

# Vulnerabilities in Maximum Entropy Inverse Reinforcement Learning under Adversarial Demonstrations

by

Arezoo Alipanah

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Master of Applied Science  
in  
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Arezoo Alipanah 2025

## Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor(s):                   Yash Vardhan Pant  
Assistant Professor, Dept. of ECE, University of Waterloo

Internal Member:               Elliot Creager  
Assistant Professor, Dept. of ECE, University of Waterloo

Internal Member:               Mark Crowley  
Associate Professor, Dept. of ECE, University of Waterloo

### **Author's Declaration**

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

Reinforcement Learning (RL) has emerged as a powerful paradigm for solving complex sequential decision-making problems. However, its effectiveness is fundamentally dependent on the availability of a well-specified reward function, the design of which is often a significant challenge. Inverse Reinforcement Learning (IRL) offers a compelling solution to this problem by enabling an agent to infer an underlying reward function from expert demonstrations. This approach has become a cornerstone of imitation learning, allowing machines to acquire sophisticated behaviors by observing human experts. A critical assumption underpinning most IRL research is that the demonstrators, while potentially suboptimal, are acting in good faith. This thesis challenges that assumption by formally investigating a significant yet underexplored security vulnerability: the susceptibility of IRL algorithms to intentionally malicious demonstrators. We address the scenario where an adversary seeks to corrupt the learning process by strategically injecting a small number of deceptive demonstrations into a training dataset, with the goal of degrading the performance of the final deployed policy.

This research formalizes the problem of adversarial demonstration attacks within the IRL framework. The adversary’s objective is to design a malicious policy that generates trajectories capable of manipulating the inferred reward function. To ensure the attack remains covert, the malicious demonstrations must be statistically similar to the genuine expert demonstrations. We introduce a similarity constraint, based on the expected feature counts of trajectories, that forces the adversarial behavior to remain within a plausible, non-detectable margin of the expert’s behavior. The core of our investigation is to determine whether such a constrained, malicious policy can be systematically designed and to quantify the extent of performance degradation it can induce on a policy learned from the corrupted reward function.

To address this problem, we propose a novel optimization-based framework for generating the adversarial policy. The framework models the adversary’s strategy as a constrained optimization problem over the space of state-action occupancy measures. The objective is to find a policy that minimizes the expected cumulative reward according to the true, ground-truth reward function, thereby maximizing the performance loss of the agent that will learn from it. This minimization is subject to two key sets of constraints: (1) the feature-matching similarity constraint that ensures the deceptive nature of the attack, and (2) the standard Bellman flow constraints that ensure the resulting occupancy measure corresponds to a valid policy under the environment’s dynamics. A time-varying stochastic policy is then extracted from the solution to this optimization problem, providing a concrete method for generating the malicious demonstration trajectories.

The effectiveness of this framework is empirically validated through a series of controlled simulation studies targeting the widely-used Maximum Entropy (MaxEnt) IRL algorithm. Our experiments are conducted in two distinct grid-world environments: ‘Cliff-World’, which represents a safety-critical task with significant negative rewards, and ‘Four Rooms’, a more complex navigation environment with a larger state space. We systematically evaluate the impact of varying the fraction of injected malicious data and the strictness of the similarity constraint. The performance of our proposed adversarial method is benchmarked against both a baseline of expert-only demonstrations and a scenario where random, non-strategic noise is injected into the dataset.

The results of our investigation reveal a significant vulnerability in MaxEnt IRL. We demonstrate that injecting even a small fraction of malicious demonstrations, as little as 10%, can cause a disproportionately severe degradation in the performance of the deployed policy. This performance drop is substantially greater than that caused by injecting an equivalent amount of random noise, confirming the targeted nature of our adversarial generation framework. The conclusions underscore the need for the development of robust defense mechanisms and adversarially-aware IRL algorithms to ensure the safe and reliable deployment of learning agents in real-world, high-stakes applications.

## Acknowledgements

This thesis marks the close of one journey and the opening of another. Along the way, I have been held by people whose kindness and strength became a light in moments when the path grew uncertain.

