffusion Model Based Constrained Policy Search for Offline Reinforcement Learning", arXiv:2310.05333, 2023.

88. Hansen-Estruch, P., I. Kostrikov, M. Janner, J. G. Kuba and S. Levine, "Idql: Implicit Q-Learning as An Actor-Critic Method with Diffusion Policies", arXiv:2304.10573, 2023.

89. Ho, J., A. Jain and P. Abbeel, "Denoising Diffusion Probabilistic Models", H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin (Editors), *Advances in Neural Information Processing Systems*, Vol. 33, pp. 6840–6851, Curran

Associates, Inc., (Online), 2020.

90. Yang, L., Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui and M.-H. Yang, "Diffusion Models: A Comprehensive Survey of Methods and Applications", arXiv:2209.00796, 2022.

91. Zhu, Z., H. Zhao, H. He, Y. Zhong, S. Zhang, Y. Yu and W. Zhang, "Diffusion Models for Reinforcement Learning: A Survey", arXiv:2311.01223, 2023.

92. Chen, J., B. Ganguly, Y. Xu, Y. Mei, T. Lan and V. Aggarwal, "Deep Generative Models for Offline Policy Learning: Tutorial, Survey, and Perspectives on Future Directions", arXiv:2402.13777, 2024.

93. Coleman, M., O. Russakovsky, C. Allen-Blanchette and Y. Zhu, "Discrete Diffusion Reward Guidance Methods for Offline Reinforcement Learning", *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, Honolulu, USA, 2023.

94. He, H., C. Bai, K. Xu, Z. Yang, W. Zhang, D. Wang, B. Zhao and X. Li, "Diffusion Model is an Effective Planner and Data Synthesizer for Multi-Task Reinforcement Learning", A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine (Editors), *Advances in Neural Information Processing Systems*, Vol. 36, pp. 64896–64917, Curran Associates, Inc., New Orleans, USA, 2023.

95. Liang, Z., Y. Mu, M. Ding, F. Ni, M. Tomizuka and P. Luo, "Adaptdiffuser: Diffusion Models as Adaptive Self-evolving Planners", arXiv:2302.01877, 2023.

96. Ackermann, J., T. Osa and M. Sugiyama, "Offline Reinforcement Learning from Datasets with Structured Non-Stationarity", *Reinforcement Learning Conference*, Amherst, MA, USA, 2024.

97. Finn, C., A. Rajeswaran, S. Kakade and S. Levine, "Online Meta-Learning", K. Chaudhuri and R. Salakhutdinov (Editors), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning*

Research, pp. 1920–1930, PMLR, Long Beach, USA, 2019.

98. Al-Shedivat, M., T. Bansal, Y. Burda, I. Sutskever, I. Mordatch and P. Abbeel, "Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments", *International Conference on Learning Representations*, Vancouver, Canada, 2018.

99. Bao, L.-L., J.-S. Zhang and C.-X. Zhang, "Spatial Multi-Attention Conditional Neural Processes", *Neural Networks*, Vol. 173, p. 106201, 2024.

100. Bruinsma, W., S. Markou, J. Requeima, A. Y. K. Foong, T. Andersson, A. Vaughan, A. Buonomo, S. Hosking and R. E. Turner, "Autoregressive Conditional Neural Processes", *International Conference on Learning Representations*, Kigali, Rwanda, 2023.

101. Kim, H., A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals and Y. W. Teh, "Attentive Neural Processes", *International Conference on Learning Representations*, New Orleans, USA, 2019.

102. Ye, Z. and L. Yao, "Contrastive Conditional Neural Processes", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9687–9696, New Orleans, LA, USA, 2022.

103. Seker, M. Y., M. Imre, J. H. Piater and E. Ugur, "Conditional Neural Movement Primitives", *Robotics: Science and Systems*, Vol. 10, Freiburg im Breisgau, Germany, 2019.

