

FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Read-Write Reinforcement Learning: Data-Centric Experience Correction for Robust Reinforcement Learning Under Uncertainty

Pedro Rafael Castro Sousa



Mestrado em Inteligência Artificial

Supervisor: Prof. Zafeiris Kokkinogenis, PhD

Supervisor: Prof. Carlos Soares, PhD

June 24, 2026

Read-Write Reinforcement Learning: Data-Centric Experience Correction for Robust Reinforcement Learning Under Uncertainty

Pedro Rafael Castro Sousa

Mestrado em Inteligência Artificial

Approved in oral examination by the committee:

President: Prof. Inês Dutra

Referee: Prof. Zafeiris Kokkinogenis

Referee: Prof. Alberto Fernandez Gil

June 24, 2026

Resumo

Embora a Reinforcement Learning (**RL**) se destaque na resolução de tarefas complexas de tomada de decisão sequencial, o seu desempenho frequentemente degrada-se de forma catastrófica em ambientes do mundo real envolvidos em ruído sensorial e incerteza. As metodologias relacionadas a robustez em **RL** existentes são predominantemente centradas no modelo, focando-se em fortalecer a *policy* para resistir a essas perturbações em vez de abordar a qualidade dos dados subjacentes. Para colmatar esta lacuna, introduzimos o paradigma Read-Write Reinforcement Learning (**RW-RL**), centrado nos dados que desafia a premissa tradicional de um *experience replay buffer* estático. Através de extensas avaliações empíricas em múltiplos ambientes sujeitos a níveis variáveis de ruído de observação e observabilidade, demonstramos que o **RW-RL** é capaz de modificar as transições armazenadas sem desestabilizar ou colapsar o treino e, além disso, os nossos resultados estabelecem que esta metodologia produz *policies* globalmente mais fortes, equiparáveis aos métodos *state of the art*. Em última análise, o **RW-RL** tem como objetivo provar que a retificação ativa da qualidade dos dados fornece um caminho viável para a construção de sistemas robustos e auto-aperfeiçoáveis.

Abstract

While Reinforcement Learning (**RL**) excels in solving complex sequential decision-making tasks, its performance often degrades catastrophically in real-world environments plagued with sensory noise and uncertainty. Existing robust **RL** methodologies are overwhelmingly model-centric, focusing on hardening the policy to withstand these perturbations rather than addressing the quality of the underlying data. To bridge this gap, we introduced Read-Write Reinforcement Learning (**RW-RL**), a novel data-centric paradigm that challenges the traditional assumption of a static experience replay buffer. Throughout extensive empirical evaluations across multiple environments subjected to varying levels of observation noise and observability, we demonstrate that **RW-RL** is capable of modifying stored transitions without destabilizing or collapsing training, and furthermore, our results establish that this methodology yields overall stronger policies that are on par with established policy-centric methods. Ultimately, **RW-RL** aims to prove that actively rectifying data quality provides a viable pathway to building robust, self-improving systems.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals¹ to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

The specific Sustainable Development Goals mentioned have the following names:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.2	By mitigating catastrophic failures in deep reinforcement learning caused by observation noise, this research facilitates the safe and productive integration of automated AI systems into real-world physical industries.	Mean episode reward and normalized success rates across continuous control environments (e.g., HalfCheetah, Ant).
9	9.5	The True RW-RL framework advances AI infrastructure by allowing agents to self-correct noisy transition data, fundamentally improving the reliability of autonomous systems under sensor degradation.	Improvement in the Robustness Index (RI) and Conditional Value at Risk (CVaR) across escalating observation noise levels.

1. <https://sdgs.un.org/goals>

Declaration of honour

I declare that this dissertation is my original work and has not previously been submitted or assessed in any other course or curricular unit, whether at this or any other institution.

References to other authors (statements, ideas, thoughts), as well as the use of published works fully complies with the rules of attribution and are appropriately cited in the text and bibliography, in accordance with reference standards. I am aware that plagiarism and self-plagiarism constitute academic misconduct.

I further declare that I have not used Artificial Intelligence tools to produce any part(s) of this dissertation or, when such tools were used, their use and results are clearly indicated and were carefully reviewed for the detection of errors, inaccuracies, unfounded attributions, or other forms of content and argument bias, as well as for the determination of authorship.

I further declare that I have included studies carried out with multiple co-authors, having specified, with transparency and rigour, the contribution of each co-author, provided information on the possible existence of any other dissertation in which the same article has been included, reproduced the work in full with the permission of the journal in which it was published, and obtained the explicit written consent of all co-authors for the inclusion of the article.

Pedro Rafael Castro Sousa



Acknowledgements

I find this chapter really ironic, this feels like writing again in my own notebooks about how I feel, knowing that no one will really read this; its cathartic, writing this as the ending piece of this thesis is the equivalent of a sigh of relief with a smile on my face, to know I've come this far.

But I have not come this far alone, for the foundation on which I was built was first laid by my family, and for this, I owe them everything. If not for their unending support and guidance, teaching me discipline, humility, and to persevere even when things looked bleak. To my family, thank you for understanding me, and understanding my way of living and seeing the world, even despite it being so unorthodox and hard to make sense of from the outside looking in. You dealt with my antics, my obsessions, the hours lost to questions nobody asked me to answer (and to question), and you never once asked me to be anyone other than who I was. But more than anything else, thank you for understanding me. For knowing that my silences were not distance, and that my struggle to put feelings into words was never a lack of feeling, but a lack of language for something that ran too deep for me to express. You learned to read me in the ways I could not read myself out loud, and you loved me patiently through every version of myself it took to get here. This thesis exists because, long before I understood what curiosity could become, you already believed in where it would take me.

And for this journey, I also had someone who chose, every single day, to walk it beside me. To my girlfriend, Antónia, thank you for being there through the late nights and the early mornings, through the experiments that failed before the ones that worked, through the doubt that crept in more often than I let on. You listened to my incomprehensible ideas that sometimes made no sense, and somehow made me feel like they mattered anyway. We celebrated the small wins as if they were big ones, and you reminded me, on the days the big ones felt impossible, that we are always capable of reaching them. Academia usually is a lonely pursuit, but you made sure I never carried it alone, and for that, this thesis is as much a product of mine as it is of you believing in me.

So, if there is one thing I have learned through all of this, it is to keep going. Keep learning, even when the answers refuse to arrive, even when the questions multiply faster than the answers ever could. Keep fighting for what I believe in, even when it is unpopular, even when it is inconvenient, even when it would be far easier to simply agree and move on.

And to whoever reads this, don't be afraid of lunacy. Follow your obsessions, even the ones that make no sense to anyone but you. Accept the madness, because it was never something to be cured. And if life is not being lived through unyielding and unfathomable passion, then perhaps it is worth asking whether it is being lived at all.

*“We were born to pursue knowledge, for we were all born with nought.
Before we were given a language, our hands reached out, without being taught.”*

Kristen Wright

Contents

1	Introduction	1
1.1	Thesis Statement	3
1.2	Organisation of the Dissertation	3
2	Background	5
2.1	Reinforcement Learning	6
2.1.1	Policies	6
2.1.2	Return and Value Functions	7
2.1.3	Optimal Policies and Value Functions	8
2.1.4	Value-Based vs Policy-Based Methods	8
2.1.5	Online vs Offline Learning	9
2.1.6	Partially Observable Markov Decision Processs (MDPs)	10
2.2	Data-centric Artificial Intelligence (AI)	10
3	State of The Art	12
3.1	Model-Centric Robustness	12
3.2	Data-Aware Robustness	13
3.3	Data-Centric Robustness	13
4	Methodology	17
4.1	Preliminary Work	17
4.2	Meta Read-Write Reinforcement Learning	19
4.2.1	Methodology Overview	20
4.2.2	Implementation Details	21
4.3	True Read-Write Reinforcement Learning	22
4.3.1	Stabilisation techniques	25
4.3.2	Evolution of Read-Write Reinforcement Learning (RW-RL)	26
4.3.3	Mathematical Formalization: Proof of Convergence	26
5	Experimental Setup	29
5.1	Environments	29
5.2	Perturbations and Partial Observability	31
5.3	Hyperparameters	31
5.4	Evaluation	31
6	Results and Discussion	33
6.1	Vanilla application of Pastprop	33
6.2	Meta and True RW-RL	36
6.3	Discussion	43

7	Conclusions	46
7.1	Main Contributions	46
7.2	Research Questions	46
7.3	Study Limitations	47
7.4	Future Work	48
	References	50
8	Appendices	54
8.1	Appendix 1 - Complete Evaluation Result Tables	54

List of Figures

2.1	Reinforcement Learning Loop.	7
2.2	Online Reinforcement Learning training loop compared to Offline Reinforcement Learning.	10
4.1	Pastprop-RL Architecture.	18
4.2	Meta RW-RL Architecture.	20
4.3	Dynamics of an SSM between the 4 parameter matrices.	22
4.4	True RW-RL Architecture. The value function modifies the collected states to be taught to the policy function.	23
6.1	Preliminary Results: Overall Performance vs. Noise	33
6.2	Performance of each observation dimension as noise increases.	34
6.3	Reward behaviour over one episode over time through the injection of multiples kinds of noise.	35
6.4	Normalized model ranking score on all experiments.	36
6.5	Normalized Mean Reward across training settings.	37
6.6	Mean Reward and CVaR Tail on LunarLander Environment.	38
6.7	Pairwise binomial test between modes for final reward and CVaR.	41
6.8	Pairwise binomial test between modes for Robustness Index and Efficiency.	41
6.9	Learning Curves on HalfCheetah.	42

List of Tables

3.1	Overview of State-of-the-Art Approaches for Robust Reinforcement Learning. . .	16
5.1	PPO hyperparameters per environment.	32
6.1	Average mean reward $\pm$ standard deviation over training noises, at selected evaluation noise levels. Best per environment, mode, and noise level is bolded. . . .	39
6.2	Average mean Conditional Value at Risk (CVaR) $\pm$ standard deviation over training noises, at selected evaluation noise levels. Best per environment, mode, and noise level is bolded.	40
6.3	Average training and inference times relative to the baseline across environments.	43
8.1	Ant-v4, mdp, train noise 0.0 — Mean Reward	54
8.2	Ant-v4, mdp, train noise 0.0 — Cvar	55
8.3	Ant-v4, mdp, train noise 0.1 — Mean Reward	55
8.4	Ant-v4, mdp, train noise 0.1 — Cvar	55
8.5	Ant-v4, pomdp, train noise 0.0 — Mean Reward	56
8.6	Ant-v4, pomdp, train noise 0.0 — Cvar	56
8.7	Ant-v4, pomdp, train noise 0.1 — Mean Reward	56
8.8	Ant-v4, pomdp, train noise 0.1 — Cvar	57
8.9	CartPole-v1, mdp, train noise 0.0 — Mean Reward	57
8.10	CartPole-v1, mdp, train noise 0.0 — Cvar	57
8.11	CartPole-v1, mdp, train noise 0.1 — Mean Reward	58
8.12	CartPole-v1, mdp, train noise 0.1 — Cvar	58
8.13	CartPole-v1, pomdp, train noise 0.0 — Mean Reward	58
8.14	CartPole-v1, pomdp, train noise 0.0 — Cvar	59
8.15	CartPole-v1, pomdp, train noise 0.1 — Mean Reward	59
8.16	CartPole-v1, pomdp, train noise 0.1 — Cvar	59
8.17	HalfCheetah-v4, mdp, train noise 0.0 — Mean Reward	60
8.18	HalfCheetah-v4, mdp, train noise 0.0 — Cvar	60
8.19	HalfCheetah-v4, mdp, train noise 0.1 — Mean Reward	60
8.20	HalfCheetah-v4, mdp, train noise 0.1 — Cvar	61
8.21	HalfCheetah-v4, pomdp, train noise 0.0 — Mean Reward	61
8.22	HalfCheetah-v4, pomdp, train noise 0.0 — Cvar	61
8.23	HalfCheetah-v4, pomdp, train noise 0.1 — Mean Reward	62
8.24	HalfCheetah-v4, pomdp, train noise 0.1 — Cvar	62
8.25	Hopper-v4, mdp, train noise 0.0 — Mean Reward	62
8.26	Hopper-v4, mdp, train noise 0.0 — Cvar	63
8.27	Hopper-v4, mdp, train noise 0.1 — Mean Reward	63
8.28	Hopper-v4, mdp, train noise 0.1 — Cvar	63

8.29 Hopper-v4, pomdp, train noise 0.0 — Mean Reward	64
8.30 Hopper-v4, pomdp, train noise 0.0 — Cvar	64
8.31 Hopper-v4, pomdp, train noise 0.1 — Mean Reward	64
8.32 Hopper-v4, pomdp, train noise 0.1 — Cvar	65
8.33 LunarLander-v3, mdp, train noise 0.0 — Mean Reward	65
8.34 LunarLander-v3, mdp, train noise 0.0 — Cvar	65
8.35 LunarLander-v3, mdp, train noise 0.1 — Mean Reward	66
8.36 LunarLander-v3, mdp, train noise 0.1 — Cvar	66
8.37 LunarLander-v3, pomdp, train noise 0.0 — Mean Reward	66
8.38 LunarLander-v3, pomdp, train noise 0.0 — Cvar	67
8.39 LunarLander-v3, pomdp, train noise 0.1 — Mean Reward	67
8.40 LunarLander-v3, pomdp, train noise 0.1 — Cvar	67

List of Acronyms

AI	Artificial Intelligence
ATLA	Alternating Training of Learned Adversaries
CVaR	Conditional Value at Risk
DRQN	Deep Recurrent Q-Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
HER	Hindsight Experience Replay
HiPPO	High-order Polynomial Projection Operators
i.i.d	Independent and Identically Distributed
LIDAR	Light Detection And Ranging
LiDER	Lucid Dreaming for Experience Replay
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MSE	Mean Squared Error
NN	Neural Network
OOD	Out-Of-Distribution
POMDP	Partially Observable Markov Decision Process
PPO	Proximal Policy Optimization
RARL	Robust Adversarial Reinforcement Learning
RE	Robustness Efficiency
RI	Robustness Index
RL	Reinforcement Learning
RMDP	Robust Markov Decision Process
RNN	Recurrent Neural Network
RW-ML	Read-Write Machine Learning
RW-RL	Read-Write Reinforcement Learning
SARIMA	Seasonal Autoregressive Integrated Moving Average
SAC	Soft Actor-Critic
SARSA	State–Action–Reward–State–Action
SA-MDP	State Adversarial Markov Decision Process

SSM Structured State Model

TD Time Difference

AI Artificial Intelligence

Chapter 1

Introduction

For the vast majority of the modern deep learning renaissance, the scientific community has almost exclusively operated under a model-centric paradigm. In this traditional framework, the dataset is usually treated as an immutable, static resource. A fixed ground truth on which researchers iterate by developing increasingly sophisticated neural network architectures, where the default response to poor performance or distribution shift was to expand parameter counts, hyperparameter optimizations [Jakubik et al. 2024], or introduce heavy architectural regularisations.

This, however, goes with the assumption that the data should accurately and perfectly reflect reality, and these models often rely on this assumption of high-fidelity observations [Zhang et al. 2021]. As these Artificial Intelligence (AI) models transition from highly curated benchmarks into the messy and often unstructured physical world, they often encounter environments plagued with missing values, mislabelling; focusing more on the physical systems, systematic sensor noise and even the reality that physical components degrade and decay over time. Recent work has shown that Deep Reinforcement Learning (DRL) algorithms are often vulnerable to model uncertainty and external perturbations, these, in the format of noisy environments [Huang et al. 2017; Pattanaik et al. 2017].

Knowing that Reinforcement Learning (RL) models, when employed in pristine scenarios, show remarkable success in problem-solving [Mnih et al. 2015], that they necessitate high-quality data to perform accordingly, and that they are sensitive to perturbations, it is only natural that the focus has to shift from enhancing the model architecture, and the data has to be placed under the spotlight.

With this, the data-centric paradigm emerged as a discipline that shifts this burden of performance away from the model, focusing on the rigorous engineering of the data itself, yet, despite the widespread adoption of data-centric principles in many areas such as supervised learning, computer vision and natural language processing, their integration into the field of RL has been comparatively sluggish. This is not unexpected, as it stems from the fundamental structural differences between the former and the latter.

In standard supervised learning pipelines, the dataset consists of Independent and Identically Distributed (i.i.d) examples collected prior to training, and this allows researchers and developers

to employ a vast range of data-centric methodologies with the objective of producing higher-quality data. On the other hand **RL** requires the agent to generate its training data interactively by exploring the environment and acting upon it through trial and error. This creates a non-stationary, highly temporally correlated stream of data, and more critically, the quality of the data is directly dependent on the current policy exploring the environment. Consequently, the vast majority of **RL** research has focused predominantly on algorithmic enhancement, designed to extract maximum value from whatever data it happens to collect.

The successful deployment of **RL** in physical systems also remains severely bottlenecked by the sim-to-real gap, as the discrepancy between predictable simulators and entropic real-world dynamics must be addressed. Problems such as camera glare or occlusion, dead pixels, Light Detection And Ranging (**LIDAR**) sensors producing outliers, mechanical joints drifting in readings, latency between sensors, etc., all contribute to problems that are often overlooked when developing an architecture in a simulator, so in reality, many models operate within a Partially Observable Markov Decision Process (**POMDP**), and they must infer the state of the world through these challenging conditions.

Despite the proliferation of robust reinforcement learning methodologies, ranging from Domain Randomization [Tobin et al. 2017] to Adversarial Training, existing approaches overwhelmingly focus on hardening the policy against noise. Even when contemporary techniques such as Out-Of-Distribution (**OOD**) detection adopt a nominally data-centric posture by curating, filtering, or isolating anomalous datasets to ensure system reliability, they stop short of fundamentally altering the data. They merely select from what is available, treating observation corruption as an immutable environmental hazard that the model must either learn to withstand or simply ignore. Rather than actively rectifying the noisy observations to produce inherently better data, these curation strategies rely on statistical divergence methodologies, such as density estimation [Yang et al. 2024] or ensemble variance [Sedlmeier et al. 2019], which often yield overly conservative policies. More critically, these defensive strategies introduce severe operational limitations during physical deployment. In high-speed, continuous-control systems or highly constrained edge environments, such as autonomous driving, inference latency is a hard, non-negotiable constraint. Achieving robustness by saddling the deployed policy with heavy computational overhead or test-time anomaly filters is structurally incompatible with real-world, real-time control.

The realization that the quality of a model is fundamentally a reflection of its underlying data [Baptista et al. 2021] necessitates a profound paradigm shift: treating the experience replay buffer not as a static, read-only repository of raw observations, but as an active, optimizable workspace. Foundational steps have been taken in this direction; for instance, Hindsight Experience Replay (**HER**) [Andrychowicz et al. 2018] alters stored goals to re-contextualize failed trajectories as successful ones, proving that strategic data manipulation within the buffer can drastically improve learning efficiency. True Read-Write Reinforcement Learning (**RW-RL**) pushes this philosophy to its mathematical conclusion by systematically making the stored data better. By integrating data correction directly into the policy model’s backpropagation, specifically leveraging Bellman error gradients to retroactively rectify corrupted state transitions, the raw states

themselves become another variable to optimize. This allows the data and the value function to co-evolve symbiotically during the offline training loop, actively smoothing out sensor noise and ensuring that emergent, state-of-the-art robustness is achieved with absolutely zero test-time inference overhead.

1.1 Thesis Statement

Building on these ideas, we introduce Read-Write Reinforcement Learning (**RW-RL**), a data-centric paradigm that adapts the read-write philosophy to **RL**. Instead of only hardening the policy architecture or relying on auxiliary anomaly detectors, a True **RW-RL** agent uses the gradients of its own value function to retroactively correct corrupted state transitions stored in the experience replay buffer. The correction is applied online, in a self-supervised manner, using only the agent’s temporal-difference error. Importantly, as our method aims to solely modify training data in order to improve robustness, it has no active correction component during inference, meaning that, when a True **RW-RL** agent is deployed, it effectively has zero inference overhead.

The central goal of this thesis is to demonstrate that such a self-directed correction mechanism is sufficient to improve robustness under observation noise, especially in partially observable settings. To evaluate this claim, we seek to answer the following research questions:

- **Question 1 (Stability):** Can an **RL** agent interactively correct its own stored transitions – using gradients of its value loss – without destabilising training (i.e., without value divergence or policy collapse)?
- **Question 2 (Robustness):** Does True **RW-RL** produce policies that are significantly more robust to varying levels of observation noise compared to a vanilla agent, particularly when noise mimics partial observability?

By answering these questions, this thesis provides a foundational step toward data-centric, self-correcting RL agents that require no external supervision or auxiliary models.

1.2 Organisation of the Dissertation

After this introduction, we’ll start in Chapter 2 with some background on **RL**, breaking down its core building blocks such as Markov Decision Processes (**MDPs**), the distinctions between value-based and policy-based methods, online and offline **RL**, and the challenges introduced by partial observability. In this chapter we also briefly introduce the concept of data-centric AI, establishing the implications of data quality in regards to model performance.

From there, in Chapter 3, we will delve into the current state of the art surrounding robustness in **RL**. This chapter is divided in three primary categories of methodologies, beginning with the model-centric approaches that aim to make policies more robust regardless of the available data, data-aware methods that curate the available data through filtering or re-weighting of the samples,

and then data-centric approaches that aim to augment or ultimately alter existing data. We compare and discuss these methodologies and their limitations, and outline how they differ from our **RW-RL** vision.

In Chapter 4, we walk through the work carried out over the course of the thesis, tracing step by step all the developments and experiments that led to our main contribution. We begin with our initial hypothesis and early explorations, followed by the progression into Meta **RW-RL** using different backbones, and then we culminate in the formalization of True **RW-RL**.

After this, in Chapter 5, we outline the experimental setup developed to evaluate our developments, detailing the chosen environments, perturbation strategies to test model resilience and evaluation metrics to quantify performance.

Finally, Chapter 6 we end by presenting the empirical results, comparing True **RW-RLs** capabilities across the remaining baselines across all experimental modalities. The discussion analyses the achieved results, and then we draw the final conclusions regarding the proposed methodologies and future work that can be produced in Chapter 7.

Chapter 2

Background

RL is a subfield of **AI** in which an agent learns to make decisions by interacting with an environment, aiming to maximize cumulative reward over time [Sutton and Barto 1998].

Agents are not provided with explicit ground-truth answers; instead, they must learn which actions yield desirable outcomes by receiving feedback in the form of rewards. In **RL**, data is inherently sequential and interactive, in which the outcome of an agent's decision influences the future states it encounters.

This temporal dependency introduces unique challenges and opportunities for modeling complex behaviours. At its core, an **RL** agent selects actions based on observations, receives rewards, and uses this feedback to refine its strategy over time, and these interactions form trajectories of state-action pairs, which the agent analyzes to learn policies that maximize reward.

A key distinction in **RL** is the difference between observation and state. Observations are what the agent perceives (e.g., its field of view), while the state refers to the true underlying condition of the environment, which may not be directly observable. Policies that dictate the agent's actions can be either deterministic (always selecting the same action for a given state) or stochastic, assigning probabilities to possible actions.

To quantify the desirability of actions, **RL** relies on reward functions, which evaluate how good a particular action is in a given state. However, the objective is not to maximize immediate rewards alone, but rather to learn strategies that lead to high long-term rewards. This concept is formalized through **MDPs**, which models the environment as a sequence of states, actions, transitions, and rewards. Continuing, **RL** problems are often divided into finite- and infinite-horizon cases. In finite-horizon settings, the agent optimizes over a fixed number of steps. In contrast, infinite horizon problems require careful handling, as optimizing for an infinite number of steps is computationally and, realistically, intractable. This handling is typically done through discounted rewards.

The process of reinforcement learning involves generating experience by interacting with the environment, estimating expected returns from these experiences, and updating the policy accordingly. Various algorithmic approaches, ranging from value-based to policy-gradient and actor-critic methods, differ in how they represent and update these policies and value estimates, each

with its own trade-offs in terms of sample efficiency, stability, and assumptions.

DRL builds on ideas from classical deep learning by using neural networks to represent complex policies and value functions, enabling researchers to tackle more sophisticated and complex problems ([Mnih et al. 2015]).

In supervised learning, neural networks are trained on labeled data to make predictions, but in **DRL**, the learning comes from trial and error. Agents learn from feedback that is often delayed or sparse, which makes training more difficult.

Still, many deep learning techniques, such as convolutional networks for visual input and recurrent networks for handling sequences, are useful in **DRL** too. These tools enable **DRL** agents to handle large, complex state or action spaces that would be too difficult to manage with traditional **RL** methods.

2.1 Reinforcement Learning

In this section, we will formalize each relevant theoretical concept to further elaborate on the proposed work.

Reinforcement learning problems are formally modeled as **MDPs**. An **MDP** is defined by the tuple (S, A, P, R, γ) , where:

- S is the state space, representing all possible states the environment can be in
- A is the action space, representing all possible actions the agent can take
- P is the state transition probability function, where $P(s'|s, a)$ denotes the probability of transitioning to state s' after taking action a in state s
- R is the reward function, where $R(s, a)$ gives the expected immediate reward for taking action a in state s
- $\gamma \in [0, 1]$ is the discount factor, which determines the present value of future rewards

The Markov property is fundamental to this framework: the probability of transitioning to the next state depends only on the current state and action, not on the history of previous states and actions. Mathematically, this is expressed as:

$$\begin{aligned} \mathbb{P}[s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0] &= \mathbb{P}[s_{t+1} | s_t, a_t] \\ &= P(s_{t+1} | s_t, a_t) \end{aligned} \tag{2.1}$$

2.1.1 Policies

A policy π defines the agent's behaviour by mapping states to actions. For a stochastic policy, $\pi(a|s)$ represents the probability of selecting action a when in state s . For all states $s \in S$, the policy must satisfy:

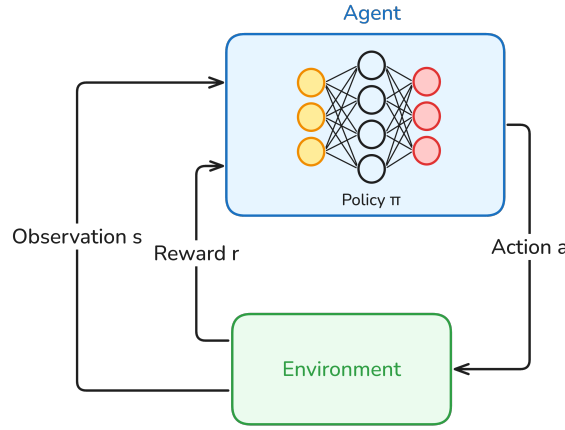


Figure 2.1: Reinforcement Learning Loop.

$$\sum_{a \in A} \pi(a|s) = 1 \quad (2.2)$$

A deterministic policy can be viewed as a special case where $\pi(a|s) \in \{0, 1\}$ for all state-action pairs.

2.1.2 Return and Value Functions

The return G_t at time step t is the cumulative (and discounted, in case of $\gamma < 1$) reward from that time onwards:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2.3)$$

where k is the time offset indexing future time steps, and r_{t+k+1} is the reward received at time $t+k+1$. The discount factor γ ensures that the infinite sum converges when $\gamma < 1$, and encodes a preference for immediate rewards over delayed ones, as k increases. Rewards further in the future are weighted less heavily due to the exponential decay factor γ^k .

The state-value function $V^\pi(s)$ represents the expected return when starting in state s and following policy π :

$$V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] \quad (2.4)$$

Similarly, the action-value function (or Q-function) $Q^\pi(s, a)$ represents the expected return when starting in state s , taking action a , and thereafter following policy π :

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \quad (2.5)$$

The state-value function $V^\pi(s)$ tells us how good it is to be in a particular state when following policy π , while the action-value function $Q^\pi(s, a)$ tells us how good it is to take a specific action

a in state s and then follow policy π thereafter. While $V^\pi(s)$ averages over all possible actions according to the policy, $Q^\pi(s, a)$ evaluates a specific action before committing to the policy. The relationship between them is:

$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s, a) \quad (2.6)$$

This equation shows that the value of being in state s is simply the weighted average of the values of all actions available in that state, where each action's value is weighted by the probability that the policy selects it.

2.1.3 Optimal Policies and Value Functions

The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected return from any starting state. A policy π^* is optimal if:

$$V^{\pi^*}(s) \geq V^\pi(s) \quad \forall s \in S, \forall \pi \quad (2.7)$$

The optimal state-value function is:

$$V^*(s) = \max_{\pi} V^\pi(s) \quad (2.8)$$

and the optimal action-value function is:

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a) \quad (2.9)$$

Once the optimal action-value function Q^* is known, an optimal deterministic policy can be extracted by:

$$\pi^*(s) = \arg \max_{a \in A} Q^*(s, a) \quad (2.10)$$

2.1.4 Value-Based vs Policy-Based Methods

Reinforcement learning algorithms can be categorized into value-based and policy-based approaches, which differ fundamentally in how they represent and optimize the agent's behaviour.

Value-based methods learn an estimate of the value function (either $V^\pi(s)$ or $Q^\pi(s, a)$) and derive a policy implicitly from these values. The policy is typically greedy with respect to the learned value function:

$$\pi(s) = \arg \max_{a \in A} Q(s, a) \quad (2.11)$$

These methods focus on learning which states or state-action pairs are valuable, and then select actions that lead to high-value states. Value-based methods are particularly effective in discrete action spaces but can struggle in continuous or high-dimensional ones.

Policy-based methods, in contrast, directly parameterize the policy $\pi_\theta(a|s)$ with parameters θ and optimize it to maximize expected return. The objective function to maximize is:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G_0] \quad (2.12)$$

where τ represents a trajectory sampled from the policy. Policy gradient methods update the parameters in the direction of the gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right] \quad (2.13)$$

There are also **Actor-critic methods**, which combine the two approaches by maintaining both a policy (the actor) and a value function (the critic). The critic evaluates the current policy, while the actor updates it based on the critic's feedback.