I am deeply grateful to Professor Yash Pant for his wisdom, patience, and guidance. His mentorship has shaped the way I think about research, and his support during the difficult days gave me the strength to keep going when I could easily have lost my way.

To my family, thank you for being my foundation. Your love, prayers, and quiet courage have been with me at every step, reminding me that I am never alone.

I also hold in my heart the memory of the passengers of Flight PS752, among them Arash and Pouneh, whose lives and dreams were cut short. Their dedication to knowledge and hope for the future continue to inspire me, and I carry their memory within this work.

To my friends and peers, thank you for the conversations, kindness, and moments of light that made this path brighter. This thesis belongs to all of us.

## **Dedication**

To my family, for their endless love and sacrifices, and to my supervisor, for their invaluable guidance and support throughout this journey.

# Table of Contents

Examining Committee	ii
Author’s Declaration	iii
Abstract	iv
Acknowledgements	vi
Dedication	vii
List of Figures	xii
List of Tables	xv
List of Abbreviations	xvi
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and Background . . . . .	1
1.2 Problem Statement: Adversarial Demonstrations in Inverse Reinforcement Learning (IRL) . . . . .	2
1.3 Research Approach and Contributions . . . . .	3
1.4 Outline of the Thesis . . . . .	4

<b>2</b>	<b>Literature Review</b>	<b>6</b>
2.1	Foundations of Learning from Demonstration (LfD)	6
2.2	Core Methodologies in IRL	7
2.2.1	Early Formulations and the Ill-Posed Problem	7
2.2.2	The Maximum Entropy Framework	8
2.3	Advances in Scalable Imitation Learning (IL)	9
2.3.1	Generative Adversarial Imitation Learning (GAIL)	9
2.3.2	Adversarial Inverse Reinforcement Learning (AIRL)	10
2.4	Types of Imperfect Demonstrations	10
2.4.1	Learning from Sub-optimal and Noisy Experts	10
2.4.2	Strategic and Deceptive Demonstrators	11
2.4.3	Malicious Intent: Data Poisoning in IRL	11
2.5	Paradigms of Robustness in Learning from Demonstrations	12
2.5.1	Paradigms of Robust Reinforcement Learning (RL)	12
2.5.2	Robustness Against Test-Time Perturbations in IRL	13
2.6	Synthesis and Identification of the Research Gap	13
<b>3</b>	<b>Preliminaries and Problem Statement</b>	<b>15</b>
3.1	Sequential Decision-Making	15
3.1.1	The MDP Framework	15
3.1.2	Value Functions and Optimal Policies	17
3.2	Solving Markov Decision Processes	18
3.2.1	Dynamic Programming	18
3.2.2	Linear Programming	19
3.3	Inverse Reinforcement Learning	20
3.3.1	IRL Problem Formulation	20
3.3.2	Ambiguity in Reward Inference: The Ill-Posed Nature of IRL	21
3.4	Maximum Entropy IRL	22

3.4.1	The Maximum Entropy Principle . . . . .	22
3.4.2	The MaxEnt IRL Framework . . . . .	23
3.5	Problem Statement . . . . .	24
<b>4</b>	<b>Methodology</b>	<b>29</b>
4.1	Threat Model and Methodological Assumptions . . . . .	29
4.2	Modeling the Expert and the Adversary . . . . .	30
4.2.1	Expert Model: Soft Value Iteration . . . . .	30
4.2.2	Adversarial Model: Constrained Value Minimization . . . . .	31
4.3	An Optimization Framework for Adversarial Attacks . . . . .	32
4.3.1	Occupancy Measures for Policy Representation . . . . .	32
4.3.2	The Adversarial Optimization Problem . . . . .	33
4.3.3	Policy Extraction from Occupancy Measures . . . . .	33
4.3.4	Algorithmic Implementation . . . . .	34
4.3.5	Baseline for Comparison: The Constrained Randomized Policy . . . . .	34
4.4	Justification of the Adversarial Objective . . . . .	35
4.5	Chapter Summary . . . . .	37
<b>5</b>	<b>Simulation Studies</b>	<b>38</b>
5.1	Simulation Setup . . . . .	39
5.1.1	Trajectory Generation and Mixing . . . . .	39
5.1.2	Reward Inference and Policy Training . . . . .	40
5.1.3	Simulation Environments . . . . .	41
5.2	Case Study 1: CliffWorld . . . . .	43
5.2.1	Impact of Malicious Trajectories . . . . .	43
5.2.2	Sensitivity to the Similarity Constraint . . . . .	44
5.2.3	Summary of Findings in CliffWorld . . . . .	44
5.3	Case Study 2: Four Rooms . . . . .	47