104. Seker, M. Y., A. Ahmetoglu, Y. Nagai, M. Asada, E. Oztop and E. Ugur, "Imitation and Mirror Systems in Robots Through Deep Modality Blending Networks", *Neural Networks*, Vol. 146, pp. 22–35, 2022.

105. Ada, S. E. and M. Y. Seker, "Generative Adversarial Networks with Conditional Neural Movement Primitives for An Interactive Generative Drawing Tool",

arXiv:2111.14934, 2021.

106. Hassabis, D., D. Kumaran, C. Summerfield and M. Botvinick, "Neuroscience-Inspired Artificial Intelligence", *Neuron*, Vol. 95, No. 2, pp. 245–258, July 2017.

107. Niv, Y., "Reinforcement Learning in the Brain", *Journal of Mathematical Psychology*, Vol. 53, No. 3, p. 139–154, 2009.

108. Brady, F., "The Contextual Interference Effect and Sport Skills", *Perceptual and motor skills*, Vol. 106, No. 2, pp. 461–472, 2008.

109. Cross, E. S., P. J. Schmitt and S. T. Grafton, "Neural Substrates of Contextual Interference During Motor Learning Support a Model of Active Preparation", *Journal of Cognitive Neuroscience*, Vol. 19, No. 11, pp. 1854–1871, 2007.

110. Lin, C.-H. J., B. J. Knowlton, M.-C. Chiang, M. Iacoboni, P. Udompholkul and A. D. Wu, "Brain–behavior Correlates of Optimizing Learning Through Interleaved Practice", *Neuroimage*, Vol. 56, No. 3, pp. 1758–1772, 2011.

111. Schorn, J. M. and B. J. Knowlton, "Interleaved Practice Benefits Implicit Sequence Learning and Transfer", *Memory & Cognition*, Vol. 49, No. 7, pp. 1436–1452, 2021.

112. Sharma, S., A. Jha, P. Hegde and B. Ravindran, "Learning to Multi-task by Active Sampling", arXiv:1702.06053, 2017.

113. Jean, S., O. Firat and M. Johnson, "Adaptive Scheduling for Multi-task Learning", arXiv:1909.06434, 2019.

114. Lesort, T., V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat and N. Díaz-Rodríguez, "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges", *Information Fusion*, Vol. 58, pp. 52–68, 2020.

115. Oudeyer, P.-Y. and F. Kaplan, "What is Intrinsic Motivation? A Typology of Computational Approaches", *Frontiers in Neurorobotics*, Vol. 1, 2007.

116. Kasmarik, K., G. Baldassarre, V. G. Santucci and J. Triesch, "Guest Editorial Special Issue on Intrinsically Motivated Open-Ended Learning (IMOL)", *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 15, No. 2, pp. 321–324, 2023.

117. Santucci, V. G., P.-Y. Oudeyer, A. Barto and G. Baldassarre, "Intrinsically Motivated Open-ended Learning in Autonomous Robots", *Frontiers in Neurorobotics*, Vol. 13, p. 115, 2020.

118. Sener, M. I., Y. Nagai, E. Oztop and E. Ugur, "Exploration with Intrinsic Motivation using Object-Action-Outcome Latent Space", *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 15, No. 2, pp. 325–336, 2021.

119. Bugur, S., E. Oztop, Y. Nagai and E. Ugur, "Effect Regulated Projection of Robot's Action Space for Production and Prediction of Manipulation Primitives Through Learning Progress and Predictability-Based Exploration", *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 13, No. 2, pp. 286–297, 2021.

120. Burda, Y., H. Edwards, A. Storkey and O. Klimov, "Exploration by Random Network Distillation", arXiv:1810.12894, 2018.

121. Eysenbach, B., A. Gupta, J. Ibarz and S. Levine, "Diversity is All You Need: Learning Skills without a Reward Function", arXiv:1802.06070, 2018.

122. Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, New York, USA, 2014.

123. Agarwal, A., N. Jiang, S. M. Kakade and W. Sun, *Reinforcement Learning: Theory and Algorithms*, Department of Computer Science, University of Washington, Seattle, WA, USA, 2019.