2.1.5 Online vs Offline Learning

Online learning refers to settings where the agent learns while actively interacting with the environment. The agent collects data under its current policy, uses it to update its policy or value estimates, and immediately deploys the updated policy for further data collection. This creates a feedback loop:

$$\pi_t \rightarrow \text{collect } D_t \rightarrow \text{update} \rightarrow \pi_{t+1} \quad (2.14)$$

where $D_t = \{(s, a, r, s')\}$ represents transitions collected under policy π_t . Online learning is characterized by non-stationary data distributions, as the policy continuously evolves during training. Algorithms like Q-learning, State–Action–Reward–State–Action (**SARSA**), and on-policy policy gradient methods are designed for online settings.

A key challenge in online learning is the exploration-exploitation trade-off: the agent must balance gathering new information about the environment (exploration) with leveraging current knowledge to maximize reward (exploitation). This is often addressed through strategies such as ϵ -greedy action selection:

$$\pi(a|s) = \begin{cases} \arg \max_{a'} Q(s, a') & \text{with probability } 1 - \epsilon \\ \text{uniform random} & \text{with probability } \epsilon \end{cases} \quad (2.15)$$

or through entropy regularisation in policy gradient methods.

Offline learning (also known as batch reinforcement learning or offline **RL**) operates on a fixed dataset D collected by one or more behaviour policies π_β , without further interaction with the environment during training:

$$D = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N \sim \pi_\beta \quad (2.16)$$

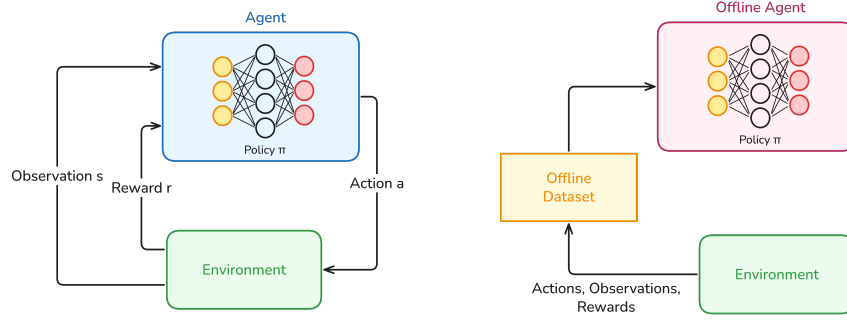


Figure 2.2: Online Reinforcement Learning training loop compared to Offline Reinforcement Learning.

The agent must learn an optimal policy π^* solely from this static dataset. This setting is crucial for domains where online interaction is costly, dangerous, or impractical, such as healthcare, robotics with real hardware, or autonomous driving.

Offline learning introduces unique challenges. The primary difficulty is the distribution shift, where the learned policy π may select actions that are not well represented in the data set, leading to unreliable value estimates. This is formally characterized by the support mismatch between the behaviour policy and the learned policy.

$$\exists(s, a) : \pi(a|s) > 0 \text{ but } \pi_{\beta}(a|s) \approx 0 \quad (2.17)$$

To address this, offline **RL** algorithms typically incorporate conservative value estimation or policy constraints to prevent the agent from selecting out-of-distribution actions. Common approaches include pessimistic value updates, policy regularisation toward the behaviour policy, or uncertainty quantification.

2.1.6 Partially Observable MDPs

In many practical scenarios, the agent does not have access to the environment's true state but only receives observations. This situation is modelled as a **POMDP**, defined by the tuple $(S, A, P, R, \gamma, O, Z)$, where O is the observation space and $Z : S \times O \rightarrow [0, 1]$ is the observation probability function, with $Z(o|s)$ representing the probability of observing o when in state s .

In **POMDPs**, policies map observation histories to actions rather than directly to states, making the learning problem significantly more challenging. The agent must infer or maintain beliefs about the underlying state from the sequence of observations.

2.2 Data-centric AI

The generally dominant paradigm in the history of machine learning has always been model-centric: Given a fixed dataset, the primary effort was always directed into producing and tuning the model: its architecture, loss functions, optimization algorithm and hyperparameters. This

approach has yielded remarkable successes, from deep neural networks to large language models; However, this implicitly treats the data as an immutable resource.

Recently, the data-centric paradigm has gained more traction, arguing that improving the dataset itself would yield a better performing model than further model tuning.

Empirical benchmarks have shown that data-centric interventions, such as cleaning mislabelled examples, augmenting scarce classes, correcting systematic biases, etc., often produce models that outperform the model-centric optimizations in practical applications. In essence, the traditional "garbage in, garbage out" is the main focus when developing a learning pipeline, and with good data curation, even a simple model can generalize well [Renggli et al. 2021].

In the case of RL, this challenge, despite analogous, is structurally more complex. Unlike supervised learning, where data points are independent and drawn from a fixed distribution i.i.d, RL agents generate their training data through interaction with the environment. This data is non-stationary and temporally correlated. Moreover, the data is often expensive, as each transition may require physical environments and time to be collected, human demonstration, or high-fidelity simulation.

Given these constraints, one might expect RL to be a natural beneficiary of data-centric approaches, but surprisingly, the vast majority of research remains model-centric, focused mainly on policy representation (such as deep policies of transformer-based architectures), value function approximation, optimization algorithms and sampling strategies. In these contexts, the data itself is always treated as read-only, and the agent has no mechanism to correct or improve its memories, may these be corrupted by sensor noise, or a reflection of poor early policies.

One particularly important dimension in RL is robustness to observation noise. In real-world deployment, sensors are often imperfect: cameras produce noisy images, and the sensor may eventually degrade to the point of producing dead pixels, LIDAR sensors may collect outliers, joints may suffer from drift, etc. A policy trained on a cleaned simulation may fail catastrophically when evaluated with slight observation noise [Zha et al. 2023].

Traditional robustness methods are very much model-centric: they add noise during training [Tobin et al. 2017], regularize the policy to be smooth, or through adversarial training to generate worst-case scenarios during training [Pinto et al. 2017]. While effective, these approaches do not alter the fundamental data the agent learns from, they change the learning objective or the environment dynamics, but the replay buffer will still contain the original and potentially suboptimal transitions.

The data-centric alternative explored in this thesis is to directly improve the quality of the stored data. If the agent can learn to correct its own experiences during training, it may become robust as an emergent property of the policy, without modifying the policy objective or requiring knowledge of the noise distribution.

Chapter 3

State of The Art

When it comes to Robust **RL**, the core objective is to produce a policy that maintains high performance when the environment deviates from the ideal conditions. All the families of work reviewed in this chapter share this same objective, and yet they differ fundamentally in where they concentrate the source of robustness, and in turn, in what they leave to be desired. We will begin by going through model-centric methodologies that enhance a policy to withstand uncertainty, to data-centric methodologies that directly manipulate data to improve sample efficiency and model robustness. Then, we'll position the proposed **RW-RL** methodology in terms of novelty compared to the current existing methodologies.

3.1 Model-Centric Robustness

The dominant strategies for building **RL** agents usually place the burden of robustness on the policy, treating the collected data/observations as a fixed, read-only resource. A foundational approach in this domain is the Robust Markov Decision Process (**RMDP**), which assumes that transition probabilities belong to a predefined uncertainty set rather than being precisely known [Nilim and Ghaoui 2005], [Suilen et al. 2024]. By optimizing for the worst-case scenario within this set, **RMDPs** provide theoretical guarantees for robustness, though they often yield overly conservative policies and scale poorly to high-dimensional state spaces.

To address these scaling issues, adversarial reinforcement learning methods such as Robust Adversarial Reinforcement Learning (**RARL**) frame the training process as a two-player zero-sum game. In this framework, a protagonist agent learns to maximize rewards while an adversary applies destabilizing forces to the system [Pinto et al. 2017]. While **RARL** improves generalization and zero-shot transfer, it still risks over-pessimism if the adversary is not carefully calibrated.

Other methodologies target specific types of noise, such as State Adversarial Markov Decision Process (**SA-MDP**), which apply a theoretically principled policy regularisation to defend against adversarial perturbations on state observations [Zhang et al. 2021].

Similarly, Action-Robust RL models uncertainty by mixing the agent’s chosen action with a random or adversarial action, promoting implicit regularisation without requiring an explicit definition of the uncertainty set [Tessler, Efroni, and Mannor 2019].

3.2 Data-Aware Robustness

Usually, models use all the data available to learn, but some of this data may not be trustworthy; the model should be able to decide which data to learn from. From this, a family of methods has emerged, that use gradients of the loss function to inform this decision. These techniques are read-only, as they never alter the data itself, but sample or re-weight entries. For instance, Step-E [Du 2025] is a framework that integrates sample selection and model learning into a single optimization process. It computes the gradient w.r.t the loss of each training point, and weights/discards samples depending on its loss; Similarly, Online Gradient-based Robust Selection [Yang, Koppel, and Zhang 2023] operates on a streaming setting, detecting noisy labels by comparing their gradient similarity across samples, excluding outliers along the training process.

While these methods show that gradient information is rich enough to assess data quality, they are confined to supervised settings, and demonstrated inability to handle nonstationary, and most importantly, they never explored the modification of data.

Regarding augmentation, some methodologies such as Model-based Counterfactual Data Augmentation [Pitis et al. 2022] learn a dynamics model from the available data, and then generates counterfactual transitions on the latent variables. These transitions are added to the replay buffer, effectively augmenting the training distribution, and has shown performance improvement in offline and low-data scenarios.

Recent advancements have further explored data-centric buffer management, such as Adaptive Replay Buffers, which address the trade-off between adding more data from diverse sources and suffering from distribution shifts or data heterogeneity, called data-mixing, by distributing sampling weights across entire sequences to prevent the policy from overfitting to noisy and corrupted transitions [Song, Lee, and Park 2026]. This approach demonstrated a consistent capability to mitigate performance degradation, but while it dynamically evaluates and selects data, RW-RL goes a step further by directly modifying and repairing the corrupted values themselves.

3.3 Data-Centric Robustness

In the scenarios where we want to close the sim-to-real gap, Domain Randomization trains policies across many simulator configurations. By sampling varying parameters like physics, lighting, etc., the policy learns invariant features, treating unmodeled real-world dynamics as just another variation of the simulator, guaranteeing a better overall performance when brought to the real world [Tobin et al. 2017].

Standard experience replay treats each stored transition as immutable, however, as the policy improves, old experiences become increasingly stale, they may have been generated by a worse

policy than the current, and are no longer representative of good behaviour. Removing them from the buffer to allow space for more recent data to be fed to the policy does happen, but this can lead to catastrophic forgetting. Instead, this old data could be further corrected to be a better representative of what should've happened.

In standard **RL**, models often struggle to learn in environment with sparse and binary rewards (e.g. a robot arm with the objective of touching an object). Hindsight Experience Replay (**HER**) aims to fix this by re-interpreting failed attempts as a successful attempt at a different goal, actively modifying the experiences in the replay buffer by relabelling the target goals of failed trajectories [Andrychowicz et al. 2018]. Lucid Dreaming for Experience Replay (**LiDER**) [Du et al. 2021], On the other hand, operates by every so often auditing the replay buffer, selecting a past state, and then "dreaming" what the current policy could've done differently back then. This dreaming simulates the complete transition model for a fixed number of steps, producing a new trajectory, and replaces the original trajectory with the new one if it yields a higher reward.

This previous methodology is very similar to the conceptual idea of **RW-RL**, in the sense that the replay buffer should not be read-only, the agent can actively improve its own memories. However, this methodology requires a learned dynamics model in order to allow the policy to "dream", or, to simulate the environment, the actions taken, the resulting state, and the produced rewards completely. The correction is made in the dream, without any interaction with the real environment, and then the correction is applied to the replay buffer. This, in turn, introduces additional training complexity and approximation errors.

To address this, the Read-Write Machine Learning (**RW-ML**) paradigm was introduced by [Baghoussi, Soares, and Mendes-Moreira 2024] in the form of the Corrector**LSTM**, a traditional Long Short-Term Memory (**LSTM**) with the ability to modify training data during learning. It uses a secondary meta-learner, which in this case is a Seasonal Autoregressive Integrated Moving Average (**SARIMA**) model, to forecast the **LSTM**s cell states. When the Euclidean distance between the forecasted and actual cell states exceeds a threshold, the methodology would trigger an iterative correction process to adjust the input data until the discrepancy fell below a correction threshold. Empirical validation showed that Corrector**LSTM** surpassed standard **LSTM**s in forecasting accuracy and anomaly detection.

Building on this foundation, the same authors later proposed Read-Write **LSTM**, which eliminates the separate **SARIMA** meta-learner and instead integrates data modification directly into the backpropagation process. Read-Write **LSTM** computes gradients of the loss function w.r.t the input data after a warm-up period, and uses an adaptive, sample-specific scaling mechanism to update the input values. This gradient-based correction is more efficient, requires no external model, and allows the data to co-evolve with the network weights. The authors demonstrated that Read-Write **LSTM** outperforms standard **LSTM** in over 70% of M4 time series and achieves competitive results on anomaly-rich NAB datasets [Baghoussi 2025].

The core idea of Read-Write **LSTM**, using the model's own gradients to directly modify training inputs, serves as the direct inspiration for the work presented in this thesis. However, Read-Write **LSTM** was designed for time-series forecasting, not for reinforcement learning. The next

chapter describes **RW-RL**, which adapts this gradient-based read-write principle to the **RL** setting, introducing state correction inside the replay buffer with per-sample adaptive scaling, Time Difference (**TD**) error gating, and an online exponential moving average correction for live observations, explicitly targeting robustness to observation noise.

Overall, these methodologies, summarized on table 3.1, aim to build **RL** agents that maintain high performance despite environmental uncertainty, sensory noise, or changing dynamics. They achieve this through model-centric approaches, to mathematically harden the policy to withstand worst-case scenarios and adversarial perturbations without altering the training data; data-aware methods use statistical or gradient-based information to filter and selectively weight trustworthy experiences while treating the underlying data as immutable; and data-centric methods actively augment or retroactively modify the experiences stored in the replay buffer using randomized physics or external dynamics models.

The overall findings of these methodologies indicate that while prior approaches work well to theoretically bound uncertainty, but they fundamentally falter when forced to handle raw observation noise without sacrificing critical sample efficiency, inference overhead or the necessity of external world/dynamics models that are assumed to be adequate for the task. Prior methods work by treating data as an immutable truth that must be cautiously navigated, filtered, or rewritten through external means.

Pushing this concept further, the emerging Read-Write paradigm breaks the assumption of data immutability by allowing the neural network to use its own internal backpropagation gradients to actively detect and mathematically correct raw, corrupted input observations during the learning process itself.

Table 3.1: Overview of State-of-the-Art Approaches for Robust Reinforcement Learning.

Method	Design	Limitations
Robust MDPs	Maximize expected return against worst-case transition dynamics within a predefined uncertainty set.	Often produces conservative policies. Scales poorly in higher dimensional environments.
Robust Adversarial RL	Employs a min-max framework where the adversary actively destabilizes the agent during training	Prone to over-pessimism, requires careful balancing of the adversarial reward structure.
SA -MDPs	Introduces policy regularisation to withstand adversarial perturbations on observations.	Enforcing this robustness during training requires calculating the worst-case perturbations and applying the regularisation penalties at every step.
Action-Robust RL	Noise injection during training.	Ignores inherent transition uncertainties.
Step-E	Uses gradient metrics to assess data quality and dynamically down-weight or discard noisy samples.	Uses the own gradient to do data sampling, but treats this data as immutable.
Adaptive Replay Buffer	Distributes sampling weights across the whole sequence instead of isolated steps.	Focus on transitioning models from online to offline settings. Alters data sampling frequency, but treats this data as immutable.
Counterfactual Data Augmentation	Trains a world model to produce counterfactual transitions to feed the agent.	Heavily dependent on the performance of the external world model.
Domain Randomization	Exposes the model to a wide distribution of physics parameters to bridge the sim-to-real gap.	Method is sensitive parameter-wise.
Hindsight Experience Replay	Modifies goals in failed trajectories to increase sample efficiency	Designed for sparse rewards, not focused on correction.
Lucid Dreaming	Learn environment dynamics, then go back into old data and "dream" better actions or latent states, then, modify the old data.	Computationally heavy; still operates on learned representations, not raw observation correction; can amplify model errors.
Read-Write LSTM	Uses the model's own gradients to directly modify and correct training inputs during backpropagation.	Designed exclusively to time-series forecasting.
Read-Write RL	Adapt gradient-based corrections for RL , modifying state observations directly during training.	Functionality and limitations are the primary subject of evaluation in this thesis.

Chapter 4

Methodology

The core objective of this thesis is to develop an efficient, data-centric correction mechanism, coupled to the policy network, that is capable of mitigating the impact of sensor degradation and anomalous behaviour during inference. To achieve this, we initially ran preliminary experiments to confirm whether this was possible, using an adapted *Pastprop* ([Baptista et al. 2021]) prediction network. After confirming the viability, our attention shifted to developing and implementing the **RW-RL** framework.

4.1 Preliminary Work

The starting point for this research was to implement the core idea of *Pastprop* [2021] into reinforcement learning. In supervised time-series forecasting, *Pastprop* demonstrated that modifying the training data using gradients of the model’s own loss can improve predictive accuracy, particularly when the data contains anomalies or disruptions. This is a distinctly data-centric intervention: instead of designing a more robust architecture, the algorithm reshapes the input to better align with the model’s understanding of the environment.

We hypothesized that a similar principle could benefit reinforcement learning. **RL** agents are notoriously sensitive to noise in observations ([Lee 2025]). A replay buffer contaminated by sensor glitches, outliers, or even task-irrelevant distractors can degrade value estimation and policy learning. *Pastprop* offers a less explored path: correcting the data itself using the agent’s own knowledge.

Pastprop extends backpropagation by assigning part of the prediction error to the input data rather than solely to the network weights. For a given input sequence, after a forward pass through an **LSTM**, the algorithm computes the gradient of the loss Err_t with respect to the hidden input h_{int} at each time step. The hidden input is a concatenation of the current input sample X_t , the previous hidden state $h_{\text{out}_{t-1}}$, and a bias term: $h_{\text{int}} = \{X_t, h_{\text{out}_{t-1}}, b\}$. Using the chain rule, *Pastprop* isolates the component of this gradient that corresponds to X_t :

$$\frac{\partial \text{Err}_t}{\partial h_{\text{int}}}[X_t] = \frac{\partial \text{Err}_t}{\partial (h_{\text{int}} W_i)} W_i + \frac{\partial \text{Err}_t}{\partial (h_{\text{int}} W_f)} W_f + \frac{\partial \text{Err}_t}{\partial (h_t W_o)} W_o + \frac{\partial \text{Err}_t}{\partial (h_{\text{int}} W_g)} W_g, \quad (4.1)$$

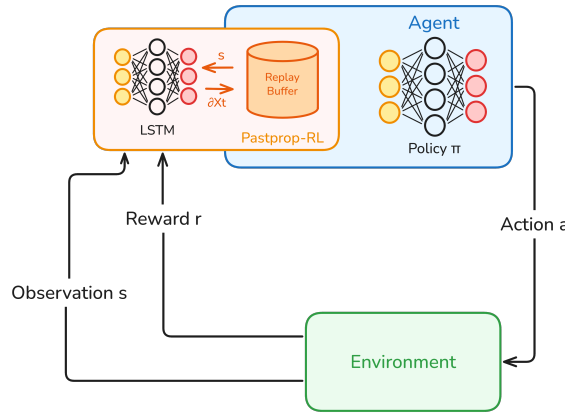


Figure 4.1: Pastprop-RL Architecture.

where W_i, W_f, W_o, W_g are the weight matrices of the input, forget, output, and candidate gates, respectively. The resulting gradient indicates how a small change to the input sample would reduce the prediction error. Pastprop then updates each input sample by subtracting a scaled version of this gradient:

$$X_t \leftarrow X_t - \eta \cdot \frac{\partial \text{Err}_t}{\partial X_t}, \quad (4.2)$$

where η is a hyperparameter called the data correction rate. The original paper proposes three variants: *epoch-wise* (corrections applied once per epoch, averaging overlapping deltas), *instance-wise* (corrections applied immediately), and *selective* (only inputs with large gradients and their temporal neighbours are modified, using a threshold and an epoch embargo to avoid early random noise). The selective variant proved most effective, reducing Mean Squared Error (MSE) by up to 48% on forecasting benchmarks.

Thus, our primary objective in this preliminary work was to answer two questions:

1. Can a correction mechanism of this style be integrated into an **RL** training loop without destabilising learning?
2. Does correcting replay buffer states with a learned environment dynamics model improve performance under increasing observation noise?

To investigate this, we implemented a prototype we call Pastprop-**RL**. As per Figure 4.1, it attaches a trainable **LSTM** component as a callback to a standard RL agent (e.g., Proximal Policy Optimization (**PPO**) or Deep Q-Network (**DQN**)). Periodically, the training loop is interrupted to audit the experience replay buffer. For a sampled batch of transitions, the **LSTM** is trained online to predict the next state given the current state and action, learning the environment’s forward dynamics. After this local training step, we compute a reconstruction residual: the vector difference between the actually observed next state and the **LSTM**’s prediction. These residuals are then used to “nudge” the stored states in the replay buffer towards the physically expected values, by

projecting them into the **LSTM**'s latent representation and back. The corrected states are written back to the buffer before the agent resumes normal policy optimisation.

Crucially, we did not backpropagate the **RL** loss (e.g., value or policy gradient) through the correction; this preliminary version follows the original Pastprop design, where the auxiliary model (the **LSTM**) is trained independently on a reconstruction loss. The **RL** loss is only used to update the policy network.

The experiments conducted in this preliminary phase focused on a controlled noise injection setup. We gradually increased the variance of the Gaussian noise added to the agent's observations and measured the degradation in cumulative reward. Our goal was to observe whether Pastprop-**RL** could slow down this degradation compared to a vanilla agent, and at what noise levels the correction mechanism becomes harmful (e.g., by over-smoothing or introducing bias).

The results are then presented in Chapter 6, revealing a mixed picture. Pastprop-**RL** provided modest protection against low-to-moderate noise, but its benefits vanished, and sometimes reversed, when noise became severe or when the environment had long temporal dependencies. More importantly, we observed that training the auxiliary **LSTM** on a pure reconstruction objective did not guarantee improvements in the **RL** task loss. This discrepancy would later become the central insight motivating our two main contributions: Meta **RW-RL**, which replaces observations with generative model predictions as a stronger baseline, and True **RW-RL**, which directly uses the **RL** loss gradient to correct data, closing the loop that Pastprop opened.

4.2 Meta Read-Write Reinforcement Learning

The concept of leveraging a history of past observations to infer an underlying true state is a classic strategy to mitigate partial observability in **RL** [Kaelbling, Littman, and Cassandra 1998]. When an environment cannot be fully perceived at a single timestep, maintaining a sequential memory allows an agent to build a more complete representation of its environment, akin to Bayesian filtering over hidden states. One of the most prominent demonstrations of this principle in **DRL** was the Deep Recurrent Q-Network (**DRQN**) [Hausknecht and Stone 2017]. They addressed the limitations of standard a **DQN** when faced against unobservability for extended periods of time by replacing the initial fully connected layer of the **DQN** with an **LSTM** network. With this architecture, it was able to accumulate and retain temporal information within its hidden state for longer time horizons, and it allowed the network to infer missing data from past context. Their findings established that a recurrent architecture could serve as a powerful online filter, turning a previously brittle model into one that creates a robust state representation in partial observability scenarios.

Building on this premise of temporal filtering, we produced Meta **RW-RL**. However, rather than embedding the recurrent filter within the agent's network, our approach explicitly decoupled the task of data correction from the task of action selection. To achieve this, Meta **RW-RL** introduces an independent, trainable auxiliary model, situated between the environment and the

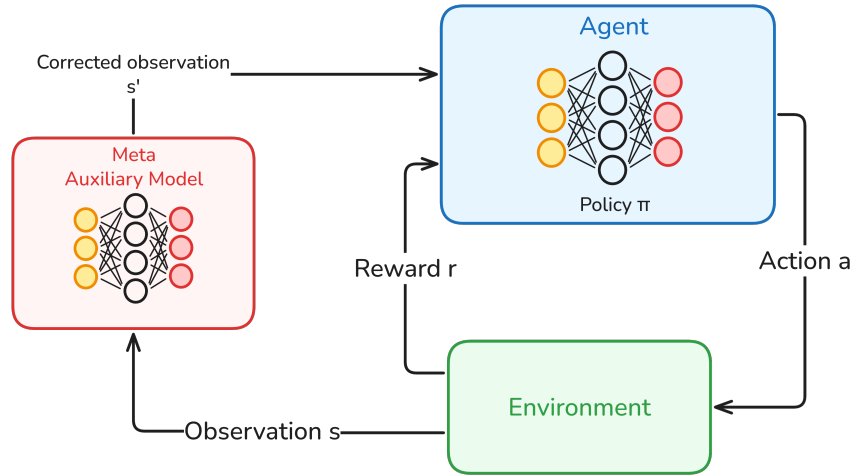


Figure 4.2: Meta RW-RL Architecture.

RL agent. We initially explore this auxiliary-based architecture to establish the viability of decoupled observation filtering, before eventually dropping the auxiliary model entirely to develop True RW-RL, a framework that relies solely on the agent’s own gradients to natively correct data without external dependencies.

4.2.1 Methodology Overview

Meta RW-RL is designed to replace raw, potentially noisy observations from the environment, with a refined prediction generated by an online-trained sequence model. By decoupling the data correction task from the policy network, we dedicate the auxiliary sequence model entirely to handle robustness, allowing the policy to solely focus on optimal action selection.

When the agent observes the environment, this potentially corrupted observation is not directly fed into the policy. Instead, we append this data to a sliding window that maintains a fixed history of recent observations. This sequence is then passed through the auxiliary sequence model, which gets processed to predict the true, underlying state of the environment. Because the efficacy of this prediction relies heavily on the model’s capacity for temporal integration, the framework was designed to utilize and experiment three distinct sequence modeling backbones: a basic Recurrent Neural Network (RNN) baseline, a LSTM network equipped with gating mechanisms to manage extended dependencies, or a continuous-time Structured State Model (SSM) for advanced, rate-invariant sequence processing.

Once the chosen model produces a corrected observation, it is forwarded to the policy network, and from here an action is chosen by the policy to be acted upon on the environment, generating a new observation. Concurrently, these observations are then stored to allow the auxiliary agent to train in a self-supervised manner to predict the next state, and it works during inference as a live correction system.

4.2.2 Implementation Details

Our implementation wraps the standard Gymnasium environment to intercept observations. This wrapper maintains a sliding window of the most recent observations of configurable size, and a separate replay buffer that stores $(window, next_obs)$ pairs for training the sequence model.

During training, for each set of determined steps, we perform AdamW optimization on a random batch of these pairs, masking the last element and setting the model to predict that observation. After this training, we flush half the buffer to make room for new data while keeping some old data for distribution consistency.

During live inference, at every environment step, the wrapper updates the sliding window with the latest raw observation, then calls the model in evaluation mode to predict the next observation. Because we train one step ahead, the cleaned observation used by the policy for time t is actually the prediction made from the window ending at $t - 1$. In practice, we implement this by shifting the window and calling the model to predict the current observation, which returns the model’s prediction for the last element. This yields a causal, low-latency filter.

4.2.2.1 Evolution of the Backbone: From RNN to SSM

As Meta **RW-RL** requires an auxiliary model to correct the data, the choice of the backbone architecture is critical. During the development of this methodology, we evaluated three different sequence models, each addressing a weakness of the previous one.

RNN The initial choice and the simplest, a one-layer **RNN** backbone with a linear decoder. While fast to train, it suffered from the vanishing gradient problem ([Bengio, Frasconi, and Simard 1993]), usually when the horizon exceeded 20-30 steps.

Many environments require the agent to handle delayed rewards, which occur when an agent’s actions yield feedback after hundreds of steps; in the case of the **RNN**, this information would’ve been lost. Consequently, the **RNNs** predictions become noisy, and the filtered observations degrade performance under moderate noise.

LSTM The **LSTM** introduces gating mechanisms (input, forget, and output gates) and a separate cell state that controls the flow of gradients. This alleviates the vanishing gradient problem and allows the model to retain information over hundreds of timesteps. The **LSTM** outperformed the **RNN**, but it has one practical and one theoretical drawback: in practice, training an **LSTM** is slow; moreover, when applied to a continuous-time environment where physical system observations are not sampled at perfectly regular intervals, the discrete nature of the **LSTM** can be a limitation.

SSM With this in mind, **SSMs** were the ideal choice for the backbone, as they effectively solve the long-horizon problem while allowing the Meta **RW-RL** framework to learn a continuous process, owing to their invariance to the sampling rate.

A continuous-time **SSM** assumes that the dynamics of a system can be predicted at time t through two equations:

$$h'(t) = Ah(t) + Bx(t) \quad (4.3)$$

$$y(t) = Ch(t) + Dx(t) \quad (4.4)$$

Equation 4.3 is called the state equation and represents the latent state representation of the environment, while equation 4.4 is the output equation, which produces an output based on the input at time t . At the core of these equations, h is the current state representation, x is the input, and the matrices A, B, C, D are the learnable parameters of the **SSM**.

As shown in Figure 4.3, the state equation describes how the state changes (through matrix A) and how the input influences the state (matrix B); the output equation describes how the state is translated to an output (matrix C) and how the input directly influences the output (matrix D) [Grootendorst 2024].

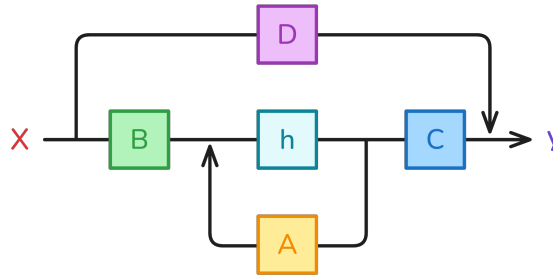


Figure 4.3: Dynamics of an SSM between the 4 parameter matrices.