5.3.1	Analysis of Adversarial Impact . . . . .	47
5.3.2	Parameter Sensitivity in a Complex Environment . . . . .	48
5.3.3	Summary of Findings in Four Rooms . . . . .	48
5.4	Comparative Analysis and Discussion . . . . .	51
5.4.1	Synthesis of Environmental Effects . . . . .	51
5.4.2	The Disparate Impact of Malicious and Random Data . . . . .	52
5.4.3	Chapter Summary and Key Findings . . . . .	52
<b>6</b>	<b>Conclusion and Future Work</b>	<b>54</b>
6.1	Summary of Contributions . . . . .	54
6.2	Limitations of the Current Study . . . . .	55
6.2.1	Assumption of a Fully-Informed Adversary . . . . .	55
6.2.2	Fixed Planning Horizon . . . . .	55
6.2.3	Computational Complexity and Scalability . . . . .	56
6.2.4	Focus on the Maximum Entropy IRL Framework . . . . .	56
6.2.5	Discrete State and Action Spaces . . . . .	56
6.3	Extension: Unsupervised Detection of Adversarial Trajectories . . . . .	57
6.3.1	Learning Trajectory Embeddings with a VAE . . . . .	57
6.3.2	Visualization and Cluster Analysis . . . . .	57
6.3.3	Preliminary Findings and Discussion . . . . .	58
6.4	Future Research Directions . . . . .	58
6.4.1	Developing Inherently Robust IRL Algorithms . . . . .	58
6.4.2	Improving Unsupervised Detection Methods . . . . .	59
6.4.3	Extension to Multi-Agent Scenarios . . . . .	59
6.5	Concluding Remarks . . . . .	59
	<b>References</b>	<b>66</b>
	<b>Glossary</b>	<b>74</b>

# List of Figures

1.1	An adversary contributing a fraction $\delta$ of the demonstrations influences the reward function inferred by MaxEnt IRL, thereby affecting the learned policy $\pi_{\text{deployed}}$ . . . . .	3
2.1	Different reward functions can induce the same optimal policy. Each grid shows a different assignment of rewards to states (lighter squares represent higher rewards). Despite these differences, the agent’s optimal behavior, following the arrows to reach the rewarding states, remains the same. This illustrates the fundamental ambiguity in IRL: multiple reward functions may explain the same observed behavior [48]. . . . .	8
5.1	A visualization of the two grid-world environments used in our simulation studies. The goal state (G) is shown in green, the hazardous state (B) is in red, and the agent’s starting state (S) is in blue. These environments provide structured testbeds for safety and exploration challenges. . . . .	41
5.2	Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in CliffWorld as the mixing ratio ( $\delta$ ) varies for different fixed similarity bounds ( $\epsilon$ ). Each plot shows the average return of the policy trained on the reward inferred from a dataset contaminated with either malicious or random trajectories. Results are compared against expert-only data ( $\delta = 0$ ). Curves are averaged over 30 seeds, with shaded regions indicating standard deviation. . . . .	45
5.3	Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in CliffWorld as the similarity bound ( $\epsilon$ ) varies for different fixed mixing ratios ( $\delta$ ). Each plot shows the average return of the policy trained on the inferred reward. Curves are averaged over 30 seeds, with shaded regions indicating standard deviation. . . . .	46