124. Ada, S. E., E. Ugur and H. L. Akin, "Generalization in Transfer Learning: Robust Control of Robot Locomotion", *Robotica*, Vol. 40, No. 11, p. 3811–3836, 2022.

125. Kaelbling, L. P., M. L. Littman and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains", *Artificial Intelligence*, Vol. 101, No. 1-2, pp. 99–134, 1998.

126. Bonet, B., "Deterministic POMDPs Revisited", arXiv:1205.2659, 2012.

127. Song, Y. and S. Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution", H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett (Editors), *Advances in Neural Information Processing Systems*, Vol. 32, p. 6392, Curran Associates, Inc., Vancouver, Canada, 2019.

128. Duan, Y., J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever and P. Abbeel, "RLˆ 2: Fast Reinforcement Learning via Slow Reinforcement Learning", arXiv:1611.02779, 2016, 2016.

129. Clavera, I., A. Nagabandi, R. S. Fearing, P. Abbeel, S. Levine and C. Finn, "Learning to Adapt: Meta-Learning for Model-Based Control", *International Conference on Learning Representations*, Vancouver, BC, Canada, 2019.

130. Williams, R. J., "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning", *Machine Learning*, Vol. 8, No. 3, pp. 229–256, 1992.

131. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization.", Y. Bengio and Y. LeCun (Editors), *ICLR*, San Diego, USA, 2015.

132. Dugas, C., Y. Bengio, F. Bélisle, C. Nadeau and R. Garcia, "Incorporating Second-Order Functional Knowledge for Better Option Pricing", *Advances in Neural Information Processing Systems*, Vol. 13, Denver, USA, 2000.

133. Chua, K., R. Calandra, R. McAllister and S. Levine, "Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models", *NIPS*, Vol. 31, Montreal, Canada, 2018.

134. Seo, Y., "Context-aware Dynamics Model for Generalization in Model-based Reinforcement Learning Code Repository", https://github.com/younggyoseo/CaDM, accessed on June 05, 2024.

135. Garcia, C. E., D. M. Prett and M. Morari, "Model Predictive Control: Theory and Practice—A Survey", *Automatica*, Vol. 25, No. 3, pp. 335–348, 1989.

136. Botev, Z. I., D. P. Kroese, R. Y. Rubinstein and P. L'Ecuyer, "The Cross-entropy Method for Optimization", *Handbook of Statistics*, Vol. 31, pp. 35–59, Elsevier, Amsterdam, Netherlands, 2013.

137. Oztop, E. and E. Ugur, "Lifelong Robot Learning", M. H. Ang, O. Khatib and B. Siciliano (Editors), *Encyclopedia of Robotics*, pp. 1–12, Springer Berlin Heidelberg, 2020.

138. Parisi, G. I., R. Kemker, J. L. Part, C. Kanan and S. Wermter, "Continual Lifelong Learning with Neural Networks: A Review", *Neural Networks*, Vol. 113, pp. 54–71, 2019.

139. Thrun, S. and T. M. Mitchell, "Lifelong Robot Learning", *Robotics and autonomous systems*, Vol. 15, No. 1-2, pp. 25–46, 1995.

140. Say, H. and E. Oztop, "A Model for Cognitively Valid Lifelong Learning", *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1–7, Chengdu, Sichuan Province, China, 2023.

141. Lin, C.-H., M.-C. Chiang, B. J. Knowlton, M. Iacoboni, P. Udompholkul and A. D. Wu, "Interleaved Practice Enhances Skill Learning and the Functional Connectivity of Fronto-Parietal Networks", *Human Brain Mapping*, Vol. 34, No. 7,

pp. 1542–1558, 2013.

142. Samani, J. and S. C. Pan, "Interleaved Practice Enhances Memory and Problem-solving Ability in Undergraduate Physics", *NPJ Science of Learning*, Vol. 6, No. 1, p. 32, 2021.

143. Park, J., K. Varma and S. Varma, "The Role of Executive Function Abilities in Interleaved vs. Blocked Learning of Science Concepts", *Frontiers in Psychology*, Vol. 14, 2023.