The initial implementation of our **SSM** performed very poorly. Analysis showed that it suffered from exploding gradients. To address this, we upgraded the implementation to incorporate the High-order Polynomial Projection Operators (**HiPPO**) theory of continuous-time memorization [Gu et al. 2020]. When this class of matrices is used to initialise equations 4.3 and 4.4, the state $h(t)$ can memorise the complete history of the input $x(t)$. Although these equations describe a continuous-time system, RL environments often provide data at discrete time steps. Therefore, to use SSMs in a discrete setting, we discretise their continuous parameters using a pre-determined time step Δ , which represents the resolution of the input signal.

4.3 True Read-Write Reinforcement Learning

Instead of employing a separate meta-learner to identify and correct anomalies, and analogous to the previously proposed method Read-Write **LSTM** [Baghoussi 2025], True **RW-RL** incorporates the data correction mechanism directly into the backpropagation of the original policy model, by modifying the transitions (s, a, r, s', d) stored in the replay buffer before training. It utilizes the

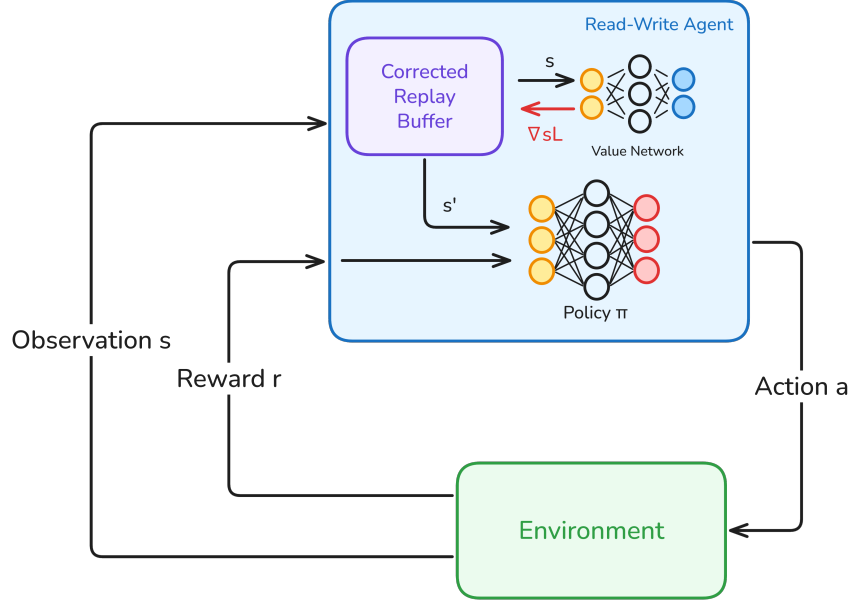


Figure 4.4: True **RW-RL** Architecture. The value function modifies the collected states to be taught to the policy function.

gradients of the agent's own value function to correct the stored states, ensuring the buffer aligns with the perceived reality learned by the policy.

We compute the **TD** target and the **MSE** loss:

$$L = \frac{1}{N} \sum (V(s) - (r + \gamma V(s')(1 - d)))^2, \quad (4.5)$$

where the $V(\cdot)$ is the current value function, γ is the discount factor, and d dictates if it is a terminal state.

We then compute how the loss changes with respect to each stored states s :

$$\nabla_s L. \quad (4.6)$$

Moving s in the negative direction of this gradient reduces the loss:

$$s \leftarrow s - \eta \nabla_s L. \quad (4.7)$$

The initial implementation applied this rule uniformly to all transitions with a fixed step size η . However, that naive approach often led to harmful feedback loops: the buffer and the value function could drift jointly, amplifying spurious corrections and degrading stability.

At first glance, moving a stored state s so that $V(s)$ approaches $r + \gamma V(s')$ seems circular, as we are changing the data to fit a value function that will fit the data again, which on itself could be imperfect. In reality, the crucial detail is that the target $r + \gamma V(s')$ contains the real observed reward r and the observed next state s' , neither of which is ever modified. Hence, in a way, the

changes we are doing to the data becomes analogous to *"What state s would better explain the outcome s' with reward r ?"*.

If the value function is already reasonable, deviations between the raw observation and the Bellman equation are often due to sensor noise and irrelevant variations. By nudging the state back towards Bellman consistency, the algorithm effectively acts as a self-consistency filter.

Now, why should a policy trained on these corrected states generalize well to uncorrected ones during inference? The answer lies in the combination of keeping the corrected state s consistent with the outcome (r, s') . Over thousands of updates, the policy and value function are exposed to these self-consistent versions of observations, meaning that it will build a representation of decisions based on the structure of the data that matters for the task, not on the brittle superficial features that could influence decision-making.

During inference, the noise appears again, but as the model has already learned to map these noisy inputs onto the same representation that the "ideal" s would have, it effectively internalizes the correction process into its own forward pass. In the end, we produce a policy that is naturally robust to the kind of distortion the buffer corrections were designed to ignore, without requiring any test-time intervention.

Consider the following example:

We have two noisy states s_{raw1} and s_{raw2} , both close in Euclidean space, but whose noise causes the TD error to fluctuate wildly in opposite directions (e.g, one would indicate a sensor reading to be very high, and the other would indicate the sensor reading to be very low). Training on these two s_{raw} would mean the value function sees conflicting information, and fitting both would lead to a high variance, fragile representations, and in turn it would try to latch onto the noise (the only varying factor between both s_{raw}) to explain the inconsistency between states.

Applying the correction on both s_{raw} , s_{corr1} and s_{corr2} would now become versions of the states that make the same underlying state more consistent with the Bellman target. The noise-induced TD error shrinks, and as a result:

- The value function represents the problem in a clearer, more self-consistent way.
- The policy receives value estimates that are more stable, learning and producing actions with lower-variance.

In the event that a particular correction pushes s_{corr} away from the true underlying state, the resulting policy would likely make worse decisions, led by a weaker value function vantage estimation, producing lower rewards and poorer outcomes. These environment failures are stored in the replay buffer alongside the real rewards and next states, which again, are never altered.

The value function then must be updated to fit this new, objectively negative data, causing the estimate of the affected regions to become more accurate (or at least more pessimistic). Consequently, the TD errors of those transitions will change, and the future corrections will be pulled in a direction that better represents what s should've been when it got corrected. This way, changing

s while not modifying (r, s') serves as a persistent anchor that keeps the value function from over-correcting, and allows the system to self-stabilize towards a buffer that is internally self-consistent and grounded to reality.

4.3.1 Stabilisation techniques

To make the corrections safe and effective, the following mechanisms are added.

TD-error gating Only transitions whose absolute TD error exceeds a threshold τ are eligible for correction. Let $\delta_i = V(s) - (r + \gamma V(s')(1 - d))$, whenever $|\delta| > \tau$, we would accept the correction, and it would be 0 otherwise. This focuses corrections on “surprising” or poorly predicted transitions, while leaving well-fitted data untouched.

Adaptive correction rate To keep the magnitude of the state change under control, the step size is scaled automatically. Let $\bar{g}$ be the mean L_2 norm of $\nabla_s L$ over the batch,

$$\bar{g} = \frac{1}{N} \sum_{i=1}^N \|\nabla_s L\|_2. \quad (4.8)$$

The global step size is set as

$$\eta = \frac{\gamma_{\text{target}}}{\bar{g} + \varepsilon}, \quad (4.9)$$

where ε is a small constant for numerical stability. γ_{target} is a *target norm* that decays linearly from γ_{start} to γ_{end} over the first T_{decay} environment steps. This schedule allows larger corrections early in training, when the value function is still evolving, and progressively tighter updates later on. The raw update is then

$$\Delta s_{\text{raw}} = -\eta \nabla_s L. \quad (4.10)$$

The negative sign moves states in the direction that reduces the value loss, i.e., toward regions that the current value function judges as higher-value.

Momentum smoothing To dampen oscillations, we maintain a momentum buffer Δs_{prev} (initialised to zero) and update it with coefficient μ :

$$\Delta s = \mu \Delta s_{\text{prev}} + (1 - \mu) \Delta s_{\text{raw}}. \quad (4.11)$$

We use $\mu = 0.9$ in all experiments. The final increment is clipped per dimension to a maximum absolute change Δ_{max} and applied to the stored states:

$$s \leftarrow s + \text{clip}(\Delta s, -\Delta_{\text{max}}, \Delta_{\text{max}}). \quad (4.12)$$

Warmup Corrections are enabled only after a warmup phase of 50000 timesteps. During this period the value function can develop a reasonable estimate of the value landscape, preventing early, uninformed modifications of the replay buffer.

Asymmetric correction Empirically, correcting only the current states s and leaving the next states s' unchanged proved more stable. Modifying both s and s' introduced coupled perturbations that frequently destabilised training.

4.3.2 Evolution of RW-RL

Before arriving at the final formulation of True RW-RL described previously, we experimented several alternative design choices. These earlier implementations, despite conceptually aligned with the objectives of the read-write paradigm, proved inconsistent and sometimes detrimental to performance. Understanding their failures motivated the implemented stabilizations that define the final method.

4.3.2.1 Policy Network vs. Value Network

Our initial implementation used the policy gradient loss instead of the value loss. Specifically, we computed the log-probability of the taken action:

$$L_{policy} = -\log \pi(a | s) \quad (4.13)$$

and corrected the stored states by following $\nabla_s \log(\pi | s)$. The intuition was that moving states in the direction that maximized action certainty would yield higher robustness when the same scenarios appeared during inference. However, this approach proved highly unstable. The method failed to improve robustness in most settings, often causing the policy to land on suboptimal local minima instead, as achieving expert performance on each environment seemed impossible.

4.3.2.2 Correcting The Whole Transition

The initial idea was to propagate the correction along both the current state s , and the following state s' . The intended behaviour was to correct the entire transition, however, correcting both states introduced a cascading effect where the target $r + \gamma V(s')$ became dependent on a corrected state, creating a causal feedback loop where corrections would further influence corrections, which in practice would lead the value estimation to drift and collapse training.

4.3.3 Mathematical Formalization: Proof of Convergence

In order to prove that True RW-RL successfully converges, we need to prove that the stored states that are corrected through $V(s)$ move closer to the target values $r + \gamma V(s')$, and that these states update the value function parameters θ to better fit the value function.

Consider a fixed batch of N transitions. The current value function is $V_\theta(s) = \Phi(s)^\top \theta$. For each transition i we compute the TD Error:

$$\delta_i = V_\theta(s_i) - (r_i + \gamma V_\theta(s'_i)(1 - d_i)) \quad (4.14)$$

And the loss we want to minimize is the Mean Squared TD Error over the batch:

$$\mathcal{L}(\theta, S) = \frac{1}{N} \sum_{i=1}^N \delta_i^2 \quad (4.15)$$

With $S = (s_1, \dots, s_N)$ being the states of the stored transitions that we will modify.

During the correction step, we calculate the gradient w.r.t a state s_i :

$$g_i = \nabla_{s_i} \mathcal{L} = 2\delta_i \nabla_s V_\theta(s_i) \quad (4.16)$$

And the initial naive approach would be to just $s_{i_{corr}} \leftarrow s_i - \eta g_i$, where η is a fixed step size. However, raw gradients can vary wildly in magnitude across transitions, so in order to keep the size of the update under control, we calculate an adaptive step size through the hyperparameter γ_{target} , controlling the effective value change:

$$\bar{g} = \frac{1}{N} \sum_{i=1}^N \|g_i\| \quad \eta = \frac{\gamma_{target}}{\bar{g} + \epsilon} \quad (4.17)$$

Now, assuming that the gradient of the loss w.r.t the stored states is Lipschitz continuous, a Taylor expansion tells us how the loss changes after a small state update:

$$\mathcal{L}(S_{corr}) \approx \mathcal{L}(S) + \sum_i g_i^\top \Delta s_i + \frac{1}{2} \Delta S^\top H \Delta S, \quad (4.18)$$

where H is the Hessian. As $\nabla \mathcal{L}(S)$ is Lipschitz continuous, when the step size is small, the change in gradient is capped by L_s and, consequently, the quadratic term is bounded by $\frac{L_s}{2} \|\Delta S\|^2$.

Substituting Δs_i with the previously calculated correction step size $-\eta g_i$:

$$\mathcal{L}(S_{corr}) \leq \mathcal{L}(S) - \eta \sum_i \|g_i\|^2 + \frac{L_s}{2} \eta^2 \sum_i \|g_i\|^2 \quad (4.19)$$

Factorizing:

$$\mathcal{L}(S_{corr}) \leq \mathcal{L}(S) - \eta \left(1 - \frac{L_s \eta}{2}\right) \sum_i \|g_i\|^2 \quad (4.20)$$

As the term $\sum \|g\|^2$ is always ≥ 0 , the loss decreases when:

$$1 - \frac{L_s \eta}{2} \geq 0 \iff \eta \leq \frac{2}{L_s}. \quad (4.21)$$

Thus, to get a loss decrease, we need η to be smaller than $\frac{2}{L_s}$. This can be further bounded to produce a sufficient decrease by using the stricter condition $\eta \leq \frac{1}{L_s}$.

Trivially, the value function update w.r.t θ also reduces the same loss $\mathcal{L}(S, \theta)$, now treating the (possibly) corrected states S as fixed. For the function $V_\theta(s) = \phi^\top \theta$, this is a standard convex quadratic minimisation: a gradient step on θ with a suitably small learning rate never increases $\mathcal{L}$. Similarly, a suitably small γ_{target} such that $\eta \leq \frac{1}{L_s}$ holds, every state correction step is guaranteed to be non-increasing.

Because both the state-correction and the value-update phases operate on the same objective $\mathcal{L}(S, \theta)$, and each step is guaranteed to be non-increasing, the sequence of losses produced by alternating the two phases is monotonically non-increasing, and bounded below zero. $\square$

Chapter 5

Experimental Setup

To evaluate the proposed methodologies, a training and evaluation pipeline was established. The experiments are designed to compare baseline reinforcement learning and robust reinforcement agents against those equipped with a Meta **RW-RL** model, and True **RW-RL** mechanism, both in standard Gymnasium environments [Towers et al. 2025] and varying degrees of noise. The main objectives are to see if agents with **RW-RL** can successfully converge, and if this mechanism provides any degree of robustness.

The base policy optimization algorithm was **PPO** [Schulman et al. 2017], used uniformly across all agents to ensure a fair comparison. To assess the performance of our methodologies, we compare them with the following agent types:

- **Baseline:** Standard "out of the box" **PPO**.
- **Noisy PPO:** A **PPO** that adds zero-mean gaussian noise to observations during training.
- **Alternating Training of Learned Adversaries (ATLA):** Adversarially trained **PPO** using an auxiliary adversary network that perturbs observations to maximize the policy's loss [Zhang et al. 2021]. It casts the training process as a two-player zero-sum game, alternating between optimizing the adversary to discover worst-case observational perturbations and updating the protagonist policy to resist those specific blind spots.
- **Meta RW-RL:** **PPO** combined with a meta-learner (with the three backbones previously mentioned) that replaces raw observations with corrected ones.
- **True RW-RL:** **PPO** augmented with the self-directed correction mechanism directly applied on the replay buffer through backpropagation of the value network.

5.1 Environments

Experiments were conducted across five environments of increasing complexity, state dimensionality, and dynamic instability, each presenting a unique mechanical challenge designed to expose different vulnerabilities in reinforcement learning policies when subjected to observation noise.

- **CartPole-v1:** A classic, low-dimensional control problem featuring a 4-dimensional continuous state space and discrete actions. The objective is to keep an unactuated pole upright by applying lateral forces to a cart. CartPole allows for the isolated evaluation of the algorithm’s base mechanics, verifying that noise injection and gradient-based data correction function correctly without being confounded by complex, nonlinear physics. If a robust methodology fails or diverges here, it is fundamentally flawed or inadequate.
- **LunarLander-v3:** A 2D navigation task with an 8-dimensional state space and discrete actions, requiring the agent to manage momentum, gravity, and orientation to land softly on a designated pad. Unlike CartPole, LunarLander introduces spatial navigation and strict terminal states (crashing versus landing safely). Observation noise in this environment severely distorts the agent’s perception of the ground and its own velocity, particularly in the critical final frames of the episode. This tests the algorithm’s ability to maintain value-function accuracy and correct trajectory states when the physical margin for error is exceptionally narrow.
- **Hopper-v4:** A continuous control locomotion task with 11 dimensions of observable state and 3 degrees of actuation. The agent must balance on a single articulated leg while maximizing forward velocity. Hopper is characterized by its extreme dynamic instability and inherent fragility. The policy must operate continuously on the edge of a failure state. Even slight observational noise injected into the joint angles or angular velocities will cause immediate collapse. This environment is vital for analyzing whether the methodology can stabilize a highly fragile, precarious physical system against adversarial or random perturbations.
- **HalfCheetah-v4:** A planar bipedal robot with 17 dimensions of observable state and 6 degrees of actuation. In contrast to Hopper, HalfCheetah is inherently stable and cannot "fall over" in a way that prematurely terminates the episode, and, because survival is not the primary challenge, HalfCheetah shifts the analytical focus from sheer stability to efficiency and optimization under noise. When sensors are corrupted, policies typically adopt highly suboptimal, energy-wasting, or overly conservative gaits. This environment tests whether the algorithm can not only survive but also reconstruct the optimal, high-speed locomotion strategy despite persistent sensory degradation.
- **Ant-v4:** A complex, 3D quadrupedal robot with 27 dimensions of observable state and 8 degrees of actuation, requiring highly coordinated, multi-joint continuous control to navigate a planar surface. Ant-v4 represents the ultimate test of high-dimensional scalability and serves as the closest proxy for the physical sim-to-real gap. The high-dimensional observation space makes it mathematically challenging to enforce consistency across all state variables simultaneously. Success in this environment demonstrates that the proposed data-correction methodology can scale to the complex, redundant sensor arrays typical of modern physical robotics without suffering from gradient divergence or the curse of dimensionality.

And all environments were normalized to zero-mean, with variance per observation dimension, and clipped to $[-10, 10]$ to mitigate outliers.

5.2 Perturbations and Partial Observability

Two types of modifications were also implemented in each environment setting, to allow further analysis into the strenghts and weaknesses of each agent, and each scenario was run for each model:

- **Gaussian Environment:** Added to the environment’s observations (before normalization) with a configurable standard deviation ($\sigma_{train} = \{0.0, 0.1\}$)
- **Partial Observability:** Added to the environment a mask that would zero out a configurable subset of the environment’ dimensions. The standard value used was 30%.

These modifications explicitly target the observation vector, deliberately corrupting the information pipeline before it reaches the agent’s neural networks. The Gaussian perturbation affects the agent’s raw sensory input, serving as a proxy for pervasive real-world sensor inaccuracies such as noise or jitter. By injecting continuous zero-mean noise, the true Markovian state is obscured. This forces the value network to calculate expected returns based on volatile data, leading to erratic **TD** errors.

On the other hand, the partial observability modification attacks the observation space, simulating failures such as a disconnected sensor, communication loss, or a physically occluded camera lens. By masking out 30% of the state, the underlying **MDP** environment turns into a **POMDP**. Deprived of physical context, such as knowing its velocity but not its spatial orientation, the agent is forced to act with blind spots.

With these two types of perturbations, the policy network struggles to map actions consistently, which frequently results in unstable control, severe overcompensation, and usually more conservative policies.

5.3 Hyperparameters

Each set of hyperparameters, detailed in Table 5.1, was individually tuned for its respective environment and empirically tested to ensure it reliably produces an average expected expert agent. To guarantee a fair and rigorous comparison, these foundational settings were kept strictly identical across all agent types within a given environment.

5.4 Evaluation

The core objective of this framework is to demonstrate the resilience of **RW-RL** against sensory degradation. Given this, each model previously mentioned will undergo robustness evaluation, where they are subjected to their environment in a wide spectrum of noise levels ($\sigma \in$

Table 5.1: PPO hyperparameters per environment.

Environment	Timesteps	n_steps	Batch Size	Epochs	Learning R.	Network Arch.
CartPole	200 000	1024	32	10	3×10^{-4}	[8]
LunarLander	800 000	2048	128	4	1×10^{-3}	[128,128]
HalfCheetah	1 000 000	4096	256	10	1×10^{-4}	[256,256]
Hopper	1 000 000	2048	64	10	3×10^{-4}	[256,256]
Ant	2 000 000	4096	128	10	3×10^{-4}	[256,256]

{0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.3, 1.5, 1.8, 2.0}). For each noise level, we ran 50 episodes to compute the following metrics:

- **Mean and Standard Deviation of Reward** - Policy performance in terms of reward under specified environmental conditions.
- **Conditional Value at Risk (CVaR)**: To quantify the worst-case scenario robustness, we calculate the average reward of the worst 25% of episodes. CVaR at the 25% level (also known as CVaR25) provides information into the agent’s performance under it’s worse performing state, capturing the tail risk that is not reflected by average performance alone. The higher the CVaR in terms of reward, the better [Ni and Lai 2024].
- **Robustness Index (RI)**: Represents the area under the performance-perturbation curve. Higher values of RI indicate smoother degradation and greater resistance to perturbations [Rajhans and Khawarey 2026].
- **Robustness Efficiency (RE)**: This metric further expands on RI by normalizing it against the perfect theoretical scenario, where the agent does not degrade in performance, regardless of noise. A score of 1.0 represents an agent perfectly robust to noise (the reward-perturbation curve is a flat horizontal line), while a score near 0.0 means that the agent is extremely fragile.
- **Pairwise Permutation Test**: To assess model performance across the full set of conditions, we conducted a paired permutation test on the win rate. For every pair of models we first computed, for each condition, which model achieved the higher average metric. The observed win count (how many conditions favoured model A over model B) was then compared against a null distribution constructed by randomly swapping the model labels within each condition over 10 000 iterations, which simulates the hypothesis that both models are indistinguishable and that any observed advantage is due to chance. The two-sided p-value is the fraction of permutations where the departure from a 50% win rate was at least as extreme as that observed. A win rate above 0.5 indicates that a model outperforms its counterpart in the majority of settings, and a p-value below 0.05 signals that this dominance is statistically reliable.

All experiments were run and aggregated across 10 different seeds, with a total of 1400 experiments, across all environment configuration combinations, models and seeds.

Chapter 6

Results and Discussion

In this chapter, we will begin with the demonstration and analysis of the preliminary work’s results, and then expose and discuss the results of the main contributions.

6.1 Vanilla application of Pastprop

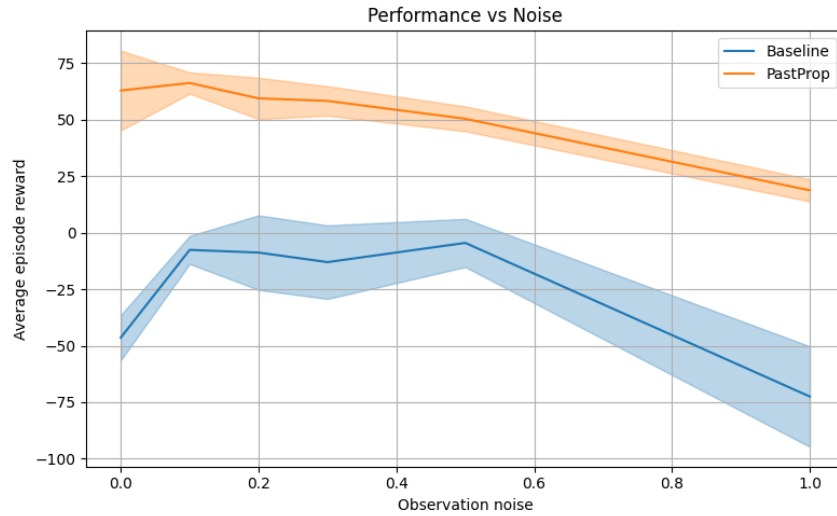


Figure 6.1: Preliminary Results: Overall Performance vs. Noise

In Experiment 1, Pastprop performed strictly above the baseline implementation, demonstrating superior resilience across all tested noise levels.

The baseline model starts with a negative reward even at the lowest noise levels, and degrades linearly as noise increases, while the Pastprop model exhibits a much gentler degradation slope.

A crucial observation is that Pastprop demonstrated a much lower reward deviation even at the highest noise values ($\sigma = 1.0$). This suggests that introducing a data correction mechanism successfully stabilized the policy against severe sensor noise.

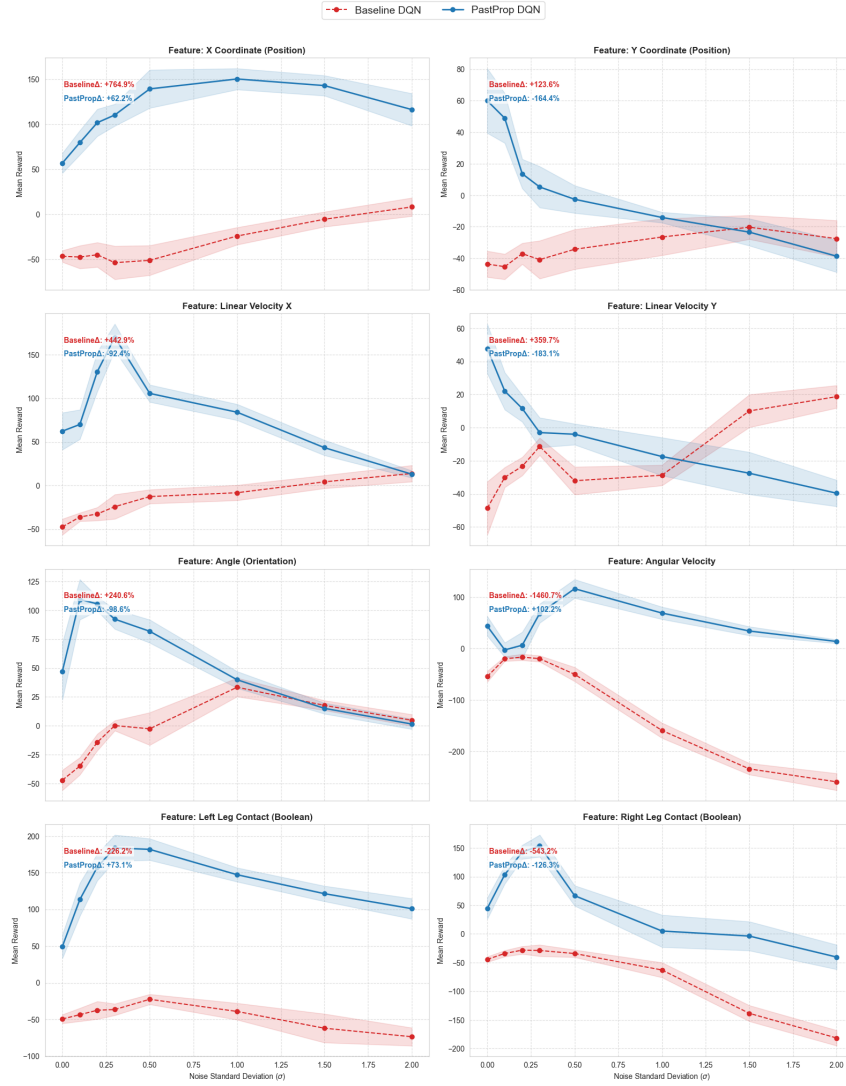


Figure 6.2: Performance of each observation dimension as noise increases.

Furthermore, in the noiseless scenario, Pastprop significantly outperformed the baseline, implying that even in noiseless environments, it is still beneficial to have a data correction mechanism, as it likely smoothes errors through the simulation, providing the policy with a much regularized input compared to the raw environment values.

As illustrated in Figure 6.2, the results of Experiment 2 indicate that, while the baseline performance degrades rapidly when critical control features (such as angular velocity or orientation) are corrupted, the PastProp agent maintains stability.

By leveraging the temporal correlations between the uncorrupted features, the data correction mechanism effectively reconstructs the noisy signal, preventing the catastrophic failure modes observed in the baseline. This is particularly prominent in binary features such as the Leg Contact observations, other features such as those correlated to the vertical components of the simulation (Linear Velocity Y, Y Coordinate) demonstrate that Pastprop might struggle to fully capture

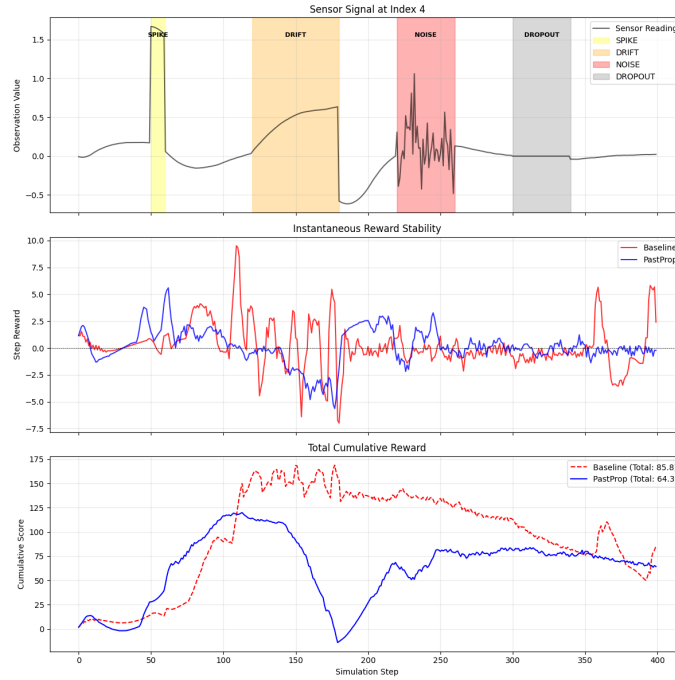


Figure 6.3: Reward behaviour over one episode over time through the injection of multiples kinds of noise.

individual variable relationship dynamics.

This experiment proved to be the most interesting, as we can observe a catastrophic failure of the Pastprop model during the drift phase.

The most plausible answer to this lies in the fundamental difference between the Markovian nature of the baseline and the temporally consistent priority of the Pastprop model. Given that the baseline model operates on a strict Markov property, where only the current observation is taken into account to predict the next state, it might observe the bad number and attempt to fix it immediately based on its policy. Despite it being a simple solution, it managed to sustained its performance during this phase, despite its extreme variation in reward.