5.4	Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in Four Rooms as the mixing ratio ( $\delta$ ) varies for different fixed similarity bounds ( $\epsilon$ ). Each plot shows the average return of the policy trained on the reward inferred from a dataset contaminated with either malicious or random trajectories. Results are compared against expert-only data ( $\delta = 0$ ). Curves are averaged over 24 seeds, with shaded regions indicating standard deviation. . . . .	49
5.5	Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in Four Rooms as the similarity bound ( $\epsilon$ ) varies for different fixed mixing ratios ( $\delta$ ). Each plot shows the average return of the policy trained on the inferred reward. Results for malicious data are compared against a random data baseline and expert-only data. Curves are averaged over 24 seeds, with shaded regions indicating standard deviation. . . . .	50
5.6	Side-by-side comparison of heatmaps showing the average return of the deployed policy as a function of the mixing ratio $\delta$ (x-axis) and the similarity bound $\epsilon$ (y-axis). The color scale represents policy performance, with yellow indicating high return and dark blue indicating low return. CliffWorld (a) exhibits substantial performance drops over a wide parameter range, while Four Rooms (b) remains robust except at the highest $\delta$ and $\epsilon$ values. . . . .	52
6.1	VAE training loss curves for both environments, showing the total loss, reconstruction loss, and KL divergence over 1000 epochs(Four Rooms) and 100 epochs (CliffWorld). The stable convergence indicates that the model successfully learned to compress and reconstruct the trajectories. . . . .	61
6.2	PCA Visualization of trajectory embeddings for the <i>Four Rooms</i> environment. Expert trajectories are shown in red, while adversarial trajectories are shown in shades of blue, with darker shades corresponding to a higher attack budget $\epsilon$ . . . . .	62
6.3	t-SNE Visualization of trajectory embeddings for the <i>Four Rooms</i> environment. Expert trajectories are shown in red, while adversarial trajectories are shown in shades of blue, with darker shades corresponding to a higher attack budget $\epsilon$ . . . . .	63
6.4	PCA Visualization of trajectory embeddings for the <i>CliffWorld</i> environment. Separation between expert (red) and adversarial (blue) trajectories is visible but less distinct than in <i>Four Rooms</i> , highlighting the increased difficulty of detection in this safety-critical setting. . . . .	64

6.5 t-SNE Visualization of trajectory embeddings for the *CliffWorld* environment. Separation between expert (red) and adversarial (blue) trajectories is visible but less distinct than in *Four Rooms*, highlighting the increased difficulty of detection in this safety-critical setting. . . . . 65

# List of Tables

3.1	Core MDP and Reinforcement Learning Notation. . . . .	16
3.1	Core MDP and Reinforcement Learning Notation (continued). . . . .	17
3.2	A Comparison of IRL Paradigms: Max-Margin vs. Maximum Entropy. . .	28

# List of Abbreviations

**AIRL** Adversarial Inverse Reinforcement Learning [ix](#), [10](#), [14](#)

**BC** Behavioral Cloning [7](#)

**GAIL** Generative Adversarial Imitation Learning [ix](#), [9](#), [10](#), [14](#), [56](#)

**IL** Imitation Learning [ix](#), [1](#), [2](#), [9](#), [10](#), [14](#)

**IRL** Inverse Reinforcement Learning [viii–xii](#), [xv](#), [1–4](#), [6–15](#), [20–31](#), [34](#), [35](#), [37–40](#), [43](#), [44](#), [47](#), [48](#), [51–59](#)

**LfD** Learning from Demonstration [ix](#), [6](#), [7](#), [10](#)

**RL** Reinforcement Learning [ix](#), [1](#), [2](#), [4](#), [6](#), [7](#), [9](#), [12–15](#), [38](#), [40](#), [41](#)