144. Nadel, L., A. Hupbach, R. Gomez and K. Newman-Smith, "Memory Formation, Consolidation and Transformation", *Neuroscience & Biobehavioral Reviews*, Vol. 36, No. 7, pp. 1640–1645, 2012.

145. Cepeda, N. J., E. Vul, D. Rohrer, J. T. Wixted and H. Pashler, "Spacing Effects in Learning: A Temporal Ridgeline of Optimal Retention", *Psychological science*, Vol. 19, No. 11, pp. 1095–1102, 2008.

146. Oudeyer, P.-Y., F. Kaplan and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development", *IEEE Transactions on Evolutionary Computation*, Vol. 11, No. 2, pp. 265–286, 2007.

147. Colas, C., P. Fournier, M. Chetouani, O. Sigaud and P.-Y. Oudeyer, "CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning", K. Chaudhuri and R. Salakhutdinov (Editors), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 1331–1340, PMLR, Long Beach, USA, 09–15 Jun 2019.

148. Forestier, S. and P.-Y. Oudeyer, "Curiosity-driven Development of Tool Use Precursors: A Computational Model", *38th Annual Conference of the Cognitive Science Society (COGSCI 2016)*, pp. 1859–1864, Philadelphia, Pennsylvania, USA, 2016.

149. Gatsoulis, Y. and T. M. McGinnity, "Intrinsically Motivated Learning Systems Based on Biologically-Inspired Novelty Detection", *Robotics and Autonomous Systems*, Vol. 68, pp. 12–20, 2015.

150. Achiam, J. and S. Sastry, "Surprise-based Intrinsic Motivation for Deep Reinforcement Learning", arXiv:1703.01732, 2017.

151. Rusu, A. A., M. Vecerík, T. Rothörl, N. Heess, R. Pascanu and R. Hadsell, "Sim-to-Real Robot Learning from Pixels with Progressive Nets", *1st Annual Conference on Robot Learning, CoRL 2017, November 13-15, 2017, Proceedings*, Vol. 78 of *Proceedings of Machine Learning Research*, pp. 262–270, PMLR, Mountain View, California, USA, 2017.

152. Huang, S., R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta and J. G. Araújo, "CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms", *Journal of Machine Learning Research*, Vol. 23, No. 274, pp. 1–18, 2022.

153. Csikszentmihalyi, M., *Flow: The Psychology of Optimal Experience*, Harper & Row, New York, USA, 1990.

154. Raffin, A., "RL Baselines3 Zoo", https://github.com/DLR-RM/rl-baselines3-zoo, accessed on June 10, 2023.

155. Engstrom, L., A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph and A. Madry, "Implementation Matters in Deep RL: A Case Study on PPO and TRPO", *International Conference on Learning Representations*, (Online), 2020.

156. Towers, M., J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen and O. G. Younis, "Gymnasium", https://zenodo.org/record/8127025, accessed on September 01, 2023.

157. Todorov, E., T. Erez and Y. Tassa, "Mujoco: A Physics Engine for Model-based Control", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

158. O'Doherty, J. P., P. Dayan, K. Friston, H. Critchley and R. J. Dolan, "Temporal Difference Models and Reward-related Learning in the Human Brain", *Neuron*, Vol. 38, No. 2, pp. 329–337, April 2003.

159. Schultz, W., P. Dayan and P. R. Montague, "A Neural Substrate of Prediction and Reward", *Science (New York, N.Y.)*, Vol. 275, No. 5306, pp. 1593–1599, March 1997.

160. Ajemian, R., A. D'Ausilio, H. Moorman and E. Bizzi, "A Theory for How Sensorimotor Skills Are Learned and Retained in Noisy and Nonstationary Neural Circuits", *Proceedings of the National Academy of Sciences*, Vol. 110, No. 52, p. E5078–E5087, 2013.