On the other hand, Pastprop prioritizes the temporal evolution of the environment state. During the drift phase, the cumulative additive error creates a smooth transition in the data, to the Pastprop component, this may look like a stable and physically possible curve based on the previous states, effectively tricking the model into failing.

Based on the results of the preliminary work, we can conclude that, as a noise correction mechanism, Pastprop through the use of a **LSTM** provides a beneficial increase in general performance, as the underlying training data becomes more and more effective. This goes in line with the works of [Jim, Horne, and Giles 1994], as it is argued that white-noise contributes to improved generalization and convergence.

[Krishnan et al. 2016] demonstrates that non-white noise, such as drifts or anomalies, in input data can significantly degrade model performance, and as seen in the experiment, plausible realities (such as the slow build-up of drift over time) seem to completely trick the model.

Given how the **LSTM** is trained using **MSE** loss, and it has a small window of 8 timesteps, the combination of the drift being a low-frequency, long-term signal change to the original value enables the model to assume this deviation is physically possible, in turn allowing this data to be successfully used in the Neural Network (**NN**) prediction step.

Moreover, as the data correction is applied through moving the noised values closer to the **LSTM**'s projection, as it allows the drift to be taken as a "correct" state, the following steps will then be slowly nudged to match the drift instead of the true environment state, effectively developing a catastrophic feedback loop. Once this drift injection is over, as seen in figure 6.3, we notice the Pastprop model effectively returning to its proper functionality, this further confirms that while the **LSTM** did not forget the underlying physics of the environment, it got trapped following a deceptive reality.

6.2 Meta and True **RW-RL**

To fairly compare across tasks with different reward scales, we min-max normalize each metric per environment to $[0, 1]$. The plot in Figure 6.4 shows the average normalized score across all seeds and environments.

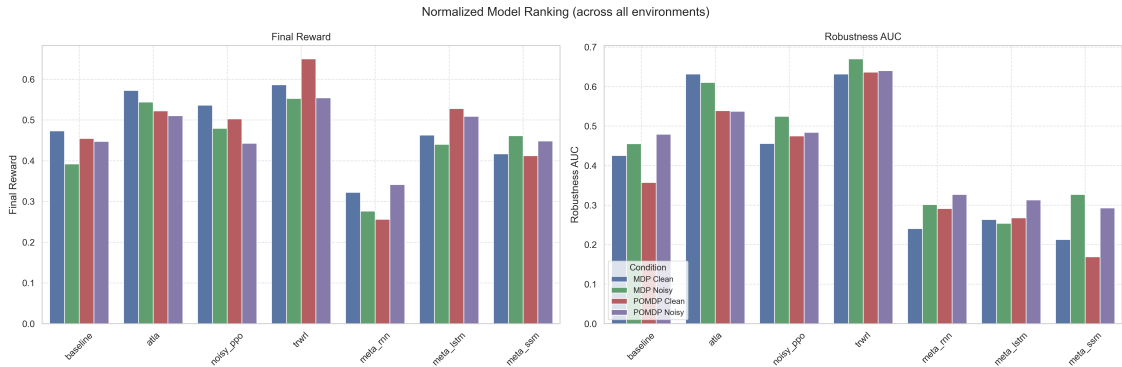


Figure 6.4: Normalized model ranking score on all experiments.

When analysing each model's ranking across all experiments, True **RW-RL** stands out as the highest scoring models in both Final Reward and Robustness Index. Trailing behind is **ATLA** in both scenarios as well. The Meta models in terms of robustness do not seem the best, but at least in terms of final reward they can keep up for the most part with the Baseline.

As seen in the Figure 6.5, overall the higher achieving models are True **RW-RL** and **ATLA**, with True **RW-RL** being consistently better specifically in the **POMDP** scenarios. Meta **RW-RL** models are the least stable ones, sometimes producing a higher robustness in higher noise scenarios, but not in a consistent and relevant way.

Figure 6.6 mean reward and **CVaR** curves of a model trained in LunarLander-v3, and then evaluated in increasing instances of noise. We can observe some interesting details, the reward plots in clean training (left side graphs) seem to all follow the save curve, only shifting vertically in relation to each other, but the worst-case scenarios shown by the **CVaR** lines show that the

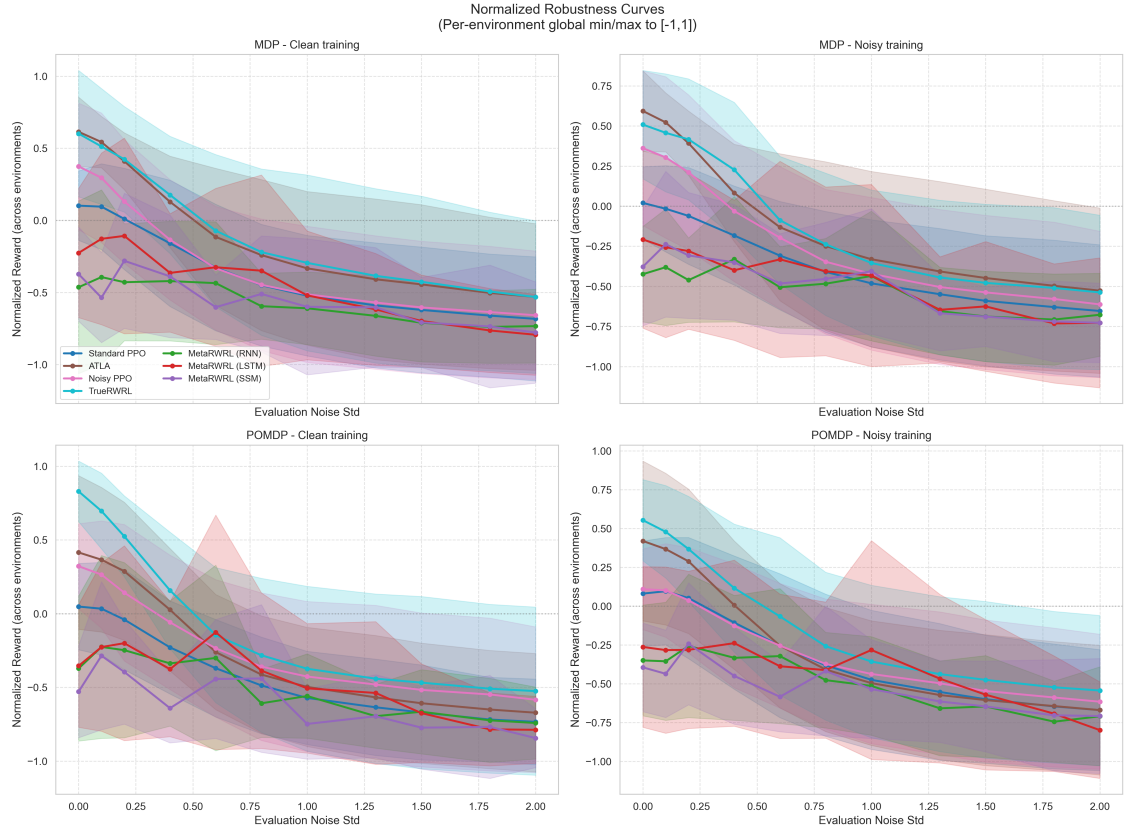


Figure 6.5: Normalized Mean Reward across training settings.

Baseline overall fails equally regardless of noise. In comparison, most robust methodologies seem to have a higher **CVaR**. On the plots of the right side, representing the robust scenario. The evaluation on the lower levels of noise usually demonstrate the highest variability, which is to be expected, as high noise levels eventually drowns out any useful information for the model to act, and in most scenarios, all models eventually merge into the same curve Reward wise. Overall, this behaviour is present across all environments.

Tables 6.1 and 6.2 contain a brief summary of each model's mean reward and **CVaR** on three levels of evaluation noise. The complete result tables are available at Appendix 8.1.

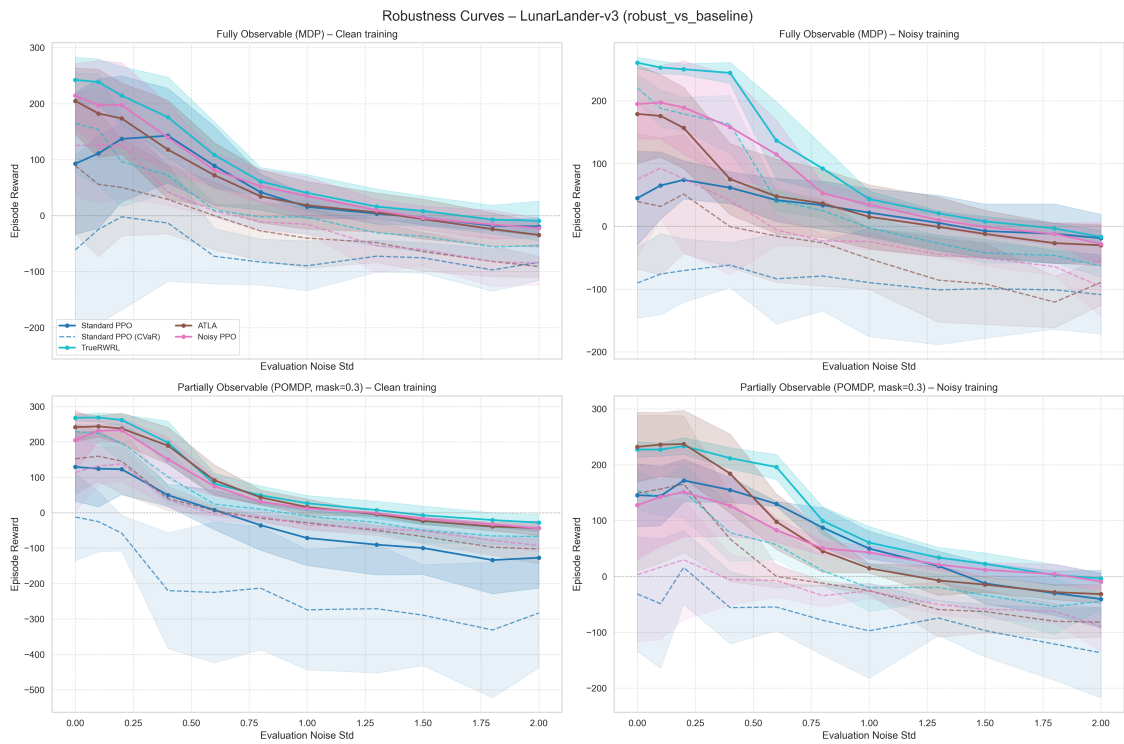


Figure 6.6: Mean Reward and CVaR Tail on LunarLander Environment.

Table 6.1: Average mean reward $\pm$ standard deviation over training noises, at selected evaluation noise levels. Best per environment, mode, and noise level is bolded.

Environment	Model	MDP			POMDP		
		0.0	0.6	2.0	0.0	0.6	2.0
Ant-v4	Baseline	1176.9 $\pm$ 197.0	434.2 $\pm$ 102.9	56.1 $\pm$ 50.2	985.2 $\pm$ 322.5	386.6 $\pm$ 87.1	67.3 $\pm$ 65.0
	Noisy PPO	1268.0 $\pm$ 318.6	416.7 $\pm$ 74.3	35.0 $\pm$ 21.6	1030.2 $\pm$ 356.9	371.2 $\pm$ 104.7	42.8 $\pm$ 62.6
	ATLA	1267.0 $\pm$ 376.0	561.7 $\pm$ 138.6	82.0 $\pm$ 62.7	1402.6 $\pm$ 408.2	585.2 $\pm$ 145.5	54.4 $\pm$ 32.4
	Meta LSTM	331.0 $\pm$ 129.5	167.5 $\pm$ 83.9	7.4 $\pm$ 19.1	396.1 $\pm$ 97.5	246.8 $\pm$ 86.0	18.2 $\pm$ 32.5
	Meta RNN	628.5 $\pm$ 149.9	543.2 $\pm$ 96.9	442.2 $\pm$ 104.1	597.3 $\pm$ 124.9	572.4 $\pm$ 102.5	410.6 $\pm$ 73.4
	Meta SSM	578.3 $\pm$ 85.0	439.6 $\pm$ 107.6	204.9 $\pm$ 73.6	488.4 $\pm$ 101.9	326.0 $\pm$ 101.6	122.2 $\pm$ 24.6
	True RW-RL	1346.3 $\pm$ 454.0	522.8 $\pm$ 85.1	66.3 $\pm$ 59.5	1607.5 $\pm$ 369.8	528.8 $\pm$ 131.7	60.0 $\pm$ 63.9
CartPole-v1	Baseline	179.7 $\pm$ 18.1	147.7 $\pm$ 14.3	93.4 $\pm$ 26.0	187.7 $\pm$ 22.7	157.3 $\pm$ 16.7	106.8 $\pm$ 27.9
	Noisy PPO	223.7 $\pm$ 103.4	202.4 $\pm$ 104.7	154.1 $\pm$ 99.7	265.0 $\pm$ 105.0	231.1 $\pm$ 99.7	180.0 $\pm$ 98.9
	ATLA	393.4 $\pm$ 164.6	337.4 $\pm$ 184.6	251.4 $\pm$ 139.3	182.5 $\pm$ 34.2	158.0 $\pm$ 29.9	115.7 $\pm$ 47.6
	Meta LSTM	306.7 $\pm$ 155.0	353.8 $\pm$ 168.9	65.2 $\pm$ 48.5	253.5 $\pm$ 109.3	370.9 $\pm$ 105.6	68.8 $\pm$ 63.4
	Meta RNN	275.5 $\pm$ 132.4	239.2 $\pm$ 129.6	69.0 $\pm$ 38.9	278.5 $\pm$ 159.0	344.9 $\pm$ 132.2	41.6 $\pm$ 30.1
	Meta SSM	189.6 $\pm$ 153.1	125.3 $\pm$ 95.3	28.5 $\pm$ 20.1	131.0 $\pm$ 75.2	183.1 $\pm$ 142.0	33.3 $\pm$ 20.9
	True RW-RL	362.3 $\pm$ 52.5	299.9 $\pm$ 89.9	230.4 $\pm$ 73.8	389.8 $\pm$ 90.6	283.1 $\pm$ 82.0	241.4 $\pm$ 88.2
HalfCheetah-v4	Baseline	1287.7 $\pm$ 337.8	802.8 $\pm$ 39.7	-54.5 $\pm$ 59.1	1330.8 $\pm$ 279.6	797.7 $\pm$ 37.3	-68.6 $\pm$ 17.3
	Noisy PPO	2440.7 $\pm$ 1292.9	695.4 $\pm$ 240.8	-158.1 $\pm$ 121.1	1533.0 $\pm$ 94.9	792.9 $\pm$ 78.0	-40.4 $\pm$ 78.6
	ATLA	1840.8 $\pm$ 927.2	784.6 $\pm$ 129.2	-87.7 $\pm$ 114.3	1363.0 $\pm$ 137.5	722.3 $\pm$ 27.2	-161.0 $\pm$ 76.1
	Meta LSTM	1428.7 $\pm$ 97.5	923.3 $\pm$ 37.4	-16.9 $\pm$ 39.9	1536.9 $\pm$ 390.0	970.1 $\pm$ 102.5	-47.0 $\pm$ 127.2
	Meta RNN	691.9 $\pm$ 234.2	503.6 $\pm$ 68.8	-23.2 $\pm$ 51.5	867.9 $\pm$ 216.5	588.9 $\pm$ 104.0	-43.1 $\pm$ 108.3
	Meta SSM	1183.5 $\pm$ 263.1	843.7 $\pm$ 170.0	83.3 $\pm$ 129.4	1104.4 $\pm$ 252.6	801.9 $\pm$ 52.3	76.2 $\pm$ 54.7
	True RW-RL	1455.2 $\pm$ 117.8	817.7 $\pm$ 72.9	-68.5 $\pm$ 51.4	2134.7 $\pm$ 854.4	783.3 $\pm$ 91.0	-104.0 $\pm$ 123.8
Hopper-v4	Baseline	1731.7 $\pm$ 831.6	616.1 $\pm$ 82.7	189.6 $\pm$ 69.0	1590.3 $\pm$ 651.8	655.7 $\pm$ 221.0	161.6 $\pm$ 26.4
	Noisy PPO	1710.8 $\pm$ 855.0	633.9 $\pm$ 67.7	207.6 $\pm$ 86.1	1810.8 $\pm$ 806.2	645.2 $\pm$ 252.2	164.5 $\pm$ 61.1
	ATLA	3233.2 $\pm$ 72.9	672.7 $\pm$ 257.3	249.9 $\pm$ 21.7	3334.1 $\pm$ 149.4	722.3 $\pm$ 156.3	234.4 $\pm$ 27.5
	Meta LSTM	305.3 $\pm$ 215.1	120.4 $\pm$ 148.1	43.1 $\pm$ 36.3	154.2 $\pm$ 130.1	90.1 $\pm$ 56.9	69.8 $\pm$ 76.0
	Meta RNN	97.0 $\pm$ 34.5	122.0 $\pm$ 54.6	68.7 $\pm$ 26.0	101.0 $\pm$ 71.1	96.7 $\pm$ 62.7	91.6 $\pm$ 37.6
	Meta SSM	234.8 $\pm$ 214.3	72.3 $\pm$ 74.6	54.5 $\pm$ 61.1	190.1 $\pm$ 275.9	252.9 $\pm$ 429.8	33.6 $\pm$ 41.6
	True RW-RL	3049.5 $\pm$ 417.0	1003.9 $\pm$ 181.9	257.6 $\pm$ 63.8	2756.6 $\pm$ 536.6	847.7 $\pm$ 290.4	244.4 $\pm$ 55.1
LunarLander-v3	Baseline	68.8 $\pm$ 100.4	65.4 $\pm$ 52.3	-19.1 $\pm$ 27.3	137.3 $\pm$ 76.7	68.8 $\pm$ 29.9	-83.8 $\pm$ 68.8
	Noisy PPO	204.7 $\pm$ 56.3	97.9 $\pm$ 42.6	-25.2 $\pm$ 25.6	166.1 $\pm$ 90.7	79.0 $\pm$ 24.4	-26.5 $\pm$ 13.2
	ATLA	191.9 $\pm$ 68.8	60.0 $\pm$ 58.9	-32.4 $\pm$ 27.1	236.7 $\pm$ 51.8	94.7 $\pm$ 46.7	-37.5 $\pm$ 18.5
	Meta LSTM	-44.1 $\pm$ 112.9	-86.4 $\pm$ 145.5	-88.0 $\pm$ 110.1	-128.7 $\pm$ 154.6	-29.6 $\pm$ 115.5	-145.0 $\pm$ 99.2
	Meta RNN	-243.4 $\pm$ 94.5	-172.4 $\pm$ 76.2	-145.6 $\pm$ 21.3	-149.8 $\pm$ 98.2	-95.6 $\pm$ 25.3	-129.6 $\pm$ 17.4
	Meta SSM	-137.4 $\pm$ 113.2	-160.2 $\pm$ 82.9	-110.6 $\pm$ 84.9	-137.1 $\pm$ 92.9	-183.5 $\pm$ 77.8	-122.2 $\pm$ 76.4
	True RW-RL	251.5 $\pm$ 24.8	122.5 $\pm$ 60.7	-13.0 $\pm$ 22.8	247.5 $\pm$ 11.2	138.8 $\pm$ 26.2	-15.8 $\pm$ 16.8

Table 6.2: Average mean **CVaR** $\pm$ standard deviation over training noises, at selected evaluation noise levels. Best per environment, mode, and noise level is bolded.

Environment	Model	MDP			POMDP		
		0.0	0.6	2.0	0.0	0.6	2.0
Ant-v4	Baseline	299.4 $\pm$ 126.7	98.6 $\pm$ 45.1	-31.2 $\pm$ 32.4	337.8 $\pm$ 145.9	76.7 $\pm$ 35.3	-24.3 $\pm$ 44.4
	Noisy PPO	393.5 $\pm$ 220.4	73.0 $\pm$ 27.1	-46.6 $\pm$ 18.8	295.1 $\pm$ 184.2	69.6 $\pm$ 33.6	-44.2 $\pm$ 57.5
	ATLA	536.7 $\pm$ 163.2	169.0 $\pm$ 67.7	-4.6 $\pm$ 20.0	629.9 $\pm$ 263.8	200.3 $\pm$ 78.6	-10.3 $\pm$ 17.6
	Meta LSTM	151.2 $\pm$ 102.8	63.7 $\pm$ 51.6	-18.1 $\pm$ 19.2	180.1 $\pm$ 90.1	95.1 $\pm$ 60.0	-4.9 $\pm$ 19.2
	Meta RNN	528.7 $\pm$ 225.2	427.4 $\pm$ 186.5	341.8 $\pm$ 157.1	477.8 $\pm$ 284.5	505.1 $\pm$ 152.3	310.1 $\pm$ 97.2
	Meta SSM	498.9 $\pm$ 116.3	310.0 $\pm$ 126.5	87.6 $\pm$ 42.8	375.2 $\pm$ 184.5	225.9 $\pm$ 139.8	35.8 $\pm$ 12.7
	True RW-RL	698.3 $\pm$ 520.3	152.4 $\pm$ 65.4	-9.5 $\pm$ 26.0	774.6 $\pm$ 356.4	127.8 $\pm$ 89.2	-20.9 $\pm$ 41.1
CartPole-v1	Baseline	167.2 $\pm$ 15.1	135.8 $\pm$ 12.2	41.9 $\pm$ 19.1	175.1 $\pm$ 19.6	144.4 $\pm$ 15.1	52.3 $\pm$ 30.8
	Noisy PPO	213.9 $\pm$ 102.8	189.8 $\pm$ 102.7	92.6 $\pm$ 76.7	246.5 $\pm$ 104.4	211.3 $\pm$ 99.1	111.4 $\pm$ 78.5
	ATLA	351.3 $\pm$ 177.7	318.3 $\pm$ 185.3	151.9 $\pm$ 91.5	170.5 $\pm$ 30.1	146.1 $\pm$ 26.7	68.0 $\pm$ 41.7
	Meta LSTM	292.4 $\pm$ 145.9	347.8 $\pm$ 176.0	40.9 $\pm$ 42.3	246.8 $\pm$ 109.7	361.8 $\pm$ 126.3	32.2 $\pm$ 27.8
	Meta RNN	264.9 $\pm$ 129.9	212.0 $\pm$ 126.1	42.1 $\pm$ 24.0	259.3 $\pm$ 172.1	318.0 $\pm$ 154.3	22.1 $\pm$ 14.9
	Meta SSM	186.4 $\pm$ 154.9	116.4 $\pm$ 97.2	14.9 $\pm$ 5.3	122.0 $\pm$ 78.7	172.5 $\pm$ 137.4	16.9 $\pm$ 9.3
	True RW-RL	315.7 $\pm$ 62.6	268.4 $\pm$ 85.7	153.9 $\pm$ 60.7	349.4 $\pm$ 103.2	253.2 $\pm$ 84.4	177.6 $\pm$ 87.6
HalfCheetah-v4	Baseline	848.1 $\pm$ 845.5	634.7 $\pm$ 138.3	-145.2 $\pm$ 62.6	1016.4 $\pm$ 657.6	646.5 $\pm$ 65.0	-163.5 $\pm$ 27.4
	Noisy PPO	2378.4 $\pm$ 1263.1	510.0 $\pm$ 291.2	-277.6 $\pm$ 146.6	1440.1 $\pm$ 150.3	696.5 $\pm$ 116.9	-136.4 $\pm$ 87.2
	ATLA	1702.6 $\pm$ 878.9	613.5 $\pm$ 242.3	-203.9 $\pm$ 166.3	1214.7 $\pm$ 223.3	622.3 $\pm$ 14.7	-253.4 $\pm$ 73.0
	Meta LSTM	1397.9 $\pm$ 117.2	908.8 $\pm$ 25.4	-48.1 $\pm$ 42.3	1466.9 $\pm$ 387.4	909.1 $\pm$ 97.1	-93.6 $\pm$ 164.8
	Meta RNN	618.4 $\pm$ 254.1	444.0 $\pm$ 77.5	-66.7 $\pm$ 42.1	792.4 $\pm$ 247.4	546.4 $\pm$ 123.3	-87.7 $\pm$ 121.5
	Meta SSM	1106.5 $\pm$ 286.3	756.6 $\pm$ 128.0	32.4 $\pm$ 108.3	1062.8 $\pm$ 343.1	760.0 $\pm$ 51.7	40.6 $\pm$ 49.6
	True RW-RL	1351.7 $\pm$ 241.4	626.0 $\pm$ 218.0	-165.5 $\pm$ 53.6	1995.3 $\pm$ 885.6	558.3 $\pm$ 261.6	-210.9 $\pm$ 151.2
Hopper-v4	Baseline	1277.4 $\pm$ 558.0	390.7 $\pm$ 117.7	83.6 $\pm$ 54.3	1212.5 $\pm$ 567.9	299.2 $\pm$ 157.0	71.2 $\pm$ 9.9
	Noisy PPO	1292.4 $\pm$ 645.3	374.6 $\pm$ 60.7	110.1 $\pm$ 60.8	1579.1 $\pm$ 716.9	311.2 $\pm$ 159.3	56.5 $\pm$ 16.2
	ATLA	2770.4 $\pm$ 406.6	414.9 $\pm$ 112.6	152.2 $\pm$ 57.9	3288.4 $\pm$ 145.3	388.0 $\pm$ 41.0	128.9 $\pm$ 50.5
	Meta LSTM	138.5 $\pm$ 177.4	73.2 $\pm$ 109.1	23.9 $\pm$ 24.7	56.3 $\pm$ 30.9	44.4 $\pm$ 42.9	31.7 $\pm$ 39.2
	Meta RNN	79.9 $\pm$ 39.8	103.3 $\pm$ 46.9	33.4 $\pm$ 24.5	78.3 $\pm$ 61.0	72.8 $\pm$ 49.7	61.5 $\pm$ 20.5
	Meta SSM	93.5 $\pm$ 113.3	23.0 $\pm$ 13.7	30.4 $\pm$ 38.5	161.3 $\pm$ 266.9	242.8 $\pm$ 454.0	20.8 $\pm$ 27.6
	True RW-RL	2915.9 $\pm$ 513.9	579.5 $\pm$ 103.0	152.0 $\pm$ 76.3	2349.7 $\pm$ 772.5	507.5 $\pm$ 143.9	133.1 $\pm$ 56.7
LunarLander-v3	Baseline	-75.8 $\pm$ 112.5	-78.1 $\pm$ 60.7	-96.1 $\pm$ 47.7	-21.8 $\pm$ 113.3	-139.9 $\pm$ 120.7	-209.9 $\pm$ 117.5
	Noisy PPO	100.2 $\pm$ 106.1	1.2 $\pm$ 17.4	-90.4 $\pm$ 43.7	58.2 $\pm$ 115.0	-5.6 $\pm$ 25.2	-90.9 $\pm$ 30.9
	ATLA	64.5 $\pm$ 115.1	-7.8 $\pm$ 67.4	-90.3 $\pm$ 27.5	150.5 $\pm$ 119.0	3.0 $\pm$ 17.5	-92.1 $\pm$ 33.9
	Meta LSTM	-91.3 $\pm$ 113.1	-136.4 $\pm$ 126.4	-119.0 $\pm$ 125.2	-160.4 $\pm$ 141.3	-83.1 $\pm$ 98.5	-184.5 $\pm$ 116.0
	Meta RNN	-284.7 $\pm$ 93.9	-207.9 $\pm$ 104.5	-165.8 $\pm$ 29.4	-188.5 $\pm$ 108.0	-118.2 $\pm$ 28.4	-153.8 $\pm$ 33.7
	Meta SSM	-181.9 $\pm$ 111.1	-181.8 $\pm$ 86.9	-140.9 $\pm$ 97.8	-176.5 $\pm$ 64.3	-209.3 $\pm$ 82.3	-152.1 $\pm$ 84.6
	True RW-RL	192.9 $\pm$ 55.5	25.2 $\pm$ 13.7	-58.4 $\pm$ 27.1	191.4 $\pm$ 27.1	40.5 $\pm$ 22.2	-56.0 $\pm$ 21.0

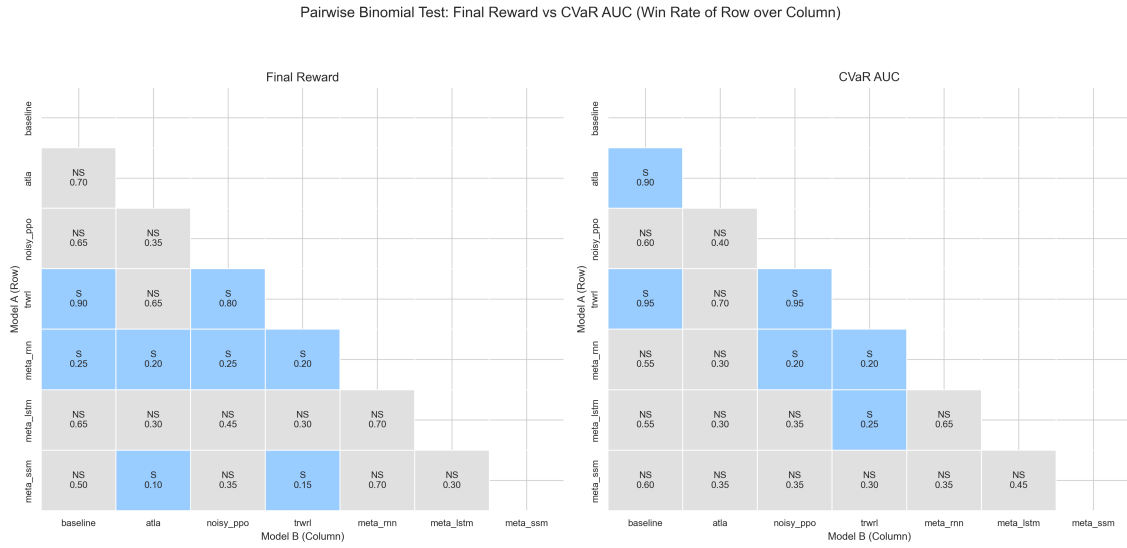


Figure 6.7: Pairwise binomial test between modes for final reward and CVaR.