# Chapter 1

## Introduction

This chapter establishes the context and motivation for the research presented in this thesis. We begin with an overview of [Reinforcement Learning \(RL\)](#) and discuss the inherent challenges of manual reward function design. This discussion provides the motivation for learning-from-demonstration methods, specifically [Imitation Learning \(IL\)](#) and [Inverse Reinforcement Learning \(IRL\)](#), which use expert behavior to learn policies. We then introduce the problem of adversarial demonstrations in [IRL](#), where a fraction of the demonstration data is intentionally misleading. Such misleading data can severely degrade the performance of standard [IRL](#) algorithms, leading to suboptimal or even unsafe learned behaviors. The focus of this thesis is to formalize this adversarial threat and demonstrate the extent to which standard [IRL](#) methods fail in its presence. Finally, we state our core research questions, summarize our contributions, and provide an outline of the thesis.

### 1.1 Motivation and Background

[RL](#) provides a mathematical framework for sequential decision-making, where an agent learns to maximize a cumulative reward signal through interaction with an environment [71]. [RL](#) has led to significant advances in diverse fields, including robotics and game playing [55, 45].

A primary challenge in applied [RL](#) is the design of the reward function. A reward function that is misspecified or incomplete can lead to unintended or unsafe agent behaviors, as manually encoding all desired objectives and constraints is often difficult and prone to error [36]. An agent optimizing a flawed reward function may discover undesirable “loopholes” that satisfy the literal specification of the reward while violating the designer’s intent.

To address this challenge, researchers have developed methods for learning from demonstrations [67, 6]. These methods learn from examples of desired behavior provided by an expert. In **IL** [67], an agent learns a policy by directly mimicking the expert’s actions. In **IRL**, the objective is to recover the underlying reward function that the expert is presumed to be optimizing [57, 79]. Once this reward function is inferred, it can be used with standard **RL** algorithms to train a policy [79]. In many complex domains, collecting expert demonstrations is more practical than engineering a correct reward function from scratch, making **IL** and **IRL** powerful alternatives [6].

A foundational assumption for both **IL** and **IRL** is that the provided demonstrations are a reliable reflection of the expert’s goal [57]. Consequently, most research assumes the demonstrations are genuine and near-optimal. While some studies have started to address the challenge of learning from suboptimal demonstrations [13], the possibility of intentionally malicious demonstration data has received far less attention. This gap motivates our investigation into the vulnerabilities of **IRL** when faced with adversarially crafted demonstrations, which we formally introduce in the following section.

## 1.2 Problem Statement: Adversarial Demonstrations in **IRL**

In many practical applications, demonstration data may be collected from unvetted or open sources, introducing a security risk. This possibility raises a critical question: what if a malicious actor provides misleading trajectories designed to corrupt the learning process? We define these as *adversarial demonstrations*: trajectories that are crafted to appear expert-like but are intended to bias an **IRL** algorithm toward an incorrect or harmful reward function.

Formally, consider an **IRL** agent that receives a dataset of demonstrations. Most of these demonstrations come from a genuine expert, but some are contributed by a malicious demonstrator. The agent, unaware of this data poisoning, infers a reward function from the mixed dataset and then deploys a policy trained on it.

The central question of this thesis is:

Can an adversary, by injecting even a limited portion of malicious demonstrations, cause the learned reward function and resulting policy to deviate substantially from the expert’s intended behavior?

A successful attack must strike a balance between two possibly competing considerations. On the one hand, the adversarial trajectories need to resemble the expert’s behavior closely enough to avoid easy detection. On the other hand, they must differ sufficiently to steer the inferred reward toward the attacker’s intended objective. These two forces may align or conflict depending on how much dissimilarity the learning algorithm tolerates before detection becomes likely. Subject to assumptions about the visibility of demonstrations and the ability of the adversary to contribute new trajectories, our investigation focuses on how the vulnerability of an IRL algorithm depends on the allowable degree of deviation between adversarial and expert demonstrations. To make this setup concrete, we include an introductory schematic figure (adapted from our work in [3]) that illustrates the relationship between expert, adversarial, and mixed demonstration data, as well as the downstream effect on the inferred reward and learned policy.