161. Dohare, S., A. R. Mahmood and R. S. Sutton, "Continual Backprop: Stochastic Gradient Descent with Persistent Randomness", arXiv:2108.06325, 2021.

162. Wolpert, D. M., R. C. Miall and M. Kawato, "Internal Models in the Cerebellum", *Trends in Cognitive Sciences*, Vol. 2, No. 9, p. 338–347, 1998.

163. Imamizu, H., "Separated Modules for Visuomotor Control and Learning in the Cerebellum: A Functional MRI Study", *NeuroImage: Third International Conference on Functional Mapping of the Human Brain*, Vol. 5, p. S598, Academic Press, Copenhagen, Denmark, 1997.

164. Imamizu, H., S. Miyauchi, T. Tamada, Y. Sasaki, R. Takino, B. PuEtz, T. Yoshioka and M. Kawato, "Human Cerebellar Activity Reflecting an Acquired Internal Model of a New Tool", *Nature*, Vol. 403, No. 6766, p. 192–195, 2000.

165. Chi, C., S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel and S. Song, "Diffusion

Policy: Visuomotor Policy Learning via Action Diffusion", *Robotics: Science and Systems*, Daegu, Republic of Korea, 2023.

166. Florence, P., C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch and J. Tompson, "Implicit Behavioral Cloning", *5th Annual Conference on Robot Learning*, pp. 158–168, London & Virtual, 2021.

167. Zhao, T. Z., V. Kumar, S. Levine and C. Finn, "Learning Fine-grained Bimanual Manipulation with Low-cost Hardware", arXiv:2304.13705, 2023.

168. Denouden, T., R. Salay, K. Czarnecki, V. Abdelzad, B. Phan and S. Vernekar, "Improving Reconstruction Autoencoder Out-of-distribution Detection with Mahalanobis Distance", arXiv:1812.02765, 2018.

169. Mutlu, U. and E. Alpaydın, "Training Bidirectional Generative Adversarial Networks with Hints", *Pattern Recognition*, Vol. 103, p. 107320, 2020.

170. Van Hasselt, H., A. Guez and D. Silver, "Deep Reinforcement Learning with Double Q-Learning", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30, Phoenix, Arizona, USA, 2016.

171. Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous Control with Deep Reinforcement Learning", Y. Bengio and Y. LeCun (Editors), *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

172. Wang, Z., "Diffusion-Policies-for-Offline-RL", https://github.com/Zhendong-Wang/Diffusion-Policies-for-Offline-RL, accessed on February 30, 2023.

173. Haarnoja, T., A. Zhou, P. Abbeel and S. Levine, "Soft Actor-Critic: Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", *International Conference on Machine Learning*, pp. 1861–1870, Stockholm, Sweden, 2018.

174. Brandfonbrener, D., W. F. Whitney, R. Ranganath and J. Bruna, "Offline RL Without Off-Policy Evaluation", A. Beygelzimer, Y. Dauphin, P. Liang and J. W. Vaughan (Editors), *Advances in Neural Information Processing Systems*, pp. 4933–4946, (Online), 2021.

175. Trabucco, B., M. Phielipp and G. Berseth, "Anymorph: Learning Transferable Polices by Inferring Agent Morphology", *International Conference on Machine Learning*, pp. 21677–21691, PMLR, Baltimore, USA, 2022.

176. Zhang, A., C. Lyle, S. Sodhani, A. Filos, M. Kwiatkowska, J. Pineau, Y. Gal and D. Precup, "Invariant Causal Prediction for Block MDPs", *International Conference on Machine Learning*, pp. 11214–11224, PMLR, (Online), 2020.

177. Xie, A., J. Harrison and C. Finn, "Deep Reinforcement Learning amidst Continual Structured Non-stationarity", *International Conference on Machine Learning*, pp. 11393–11403, PMLR, (Online), 2021.

178. Rasul, K., A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hassen, M. Biloš, S. Garg, A. Schneider, N. Chapados, A. Drouin, V. Zantedeschi, Y. Nevmyvaka and I. Rish, "Lag-Llama: Towards Foundation Models for Time series Forecasting", arXiv:2310.08278, 2023.