Going over the pairwise permutation tests, on Figure 6.7 and Figure 6.8, True *RW-RL* and *ATLA* emerge as the top-tier models in terms of Final Reward, with True *RW-RL* outperforming the Standard *PPO* baseline in 90% of training conditions and demonstrating superiority over most competitors, with *ATLA* also showing significant performance advantages over Meta-*SSM* and Meta-*RNN* without separating from True *RW-RL*.

This pattern generally persists regarding *CVaR*, where True *RW-RL* and *ATLA* continue to dominate Standard *PPO*, though the gap narrows as *ATLA* and Meta-*SSM* partially bridge the performance difference compared to True *RW-RL*.

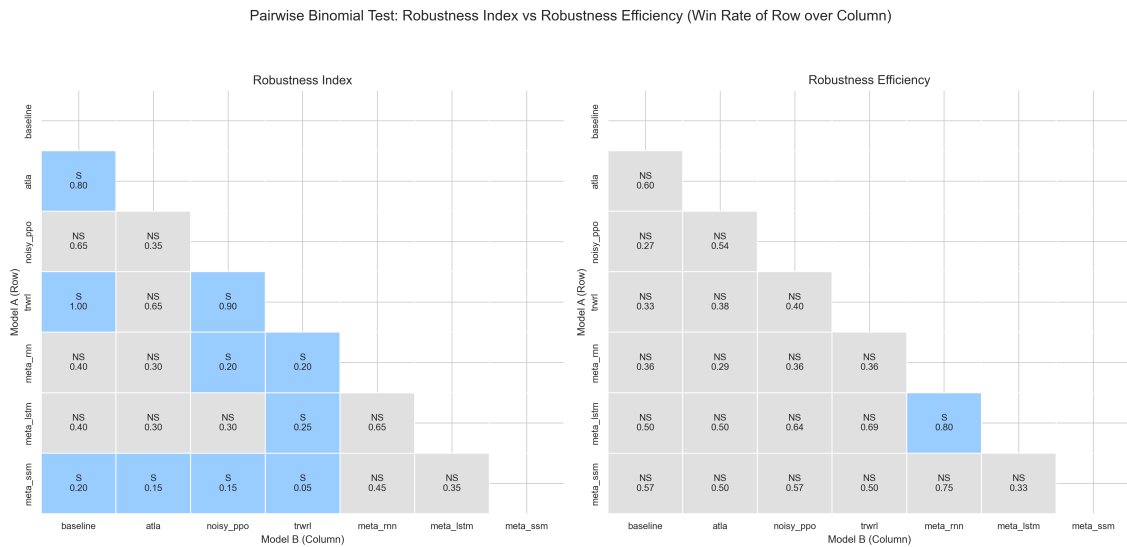


Figure 6.8: Pairwise binomial test between modes for Robustness Index and Efficiency.

Regarding the Robustness Index, which captures overall resistance to evaluation noise, True

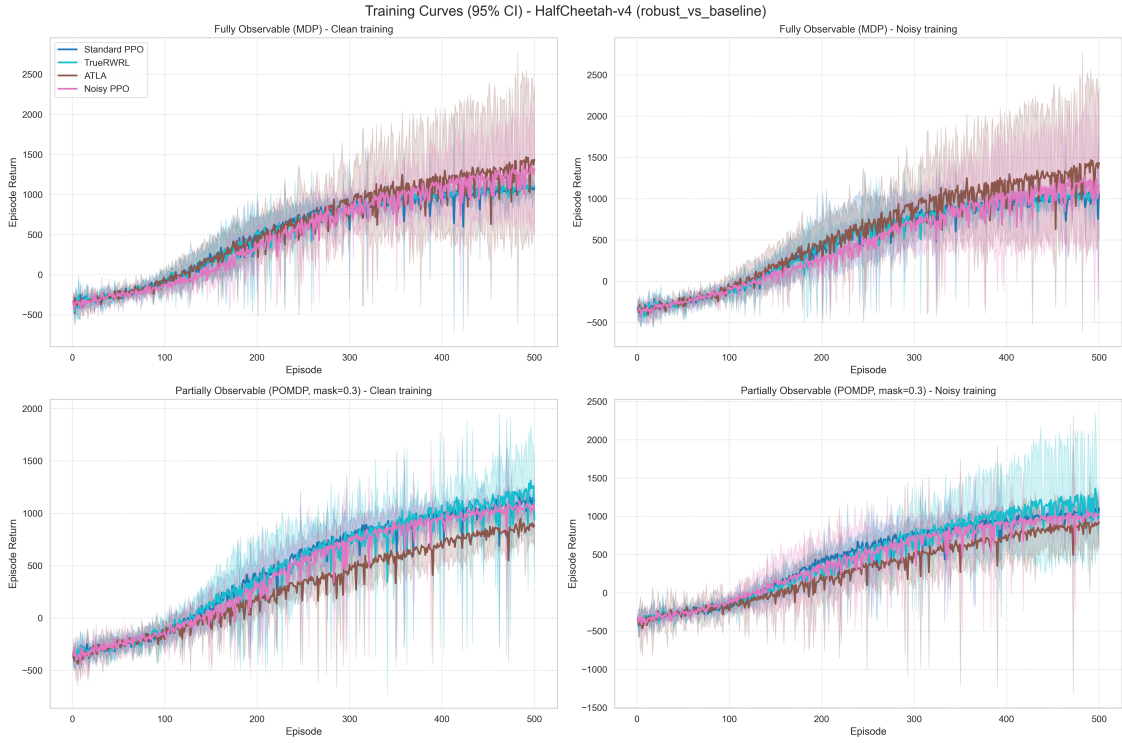


Figure 6.9: Learning Curves on HalfCheetah.

RW-RL is overwhelmingly dominant, winning every condition against Standard **PPO** and significantly outperforming most others; **ATLA** is the only competitor not significantly worse. Interestingly, Standard **PPO** significantly outperforms Meta-**SSM** on this metric. Conversely, Robustness Efficiency yields a striking null result: once initial performance is accounted for, no method degrades proportionally slower than others, as no pairwise comparison reached significance except for Meta-**LSTM** versus Meta-**RNN**. This demonstrates that the robustness advantages of True **RW-RL** and **ATLA** stem from higher absolute performance rather than a fundamentally greater resilience to noise per unit of initial reward.

Finally, in terms of training progression, all learning curves are highly similar across evaluated models in each environment. As illustrated in Figure 6.9, all models exhibit similar convergence during the training phase. In the case of HalfCheetah, for instance, we can see that in the **POMDP** scenarios, True **RW-RL**, through the shaded area, shows an overall higher capability of achieving a stronger reward.

And in terms of training and inference overhead, we’ve done an approximation of the training and inference time in relation to the total quantity of timesteps, compared relative to the baseline, in Table 6.3.

Table 6.3: Average training and inference times relative to the baseline across environments.

Model	Training Time (%)	Inference Time (%)
baseline	100.00 $\pm$ 0.00	100.00 $\pm$ 0.00
NoisyPPO	99.70 $\pm$ 6.43	100.56 $\pm$ 4.95
True RWRL	118.01 $\pm$ 11.93	99.75 $\pm$ 11.53
ATLA	146.64 $\pm$ 67.35	101.26 $\pm$ 8.97
Meta RWRL	290.00 $\pm$ 42.45	106.89 $\pm$ 17.77

6.3 Discussion

The primary objective of this thesis was to evaluate whether a self-directed correction mechanism is sufficient to improve robustness under observation noise and partial observability.

These experiments and results clearly establish True **RW-RL** and **ATLA** as the highest performing models in terms of both reward and worst-case scenario handling, with notable performance in the **POMDP** cases. By actively correcting the underlying data, True **RW-RL** essentially acts as a self-supervised stabilizer that mitigates the severe error compounding typically seen when sequential predictions are corrupted by noise. On the other hand, Meta **RW-RL** architectures turned out to be highly unstable. This could be due to a myriad of reasons, but fundamentally, such an architecture creates an uncoordinated bilevel optimization problem. It not only has to deal with the tuning of the policy model, but also with the meta-model itself across highly non-stationary data distributions. This lack of convergence from either side often causes the meta-model to optimize for outdated state distributions, inducing a massive covariate shift. While in some scenarios where, depending on the backbone, they perform relatively well, which could also mean that these architectures are task-specific.

The most critical and nuanced finding of this research concerns the nature of the robustness advantage held by True **RW-RL** and **ATLA**. The **RE** metric reveals that neither method achieves a different noise-degradation rate than the baseline: once initial performance is accounted for, no method degrades proportionally slower or faster than any other under escalating noise levels in a statistically significant way, as seen in Figure 6.5, or in the Binomial test seen in 6.8. These results reflect an important information theory constraint. Beyond a certain amount of noise, the total information carried by the corrupted observation approaches zero, and no algorithm can both act correctly through it, or reconstruct the true state from pure entropy.

What **RE** does not capture is the degradation onset. The metric that does, the noise level at which performance first drops sharply, tells a different and more practically relevant story. As visible in Figure 6.6, True **RW-RL** consistently delays this onset relative to competing methods: in noisy training scenarios, the sudden performance drop occurs approximately 0.25σ later than in baseline and **ATLA** conditions. This pattern is not seen across every environments every time, but it is relatively consistent. By correcting stored transitions toward Bellman self-consistency during training, True **RW-RL** reduces the variance of **TD** errors and prevents the policy from fitting brittle, noise-sensitive features. The result is a policy with a higher absolute performance ceiling and a wider operating envelope before catastrophic degradation begins.

This is further reflected in the **CVaR** results, where True **RW-RL** demonstrates consistent advantages in worst-case episode performance even at noise levels where mean reward differences narrow. By acting as a retroactive data smoother that permanently removes erratic overcompensations from the training distribution, True **RW-RL** reduces tail risk in a way that the **RE** metric was not designed to measure. Together, these findings suggest that the robustness contribution of True **RW-RL** is real, although regime-dependent, and it could be further explored.

The conclusion regarding the increasing levels of performance of both True **RW-RL** and **ATLA** comes from the fact that both methods compel the development of a fundamentally stronger baseline policy by preventing the network from overfitting to brittle, noise-sensitive observation features. Through adversarial attacks making it learn what it fails the most, **ATLA** explicitly penalizes sensitivity to worst-case scenarios, effectively hardening the policy’s decision boundaries. Conversely, through self-correction of this data, True **RW-RL** achieves this through representation smoothing, avoiding fitting its value function to high-variance temporal difference errors. Because both methods build these highly consistent, generalized representations, it grants the models a much larger capability to optimally learn, and when data degrades enough, to perform slightly better and sustain functionality longer, simply because they have a much higher absolute operational ceiling to fall from than their baseline counterparts.

To answer our second question, our empirical results demonstrate that deliberately modifying stored transitions, while jointly training the value function, improves a model’s robustness. The mathematical formalization then proves that this joint optimization problem does not sacrifice convergence, as both the state-correction and the value-update phases operate on a perfectly aligned objective surface to strictly decrease the Bellman error. Together, these findings establish that data can be treated as an optimisable variable, while breaking the guarantees we normally rely on **i.i.d** settings. This represents a radical evolution of the data-centric reinforcement learning paradigm.

Beyond raw performance and robustness, True **RW-RL** presents a highly practical tradeoff in terms of computational efficiency. As observed in Table 6.3, while the self-directed correction mechanism introduces an approximate 18-20% overhead during training, it incurs effectively zero additional cost during live environment inference. This stands in contrast to **ATLA**, which requires massive computational overhead to continuously generate optimal minimax perturbations, suffering from an approximate 146% overhead in training and a severe 134% latency penalty during inference. Meta **RW-RL** similarly struggles, demanding nearly triple the baseline training time. Consequently, because True **RW-RL** enforces mathematical robustness strictly at the data level during offline training, the final policy network remains structurally lightweight. It achieves superior performance without sacrificing real-time execution speed, making it highly suitable for deployment in edge-computing devices, autonomous driving, or high-frequency physical systems where inference latency is a hard, non-negotiable constraint.

An important architectural variant that was explored during development consisted in using the policy’s gradient to produce corrections, with an explicit confirmation step from the value networks. This rather late experiment introduced severe instability during training, and often led to suboptimal local minima, as it overfit to "safe" actions very early. Mechanistically, updating states

to minimize policy entropy, such as following $\nabla_s \log \pi(a|s)$, creates a destructive confirmation bias; the agent modifies its stored memories to artificially match its current, potentially immature policy. This results in rapid representation collapse and severe overfitting to myopic, low-risk local minima, a failure mode heavily documented in state-adversarial literature when optimizing policies directly on perturbed input spaces without value-based grounding.

However, this experiment in some specific scenarios (less than 2%), achieved a striking stabilization effect, producing a near perfect performance, and more importantly, near perfect **RE** score, meaning that regardless of noise, its performance wouldn't budge when compared to the optimal scenario.

This exceptional edge-case suggests that perhaps exploring this path would lead to a methodology that would achieve the original intended objectives with stronger statistical significance. Specifically, future work should investigate formalizing this mechanism, which unifies state correction with policy constraints. By constraining the policy's state-modification gradients using a mathematically bounded Lipschitz-regularized value network, we can systematically eliminate representation collapse, transforming this 2% edge case into a robust and provably stable paradigm for data-centric reinforcement learning.

Chapter 7

Conclusions

This thesis introduced Read-Write Reinforcement Learning (**RW-RL**), a data-centric paradigm that shifts the burden of robustness from model architecture to online self-supervising data refinement, by treating the experience replay buffer as an active, optimizable variable rather than a static, read-only resource.

7.1 Main Contributions

The True RW-RL Correction Mechanism Our central contribution is True **RW-RL**, a mechanism that integrates data correction directly into the backpropagation of the policy’s own value network. Rather than relying on an external dynamics model or meta-learner, the agent computes the gradient of its temporal-difference loss with respect to each stored state and nudges that state toward Bellman self-consistency. To make this correction safe in practice, we also introduced a set of stabilisation techniques: TD-error gating, an adaptive correction rate, momentum smoothing, a training warmup period, and an asymmetric updates, resulting in a correction rule that requires no auxiliary model and adds no computation to the policy’s forward pass.

Proof of Convergence We provided a formal convergence argument showing that alternating between state correction and value-function updates on the same Bellman objective produces a monotonically non-increasing loss, under a bound on the correction step size relative to the Lipschitz constant of the value gradient. This establishes that treating stored states as an optimisable variable does not, by itself, sacrifice the convergence guarantees the rest of the training procedure relies on.

7.2 Research Questions

The central goal of the thesis was to demonstrate that such a self-directed correction mechanism is sufficient to improve robustness under observation noise, especially in partially observable settings. To evaluate this claim, we sought to answer the following research questions:

- **Question 1 (Stability):** Can an Reinforcement Learning (RL) agent interactively correct its own stored transitions – using gradients of its value loss – without destabilising training (i.e., without value divergence or policy collapse)?
 - Yes. Our primary contribution establishes that data correction through temporal-difference gradients inside its replay buffer can be achieved with a minimal $\approx 20\%$ training overhead while still guaranteeing zero additional latency during inference. This presents a highly practical breakthrough for real-world physical systems where inference speed is critical.
- **Question 2 (Robustness):** Does True RW-RL produce policies that are significantly more robust to varying levels of observation noise compared to a vanilla agent, particularly when noise mimics partial observability?
 - Yes. By shifting the burden of robustness to the data through its refinement, we effectively reduce the Bellman error, moving data to regions where the model’s predictions align with the target. Experiments strongly suggested that the models could withstand degradation longer under increasing levels of noise, and demonstrated higher levels of overall performance, reflecting in higher robustness.

7.3 Study Limitations

While the results presented in Chapter 6 support the central claims and research questions of this thesis, several limitations on the depth of this study should be acknowledged.

First, while standard in robust RL literature, zero-mean Gaussian noise remains a simplified proxy for real-world sensor degradation. Real sensors, however, are rarely this well-behaved. As demonstrated in our preliminary experiments in 6.1, non-stationary perturbations produce considerably more damage to the policy’s behaviour than standard Gaussian noise. The RW-RL experimental pipeline was not implemented with this same battery of structured perturbations, so its behaviour under drift, sensor dropout, or other temporally correlated corruption patterns that are arguably more representative of physical sensor degradation remains untested.

All these experiments were built on top of Proximal Policy Optimization (PPO), an on-policy actor-critic algorithm, but this leaves open the question of whether the True RW-RL mechanism generalizes to other classes of actor-critic algorithms, particularly off-policy algorithms such as Soft Actor-Critic (SAC) [Haarnoja et al. 2018]. As the value function is decoupled from the immediate rollout and the data in the replay buffer is reused over more update steps, the dynamics of state correction and stability could behave differently, and this remains unverified.

7.4 Future Work

With this established, several implications and lines of work radiate from this thesis, touching many challenges in the field:

Real-Time Streaming Data Correction A natural extension of this work is to adapt the **RW-RL** gradient correction mechanism into an active, test-time streaming filter. By computing lightweight, localized value-function approximations on live observations, the agent could intercept and clean sensor anomalies on the fly, providing an active layer of defence against non-white noise (such as sensor drift or temporary dropouts) without losing the zero-overhead benefits established during baseline training. Given the current implementations make use of small, controlled step sizes to limit network updates [Elsayed, Vasan, and Mahmood 2024] to enforce stability, **RW-RL** seems like an adequate fit as a direct continuation of this thesis.

Offline RL Data Refinement In offline **RL**, the dataset is fixed and often suboptimal. Our method suggests a principled way to improve the dataset from within: one could project states back to the support of the data (ensuring realism) while moving them to reduce Bellman error. This could shrink the distributional shift that plagues offline algorithms, all while maintaining convergence guarantees.

Self-consistency and Robustness Classical adversarial settings seek to produce a perturbation to maximise loss, expanding the model’s robust region. Our state correction mechanism, by contrast, moves the data towards the regions where the model’s predictions align with the target, a kind of “self-supervised augmentation” that reduces Bellman error. This method proves that a simple gradient descent on data still yields convergence, raising the question: can we design more data-modification operators that provably enlarge the model’s robust region while still guaranteeing convergence?

Neuromorphic Computing This brain-inspired paradigm aims to develop systems that match the flexibility and energy efficiency of the human brain by replicating its neural architecture in physical hardware [Gerven 2025]. As this field matures, it faces two related challenges: the inherent noise and analogue imperfections of these physical devices necessitate highly robust learning algorithms, yet their mandate for extreme energy efficiency excludes computationally heavy regularisation. The stability and self-consistency benefits introduced by **RW-RL** offer a compelling solution to both constraints. By employing lightweight state corrections, such data-modification operators can provide localized, low-overhead error correction, counteracting physical hardware instability while perfectly aligning with the strict energy and computational constraints of neuromorphic systems.

In conclusion, the paradigm of Read-Write Reinforcement Learning challenges the traditional assumption that an agent's data must remain immutable. By treating memory as an active workspace for optimization, this thesis points toward a broader paradigm shift in robust machine learning as a whole: instead of passively hardening a model's architecture against adversarial input disturbances, we can equip the system to actively question and realign its own data. The result is a self-repairing agent that dynamically brings its perception of the world into agreement with its internal values, and vice versa. As systems transition from controlled simulations into the open, unpredictable physical world, this data-centric, zero-latency approach to robustness will be essential to bridging the gap between simulated safety and real-world resilience.

References

- Andrychowicz, Marcin, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2018. “Hindsight Experience Replay,” arXiv: 1707.01495 [cs.LG]. <https://arxiv.org/abs/1707.01495>.
- Baghoussi, Yassine, Carlos Soares, and João Mendes-Moreira. 2024. “Corrector LSTM: built-in training data correction for improved time-series forecasting” [in en]. *Neural Computing and Applications* 36 (26): 16213–16231. ISSN: 1433-3058. <https://doi.org/10.1007/s00521-024-09962-x>.
- Baghoussi, Yassine et al. 2025. “Read-write LSTM: A Novel Approach Integrating Backpropagation to Data in LSTM.” In *2025 IEEE International Conference on Data Mining (ICDM)*, 51–60. ISSN: 2374-8486. November. <https://doi.org/10.1109/ICDM65498.2025.00012>. <https://ieeexplore.ieee.org/abstract/document/11391874>.
- Baptista, André, Yassine Baghoussi, Carlos Soares, João Mendes-Moreira, and Miguel Arantes. 2021. “Pastprop-RNN: improved predictions of the future by correcting the past.” *arXiv preprint arXiv:2106.13881*.
- Bengio, Y., P. Frasconi, and P. Simard. 1993. “The problem of learning long-term dependencies in recurrent networks.” In *IEEE International Conference on Neural Networks*, 1183–1188 vol.3. March. <https://doi.org/10.1109/ICNN.1993.298725>. <https://ieeexplore.ieee.org/document/298725>.
- Du, Wenzhang. 2025. “Step-E: A Differentiable Data Cleaning Framework for Robust Learning with Noisy Labels” (November). <https://doi.org/10.48550/arXiv.2511.17040>. https://www.researchgate.net/publication/397895494_Step-E_A_Differentiable_Data_Cleaning_Framework_for_Robust_Learning_with_Noisy_Labels.
- Du, Yunshu, Garrett Warnell, Assefaw Gebremedhin, Peter Stone, and Matthew E. Taylor. 2021. “Lucid Dreaming for Experience Replay: Refreshing Past States with the Current Policy.” ArXiv:2009.13736 [cs.LG], no. arXiv:2009.13736 (April). <https://doi.org/10.48550/arXiv.2009.13736>. <http://arxiv.org/abs/2009.13736>.
- Elsayed, Mohamed, Gautham Vasan, and A. Rupam Mahmood. 2024. *Streaming Deep Reinforcement Learning Finally Works*. arXiv: 2410.14606 [cs.LG]. <https://arxiv.org/abs/2410.14606>.
- Gerven, Marcel van. 2025. *Neuromorphic Intelligence*. arXiv: 2509.11940 [cs.AI]. <https://arxiv.org/abs/2509.11940>.
- Grootendorst, Maarten. 2024. *A Visual Guide to Mamba and State Space Models* [in en]. <https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state>.

- Gu, Albert, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Re. 2020. “HiPPO: Recurrent Memory with Optimal Polynomial Projections.” ArXiv:2008.07669 [cs], no. arXiv:2008.07669 (October). <https://doi.org/10.48550/arXiv.2008.07669>. <http://arxiv.org/abs/2008.07669>.
- Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. arXiv: 1801.01290 [cs.LG]. <https://arxiv.org/abs/1801.01290>.
- Hausknecht, Matthew, and Peter Stone. 2017. *Deep Recurrent Q-Learning for Partially Observable MDPs*. eprint: 1507.06527. https://www.researchgate.net/publication/280329735_Deep_Recurrent_Q-Learning_for_Partially_Observable_MDPs.
- Huang, Sandy, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. *Adversarial Attacks on Neural Network Policies*. arXiv: 1702.02284 [cs.LG]. <https://arxiv.org/abs/1702.02284>.
- Jakubik, Johannes, Michael Vössing, Niklas Kühl, Jannis Walk, and Gerhard Satzger. 2024. *Data-Centric Artificial Intelligence*. arXiv: 2212.11854 [cs.AI]. <https://arxiv.org/abs/2212.11854>.
- Jim, Kam, Bill Horne, and C. Giles. 1994. “Effects of Noise on Convergence and Generalization in Recurrent Networks.” In *Advances in Neural Information Processing Systems*, vol. 7. MIT Press. <https://proceedings.neurips.cc/paper/1994/hash/6c3cf77d52820cd0fe646d38bc2145ca-Abstract.html>.
- Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra. 1998. “Planning and acting in partially observable stochastic domains.” *Artificial Intelligence* 101 (1): 99–134. ISSN: 0004-3702. [https://doi.org/https://doi.org/10.1016/S0004-3702\(98\)00023-X](https://doi.org/https://doi.org/10.1016/S0004-3702(98)00023-X). <https://www.sciencedirect.com/science/article/pii/S000437029800023X>.
- Krishnan, Sanjay, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. “ActiveClean: interactive data cleaning for statistical modeling.” *Proc. VLDB Endow.* 9, no. 12 (August): 948–959. ISSN: 2150-8097. <https://doi.org/10.14778/2994509.2994514>.
- Lee, Hyunsu. 2025. “Noise Resilience of Successor and Predecessor Feature Algorithms in One- and Two-Dimensional Environments.” *Sensors* 25 (3). ISSN: 1424-8220. <https://doi.org/10.3390/s25030979>. <https://www.mdpi.com/1424-8220/25/3/979>.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Belle-mare, Alex Graves, et al. 2015. “Human-level control through deep reinforcement learning” [in en]. *Nature* 518, no. 7540 (February): 529–533. ISSN: 0028-0836, 1476-4687. <https://doi.org/10.1038/nature14236>.
- Ni, Xinyi, and Lifeng Lai. 2024. *Robust Risk-Sensitive Reinforcement Learning with Conditional Value-at-Risk*. arXiv: 2405.01718 [cs.LG]. <https://arxiv.org/abs/2405.01718>.
- Nilim, Arnab, and Laurent Ghaoui. 2005. “Robust Control of Markov Decision Processes with Uncertain Transition Matrices.” *Operations Research* 53 (October): 780–798. <https://doi.org/10.1287/opre.1050.0216>.
- Pattanaik, Anay, Zhenyi Tang, Shuijing Liu, Gautham Bommanan, and Girish Chowdhary. 2017. *Robust Deep Reinforcement Learning with Adversarial Attacks*. arXiv: 1712.03632 [cs.LG]. <https://arxiv.org/abs/1712.03632>.

- Pinto, Lerrel, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. “Robust Adversarial Reinforcement Learning.” ArXiv:1703.02702 [cs], no. arXiv:1703.02702 (March). <https://doi.org/10.48550/arXiv.1703.02702>. <http://arxiv.org/abs/1703.02702>.
- Pitis, Silviu, Elliot Creager, Ajay Mandlekar, and Animesh Garg. 2022. “MoCoDA: Model-based Counterfactual Data Augmentation.” ArXiv:2210.11287 [cs.LG], no. arXiv:2210.11287 (October). <https://doi.org/10.48550/arXiv.2210.11287>. <http://arxiv.org/abs/2210.11287>.
- Rajhans, Mona, and Vishal Khawarey. 2026. *Empirical Analysis of Adversarial Robustness and Explainability Drift in Cybersecurity Classifiers*. arXiv: 2602.06395 [cs.CR]. <https://arxiv.org/abs/2602.06395>.
- Renggli, Cedric, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlaš, Wentao Wu, and Ce Zhang. 2021. *A Data Quality-Driven View of MLOps*. arXiv: 2102.07750 [cs.LG]. <https://arxiv.org/abs/2102.07750>.
- Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. “Proximal Policy Optimization Algorithms.” ArXiv abs/1707.06347. <https://api.semanticscholar.org/CorpusID:28695052>.
- Sedlmeier, Andreas, Thomas Gabor, Thomy Phan, and Lenz Belzner. 2019. “Uncertainty-Based Out-of-Distribution Detection in Deep Reinforcement Learning.” *Digitale Welt* 4 (November): 74–78. <https://doi.org/10.1007/s42354-019-0238-z>.
- Song, Chihyeon, Jaewoo Lee, and Jinkyoo Park. 2026. *Adaptive Replay Buffer for Offline-to-Online Reinforcement Learning*. arXiv: 2512.10510 [cs.LG]. <https://arxiv.org/abs/2512.10510>.
- Suilen, Marnix, Thom Badings, Eline M. Bovy, David Parker, and Nils Jansen. 2024. *Robust Markov Decision Processes: A Place Where AI and Formal Methods Meet*. arXiv: 2411.11451 [cs.AI]. <https://arxiv.org/abs/2411.11451>.
- Sutton, R.S., and A.G. Barto. 1998. *Reinforcement Learning: An Introduction*. A Bradford book. MIT Press. ISBN: 978-0-262-19398-6. <https://books.google.pt/books?id=CAFR6IBF4xYC>.
- Tessler, Chen, Yonathan Efroni, and Shie Mannor. 2019. “Action Robust Reinforcement Learning and Applications in Continuous Control.” ArXiv:1901.09184 [cs], no. arXiv:1901.09184 (May). <https://doi.org/10.48550/arXiv.1901.09184>. <http://arxiv.org/abs/1901.09184>.
- Tobin, Josh, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. arXiv: 1703.06907 [cs.RO]. <https://arxiv.org/abs/1703.06907>.
- Towers, Mark, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, et al. 2025. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. arXiv: 2407.17032 [cs.LG]. <https://arxiv.org/abs/2407.17032>.
- Yang, Jinggang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2024. *Generalized Out-of-Distribution Detection: A Survey*. arXiv: 2110.11334 [cs.CV]. <https://arxiv.org/abs/2110.11334>.
- Yang, Yifan, Alec Koppel, and Zheng Zhang. 2023. “A Gradient-based Approach for Online Robust Deep Neural Network Training with Noisy Labels.” ArXiv:2306.05046 [cs.LG], no. arXiv:2306.05046, <https://doi.org/10.48550/arXiv.2306.05046>. <http://arxiv.org/abs/2306.05046>.

- Zha, Daochen, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. 2023. “Data-centric Artificial Intelligence: A Survey.” ArXiv:2303.10158 [cs.LG], no. arXiv:2303.10158, <https://doi.org/10.48550/arXiv.2303.10158>. <http://arxiv.org/abs/2303.10158>.
- Zhang, Huan, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. 2021. *Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations*. arXiv: 2003.08938 [cs.LG]. <https://arxiv.org/abs/2003.08938>.