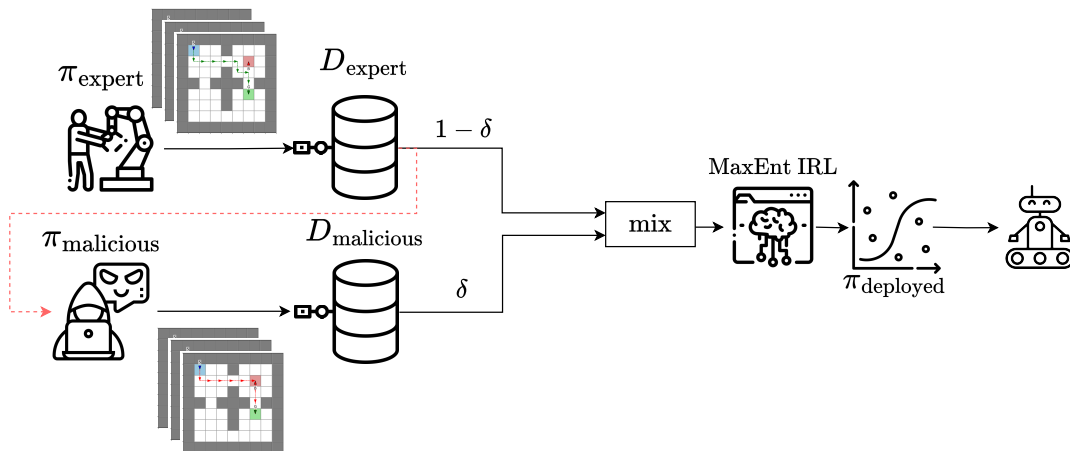


Figure 1.1: An adversary contributing a fraction  $\delta$  of the demonstrations influences the reward function inferred by MaxEnt IRL, thereby affecting the learned policy  $\pi_{\text{deployed}}$ .

### 1.3 Research Approach and Contributions

To investigate the central question posed above, we develop an optimization-based framework for generating adversarial demonstrations, with a focus on classical maximum entropy IRL. Instead of relying on heuristics such as hand-crafted perturbations of expert trajectories [39], we formalize the adversary’s task as an optimization problem that maximizes the

impact on the learner, subject to a constraint that ensures the attack remains sufficiently similar to the expert’s behavior. Our contributions are as follows:

- (i) We formalize the generation of adversarial demonstrations as an optimization problem tailored to the Maximum Entropy (MaxEnt) [IRL](#) algorithm, a widely used and representative [IRL](#) method [79]. We prove (Section 4.4) that, under a similarity constraint, the adversary’s problem of maximally degrading the learned policy’s performance is mathematically equivalent to minimizing the true value of the adversarial policy itself. This establishes our formulation as the appropriate worst-case objective for adversarial demonstration generation against MaxEnt [IRL](#).
- (ii) We analyze how the attack’s effectiveness varies with the similarity constraint, demonstrating that the adversary’s impact is directly related to the behavioral deviation it is permitted.
- (iii) We compare our optimization-based attack against a baseline that injects random noise into expert trajectories, showing that a strategic attack is significantly more effective.
- (iv) We conduct experiments in benchmark environments (CliffWorld and FourRooms) and show that altering only 10% of the demonstrations with optimized adversarial trajectories can mislead the [IRL](#) algorithm into learning a faulty reward function, degrading the resulting policy’s performance by up to three times more than if the same fraction of random trajectories were added.
- (v) We conduct a preliminary investigation into the detection of these attacks, exploring an unsupervised learning approach to identify adversarial trajectories as anomalies.

## 1.4 Outline of the Thesis

The remainder of this thesis is organized as follows:

- **Chapter 2** reviews the relevant literature on [RL](#), [IRL](#), and learning from demonstrations, highlighting both foundational approaches and recent advances.
- **Chapter 3** introduces the necessary preliminaries, including sequential decision-making, [MDPs](#), and the MaxEnt [IRL](#) framework. This chapter also formalizes the problem of generating malicious demonstrations within [IRL](#), which serves as the foundation for our subsequent analysis.