179. Song, Y., J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations", *International Conference on Learning Representations*, (Online), 2021.

180. Xiao, Z., K. Kreis and A. Vahdat, "Tackling the Generative Learning Trilemma with Denoising Diffusion GANs", *International Conference on Learning Representations*, (Online), 2022.

181. Luo, C., "Understanding Diffusion Models: A Unified Perspective",

arXiv:2208.11970, 2022.

182. Fujimoto, S., H. Hoof and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods", *International Conference on Machine Learning*, pp. 1587–1596, PMLR, Stockholm, Sweden, 2018.

183. Kostrikov, I., A. Nair and S. Levine, "Offline Reinforcement Learning with Implicit Q-Learning", *International Conference on Learning Representations*, (Online), 2022.

184. Peters, J. and S. Schaal, "Reinforcement Learning by Reward-weighted Regression for Operational Space Control", *Proceedings of the 24th International Conference on Machine Learning*, pp. 745–750, Oregon, USA, 2007.

185. Wang, Q., J. Xiong, L. Han, p. sun, H. Liu and T. Zhang, "Exponentially Weighted Imitation Learning for Batched Historical Data", *Advances in Neural Information Processing Systems*, Vol. 31, pp. 6288–6297, Montreal, Canada, 2018.

186. Tarasov, D., A. Nikulin, D. Akimov, V. Kurenkov and S. Kolesnikov, "CORL: Research-oriented Deep Offline Reinforcement Learning Library", *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, New Orleans, USA, 2022.

187. Thomas, G., "Implicit Q-Learning (IQL) in PyTorch", https://github.com/gwthomas/IQL-PyTorch, accessed on July 26, 2025.

188. Park, S., Q. Li and S. Levine, "Flow Q-Learning", *International Conference on Machine Learning*, Vancouver, Canada, 2025.

189. Park, S., "FQL: Flow Q-Learning Official Implementation", https://github.com/seohongpark/fql, accessed on July 26, 2025.

190. Liu, X., C. Gong and Q. Liu, "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow", *International Conference on Learning*

Representations, Kigali, Rwanda, 2023.

191. Liu, X., X. Zhang, J. Ma, J. Peng and Q. Liu, "Instaflow: One Step is Enough for High-quality Diffusion-based Text-to-Image Generation", *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024.

192. Frans, K., D. Hafner, S. Levine and P. Abbeel, "One Step Diffusion via Shortcut Models", arXiv:2410.12557, 2024.

193. Ding, Z., C. Jin, D. Liu, H. Zheng, K. K. Singh, Q. Zhang, Y. Kang, Z. Lin and Y. Liu, "Dollar: Few-step Video Generation via Distillation and Latent Reward Optimization", arXiv:2412.15689, 2024.

194. Li, J., W. Feng, W. Chen and W. Y. Wang, "Reward Guided Latent Consistency Distillation", arXiv:2403.11027, 2024.

195. Godahewa, R. W., C. Bergmeir, G. I. Webb, R. Hyndman and P. Montero-Manso, "Monash Time Series Forecasting Archive", *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, (Online), 2021.

196. Dheeru, D. and E. Karra Taniskidou, "UCI Machine Learning Repository", http://archive.ics.uci.edu/ml, accessed on July 10, 2024.

197. Salinas, D., M. Bohlke-Schneider, L. Callot, R. Medico and J. Gasthaus, "High-dimensional Multivariate Forecasting with Low-rank Gaussian Copula Processes", *Advances in Neural Information Processing Systems*, Vol. 32, Vancouver, Canada, 2019.

198. Lai, G., W.-C. Chang, Y. Yang and H. Liu, "Modeling Long-and Short-term Temporal Patterns with Deep Neural Networks", *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, Ann Arbor, MI, USA, 2018.

199. Ronneberger, O., P. Fischer and T. Brox, "U-net: Convolutional Networks for Biomedical Image Segmentation", *Medical image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, Munich, Germany, 2015.