Chapter 8

Appendices

8.1 Appendix 1 - Complete Evaluation Result Tables

Ant-v4

Table 8.1: Ant-v4, mdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1171.7 ± 295.4	1496.7 ± 401.3	1290.2 ± 348.9	346.9 ± 93.0	699.6 ± 137.8	515.5 ± 105.3	1059.4 ± 215.9
0.1	1196.8 ± 227.8	1446.9 ± 419.1	1327.9 ± 417.0	275.5 ± 73.6	634.1 ± 79.6	398.3 ± 148.8	1024.6 ± 135.6
0.2	967.2 ± 207.2	1134.4 ± 224.2	1174.0 ± 283.4	269.4 ± 56.6	620.9 ± 105.8	466.6 ± 200.3	974.8 ± 222.8
0.4	582.6 ± 58.0	714.1 ± 100.1	856.5 ± 161.3	177.9 ± 85.2	645.6 ± 105.5	362.4 ± 50.6	720.4 ± 110.6
0.6	343.5 ± 85.0	352.8 ± 61.5	565.5 ± 146.0	195.8 ± 93.5	577.1 ± 116.2	315.6 ± 105.2	478.8 ± 69.6
0.8	197.3 ± 79.5	157.1 ± 40.4	384.4 ± 52.9	117.2 ± 13.3	549.4 ± 65.9	261.7 ± 77.1	279.5 ± 69.6
1.0	130.7 ± 78.6	73.0 ± 35.6	277.1 ± 31.5	54.5 ± 15.3	513.9 ± 23.0	261.2 ± 121.5	194.6 ± 43.2
1.3	61.9 ± 56.8	31.2 ± 23.7	165.6 ± 45.1	40.0 ± 21.8	500.4 ± 51.0	183.6 ± 77.4	111.4 ± 30.2
1.5	39.4 ± 72.8	9.4 ± 20.6	134.0 ± 62.1	27.8 ± 8.2	515.1 ± 66.0	145.3 ± 22.9	83.0 ± 27.0
1.8	20.2 ± 31.6	−13.0 ± 14.3	94.1 ± 62.3	4.6 ± 9.0	408.2 ± 68.2	133.0 ± 40.5	58.7 ± 32.7
2.0	14.7 ± 47.1	−7.1 ± 10.7	87.0 ± 65.2	−9.5 ± 13.0	454.4 ± 99.0	140.6 ± 81.0	35.6 ± 11.7

Table 8.2: Ant-v4, mdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	294.0 ± 150.4	525.4 ± 313.9	551.2 ± 44.3	167.9 ± 91.9	659.2 ± 121.9	429.8 ± 111.2	333.7 ± 192.1
0.1	332.3 ± 132.6	530.4 ± 417.4	577.7 ± 167.8	82.8 ± 74.4	497.2 ± 178.3	295.1 ± 194.4	370.5 ± 153.1
0.2	213.1 ± 119.4	256.6 ± 122.9	361.0 ± 127.9	80.0 ± 43.3	475.7 ± 193.7	370.4 ± 266.6	351.4 ± 244.2
0.4	119.2 ± 46.2	116.9 ± 15.7	283.3 ± 93.1	49.7 ± 36.3	540.5 ± 131.2	209.8 ± 106.4	168.8 ± 144.1
0.6	60.3 ± 38.0	57.2 ± 33.0	149.8 ± 76.9	89.1 ± 83.2	469.8 ± 166.0	198.4 ± 76.4	95.8 ± 55.4
0.8	22.9 ± 33.2	8.6 ± 17.7	83.2 ± 23.1	28.3 ± 16.0	499.9 ± 32.9	131.9 ± 81.3	38.5 ± 26.7
1.0	13.6 ± 26.7	-41.5 ± 44.7	54.8 ± 6.2	11.5 ± 11.4	462.2 ± 62.5	163.4 ± 114.3	30.8 ± 10.8
1.3	-34.4 ± 38.1	-43.0 ± 32.8	22.8 ± 15.2	16.5 ± 9.5	443.6 ± 40.0	68.6 ± 41.3	13.0 ± 7.3
1.5	-46.1 ± 52.0	-71.4 ± 29.9	17.0 ± 21.8	4.3 ± 10.7	434.7 ± 119.3	50.3 ± 12.0	4.5 ± 20.6
1.8	-49.2 ± 39.1	-98.5 ± 33.0	0.9 ± 28.7	-14.1 ± 18.9	249.8 ± 151.0	55.6 ± 20.5	-7.2 ± 18.4
2.0	-63.2 ± 49.0	-87.7 ± 26.3	-9.3 ± 22.3	-35.5 ± 28.0	360.5 ± 149.3	50.5 ± 26.1	-21.0 ± 10.7

Table 8.3: Ant-v4, mdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1182.2 ± 98.7	1039.3 ± 235.9	1243.9 ± 403.0	315.1 ± 166.1	557.5 ± 162.1	641.1 ± 64.8	1633.1 ± 692.1
0.1	1118.5 ± 88.1	965.3 ± 223.9	1260.4 ± 453.0	325.7 ± 121.7	529.6 ± 107.3	645.0 ± 108.5	1454.9 ± 545.8
0.2	1047.7 ± 65.6	868.8 ± 143.6	1106.0 ± 308.9	316.1 ± 170.4	583.5 ± 184.1	614.6 ± 154.5	1389.3 ± 474.5
0.4	744.0 ± 42.6	681.2 ± 97.1	807.7 ± 166.5	219.2 ± 69.6	620.1 ± 75.8	630.0 ± 166.8	909.0 ± 87.0
0.6	524.9 ± 120.8	480.6 ± 87.1	557.9 ± 131.3	139.1 ± 74.4	509.3 ± 77.5	563.7 ± 110.0	566.7 ± 100.6
0.8	330.2 ± 85.0	326.8 ± 67.4	370.7 ± 60.6	103.8 ± 38.8	534.3 ± 73.7	313.2 ± 99.7	356.7 ± 129.7
1.0	259.0 ± 71.6	243.9 ± 90.5	245.3 ± 37.1	85.8 ± 32.6	532.0 ± 101.3	409.0 ± 37.6	239.4 ± 125.7
1.3	177.5 ± 57.5	158.7 ± 78.0	160.9 ± 60.8	58.7 ± 40.2	527.1 ± 76.1	306.3 ± 94.0	149.9 ± 109.0
1.5	142.5 ± 74.3	125.7 ± 57.3	120.6 ± 54.4	37.7 ± 35.2	503.3 ± 45.5	362.1 ± 33.8	122.8 ± 114.3
1.8	103.3 ± 60.8	92.2 ± 47.8	99.0 ± 52.5	21.4 ± 31.2	458.9 ± 110.8	285.1 ± 80.6	96.3 ± 89.9
2.0	97.6 ± 53.3	77.1 ± 32.6	77.0 ± 60.1	24.3 ± 25.2	429.9 ± 109.2	269.3 ± 66.3	97.0 ± 107.3

Table 8.4: Ant-v4, mdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	304.8 ± 103.0	261.6 ± 127.0	522.2 ± 282.2	134.6 ± 113.7	398.3 ± 328.4	568.0 ± 121.5	1062.9 ± 848.5
0.1	312.4 ± 133.9	257.0 ± 131.9	510.9 ± 230.2	129.7 ± 87.7	353.9 ± 210.1	535.2 ± 258.1	626.1 ± 378.7
0.2	319.3 ± 65.6	165.9 ± 32.9	485.8 ± 132.2	173.8 ± 237.2	477.3 ± 291.1	553.9 ± 259.9	664.1 ± 421.2
0.4	153.4 ± 42.2	136.0 ± 10.9	257.1 ± 93.3	65.7 ± 31.6	506.6 ± 188.4	554.2 ± 223.9	360.8 ± 39.6
0.6	137.0 ± 52.2	88.8 ± 21.2	188.3 ± 58.4	38.2 ± 20.0	385.0 ± 207.0	421.7 ± 176.6	209.0 ± 75.3
0.8	59.3 ± 23.8	61.2 ± 3.3	77.0 ± 14.2	26.1 ± 11.4	453.8 ± 122.7	169.1 ± 83.6	100.2 ± 45.4
1.0	44.3 ± 23.0	53.8 ± 36.9	45.2 ± 21.1	20.3 ± 12.8	473.4 ± 173.5	310.4 ± 47.4	47.9 ± 45.1
1.3	17.2 ± 13.5	16.5 ± 22.4	20.1 ± 16.2	8.9 ± 13.5	459.2 ± 140.6	164.2 ± 107.8	25.0 ± 35.9
1.5	23.4 ± 14.4	12.2 ± 13.9	7.0 ± 22.9	-1.2 ± 18.2	476.4 ± 103.5	247.8 ± 35.1	11.5 ± 50.0
1.8	6.8 ± 22.9	6.0 ± 21.3	-0.4 ± 7.5	-16.7 ± 47.6	375.9 ± 156.0	135.5 ± 24.3	-4.3 ± 29.8
2.0	0.8 ± 15.8	-5.5 ± 11.3	0.1 ± 17.8	-0.8 ± 10.3	323.0 ± 164.9	124.6 ± 59.6	1.9 ± 41.4

Table 8.5: Ant-v4, pomdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	844.2 ± 155.8	985.3 ± 56.5	1368.8 ± 434.3	429.7 ± 115.4	615.9 ± 132.9	333.2 ± 90.3	1981.3 ± 435.3
0.1	806.7 ± 173.3	928.8 ± 180.7	1410.8 ± 422.1	447.0 ± 124.6	712.9 ± 101.1	400.3 ± 108.7	1870.0 ± 435.0
0.2	783.4 ± 107.8	801.8 ± 91.9	1271.3 ± 327.4	399.4 ± 146.5	694.4 ± 105.5	378.5 ± 55.0	1563.4 ± 352.2
0.4	472.1 ± 52.1	517.0 ± 138.4	942.6 ± 290.7	368.2 ± 52.5	703.1 ± 47.5	287.8 ± 55.8	935.2 ± 228.2
0.6	307.0 ± 99.9	329.9 ± 76.3	581.1 ± 156.8	290.3 ± 68.3	594.1 ± 53.0	245.7 ± 107.8	496.5 ± 166.9
0.8	193.4 ± 83.6	173.0 ± 72.1	328.6 ± 79.7	235.4 ± 33.1	542.5 ± 46.0	182.4 ± 64.3	280.3 ± 126.4
1.0	135.8 ± 71.9	122.9 ± 45.7	220.6 ± 66.9	153.1 ± 35.6	548.8 ± 79.9	142.4 ± 42.4	139.9 ± 96.1
1.3	59.0 ± 50.2	47.2 ± 36.7	136.6 ± 66.0	78.4 ± 28.1	501.8 ± 79.0	106.6 ± 81.4	82.4 ± 71.1
1.5	41.8 ± 53.5	30.1 ± 48.7	109.0 ± 36.9	70.4 ± 17.5	494.6 ± 52.1	63.5 ± 17.8	55.0 ± 60.9
1.8	33.6 ± 56.7	19.8 ± 30.2	70.7 ± 30.7	39.6 ± 12.9	484.8 ± 38.9	64.3 ± 31.2	33.0 ± 50.8
2.0	21.4 ± 48.7	7.1 ± 24.3	54.8 ± 35.6	26.7 ± 22.9	418.8 ± 67.3	60.6 ± 38.7	22.0 ± 38.8

Table 8.6: Ant-v4, pomdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	236.1 ± 61.8	203.2 ± 47.2	545.0 ± 359.1	225.1 ± 136.4	487.0 ± 310.4	201.0 ± 132.5	1088.7 ± 446.6
0.1	176.9 ± 76.0	183.7 ± 89.3	614.1 ± 281.2	329.3 ± 164.1	634.0 ± 111.4	259.4 ± 91.2	1002.7 ± 397.0
0.2	154.3 ± 22.5	167.7 ± 29.3	541.4 ± 172.9	249.2 ± 216.9	528.7 ± 218.0	236.1 ± 42.2	614.5 ± 346.9
0.4	86.5 ± 15.4	89.3 ± 46.9	394.5 ± 209.5	216.2 ± 91.1	620.2 ± 77.3	149.4 ± 66.3	287.9 ± 180.8
0.6	55.4 ± 20.3	45.6 ± 15.5	201.3 ± 104.5	152.6 ± 99.6	524.2 ± 56.0	131.8 ± 116.8	109.4 ± 89.7
0.8	24.2 ± 27.5	11.5 ± 31.0	81.3 ± 28.2	105.0 ± 63.3	450.4 ± 117.0	77.9 ± 48.9	46.7 ± 30.6
1.0	16.5 ± 24.4	4.9 ± 29.0	40.6 ± 34.2	66.8 ± 12.6	474.3 ± 131.6	65.4 ± 12.8	16.8 ± 45.0
1.3	−30.0 ± 56.1	−29.7 ± 46.6	18.8 ± 39.8	5.6 ± 32.2	417.8 ± 104.4	26.1 ± 38.7	−20.5 ± 53.3
1.5	−34.7 ± 60.8	−55.6 ± 71.3	11.5 ± 4.2	11.8 ± 16.8	362.8 ± 149.9	30.7 ± 12.0	−28.2 ± 44.5
1.8	−49.1 ± 62.4	−48.4 ± 50.0	−6.1 ± 15.1	3.3 ± 20.4	435.4 ± 56.4	17.8 ± 10.6	−42.0 ± 62.1
2.0	−56.1 ± 57.9	−62.4 ± 57.1	−7.6 ± 18.4	2.9 ± 17.3	314.5 ± 117.1	13.2 ± 6.8	−46.5 ± 48.6

Table 8.7: Ant-v4, pomdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1126.1 ± 489.1	1075.1 ± 657.3	1436.3 ± 382.0	362.5 ± 79.5	578.7 ± 117.0	643.6 ± 113.4	1233.6 ± 304.3
0.1	1164.1 ± 619.3	1103.4 ± 667.0	1439.1 ± 377.1	428.5 ± 175.3	514.9 ± 87.6	666.6 ± 86.7	1148.3 ± 251.1
0.2	974.9 ± 465.3	925.4 ± 523.3	1301.3 ± 351.9	338.7 ± 125.9	593.1 ± 113.0	490.6 ± 207.8	1052.5 ± 254.0
0.4	676.3 ± 80.4	570.4 ± 163.9	917.0 ± 261.8	278.1 ± 118.6	535.7 ± 100.1	411.0 ± 49.4	774.6 ± 112.1
0.6	466.2 ± 74.4	412.5 ± 133.0	589.4 ± 134.3	203.3 ± 103.7	550.7 ± 152.0	406.3 ± 95.4	561.2 ± 96.6
0.8	351.5 ± 67.1	283.3 ± 135.6	338.3 ± 97.0	191.1 ± 106.6	534.4 ± 89.8	319.8 ± 54.5	348.0 ± 38.0
1.0	252.1 ± 84.4	208.2 ± 130.6	238.5 ± 59.4	107.2 ± 76.8	493.5 ± 120.7	289.3 ± 68.0	232.4 ± 60.8
1.3	172.6 ± 97.0	138.9 ± 118.1	128.9 ± 32.9	55.6 ± 64.4	477.1 ± 85.1	254.6 ± 59.5	163.0 ± 97.2
1.5	156.9 ± 89.0	109.9 ± 102.8	91.1 ± 41.7	24.5 ± 43.0	465.9 ± 53.3	206.5 ± 39.0	124.5 ± 88.7
1.8	121.2 ± 78.8	93.8 ± 123.6	76.4 ± 34.6	29.8 ± 49.7	385.8 ± 107.3	158.1 ± 26.4	98.5 ± 68.7
2.0	113.2 ± 81.2	78.4 ± 100.8	53.9 ± 29.3	9.8 ± 42.1	402.4 ± 79.5	183.9 ± 10.5	98.0 ± 89.1

Table 8.8: Ant-v4, pomdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	439.4 ± 230.0	387.0 ± 321.2	714.8 ± 168.6	135.2 ± 43.8	468.5 ± 258.6	549.5 ± 236.5	460.6 ± 266.2
0.1	478.5 ± 290.4	366.0 ± 226.3	742.7 ± 180.7	247.8 ± 236.0	290.7 ± 190.8	580.5 ± 121.8	342.0 ± 154.1
0.2	298.6 ± 172.4	257.8 ± 206.3	573.0 ± 198.3	135.5 ± 82.6	441.1 ± 241.5	300.1 ± 265.9	289.7 ± 167.3
0.4	166.7 ± 37.9	102.8 ± 40.7	434.7 ± 206.2	107.0 ± 64.4	413.3 ± 249.7	247.3 ± 104.6	175.7 ± 22.9
0.6	98.1 ± 50.4	93.5 ± 51.8	199.3 ± 52.7	37.6 ± 20.5	486.1 ± 248.7	319.9 ± 162.7	146.2 ± 88.8
0.8	70.6 ± 16.5	62.3 ± 41.4	77.3 ± 64.9	44.2 ± 13.7	497.2 ± 151.2	194.8 ± 91.3	72.1 ± 10.7
1.0	48.2 ± 19.7	33.9 ± 31.8	55.6 ± 51.3	6.3 ± 17.8	420.2 ± 217.3	120.2 ± 90.2	45.0 ± 24.7
1.3	29.1 ± 29.8	9.1 ± 36.1	17.2 ± 17.7	-1.9 ± 6.5	386.5 ± 159.1	120.6 ± 70.1	9.5 ± 28.8
1.5	22.6 ± 27.8	-6.8 ± 36.1	1.1 ± 14.8	-19.8 ± 36.9	402.1 ± 125.0	105.1 ± 55.0	6.8 ± 35.6
1.8	7.0 ± 28.2	-17.5 ± 73.2	-2.5 ± 13.1	-1.9 ± 13.4	269.0 ± 123.7	59.5 ± 42.2	4.7 ± 22.8
2.0	7.6 ± 30.9	-26.0 ± 57.9	-13.0 ± 16.8	-12.7 ± 21.1	305.8 ± 77.4	58.5 ± 18.7	4.6 ± 33.5

CartPole-v1

Table 8.9: CartPole-v1, mdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	186.2 ± 20.5	195.0 ± 39.6	391.5 ± 164.2	298.0 ± 157.1	342.7 ± 186.1	163.9 ± 99.8	429.7 ± 67.7
0.1	168.6 ± 18.7	180.9 ± 33.8	353.2 ± 176.3	391.5 ± 127.9	376.5 ± 167.0	90.7 ± 64.5	369.2 ± 124.3
0.2	158.8 ± 17.4	176.9 ± 32.6	344.5 ± 183.2	442.1 ± 71.2	275.2 ± 157.3	316.3 ± 231.8	359.5 ± 132.1
0.4	154.0 ± 16.4	173.7 ± 31.6	340.3 ± 183.7	250.8 ± 163.5	282.3 ± 230.4	276.0 ± 139.6	354.1 ± 138.6
0.6	149.1 ± 16.3	169.5 ± 32.7	337.1 ± 184.6	344.3 ± 179.9	290.1 ± 177.5	133.4 ± 118.9	358.1 ± 147.3
0.8	142.0 ± 15.6	167.5 ± 31.5	333.7 ± 183.6	393.1 ± 60.8	137.2 ± 56.8	261.5 ± 216.1	349.5 ± 152.3
1.0	137.4 ± 14.7	165.8 ± 35.6	311.5 ± 167.6	214.7 ± 135.3	147.5 ± 179.1	45.0 ± 69.1	347.9 ± 152.7
1.3	127.6 ± 18.7	158.7 ± 35.0	299.4 ± 165.8	167.6 ± 54.7	66.1 ± 22.3	114.7 ± 98.1	324.2 ± 131.7
1.5	119.1 ± 24.6	153.7 ± 32.9	290.1 ± 154.0	134.7 ± 61.7	43.1 ± 20.9	69.4 ± 81.5	307.3 ± 119.5
1.8	105.0 ± 22.2	146.4 ± 45.6	263.4 ± 149.5	78.5 ± 47.5	42.5 ± 14.1	17.6 ± 6.9	267.5 ± 107.1
2.0	90.3 ± 24.9	132.4 ± 46.0	250.3 ± 140.7	70.5 ± 54.5	46.5 ± 25.5	16.0 ± 7.3	241.1 ± 100.6

Table 8.10: CartPole-v1, mdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	172.2 ± 16.2	183.2 ± 34.5	350.9 ± 178.0	293.4 ± 158.9	336.3 ± 193.1	155.6 ± 103.1	369.2 ± 98.5
0.1	155.9 ± 15.6	170.3 ± 30.8	339.6 ± 187.8	385.6 ± 136.6	363.7 ± 173.7	74.9 ± 76.1	335.0 ± 125.9
0.2	147.1 ± 14.6	166.8 ± 28.7	335.3 ± 191.9	421.8 ± 94.4	261.0 ± 161.8	311.7 ± 236.3	329.2 ± 132.9
0.4	142.5 ± 12.7	160.7 ± 26.2	325.9 ± 186.4	248.9 ± 173.1	272.7 ± 229.5	263.2 ± 131.4	321.6 ± 140.7
0.6	136.6 ± 14.9	155.4 ± 27.0	316.9 ± 185.4	337.2 ± 188.5	251.6 ± 187.4	125.8 ± 112.6	323.7 ± 147.4
0.8	128.1 ± 13.6	151.3 ± 26.2	305.0 ± 178.1	353.9 ± 101.5	113.5 ± 54.6	245.6 ± 201.0	308.6 ± 151.7
1.0	123.2 ± 13.2	149.4 ± 30.6	251.5 ± 125.6	162.2 ± 105.5	109.8 ± 122.6	40.6 ± 63.1	309.0 ± 156.8
1.3	100.7 ± 28.0	135.5 ± 29.0	222.4 ± 111.8	126.3 ± 65.2	30.6 ± 20.4	64.3 ± 63.8	224.4 ± 66.3
1.5	85.3 ± 33.2	130.6 ± 28.4	201.4 ± 97.5	77.9 ± 66.5	17.3 ± 0.7	44.6 ± 63.1	193.9 ± 59.0
1.8	55.4 ± 20.4	102.0 ± 44.7	159.5 ± 101.5	35.5 ± 22.2	27.1 ± 10.4	10.7 ± 1.8	155.1 ± 88.6
2.0	40.4 ± 13.6	82.1 ± 52.9	154.8 ± 98.3	46.8 ± 53.2	27.8 ± 12.2	10.3 ± 1.2	148.7 ± 85.3

Table 8.11: CartPole-v1, mdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	173.2 ± 15.7	252.4 ± 167.2	395.2 ± 165.0	315.4 ± 153.0	208.3 ± 78.8	215.2 ± 206.4	294.9 ± 37.3
0.1	161.1 ± 12.7	246.2 ± 171.3	357.3 ± 173.8	340.5 ± 146.5	256.5 ± 168.6	254.2 ± 166.3	261.3 ± 32.8
0.2	156.2 ± 11.2	243.9 ± 173.1	347.4 ± 181.0	266.4 ± 132.8	157.0 ± 36.8	204.7 ± 106.4	253.8 ± 33.7
0.4	152.0 ± 12.7	240.3 ± 174.6	342.8 ± 183.3	204.0 ± 82.1	279.5 ± 159.6	218.7 ± 142.3	249.3 ± 31.8
0.6	146.4 ± 12.3	235.3 ± 176.7	337.6 ± 184.7	363.2 ± 157.9	188.3 ± 81.7	117.1 ± 71.6	241.7 ± 32.5
0.8	143.8 ± 11.4	233.2 ± 178.1	330.1 ± 177.1	226.4 ± 140.0	194.6 ± 120.4	208.7 ± 203.9	243.3 ± 38.8
1.0	137.5 ± 12.4	227.3 ± 178.5	320.9 ± 178.7	213.6 ± 165.0	270.4 ± 117.4	229.4 ± 190.0	240.1 ± 39.9
1.3	132.2 ± 14.4	212.7 ± 169.8	305.8 ± 162.2	136.3 ± 106.3	55.2 ± 11.5	66.7 ± 69.6	232.1 ± 34.8
1.5	120.7 ± 18.0	204.1 ± 171.8	292.9 ± 155.2	145.1 ± 83.9	40.6 ± 20.1	69.0 ± 52.7	232.1 ± 46.2
1.8	109.4 ± 25.8	192.0 ± 168.4	270.8 ± 148.6	70.3 ± 42.3	38.1 ± 18.5	42.3 ± 48.2	232.1 ± 52.3
2.0	96.6 ± 27.0	175.7 ± 153.4	252.5 ± 137.8	59.9 ± 42.4	91.5 ± 52.3	41.0 ± 32.9	219.6 ± 47.0

Table 8.12: CartPole-v1, mdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	162.2 ± 14.1	244.5 ± 171.2	351.7 ± 177.4	291.4 ± 132.9	193.6 ± 66.6	217.2 ± 206.6	262.1 ± 26.7
0.1	151.2 ± 11.2	238.6 ± 174.6	340.6 ± 187.1	331.6 ± 151.8	250.5 ± 171.8	260.2 ± 194.5	236.2 ± 23.7
0.2	146.4 ± 9.9	237.5 ± 177.3	337.2 ± 190.3	241.3 ± 111.5	148.5 ± 33.2	185.6 ± 123.5	231.8 ± 27.8
0.4	141.3 ± 11.6	232.0 ± 177.1	329.6 ± 190.4	189.2 ± 70.8	270.4 ± 160.7	196.9 ± 124.6	222.2 ± 24.3
0.6	135.1 ± 9.5	224.3 ± 178.4	319.6 ± 185.1	358.3 ± 163.6	172.5 ± 64.7	107.0 ± 81.9	213.0 ± 23.9
0.8	130.5 ± 9.8	218.4 ± 180.9	287.3 ± 163.0	195.4 ± 106.3	155.7 ± 105.4	200.9 ± 209.3	209.9 ± 26.8
1.0	121.9 ± 14.5	205.7 ± 173.9	270.1 ± 146.8	188.1 ± 160.9	240.6 ± 110.2	220.5 ± 208.3	202.2 ± 30.7
1.3	110.0 ± 17.4	170.0 ± 137.4	227.4 ± 105.4	106.9 ± 89.8	20.4 ± 7.6	25.6 ± 16.0	187.4 ± 20.1
1.5	79.2 ± 36.2	153.0 ± 132.6	204.7 ± 102.9	112.1 ± 70.4	21.9 ± 12.0	44.7 ± 34.4	184.1 ± 29.8
1.8	62.8 ± 39.5	122.3 ± 114.0	168.6 ± 103.1	39.5 ± 24.7	20.2 ± 7.3	19.1 ± 16.8	172.0 ± 33.0
2.0	43.4 ± 24.6	103.1 ± 100.5	149.0 ± 84.6	35.0 ± 31.4	56.4 ± 35.7	19.4 ± 9.5	159.2 ± 36.1

Table 8.13: CartPole-v1, pomdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	196.3 ± 29.1	296.3 ± 144.3	182.5 ± 34.5	239.1 ± 62.6	328.9 ± 128.6	129.5 ± 70.4	421.2 ± 83.8
0.1	180.0 ± 23.8	280.9 ± 151.9	170.1 ± 31.0	359.4 ± 163.1	412.9 ± 86.8	337.4 ± 180.6	338.9 ± 108.0
0.2	172.6 ± 22.3	277.3 ± 154.1	166.1 ± 29.7	415.2 ± 162.9	396.3 ± 190.3	235.0 ± 183.8	326.2 ± 115.9
0.4	167.8 ± 21.5	273.3 ± 156.4	163.3 ± 30.1	288.8 ± 141.1	297.9 ± 150.4	86.8 ± 61.5	316.8 ± 122.3
0.6	164.2 ± 22.8	271.0 ± 158.3	158.2 ± 31.0	465.7 ± 55.7	407.6 ± 106.8	272.0 ± 214.3	310.9 ± 126.2
0.8	158.4 ± 21.9	270.2 ± 159.4	154.7 ± 31.1	322.9 ± 205.0	108.9 ± 111.2	282.9 ± 85.3	306.8 ± 129.2
1.0	153.1 ± 24.5	260.8 ± 162.6	150.2 ± 29.1	201.4 ± 183.6	176.4 ± 92.9	41.5 ± 61.9	300.5 ± 133.6
1.3	148.5 ± 29.1	261.9 ± 163.5	147.4 ± 35.2	195.3 ± 162.8	78.6 ± 46.9	82.0 ± 79.2	292.1 ± 139.3
1.5	137.2 ± 33.8	249.8 ± 160.1	138.2 ± 40.8	122.3 ± 73.5	75.9 ± 61.6	43.9 ± 28.0	294.9 ± 137.8
1.8	126.1 ± 40.6	246.2 ± 163.4	125.9 ± 40.7	77.0 ± 70.6	35.5 ± 11.0	26.8 ± 17.4	280.4 ± 140.5
2.0	111.3 ± 37.3	226.4 ± 140.5	118.0 ± 47.7	96.5 ± 94.1	46.5 ± 34.2	32.2 ± 15.0	277.3 ± 138.4

Table 8.14: CartPole-v1, pomdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	182.3 ± 25.6	281.8 ± 151.3	171.1 ± 31.1	228.5 ± 61.5	308.2 ± 136.6	119.8 ± 70.0	362.4 ± 95.9
0.1	168.4 ± 21.0	269.2 ± 157.9	160.0 ± 28.9	356.1 ± 166.8	393.4 ± 96.2	326.1 ± 178.8	312.3 ± 125.2
0.2	163.1 ± 21.0	265.0 ± 160.6	156.4 ± 26.9	416.3 ± 167.4	402.6 ± 194.9	227.6 ± 188.8	302.8 ± 131.5
0.4	155.8 ± 18.8	262.0 ± 162.6	151.4 ± 26.8	285.1 ± 143.5	283.9 ± 154.5	78.9 ± 67.1	294.4 ± 137.2
0.6	150.1 ± 19.1	256.9 ± 165.8	146.6 ± 27.4	453.1 ± 93.9	377.7 ± 142.0	260.3 ± 209.3	285.6 ± 143.0
0.8	143.7 ± 17.7	251.7 ± 169.0	140.6 ± 25.0	321.8 ± 206.4	83.6 ± 91.9	246.0 ± 67.5	283.0 ± 145.3
1.0	132.7 ± 23.3	242.6 ± 167.0	134.8 ± 27.7	194.5 ± 208.3	123.3 ± 88.5	40.0 ± 61.2	270.9 ± 153.3
1.3	121.7 ± 40.0	235.8 ± 173.1	124.1 ± 37.8	143.2 ± 129.0	36.2 ± 22.1	39.2 ± 53.7	259.5 ± 161.5
1.5	106.4 ± 45.9	201.2 ± 152.3	109.3 ± 56.3	99.0 ± 75.8	27.4 ± 18.6	20.4 ± 9.5	260.2 ± 160.3
1.8	87.5 ± 47.3	188.0 ± 154.1	82.4 ± 49.2	46.6 ± 48.6	19.0 ± 1.9	15.9 ± 11.6	236.9 ± 148.2
2.0	59.5 ± 43.4	139.9 ± 90.7	70.9 ± 42.6	42.9 ± 44.5	21.2 ± 12.5	12.8 ± 2.8	220.9 ± 140.8