200. Yang, Z. and Y. Xu, "DMBP: Diffusion Model-based Predictor for Robust Offline Reinforcement Learning Against State Observation Perturbations", https://github.com/zhyang2226/DMBP/tree/main, accessed on May 30, 2024.

201. Ronchetti, E. M. and P. J. Huber, *Robust Statistics*, John Wiley & Sons Hoboken, NJ, USA, 2009.

202. Fischler, M. A. and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, Vol. 24, No. 6, p. 381–395, June 1981.

203. Welford, B. P., "Note on a Method for Calculating Corrected Sums of Squares and Products", *Technometrics*, Vol. 4, No. 3, pp. 419–420, 1962.

204. Knuth, D. E., *The Art of Computer Programming*, Vol. 3, Pearson Education, Massachusetts, USA, 1997.

205. Scott, D. W., *Multivariate Density Estimation: Theory, Practice, and Visualization*, John Wiley & Sons, New York, USA, 2015.

206. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is All You Need", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008, Long Beach, USA, 2017.

207. Radford, A., K. Narasimhan, T. Salimans and I. Sutskever, "Improving Language Understanding by Generative Pre-training", https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf, accessed on

May 10, 2022.

208. Radford, A., J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language Models are Unsupervised Multitask Learners", *OpenAI Blog*, Vol. 1, No. 8, p. 9, 2019.

209. Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, "Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 1877–1901, (Online), 2020.

210. Ouyang, L., J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike and R. J. Lowe, "Training Language Models to Follow Instructions With Human Feedback", *Advances in Neural Information Processing Systems*, pp. 27730–27744, New Orleans, LA, USA, 2022.

211. Liu, Y., T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, Z. Wu, L. Zhao, D. Zhu, X. Li, N. Qiang, D. Shen, T. Liu and B. Ge, "Summary of Chatgpt-related Research and Perspective Towards the Future of Large Language Models", *Meta-radiology*, Vol. 1, No. 2, p. 100017, 2023.

212. Biewald, L., "Experiment Tracking with Weights and Biases", https://www.wandb.com, accessed on Jan 01, 2023.

213. Yang, R., "Robust Offline Reinforcement Learning Code Repository", https://github.com/YangRui2015/RORL, accessed on February 01, 2025.

214. Kostrikov, I., "Implicit Q-Learning (Official Implementation)", https://github.com/ikostrikov/implicit_q_Learning, accessed on July 26, 2025.

215. Fujimoto, S., "Twin Delayed Deep Deterministic Behavioral Cloning Code Repository", https://github.com/sfujim/TD3_BC, accessed on February 01, 2025.

216. Tinkoff AI, "CORL: Clean Offline Reinforcement Learning Code Repository", https://github.com/tinkoff-ai/CORL, accessed on July 26, 2025.

APPENDIX A: HYPERPARAMETERS USED IN FORL

We select the hyperparameters in Table A.1 based on the validation loss of the FORL diffusion model in D4RL and OGbench standard offline RL datasets. The validation loss is computed using the DM loss function in Equation (7.7). Hyperparameter optimization was conducted using a grid search, with the following ranges for Maze2d and Antmaze: diffusion timesteps $N = \{10, 20\}$, the number of hidden layers following the Temporal Unet model $\#layers = \{1, 2, 3\}$, window size $w = \{128, 256\}$, and learning rate $lr = \{0.0004, 0.0006, 0.0009\}$. For the Kitchen and Cube-S-P *embedding dimension* $= \{64, 128\}$, learning rate $lr = \{0.0004, 0.0009\}$ and $w = \{32, 64\}$ are used for grid search. We used the Weights & Biases platform to track our experiments [212].

Table A.1. Hyperparameters for FORL.

Hyperparameters	**Value**				
Batch Size	128				
Hidden Size	128				
# Denoiser samples	50				
Optimizer	Adam[131]				
Maximum Timesteps	300 000				
Hyperparameters	**Kitchen**	**Antmaze**	**Maze2d**	**AntM-L-Nav**	**Cube-S-P**
Embedding Dimension	128	64	64	64	128
w	32	256	128	256	64
Learning rate	4×10^{-4}	4×10^{-4}	9×10^{-4}	4×10^{-4}	9×10^{-4}
Observation Scale	1	1	100	1	1
Time Concatenation	true	true	false	true	true
# Middle Layers	1	1	`large:1` `medium:3`	1	1
N	10	10	`large:10` `medium:20`	20	20

We use the established source codes and hyperparameter settings for DMBP [200], DQL [29, 172, 200], RORL [77, 213, 200], IQL [183, 214, 187, 186], TD3+BC [182, 215, 200], and FQL [188, 189].

Table A.2. Hyperparameters for DQL [29, 172, 200].

Hyperparameters	**Value**		
Maximum Timesteps	1 000 000		
γ	0.99		
τ	0.005		
Learning rate decay	True		
T	10		
β Schedule	`VP`		
Learning rate	3×10^{-4}		
α	0.2		
Batch Size	256		
Hidden Size	256		
Reward tune	No		
Normalize	False		
Optimizer	Adam[131]		
Hyperparameters	**Kitchen**	**Maze2d**	**Antmaze**

gn	10.0	`umaze-diverse: 3.0`
		`medium-diverse: 1.0`
		`large-diverse: 7.0`
			`umaze-diverse: 3.0`
			`medium-diverse: 1.0`
			`large-diverse: 7.0`

η	0.005	`umaze-diverse: 2.0`
		`medium-diverse: 3.0`
		`large-diverse: 3.5`
			`umaze-diverse: 2.0`
			`medium-diverse: 3.0`
			`large-diverse: 3.5`

MaxQ Backup	False	True	True

Table A.3. Hyperparameters for Implicit Q-Learning (IQL) [183, 214, 187, 216].

Hyperparameters	**Value**
Batch Size	256
Discount (γ)	0.99
Target Network Update (τ)	0.005
Maze2d β	3.0
Maze2d τ_{IQL}	0.7
Maze2d Normalize Rewards	false
Antmaze β	10.0
Antmaze τ_{IQL}	0.9
Antmaze Normalize Rewards	true

Table A.4. Hyperparameters for RORL [77, 213, 200].

Hyperparameters	**Value**
γ	0.99
$\textbf{soft}_\tau$	0.005
Q Learning Rate	3×10^{-4}
Policy Learning Rate	3×10^{-4}
α	1.0
Auto-tune entropy	true
MaxQ Backup	false
Deterministic Backup	false
η	-1
Batch Size	256
Hidden Size	256
Target Update Interval	1
τ	0.2
Normalize	False
n sample	20
β_Q	0.0001
β_P	1.0
ϵ_{ood}	0.01
Maximum Timesteps	3 000 000
Optimizer	Adam[131]
β_{OOD}	0.5
ϵ_Q	0.01
ϵ_P	0.03
λ_{max}	1.0
λ_{min}	0.5
λ_{decay}	10^{-6}

Table A.5. Hyperparameters for Flow Q-Learning (FQL) [188, 189].

Hyperparameters	**Value**
Batch Size	256
Learning Rate	0.0003
Discount factor (γ)	0.99
Target network smoothing coefficient (τ)	0.005
BC Coefficient (α)	10.0
Flow Steps	10
Actor Hidden Dimensions	(512, 512, 512, 512)
Value Hidden Dimensions	(512, 512, 512, 512)
AntM-L-Nav	BC Coefficient (α) = 10.0
Cube-S-P	BC Coefficient (α) = 300.0

Table A.6. Hyperparameters for TD3+BC [182, 215, 200].

Hyperparameters	**Value**
Maximum Timesteps	1 000 000
Exploration noise	0.1
Batch Size	256
Discount factor	0.99
τ	0.005
Policy Noise	0.2
Policy Noise Clipping	0.5
Policy update frequency	2
α	2.5
Normalize	true
Optimizer	Adam[131]