Table 8.15: CartPole-v1, pomdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	179.1 ± 16.3	233.7 ± 65.8	182.6 ± 33.8	268.0 ± 156.0	228.0 ± 189.4	132.4 ± 80.1	358.5 ± 97.4
0.1	164.7 ± 10.9	203.3 ± 44.2	171.5 ± 31.4	299.9 ± 135.9	211.7 ± 160.7	119.6 ± 62.3	309.4 ± 96.3
0.2	159.4 ± 11.3	194.3 ± 40.9	167.2 ± 31.6	300.8 ± 171.0	315.3 ± 213.4	282.2 ± 180.5	270.2 ± 43.0
0.4	154.0 ± 10.5	197.3 ± 43.2	164.2 ± 30.9	341.8 ± 182.6	265.5 ± 163.8	143.6 ± 49.8	261.7 ± 41.2
0.6	150.5 ± 10.5	191.2 ± 41.2	157.9 ± 28.7	276.1 ± 155.5	282.1 ± 157.7	94.2 ± 69.6	255.2 ± 37.7
0.8	145.7 ± 9.4	184.6 ± 40.3	157.0 ± 31.5	238.4 ± 139.9	155.1 ± 107.4	252.7 ± 270.0	246.0 ± 34.4
1.0	141.0 ± 11.5	182.1 ± 38.9	150.8 ± 27.1	400.4 ± 199.3	141.6 ± 138.5	149.6 ± 208.9	244.0 ± 41.0
1.3	135.4 ± 12.6	186.8 ± 48.3	145.9 ± 35.4	260.0 ± 179.6	41.7 ± 43.4	137.9 ± 167.3	233.1 ± 32.2
1.5	129.2 ± 13.9	167.5 ± 50.5	138.2 ± 39.5	275.6 ± 146.6	56.1 ± 42.3	112.6 ± 106.3	228.2 ± 32.2
1.8	115.2 ± 11.7	141.9 ± 51.3	125.4 ± 40.8	109.7 ± 110.3	26.1 ± 10.9	38.3 ± 26.1	213.1 ± 31.5
2.0	102.4 ± 18.4	133.5 ± 57.3	113.4 ± 47.4	41.0 ± 32.7	36.7 ± 26.0	34.4 ± 26.8	205.4 ± 38.0

Table 8.16: CartPole-v1, pomdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	167.8 ± 13.5	211.2 ± 57.5	169.9 ± 29.1	265.2 ± 157.9	210.4 ± 207.7	124.2 ± 87.3	336.4 ± 110.4
0.1	154.2 ± 8.2	186.2 ± 40.8	161.1 ± 29.2	293.8 ± 139.7	185.6 ± 160.6	108.7 ± 63.4	271.2 ± 67.5
0.2	149.7 ± 9.2	181.7 ± 39.4	156.8 ± 27.8	293.0 ± 168.7	309.8 ± 219.7	274.2 ± 183.6	246.6 ± 37.1
0.4	142.7 ± 8.0	173.7 ± 33.6	152.2 ± 28.2	338.8 ± 186.2	254.2 ± 168.0	137.2 ± 49.0	232.3 ± 32.3
0.6	138.6 ± 11.2	165.7 ± 32.5	145.6 ± 26.0	270.5 ± 158.8	258.2 ± 166.6	84.6 ± 65.6	220.7 ± 25.7
0.8	133.0 ± 7.8	159.7 ± 32.2	142.4 ± 26.1	221.9 ± 127.9	130.4 ± 85.4	241.9 ± 264.0	213.4 ± 24.3
1.0	123.5 ± 9.5	153.5 ± 28.2	136.6 ± 24.9	401.6 ± 196.9	85.0 ± 74.7	141.0 ± 197.0	203.6 ± 25.5
1.3	114.8 ± 14.2	131.8 ± 43.5	122.2 ± 40.7	246.4 ± 187.0	22.4 ± 12.1	117.3 ± 144.1	192.2 ± 21.0
1.5	99.1 ± 26.6	107.5 ± 61.5	107.4 ± 55.7	188.2 ± 133.5	34.9 ± 30.1	80.9 ± 77.7	180.1 ± 21.0
1.8	64.3 ± 22.6	90.2 ± 66.3	81.9 ± 49.6	71.0 ± 92.1	14.6 ± 2.4	20.4 ± 15.0	161.0 ± 29.8
2.0	45.1 ± 18.2	83.0 ± 66.3	65.2 ± 40.9	21.4 ± 11.1	22.9 ± 17.4	21.0 ± 15.8	134.3 ± 34.4

HalfCheetah-v4

Table 8.17: HalfCheetah-v4, mdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1328.6 ± 415.9	2227.5 ± 1258.0	1845.0 ± 952.5	1435.6 ± 129.0	530.4 ± 265.2	1066.3 ± 91.2	1492.4 ± 183.2
0.1	1419.4 ± 131.4	2092.4 ± 1125.9	1825.4 ± 912.9	1420.8 ± 96.1	528.8 ± 388.1	1038.2 ± 88.7	1463.5 ± 118.6
0.2	1312.0 ± 101.8	1612.2 ± 424.0	1593.9 ± 603.0	1381.7 ± 84.7	667.6 ± 18.8	1021.6 ± 26.7	1369.8 ± 76.1
0.4	1065.4 ± 70.6	1005.4 ± 209.8	1128.9 ± 123.5	1150.2 ± 82.6	563.0 ± 64.5	884.3 ± 23.7	1105.3 ± 46.0
0.6	779.9 ± 17.3	651.9 ± 302.1	783.1 ± 125.5	891.6 ± 31.3	395.9 ± 52.7	655.0 ± 80.6	805.0 ± 39.3
0.8	498.3 ± 32.5	377.9 ± 294.9	501.3 ± 185.8	619.2 ± 21.5	269.1 ± 45.3	490.4 ± 70.7	536.9 ± 48.1
1.0	302.9 ± 72.9	192.9 ± 245.1	280.9 ± 196.5	427.5 ± 28.2	124.1 ± 95.3	350.1 ± 85.5	334.6 ± 49.6
1.3	106.6 ± 77.3	53.1 ± 195.9	96.0 ± 154.0	179.8 ± 54.8	55.0 ± 50.5	182.3 ± 66.5	141.2 ± 55.3
1.5	25.5 ± 80.2	-31.1 ± 177.9	30.0 ± 128.7	95.0 ± 21.6	13.2 ± 50.9	120.5 ± 74.8	57.9 ± 52.5
1.8	-49.6 ± 86.2	-104.1 ± 145.4	-51.8 ± 123.4	17.5 ± 32.4	-29.3 ± 52.4	17.5 ± 62.1	-38.7 ± 41.3
2.0	-80.0 ± 76.6	-141.3 ± 130.1	-90.8 ± 116.9	-39.4 ± 43.6	-39.3 ± 44.5	-7.8 ± 75.8	-82.5 ± 37.4

Table 8.18: HalfCheetah-v4, mdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1042.1 ± 897.3	2166.8 ± 1224.0	1685.2 ± 935.0	1389.0 ± 163.2	443.6 ± 290.1	1049.1 ± 127.1	1330.9 ± 429.4
0.1	1326.3 ± 216.0	1954.5 ± 932.8	1671.7 ± 693.4	1395.6 ± 97.2	466.7 ± 377.5	926.5 ± 233.1	1377.0 ± 194.7
0.2	1228.6 ± 163.5	1313.4 ± 71.2	1284.9 ± 95.9	1348.2 ± 108.0	577.7 ± 43.9	972.5 ± 41.2	1319.7 ± 81.3
0.4	947.3 ± 160.1	900.4 ± 289.3	906.3 ± 148.9	1121.1 ± 64.7	517.2 ± 90.7	842.0 ± 34.6	1040.7 ± 49.7
0.6	650.5 ± 96.8	537.7 ± 359.8	590.8 ± 257.6	886.3 ± 23.3	325.3 ± 61.8	597.1 ± 61.2	676.1 ± 36.2
0.8	331.2 ± 102.7	247.4 ± 344.1	357.2 ± 276.2	586.2 ± 18.5	235.7 ± 28.6	443.0 ± 50.5	423.7 ± 40.2
1.0	169.9 ± 49.1	73.4 ± 250.7	169.6 ± 216.9	397.9 ± 34.8	101.7 ± 93.5	303.2 ± 67.0	235.6 ± 48.3
1.3	-32.6 ± 85.3	-54.6 ± 212.8	-35.7 ± 180.7	144.4 ± 52.4	12.6 ± 40.5	141.2 ± 61.2	34.1 ± 52.7
1.5	-97.3 ± 92.6	-142.1 ± 193.0	-97.9 ± 178.6	53.8 ± 18.9	-27.1 ± 57.9	86.1 ± 79.0	-31.5 ± 59.3
1.8	-152.8 ± 86.0	-208.0 ± 176.2	-185.3 ± 173.8	-3.4 ± 40.9	-82.7 ± 63.9	-25.4 ± 43.8	-140.4 ± 46.1
2.0	-168.8 ± 65.4	-258.6 ± 167.8	-208.7 ± 170.1	-82.7 ± 47.1	-98.2 ± 25.8	-50.6 ± 89.5	-181.1 ± 43.0

Table 8.19: HalfCheetah-v4, mdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1246.9 ± 259.7	2654.0 ± 1327.8	1836.6 ± 902.0	1421.8 ± 66.0	853.4 ± 203.2	1300.7 ± 435.0	1417.9 ± 52.4
0.1	1232.9 ± 228.1	2548.8 ± 1227.2	1826.6 ± 925.7	1323.7 ± 149.3	819.2 ± 232.8	1552.8 ± 455.0	1392.5 ± 49.5
0.2	1197.7 ± 181.3	2247.8 ± 971.6	1609.0 ± 698.8	1344.8 ± 72.3	858.7 ± 168.5	1411.8 ± 478.2	1325.3 ± 46.9
0.4	1039.6 ± 90.3	1297.7 ± 269.3	1138.5 ± 139.7	1213.8 ± 43.7	785.6 ± 118.6	1234.1 ± 287.5	1136.1 ± 22.1
0.6	825.8 ± 62.0	738.9 ± 179.5	786.2 ± 132.8	954.9 ± 43.6	611.3 ± 84.8	1032.4 ± 259.5	830.5 ± 106.5
0.8	564.7 ± 52.2	371.0 ± 179.3	487.0 ± 210.6	728.3 ± 31.8	464.2 ± 67.1	800.0 ± 128.4	609.4 ± 45.0
1.0	357.6 ± 46.6	187.2 ± 207.7	296.1 ± 187.1	489.1 ± 30.3	290.7 ± 19.1	685.7 ± 225.1	403.1 ± 57.0
1.3	189.5 ± 35.3	8.4 ± 157.9	111.4 ± 145.8	276.6 ± 33.5	163.6 ± 66.5	455.4 ± 184.9	173.4 ± 56.6
1.5	97.4 ± 55.1	-65.2 ± 148.1	38.6 ± 146.6	167.2 ± 47.0	99.7 ± 68.7	348.4 ± 216.8	73.1 ± 53.1
1.8	3.3 ± 42.8	-139.6 ± 131.6	-50.8 ± 132.1	38.3 ± 37.3	16.9 ± 75.6	246.9 ± 209.7	-14.3 ± 56.1
2.0	-29.0 ± 41.6	-174.8 ± 112.1	-84.5 ± 111.8	5.7 ± 36.1	-7.0 ± 58.6	174.4 ± 183.0	-54.5 ± 65.5

Table 8.20: HalfCheetah-v4, mdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	654.1 ± 793.7	2590.1 ± 1302.1	1720.0 ± 822.8	1406.9 ± 71.3	793.2 ± 218.1	1163.9 ± 445.5	1372.4 ± 53.4
0.1	690.1 ± 691.6	2455.3 ± 1166.5	1664.4 ± 759.1	1234.5 ± 272.3	746.4 ± 247.1	1509.5 ± 375.3	1341.2 ± 45.7
0.2	780.2 ± 560.4	1964.5 ± 769.2	1287.6 ± 392.7	1317.2 ± 61.7	836.7 ± 165.0	1297.7 ± 341.5	1276.2 ± 49.7
0.4	739.3 ± 366.1	801.4 ± 396.9	929.2 ± 146.4	1196.0 ± 43.0	737.8 ± 101.3	1100.4 ± 183.3	1052.9 ± 38.2
0.6	618.8 ± 179.8	482.3 ± 222.7	636.3 ± 227.0	931.3 ± 27.5	562.6 ± 93.2	916.1 ± 194.7	575.8 ± 399.7
0.8	342.1 ± 153.2	142.4 ± 232.9	353.5 ± 288.1	695.7 ± 38.7	408.3 ± 44.2	734.9 ± 109.4	456.2 ± 107.8
1.0	162.3 ± 92.2	70.1 ± 214.9	175.4 ± 210.4	456.8 ± 39.6	263.8 ± 25.0	581.7 ± 118.8	261.2 ± 21.3
1.3	72.7 ± 42.7	-106.4 ± 162.6	-24.8 ± 170.1	228.9 ± 37.8	130.2 ± 69.3	396.0 ± 148.7	33.4 ± 25.2
1.5	-32.5 ± 64.4	-163.7 ± 173.2	-70.2 ± 192.0	137.1 ± 54.8	49.4 ± 50.5	295.9 ± 190.2	-36.4 ± 45.8
1.8	-110.0 ± 48.1	-252.6 ± 163.3	-158.7 ± 182.8	13.8 ± 33.4	-19.6 ± 68.2	183.9 ± 155.7	-119.9 ± 54.9
2.0	-121.5 ± 59.8	-296.5 ± 125.5	-199.0 ± 162.4	-13.5 ± 37.5	-35.1 ± 58.3	115.4 ± 127.1	-150.0 ± 64.1

Table 8.21: HalfCheetah-v4, pomdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1267.0 ± 505.7	1690.6 ± 79.0	1370.7 ± 140.7	1293.7 ± 156.7	752.6 ± 256.4	1067.0 ± 304.8	1941.2 ± 826.5
0.1	1419.3 ± 165.9	1617.1 ± 62.9	1369.0 ± 110.4	1320.7 ± 104.2	811.4 ± 207.4	1161.5 ± 91.5	1859.6 ± 771.0
0.2	1288.0 ± 132.7	1458.5 ± 44.8	1305.4 ± 106.3	1231.2 ± 109.3	803.7 ± 152.4	1091.9 ± 41.7	1646.9 ± 601.8
0.4	1017.7 ± 68.1	1071.9 ± 76.1	1052.8 ± 55.7	1085.3 ± 34.9	628.6 ± 144.8	894.1 ± 32.8	1134.0 ± 99.0
0.6	765.1 ± 14.3	722.2 ± 126.1	717.7 ± 17.3	833.2 ± 26.1	482.8 ± 62.9	670.2 ± 50.9	711.6 ± 133.6
0.8	517.5 ± 38.8	464.0 ± 135.9	436.1 ± 77.7	589.5 ± 22.4	309.1 ± 104.3	457.1 ± 180.3	410.5 ± 147.9
1.0	294.2 ± 14.8	285.4 ± 141.7	227.9 ± 89.0	400.3 ± 21.5	207.7 ± 130.8	336.6 ± 94.9	217.9 ± 170.8
1.3	105.8 ± 18.3	120.3 ± 131.1	27.2 ± 94.2	212.4 ± 17.9	68.1 ± 185.0	167.5 ± 116.3	27.2 ± 167.5
1.5	36.0 ± 14.9	45.0 ± 116.4	-41.2 ± 76.4	139.6 ± 13.4	17.9 ± 156.4	116.1 ± 56.9	-40.4 ± 141.2
1.8	-35.1 ± 17.9	-28.3 ± 118.2	-119.1 ± 86.7	5.1 ± 42.8	-45.8 ± 146.5	44.1 ± 74.3	-114.3 ± 146.1
2.0	-70.8 ± 22.2	-81.6 ± 106.3	-156.6 ± 70.7	-17.5 ± 36.9	-69.4 ± 175.9	9.9 ± 42.2	-147.6 ± 135.8

Table 8.22: HalfCheetah-v4, pomdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1018.3 ± 908.3	1648.5 ± 75.3	1204.6 ± 228.8	1276.4 ± 174.7	665.1 ± 325.6	953.7 ± 452.4	1875.4 ± 790.2
0.1	1372.0 ± 166.0	1570.9 ± 63.4	1265.1 ± 117.8	1293.7 ± 148.8	754.3 ± 242.7	1100.5 ± 52.2	1782.3 ± 712.3
0.2	1134.7 ± 121.4	1405.0 ± 40.2	1237.0 ± 132.2	1164.4 ± 130.4	783.6 ± 197.6	1055.6 ± 19.9	1476.0 ± 509.1
0.4	783.7 ± 255.5	1000.6 ± 76.2	971.1 ± 77.4	1039.5 ± 31.1	582.3 ± 144.3	866.4 ± 49.4	863.7 ± 285.6
0.6	592.5 ± 30.7	642.8 ± 125.4	616.4 ± 20.7	789.9 ± 26.0	454.8 ± 79.0	619.6 ± 46.1	445.4 ± 366.5
0.8	352.2 ± 98.3	380.0 ± 128.1	339.5 ± 82.0	555.1 ± 27.4	265.2 ± 114.8	437.3 ± 186.5	195.8 ± 192.3
1.0	83.2 ± 82.2	197.5 ± 137.7	125.9 ± 98.2	356.5 ± 35.2	168.2 ± 136.5	297.7 ± 110.2	57.7 ± 215.0
1.3	-48.5 ± 41.7	38.7 ± 135.2	-79.1 ± 100.6	165.6 ± 3.0	42.3 ± 187.0	131.1 ± 112.0	-109.6 ± 207.2
1.5	-78.4 ± 19.2	-50.8 ± 106.7	-145.0 ± 82.3	98.0 ± 12.8	-34.6 ± 161.2	102.3 ± 62.3	-167.0 ± 157.6
1.8	-135.8 ± 31.9	-105.3 ± 122.1	-218.8 ± 83.8	-29.0 ± 46.7	-74.0 ± 145.4	-4.8 ± 74.2	-239.0 ± 194.7
2.0	-168.7 ± 29.5	-168.0 ± 112.3	-252.8 ± 62.7	-48.2 ± 54.3	-109.3 ± 181.9	-38.1 ± 33.6	-252.3 ± 171.8

Table 8.23: HalfCheetah-v4, pomdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1394.6 ± 53.6	1375.4 ± 110.9	1355.3 ± 134.3	1780.1 ± 623.3	983.1 ± 176.6	1141.8 ± 200.4	2328.1 ± 882.3
0.1	1437.1 ± 174.2	1378.2 ± 86.5	1340.5 ± 127.3	1699.0 ± 685.6	939.6 ± 169.1	1153.2 ± 206.3	2260.8 ± 760.2
0.2	1359.8 ± 166.8	1300.7 ± 69.5	1288.6 ± 120.1	1563.6 ± 485.2	905.8 ± 178.9	1163.6 ± 135.9	1992.5 ± 562.8
0.4	1133.8 ± 94.5	1096.3 ± 32.4	1055.9 ± 50.6	1366.0 ± 277.7	837.6 ± 128.9	1072.6 ± 107.7	1320.5 ± 120.6
0.6	830.2 ± 60.3	863.6 ± 29.9	726.8 ± 37.2	1107.0 ± 179.0	695.0 ± 145.1	933.6 ± 53.8	854.9 ± 48.4
0.8	532.7 ± 54.5	601.0 ± 34.5	425.8 ± 100.0	802.9 ± 121.0	526.0 ± 158.4	742.4 ± 20.3	511.3 ± 127.8
1.0	330.0 ± 45.0	415.1 ± 43.4	208.1 ± 84.2	561.9 ± 7.8	362.3 ± 110.9	576.0 ± 48.4	331.9 ± 123.1
1.3	144.8 ± 44.2	214.2 ± 56.0	21.6 ± 105.9	290.1 ± 97.1	213.0 ± 60.9	394.2 ± 56.2	132.8 ± 124.4
1.5	52.1 ± 29.9	112.9 ± 54.6	-46.3 ± 86.5	123.4 ± 184.0	120.8 ± 30.5	227.8 ± 91.4	52.6 ± 129.2
1.8	-41.4 ± 20.8	47.7 ± 64.7	-125.7 ± 107.6	18.0 ± 149.6	50.8 ± 54.4	210.0 ± 93.4	-28.7 ± 125.2
2.0	-66.5 ± 12.4	0.7 ± 50.9	-165.4 ± 81.5	-76.5 ± 217.5	-16.9 ± 40.7	142.5 ± 67.2	-60.3 ± 111.8

Table 8.24: HalfCheetah-v4, pomdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1014.5 ± 406.8	1231.7 ± 225.3	1224.8 ± 217.7	1657.4 ± 600.1	919.7 ± 169.3	1171.9 ± 233.7	2115.3 ± 981.0
0.1	1305.9 ± 256.8	1334.2 ± 87.4	1251.7 ± 164.0	1542.3 ± 553.8	890.5 ± 236.2	1177.2 ± 238.4	2138.9 ± 699.3
0.2	1227.8 ± 210.8	1251.1 ± 77.0	1219.4 ± 131.2	1458.3 ± 320.9	859.6 ± 164.8	1125.9 ± 108.3	1701.8 ± 318.2
0.4	1035.8 ± 113.2	1004.9 ± 69.8	979.0 ± 54.1	1260.8 ± 156.0	774.1 ± 127.1	1025.8 ± 127.2	1030.6 ± 177.0
0.6	700.4 ± 99.2	750.2 ± 108.5	628.1 ± 8.8	1028.3 ± 168.3	638.1 ± 167.6	900.3 ± 57.3	671.3 ± 156.6
0.8	373.8 ± 77.3	423.8 ± 144.6	283.7 ± 129.5	731.8 ± 44.4	488.9 ± 150.0	715.0 ± 22.7	342.0 ± 213.7
1.0	193.7 ± 46.4	286.5 ± 88.7	35.6 ± 38.7	471.9 ± 100.5	328.0 ± 73.0	526.9 ± 64.0	213.9 ± 159.9
1.3	32.1 ± 53.4	83.0 ± 95.6	-91.2 ± 89.6	140.7 ± 212.6	194.8 ± 62.0	367.9 ± 61.4	28.1 ± 125.6
1.5	-63.7 ± 39.7	-13.7 ± 80.7	-139.6 ± 79.3	8.2 ± 267.8	73.4 ± 41.0	160.3 ± 151.8	-41.9 ± 145.3
1.8	-149.8 ± 31.2	-40.7 ± 65.4	-232.2 ± 115.6	-77.7 ± 224.5	2.0 ± 89.2	171.5 ± 93.9	-125.9 ± 130.9
2.0	-158.2 ± 25.3	-104.8 ± 62.0	-254.0 ± 83.2	-139.0 ± 275.3	-66.1 ± 61.0	119.3 ± 65.7	-169.5 ± 130.6

Hopper-v4

Table 8.25: Hopper-v4, mdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1857.5 ± 626.7	1794.6 ± 626.0	3218.1 ± 65.6	461.7 ± 196.2	44.1 ± 23.9	229.7 ± 214.7	3430.6 ± 169.0
0.1	1677.0 ± 447.1	1571.0 ± 345.8	2985.9 ± 145.7	482.0 ± 356.7	73.5 ± 75.6	102.3 ± 63.8	3245.6 ± 190.1
0.2	1404.2 ± 320.9	1311.2 ± 230.0	2508.5 ± 390.8	434.0 ± 321.6	55.7 ± 43.8	154.9 ± 173.6	2894.9 ± 320.3
0.4	918.0 ± 116.9	846.3 ± 50.3	1554.8 ± 705.1	358.9 ± 244.7	62.9 ± 31.0	74.7 ± 68.9	1796.0 ± 299.4
0.6	578.6 ± 80.7	560.5 ± 49.5	673.9 ± 242.3	190.2 ± 256.7	95.0 ± 51.3	79.4 ± 90.0	800.1 ± 244.8
0.8	439.5 ± 78.5	424.5 ± 58.7	483.4 ± 174.6	64.8 ± 43.8	88.6 ± 44.1	40.5 ± 28.2	527.6 ± 133.9
1.0	347.5 ± 90.4	337.8 ± 88.4	406.9 ± 92.6	71.5 ± 21.6	80.8 ± 41.7	48.7 ± 31.4	408.4 ± 52.6
1.3	271.3 ± 81.6	289.7 ± 94.8	334.8 ± 32.9	75.3 ± 7.5	112.4 ± 41.1	34.7 ± 25.3	332.2 ± 16.2
1.5	244.6 ± 71.0	255.3 ± 87.2	304.9 ± 22.4	37.4 ± 30.5	73.5 ± 32.1	57.8 ± 45.8	300.6 ± 26.2
1.8	218.6 ± 58.5	224.9 ± 82.1	270.2 ± 18.1	65.1 ± 16.6	80.7 ± 19.7	43.5 ± 47.3	271.2 ± 51.6
2.0	189.9 ± 45.8	214.5 ± 75.6	247.6 ± 27.8	39.5 ± 15.1	60.1 ± 32.4	17.8 ± 9.7	249.9 ± 56.0

Table 8.26: Hopper-v4, mdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1472.7 ± 346.9	1374.5 ± 310.5	2726.7 ± 412.5	159.5 ± 125.9	32.6 ± 16.4	110.3 ± 160.3	3397.3 ± 167.8
0.1	1203.9 ± 185.9	1145.0 ± 156.7	2087.3 ± 613.3	329.6 ± 448.0	71.7 ± 83.3	43.5 ± 31.5	2963.4 ± 479.5
0.2	1017.0 ± 147.5	935.6 ± 166.3	1413.4 ± 547.4	325.8 ± 451.0	34.6 ± 15.1	56.6 ± 43.9	2169.8 ± 698.3
0.4	618.9 ± 73.4	527.6 ± 83.7	748.2 ± 321.9	120.1 ± 109.7	42.0 ± 13.4	29.7 ± 19.2	873.1 ± 160.7
0.6	422.6 ± 93.1	364.7 ± 38.7	409.8 ± 99.9	114.0 ± 182.0	71.6 ± 35.6	22.5 ± 13.9	461.4 ± 146.1
0.8	270.6 ± 101.2	300.7 ± 88.2	355.2 ± 75.9	24.4 ± 11.1	48.5 ± 10.6	13.3 ± 8.1	359.8 ± 33.4
1.0	239.0 ± 105.0	252.2 ± 121.6	311.7 ± 49.4	41.4 ± 15.0	49.2 ± 29.2	18.4 ± 16.2	304.5 ± 35.4
1.3	168.2 ± 67.4	212.0 ± 104.6	248.3 ± 38.1	31.8 ± 12.6	87.5 ± 43.4	12.0 ± 6.5	243.6 ± 80.5
1.5	119.4 ± 49.1	165.3 ± 91.0	194.7 ± 41.4	20.4 ± 18.1	36.0 ± 17.8	28.4 ± 28.8	201.6 ± 70.6
1.8	103.2 ± 36.1	138.3 ± 73.9	167.9 ± 54.9	33.3 ± 7.8	53.6 ± 15.7	21.6 ± 28.8	164.9 ± 92.2
2.0	76.3 ± 41.1	120.3 ± 43.4	149.2 ± 66.3	22.3 ± 10.7	24.8 ± 16.4	7.7 ± 5.0	158.8 ± 90.7

Table 8.27: Hopper-v4, mdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1605.8 ± 1036.6	1626.9 ± 1084.0	3248.3 ± 80.3	148.9 ± 234.0	150.0 ± 45.1	239.9 ± 213.9	2668.3 ± 665.0
0.1	1401.3 ± 653.4	1422.4 ± 738.1	2928.1 ± 60.9	71.5 ± 58.3	133.5 ± 57.6	98.0 ± 78.9	2862.3 ± 283.3
0.2	1170.5 ± 287.5	1220.9 ± 364.5	2529.4 ± 274.7	127.9 ± 132.2	148.7 ± 57.0	129.7 ± 103.2	2780.2 ± 199.5
0.4	939.8 ± 99.9	826.1 ± 117.0	1449.2 ± 673.9	68.9 ± 80.0	158.2 ± 55.8	137.9 ± 150.8	2278.3 ± 421.1
0.6	653.6 ± 84.6	707.3 ± 85.9	671.5 ± 272.4	50.6 ± 39.5	149.1 ± 57.9	65.2 ± 59.2	1207.7 ± 119.0
0.8	502.9 ± 89.8	499.5 ± 105.3	472.1 ± 150.0	56.2 ± 35.3	141.8 ± 30.0	83.2 ± 105.9	790.6 ± 99.1
1.0	379.5 ± 77.1	367.5 ± 101.4	405.1 ± 86.1	39.8 ± 25.9	136.6 ± 39.0	62.9 ± 46.9	540.7 ± 84.5
1.3	274.2 ± 117.7	272.2 ± 105.9	327.3 ± 32.3	57.8 ± 70.7	117.5 ± 31.6	56.7 ± 39.0	390.2 ± 83.6
1.5	252.5 ± 98.6	258.7 ± 95.2	295.9 ± 25.9	60.2 ± 49.4	89.0 ± 51.3	82.6 ± 83.3	341.3 ± 60.3
1.8	211.6 ± 92.0	208.3 ± 95.4	273.9 ± 23.5	26.1 ± 23.2	106.6 ± 61.0	45.9 ± 34.6	283.6 ± 70.6
2.0	189.4 ± 92.1	200.7 ± 96.7	252.3 ± 15.5	46.7 ± 57.4	77.3 ± 19.6	91.2 ± 112.6	265.3 ± 71.5

Table 8.28: Hopper-v4, mdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1082.0 ± 769.1	1210.3 ± 980.1	2814.1 ± 400.6	117.4 ± 228.9	127.3 ± 63.3	76.8 ± 66.2	2434.5 ± 860.0
0.1	628.3 ± 109.9	620.1 ± 189.5	1868.2 ± 641.9	50.0 ± 55.3	97.1 ± 88.8	47.2 ± 44.1	2330.8 ± 721.5
0.2	539.6 ± 199.6	550.5 ± 203.7	1382.1 ± 237.8	95.4 ± 143.6	134.9 ± 59.7	57.1 ± 67.4	2160.1 ± 658.8
0.4	483.8 ± 167.5	437.0 ± 177.2	731.5 ± 270.9	30.9 ± 29.6	139.0 ± 58.4	54.7 ± 61.1	1447.8 ± 482.8
0.6	358.8 ± 142.2	384.6 ± 82.7	420.0 ± 125.2	32.4 ± 36.2	135.0 ± 58.2	23.6 ± 13.6	697.6 ± 59.9
0.8	223.9 ± 107.7	248.9 ± 127.1	349.5 ± 48.0	28.5 ± 23.7	115.1 ± 28.5	32.6 ± 27.2	458.7 ± 41.2
1.0	192.2 ± 146.3	157.0 ± 124.3	314.6 ± 37.4	18.9 ± 13.8	106.5 ± 47.6	36.7 ± 30.2	314.3 ± 80.0
1.3	137.7 ± 120.2	136.7 ± 132.2	239.5 ± 23.0	37.2 ± 52.7	76.4 ± 52.6	31.3 ± 17.4	251.1 ± 95.7
1.5	132.8 ± 124.4	126.7 ± 113.0	193.9 ± 61.4	32.7 ± 34.8	57.5 ± 34.8	29.9 ± 25.9	229.9 ± 95.6
1.8	104.8 ± 81.4	98.8 ± 75.2	177.8 ± 53.7	15.1 ± 17.8	71.4 ± 57.0	26.8 ± 25.3	168.5 ± 86.9
2.0	90.9 ± 67.4	99.8 ± 78.2	155.1 ± 49.6	25.5 ± 38.7	41.9 ± 32.6	53.1 ± 72.0	145.3 ± 62.0

Table 8.29: Hopper-v4, pomdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1749.3 ± 965.8	2150.8 ± 1261.6	3335.6 ± 150.9	76.8 ± 49.2	91.9 ± 51.5	71.3 ± 88.2	3173.9 ± 276.2
0.1	1658.6 ± 674.7	1819.8 ± 888.9	2927.6 ± 351.4	65.3 ± 32.9	100.3 ± 73.8	227.8 ± 426.6	2944.9 ± 505.2
0.2	1290.5 ± 476.7	1198.4 ± 321.2	2638.2 ± 591.4	53.5 ± 30.3	86.0 ± 40.6	225.9 ± 427.0	2397.0 ± 452.0
0.4	984.7 ± 505.5	929.4 ± 543.8	1569.3 ± 610.8	56.9 ± 38.7	125.3 ± 94.4	232.1 ± 428.9	1392.2 ± 496.3
0.6	629.6 ± 313.3	604.7 ± 394.8	729.4 ± 140.9	69.6 ± 58.2	91.5 ± 42.5	194.1 ± 390.2	759.1 ± 338.2
0.8	410.9 ± 149.0	380.6 ± 178.3	495.8 ± 94.3	70.8 ± 41.1	66.2 ± 22.7	34.9 ± 48.1	537.4 ± 204.9
1.0	299.4 ± 92.9	285.3 ± 86.1	406.7 ± 36.5	40.3 ± 24.4	62.7 ± 33.6	42.9 ± 48.0	406.0 ± 86.6
1.3	255.6 ± 85.3	218.4 ± 82.9	322.2 ± 36.4	41.8 ± 36.7	67.9 ± 15.7	157.3 ± 317.2	329.8 ± 64.6
1.5	209.6 ± 64.9	206.2 ± 73.2	285.8 ± 50.4	41.9 ± 19.5	67.1 ± 29.3	41.1 ± 44.9	291.7 ± 59.5
1.8	173.1 ± 37.3	163.0 ± 62.2	261.7 ± 33.3	26.5 ± 13.2	88.3 ± 29.3	66.2 ± 79.6	251.4 ± 67.8
2.0	178.8 ± 43.0	155.1 ± 58.6	236.5 ± 30.8	42.5 ± 47.3	104.9 ± 27.4	18.2 ± 16.1	236.8 ± 64.4

Table 8.30: Hopper-v4, pomdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1257.0 ± 876.5	1961.6 ± 1159.0	3289.6 ± 147.0	36.8 ± 11.9	74.5 ± 35.7	38.0 ± 61.2	2846.5 ± 580.6
0.1	983.4 ± 155.1	1068.7 ± 332.2	2324.6 ± 999.1	36.2 ± 15.1	79.8 ± 55.1	208.7 ± 436.2	2515.2 ± 897.5
0.2	724.1 ± 251.9	675.4 ± 240.2	1958.7 ± 985.7	26.7 ± 16.1	45.5 ± 24.3	209.9 ± 434.5	1383.1 ± 608.4
0.4	520.1 ± 281.0	410.8 ± 221.3	831.5 ± 196.7	27.6 ± 7.4	112.7 ± 96.0	215.3 ± 436.1	651.0 ± 156.1
0.6	315.9 ± 160.0	294.7 ± 155.3	403.6 ± 21.0	41.1 ± 47.2	72.6 ± 31.8	203.0 ± 437.3	416.7 ± 119.1
0.8	227.0 ± 113.7	184.0 ± 108.3	299.4 ± 123.8	32.1 ± 10.6	48.6 ± 18.7	17.4 ± 19.3	333.9 ± 66.8
1.0	151.0 ± 89.1	117.7 ± 65.1	242.8 ± 88.6	25.6 ± 15.7	47.8 ± 26.8	16.5 ± 10.9	285.0 ± 74.7
1.3	102.3 ± 25.8	53.0 ± 28.8	200.6 ± 108.8	23.6 ± 26.1	47.6 ± 22.1	138.3 ± 295.8	254.5 ± 83.5
1.5	65.4 ± 34.4	59.2 ± 23.7	160.4 ± 81.6	24.2 ± 12.2	45.8 ± 27.1	22.8 ± 27.9	193.5 ± 69.2
1.8	61.4 ± 27.4	42.2 ± 28.9	145.6 ± 58.1	19.5 ± 9.2	55.3 ± 21.2	32.5 ± 46.4	148.2 ± 86.8
2.0	70.9 ± 15.2	41.6 ± 28.6	132.2 ± 52.1	26.0 ± 37.1	74.9 ± 9.5	4.7 ± 3.0	123.9 ± 65.0

Table 8.31: Hopper-v4, pomdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1431.2 ± 337.8	1470.7 ± 350.8	3332.6 ± 147.8	231.6 ± 211.1	110.1 ± 90.7	308.9 ± 463.6	2339.3 ± 797.0
0.1	1564.9 ± 350.5	1495.5 ± 375.7	2959.2 ± 313.6	335.3 ± 347.2	80.8 ± 82.4	316.3 ± 455.4	2299.9 ± 718.2
0.2	1488.9 ± 337.4	1372.9 ± 305.4	2612.0 ± 642.6	291.1 ± 392.7	141.9 ± 79.6	372.3 ± 437.1	2102.4 ± 632.2
0.4	1060.8 ± 241.4	940.4 ± 185.2	1459.1 ± 608.4	317.9 ± 291.2	106.5 ± 59.6	179.7 ± 119.2	1425.0 ± 473.7
0.6	681.9 ± 128.8	685.7 ± 109.6	715.3 ± 171.6	110.7 ± 55.6	101.9 ± 82.9	311.7 ± 469.3	936.4 ± 242.6
0.8	436.5 ± 65.2	453.9 ± 86.8	502.3 ± 88.5	113.2 ± 109.1	102.0 ± 44.5	30.8 ± 27.8	687.2 ± 201.4
1.0	294.6 ± 27.2	316.7 ± 88.4	407.5 ± 47.3	146.3 ± 133.2	105.8 ± 36.9	63.5 ± 48.3	490.9 ± 108.1
1.3	205.2 ± 10.8	250.4 ± 76.6	323.7 ± 49.9	81.5 ± 62.1	80.3 ± 47.2	39.0 ± 29.2	390.4 ± 43.9
1.5	164.9 ± 14.3	198.5 ± 86.1	292.1 ± 35.6	106.2 ± 153.1	79.7 ± 41.4	61.9 ± 68.6	336.1 ± 12.2
1.8	148.2 ± 16.4	184.1 ± 61.5	255.9 ± 36.0	64.1 ± 51.8	71.4 ± 49.1	55.3 ± 38.7	298.9 ± 38.7
2.0	144.5 ± 9.7	173.9 ± 63.7	232.3 ± 24.1	97.1 ± 104.7	78.2 ± 47.8	49.0 ± 67.1	252.1 ± 45.9

Table 8.32: Hopper-v4, pomdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	1168.0 ± 259.4	1196.7 ± 274.8	3287.3 ± 143.6	75.7 ± 49.8	82.0 ± 86.3	284.7 ± 472.7	1852.8 ± 964.3
0.1	1126.2 ± 258.0	973.9 ± 270.4	2424.2 ± 890.8	277.3 ± 399.4	73.2 ± 91.8	275.6 ± 476.6	1740.1 ± 820.9
0.2	898.1 ± 301.8	731.6 ± 211.2	2018.0 ± 925.0	268.9 ± 402.9	108.7 ± 74.1	293.4 ± 464.0	1415.9 ± 440.4
0.4	534.1 ± 250.6	440.0 ± 212.5	694.6 ± 135.9	242.1 ± 362.7	72.6 ± 56.7	127.1 ± 90.6	832.4 ± 130.3
0.6	282.4 ± 153.9	327.7 ± 163.4	372.3 ± 61.0	47.7 ± 38.7	72.9 ± 67.5	282.6 ± 470.7	598.4 ± 168.7
0.8	168.9 ± 64.7	204.6 ± 107.9	311.5 ± 121.5	55.9 ± 49.7	68.1 ± 40.2	15.7 ± 14.4	442.5 ± 115.6
1.0	112.3 ± 19.1	154.3 ± 99.3	247.8 ± 111.9	55.1 ± 52.9	67.4 ± 37.6	40.5 ± 32.8	307.7 ± 20.3
1.3	94.2 ± 24.5	130.3 ± 81.3	195.3 ± 111.1	37.5 ± 37.9	57.6 ± 40.7	21.5 ± 13.1	262.8 ± 29.8
1.5	79.4 ± 15.8	105.4 ± 56.5	162.6 ± 76.4	64.9 ± 117.5	50.5 ± 34.3	37.4 ± 40.5	234.5 ± 51.3
1.8	77.8 ± 14.8	82.0 ± 17.7	140.9 ± 61.3	36.0 ± 38.5	41.0 ± 36.0	25.8 ± 15.5	195.4 ± 70.9
2.0	71.5 ± 4.5	71.3 ± 3.7	125.7 ± 49.0	37.3 ± 41.4	48.1 ± 31.5	36.8 ± 52.2	142.3 ± 48.4

LunarLander-v3

Table 8.33: LunarLander-v3, mdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	92.8 ± 126.1	214.5 ± 57.3	204.7 ± 59.0	−80.3 ± 92.0	−342.3 ± 132.1	−53.4 ± 99.3	242.5 ± 40.6
0.1	111.1 ± 133.3	197.3 ± 79.6	182.4 ± 78.8	−36.8 ± 129.4	−265.7 ± 69.3	−137.7 ± 51.6	238.6 ± 41.0
0.2	137.0 ± 112.5	197.5 ± 75.9	173.6 ± 63.1	−55.4 ± 118.9	−205.3 ± 68.0	−81.4 ± 147.8	214.5 ± 51.7
0.4	142.8 ± 84.4	139.3 ± 60.6	117.9 ± 88.2	−96.1 ± 103.1	−189.1 ± 50.3	−115.4 ± 102.6	175.7 ± 71.8
0.6	89.1 ± 70.2	81.2 ± 30.8	72.2 ± 58.4	−83.0 ± 109.0	−170.3 ± 73.0	−185.2 ± 36.2	108.3 ± 59.4
0.8	41.8 ± 32.3	52.5 ± 30.0	34.4 ± 47.2	−80.0 ± 116.7	−169.8 ± 40.1	−159.6 ± 72.8	60.9 ± 25.1
1.0	15.8 ± 21.3	34.9 ± 34.8	18.5 ± 42.5	−39.6 ± 42.6	−162.2 ± 58.9	27.0 ± 96.8	40.7 ± 32.7
1.3	3.4 ± 8.4	9.6 ± 19.5	5.7 ± 34.8	−67.1 ± 75.3	−118.3 ± 24.9	−10.7 ± 24.2	16.1 ± 31.5
1.5	−3.7 ± 13.0	−4.5 ± 21.1	−6.1 ± 35.6	−116.3 ± 90.0	−149.0 ± 39.2	−88.0 ± 76.3	8.4 ± 26.6
1.8	−18.1 ± 9.1	−15.9 ± 26.0	−24.1 ± 29.2	−121.7 ± 120.6	−151.4 ± 27.3	−31.8 ± 66.2	−7.0 ± 25.4
2.0	−19.3 ± 16.4	−22.3 ± 17.3	−34.6 ± 21.4	−135.1 ± 117.3	−156.8 ± 18.8	−86.4 ± 98.8	−9.2 ± 35.4

Table 8.34: LunarLander-v3, mdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	−61.4 ± 169.1	125.4 ± 90.1	89.3 ± 122.4	−131.5 ± 73.6	−386.0 ± 129.1	−101.4 ± 86.6	165.0 ± 89.7
0.1	−24.5 ± 168.0	125.7 ± 103.1	55.8 ± 129.3	−107.2 ± 85.8	−357.4 ± 107.9	−176.1 ± 45.5	153.6 ± 99.9
0.2	−2.2 ± 166.5	126.2 ± 94.7	50.4 ± 86.4	−105.9 ± 124.9	−253.3 ± 95.3	−142.2 ± 124.5	96.2 ± 80.0
0.4	−13.3 ± 104.0	42.2 ± 24.1	28.9 ± 61.3	−131.6 ± 99.2	−223.2 ± 50.4	−161.0 ± 67.6	72.0 ± 31.2
0.6	−72.6 ± 49.4	8.0 ± 10.3	−0.1 ± 60.8	−134.0 ± 83.2	−216.3 ± 114.3	−205.4 ± 51.4	9.7 ± 11.0
0.8	−83.0 ± 41.0	−11.1 ± 16.9	−27.4 ± 52.6	−136.7 ± 97.7	−202.8 ± 79.2	−183.9 ± 52.7	−2.2 ± 21.1
1.0	−89.6 ± 44.5	−15.5 ± 16.1	−40.2 ± 53.7	−88.0 ± 51.4	−197.4 ± 65.4	−8.8 ± 136.7	−2.8 ± 30.5
1.3	−72.4 ± 28.5	−53.6 ± 48.3	−48.0 ± 35.7	−95.0 ± 85.8	−131.4 ± 25.7	−25.5 ± 27.8	−30.8 ± 23.8
1.5	−75.3 ± 27.2	−60.7 ± 29.1	−64.2 ± 34.9	−158.1 ± 86.4	−186.1 ± 57.6	−113.0 ± 75.9	−37.0 ± 28.1
1.8	−97.0 ± 38.3	−82.2 ± 44.2	−81.8 ± 27.3	−145.0 ± 136.2	−194.4 ± 41.9	−69.2 ± 78.7	−55.8 ± 23.7
2.0	−83.3 ± 32.3	−84.6 ± 39.4	−91.0 ± 18.9	−177.8 ± 128.6	−186.9 ± 32.6	−116.4 ± 131.6	−53.6 ± 37.0

Table 8.35: LunarLander-v3, mdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	44.8 ± 74.7	194.8 ± 55.3	179.1 ± 78.6	-7.8 ± 133.8	-144.5 ± 56.9	-221.5 ± 127.1	260.6 ± 8.9
0.1	65.0 ± 53.5	197.2 ± 58.5	175.9 ± 65.9	-83.9 ± 92.6	-124.7 ± 47.0	-90.0 ± 157.0	253.0 ± 9.9
0.2	73.9 ± 30.6	189.4 ± 74.3	156.8 ± 64.8	-31.4 ± 179.7	-140.2 ± 70.7	-95.6 ± 120.8	250.2 ± 8.7
0.4	61.5 ± 25.5	158.2 ± 81.9	75.1 ± 56.6	-61.0 ± 170.1	-104.8 ± 47.1	-148.7 ± 186.9	244.5 ± 16.7
0.6	41.7 ± 34.3	114.6 ± 54.5	47.8 ± 59.4	-89.9 ± 182.0	-174.6 ± 79.4	-135.3 ± 129.5	136.6 ± 62.0
0.8	33.6 ± 38.2	52.5 ± 15.7	36.4 ± 54.8	40.8 ± 189.4	-124.0 ± 25.0	-91.9 ± 75.6	92.0 ± 36.5
1.0	21.8 ± 38.2	34.4 ± 14.4	15.2 ± 51.0	78.7 ± 170.5	-114.7 ± 26.6	-48.9 ± 61.1	43.4 ± 13.1
1.3	5.4 ± 44.0	10.0 ± 12.4	-0.9 ± 47.3	-92.5 ± 93.1	-124.1 ± 13.9	-144.4 ± 102.2	20.7 ± 3.7
1.5	-7.5 ± 42.9	-1.1 ± 14.2	-12.1 ± 37.4	-41.9 ± 78.0	-129.7 ± 6.1	-178.3 ± 94.1	7.8 ± 8.0
1.8	-12.0 ± 47.6	-12.1 ± 18.0	-26.8 ± 31.8	-63.9 ± 52.2	-126.9 ± 21.7	-136.3 ± 90.9	-3.3 ± 8.7
2.0	-18.9 ± 38.1	-28.0 ± 33.9	-30.2 ± 32.7	-40.9 ± 102.9	-134.5 ± 23.9	-134.9 ± 71.1	-16.8 ± 10.1

Table 8.36: LunarLander-v3, mdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	-90.3 ± 56.0	75.0 ± 122.1	39.7 ± 107.8	-51.0 ± 152.7	-183.4 ± 58.8	-262.5 ± 135.6	220.8 ± 21.2
0.1	-76.1 ± 64.8	92.5 ± 81.8	31.7 ± 109.4	-121.6 ± 84.9	-152.1 ± 68.3	-157.5 ± 117.3	188.1 ± 29.7
0.2	-70.5 ± 50.4	75.6 ± 116.8	51.3 ± 94.8	-92.8 ± 167.5	-191.5 ± 122.6	-128.1 ± 102.9	179.2 ± 26.6
0.4	-61.7 ± 35.8	41.1 ± 118.1	-0.5 ± 64.7	-70.9 ± 181.9	-133.5 ± 56.5	-190.2 ± 191.4	162.5 ± 46.2
0.6	-83.6 ± 72.0	-5.7 ± 24.4	-15.5 ± 74.0	-138.9 ± 169.6	-199.5 ± 94.7	-158.2 ± 122.4	40.6 ± 16.5
0.8	-79.2 ± 55.7	-23.3 ± 10.0	-26.2 ± 69.0	26.5 ± 201.4	-142.7 ± 23.9	-133.4 ± 59.6	25.0 ± 12.4
1.0	-89.6 ± 86.2	-24.2 ± 13.0	-51.7 ± 48.4	45.4 ± 170.3	-135.3 ± 37.4	-79.6 ± 58.8	-2.5 ± 17.5
1.3	-101.2 ± 87.3	-44.7 ± 4.4	-85.9 ± 67.1	-110.4 ± 81.4	-156.2 ± 32.6	-173.4 ± 120.5	-27.2 ± 12.8
1.5	-99.4 ± 78.9	-50.8 ± 12.6	-91.7 ± 64.5	-60.4 ± 67.5	-146.8 ± 9.8	-194.9 ± 96.9	-42.4 ± 16.1
1.8	-101.2 ± 62.6	-64.2 ± 14.1	-120.7 ± 41.0	-108.2 ± 62.7	-151.3 ± 28.5	-172.5 ± 91.8	-46.4 ± 11.4
2.0	-108.8 ± 63.0	-96.2 ± 48.0	-89.5 ± 36.1	-60.3 ± 121.7	-144.7 ± 26.2	-165.3 ± 64.0	-63.3 ± 17.2

Table 8.37: LunarLander-v3, pomdp, train noise 0.0 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	129.2 ± 97.0	204.7 ± 84.4	241.5 ± 41.2	-122.5 ± 146.3	-214.1 ± 118.0	-160.6 ± 100.9	267.8 ± 8.2
0.1	124.3 ± 107.7	231.5 ± 28.7	243.7 ± 29.6	-100.1 ± 199.7	-149.4 ± 35.3	-140.7 ± 166.9	268.9 ± 12.8
0.2	122.7 ± 71.8	232.9 ± 33.7	237.7 ± 43.7	-99.8 ± 220.2	-150.4 ± 48.3	-147.4 ± 117.5	261.8 ± 16.2
0.4	49.6 ± 31.9	150.7 ± 61.4	189.3 ± 51.8	-156.8 ± 152.8	-125.5 ± 13.0	-252.0 ± 51.3	197.8 ± 60.5
0.6	7.8 ± 55.5	75.0 ± 27.2	91.5 ± 43.6	60.9 ± 134.8	-147.0 ± 23.1	-133.0 ± 74.2	81.7 ± 30.0
0.8	-35.4 ± 70.7	30.7 ± 6.0	43.1 ± 21.1	-73.5 ± 89.4	-153.1 ± 41.5	-48.6 ± 163.7	49.0 ± 26.6
1.0	-71.4 ± 77.5	12.1 ± 4.2	15.9 ± 22.8	-19.6 ± 115.2	-148.9 ± 20.6	-154.6 ± 165.2	26.7 ± 21.7
1.3	-90.2 ± 85.4	-1.5 ± 2.4	-4.5 ± 5.4	3.9 ± 91.2	-181.3 ± 32.5	-102.9 ± 110.4	7.4 ± 25.2
1.5	-99.6 ± 74.7	-16.9 ± 7.2	-23.1 ± 9.9	-87.3 ± 125.8	-118.4 ± 14.2	-124.2 ± 54.1	-7.2 ± 26.2
1.8	-133.8 ± 94.9	-32.7 ± 3.8	-38.9 ± 12.4	-153.0 ± 194.1	-139.8 ± 14.6	-82.5 ± 146.5	-20.8 ± 20.7
2.0	-127.1 ± 85.9	-43.7 ± 11.3	-43.4 ± 20.4	-177.9 ± 125.9	-159.7 ± 20.6	-188.4 ± 115.4	-28.0 ± 23.7

Table 8.38: LunarLander-v3, pomdp, train noise 0.0 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	-12.1 ± 123.6	113.2 ± 110.0	152.2 ± 98.7	-166.8 ± 101.2	-255.7 ± 151.9	-195.2 ± 77.7	229.2 ± 19.2
0.1	-25.2 ± 84.9	132.0 ± 52.6	159.6 ± 71.3	-128.4 ± 213.1	-170.8 ± 61.5	-195.0 ± 137.9	223.9 ± 25.5
0.2	-57.6 ± 50.4	137.8 ± 48.5	145.5 ± 91.8	-160.8 ± 237.9	-188.2 ± 68.7	-187.2 ± 92.6	197.3 ± 52.9
0.4	-220.0 ± 163.5	36.2 ± 20.8	40.4 ± 34.4	-218.8 ± 144.0	-156.7 ± 33.2	-278.1 ± 67.4	101.4 ± 77.4
0.6	-225.0 ± 198.8	-4.0 ± 20.4	5.9 ± 14.8	-1.5 ± 164.8	-175.9 ± 29.7	-162.4 ± 78.1	24.1 ± 17.3
0.8	-213.1 ± 173.9	-10.7 ± 17.0	-15.7 ± 7.8	-124.5 ± 58.9	-181.1 ± 52.3	-90.9 ± 130.1	10.0 ± 20.8
1.0	-274.0 ± 169.5	-33.7 ± 14.0	-28.2 ± 20.3	-57.6 ± 105.8	-168.4 ± 31.8	-197.5 ± 178.2	-9.6 ± 19.3
1.3	-271.0 ± 181.6	-45.8 ± 10.3	-49.9 ± 14.6	-34.2 ± 77.4	-187.5 ± 31.2	-156.1 ± 99.8	-27.8 ± 27.1
1.5	-289.3 ± 142.4	-51.5 ± 9.6	-67.2 ± 19.8	-128.1 ± 136.5	-130.6 ± 14.3	-176.8 ± 30.1	-49.2 ± 35.4
1.8	-331.0 ± 191.7	-78.3 ± 12.5	-97.4 ± 22.7	-199.1 ± 227.3	-151.5 ± 14.2	-125.6 ± 176.0	-65.5 ± 23.4
2.0	-283.1 ± 154.3	-92.5 ± 17.7	-102.3 ± 41.6	-223.5 ± 139.5	-184.8 ± 41.7	-219.2 ± 130.2	-67.6 ± 27.8

Table 8.39: LunarLander-v3, pomdp, train noise 0.1 — Mean Reward

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	145.4 ± 56.4	127.4 ± 97.1	231.8 ± 62.5	-134.9 ± 162.9	-85.5 ± 78.4	-113.5 ± 85.0	227.3 ± 14.2
0.1	143.7 ± 53.0	142.7 ± 88.7	235.9 ± 57.2	-230.4 ± 29.0	-37.5 ± 14.0	-170.4 ± 183.6	226.9 ± 11.7
0.2	171.7 ± 38.1	150.8 ± 80.0	236.9 ± 60.3	-164.5 ± 81.7	-54.6 ± 23.9	-54.3 ± 143.6	233.4 ± 15.1
0.4	154.8 ± 22.5	126.5 ± 54.7	184.2 ± 70.7	-97.9 ± 127.5	-65.7 ± 51.3	-103.9 ± 95.2	211.7 ± 18.6
0.6	129.8 ± 4.4	82.9 ± 21.7	97.9 ± 49.8	-120.1 ± 96.2	-44.2 ± 27.5	-234.0 ± 81.5	195.8 ± 22.4
0.8	87.3 ± 35.4	50.2 ± 12.6	45.3 ± 38.4	-37.7 ± 199.6	-66.6 ± 12.6	-77.4 ± 182.4	99.5 ± 25.1
1.0	49.9 ± 31.1	42.9 ± 12.3	14.6 ± 8.3	12.0 ± 100.6	-57.8 ± 19.2	-76.2 ± 170.5	60.5 ± 28.8
1.3	18.5 ± 26.6	21.1 ± 10.7	-7.4 ± 26.1	6.6 ± 30.4	-100.9 ± 55.8	-128.0 ± 191.9	33.7 ± 18.1
1.5	-12.5 ± 36.6	11.8 ± 13.0	-14.7 ± 13.8	-131.1 ± 78.9	-78.3 ± 20.2	-95.5 ± 158.8	22.6 ± 19.7
1.8	-30.0 ± 39.6	4.1 ± 17.9	-28.2 ± 13.3	-64.1 ± 104.5	-146.4 ± 130.5	-62.5 ± 129.4	2.5 ± 18.6
2.0	-40.5 ± 51.6	-9.2 ± 15.2	-31.6 ± 16.6	-112.0 ± 72.4	-99.5 ± 14.2	-55.9 ± 37.4	-3.7 ± 9.8

Table 8.40: LunarLander-v3, pomdp, train noise 0.1 — Cvar

Noise _{eval}	Baseline	Noisy PPO	ATLA	Meta LSTM	Meta RNN	Meta SSM	True RW-RL
0.0	-31.4 ± 102.9	3.2 ± 120.1	148.8 ± 139.2	-154.0 ± 181.5	-121.2 ± 64.2	-157.8 ± 50.8	153.6 ± 35.0
0.1	-48.7 ± 115.7	16.3 ± 129.7	156.4 ± 132.5	-249.9 ± 25.9	-57.0 ± 13.5	-253.9 ± 181.1	140.7 ± 31.5
0.2	16.0 ± 65.6	29.6 ± 107.2	165.2 ± 122.1	-198.0 ± 86.6	-90.0 ± 11.8	-127.5 ± 91.5	152.3 ± 47.4
0.4	-55.9 ± 64.6	-5.8 ± 45.5	67.1 ± 76.6	-125.3 ± 109.7	-83.6 ± 64.1	-151.8 ± 52.7	78.2 ± 51.0
0.6	-54.8 ± 42.6	-7.3 ± 30.0	0.1 ± 20.2	-164.7 ± 32.2	-60.6 ± 27.0	-256.2 ± 86.4	56.8 ± 27.1
0.8	-78.6 ± 62.4	-34.5 ± 20.4	-12.2 ± 5.3	-88.4 ± 224.5	-90.2 ± 26.9	-121.0 ± 151.8	10.1 ± 26.2
1.0	-97.3 ± 85.0	-25.9 ± 13.5	-25.4 ± 7.1	-27.9 ± 119.5	-71.5 ± 16.2	-94.2 ± 185.6	-20.2 ± 42.9
1.3	-74.5 ± 31.4	-50.5 ± 18.9	-59.4 ± 49.3	-52.3 ± 23.1	-122.9 ± 74.6	-180.6 ± 170.5	-19.8 ± 23.2
1.5	-97.0 ± 47.0	-58.7 ± 17.2	-62.9 ± 38.1	-165.6 ± 62.7	-98.5 ± 14.0	-116.7 ± 153.8	-33.6 ± 35.8
1.8	-121.1 ± 65.0	-62.6 ± 21.6	-80.2 ± 31.6	-100.2 ± 132.1	-171.5 ± 145.0	-100.3 ± 122.5	-53.9 ± 49.9
2.0	-136.6 ± 80.8	-89.2 ± 44.1	-82.0 ± 26.2	-145.5 ± 92.5	-122.7 ± 25.7	-85.0 ± 39.0	-44.4 ± 14.3