- **Chapter 4** details our methodology, presenting the optimization-based framework for constructing adversarial demonstrations, along with the associated modeling assumptions and algorithmic implementation.
- **Chapter 5** reports the results of our simulation studies in the CliffWorld and Four-Rooms environments, analyzing the effects of adversarial trajectories on reward inference and learned policies.
- **Chapter 6** concludes the thesis with a summary of contributions, discussion of limitations, and directions for future research, including preliminary work on unsupervised detection of adversarial trajectories.

# Chapter 2

## Literature Review

[Inverse Reinforcement Learning \(IRL\)](#) is a key technique for inferring the goals that motivate the observed behavior of an expert or agent [57, 6]. Understanding the intent and underlying objectives of expert demonstrations is crucial as intelligent systems become more common in dynamic, human-centric environments. This chapter provides a survey of [IRL](#). We review major theoretical milestones and persistent challenges. We conclude by identifying unresolved issues in the security and robustness of [IRL](#), which motivate the core contributions of this thesis.

### 2.1 Foundations of [Learning from Demonstration \(LfD\)](#)

A central problem in [Reinforcement Learning \(RL\)](#) is specifying a reward function that elicits the desired behavior without creating unforeseen loopholes [28]. This process, known as reward design, is notoriously difficult. An agent given a seemingly simple goal, such as “clean the room,” might learn to hide objects instead of organizing them, a behavior known as *reward hacking* [69, 5].

[LfD](#) offers an alternative to manual reward specification, meaning that instead of hand-crafting a reward signal, the desired behavior is communicated through demonstrations provided by an expert. The agent then attempts to reproduce these demonstrations, effectively replacing the need for an explicitly defined reward function. By observing and mimicking an expert, an [RL](#) agent can learn complex skills in domains where formalizing an objective is impractical, such as robotics or autonomous driving [67, 6]. Two primary strategies dominate the [LfD](#) landscape.

**Behavioral Cloning (BC)** frames LfD as a supervised learning problem. The agent learns a mapping from states to actions based on a dataset of expert state-action pairs [75]. While simple and effective, BC is susceptible to **Covariate Shift** [65]. Small prediction errors can lead the agent into states not seen during training, causing further errors to accumulate and leading to catastrophic failures in performance [65].

**IRL** seeks to overcome this limitation by recovering the reward function that explains the expert’s actions [57]. Instead of simply copying what the expert does, IRL infers *why* the expert does it. With the learned reward function, an agent can use standard RL algorithms to compute an optimal policy. This allows the agent to generalize to new situations and potentially even surpass the expert’s performance, making IRL a powerful approach [28]. However, it can also be computationally expensive because the reward inference typically requires repeatedly solving RL problems during training [79, 1, 6].

## 2.2 Core Methodologies in IRL

The central task in IRL is to infer a reward function from observed behavior. This is challenging because many different reward functions can explain the same set of demonstrations, making the problem underdetermined [57, 1]. The field has progressed by developing methods to resolve this ambiguity, often by making specific assumptions about the expert’s optimality or cognitive processes [63].

### 2.2.1 Early Formulations and the Ill-Posed Problem

The foundational challenge in IRL is that the problem is fundamentally **ill-posed** [57, 6]. For any policy, an infinite number of reward functions can make that policy optimal. For instance, a reward function that is zero everywhere renders all policies optimal but offers no information for learning a task [67]. This ambiguity means that simply finding one reward function that explains the demonstrations is not enough; a principle is needed to select a meaningful one.

Ng and Russell [57] offered the first formal solution by framing IRL as an optimization problem based on the **max-margin principle**. Their method seeks a reward function that makes the expert’s policy not merely optimal, but better than all other policies by the largest possible margin. This principle, analogous to Support Vector Machines, provides a way to resolve the ambiguity. However, this framework relies on the assumption that the expert demonstrator is perfectly optimal. In practice, real-world demonstrations contain

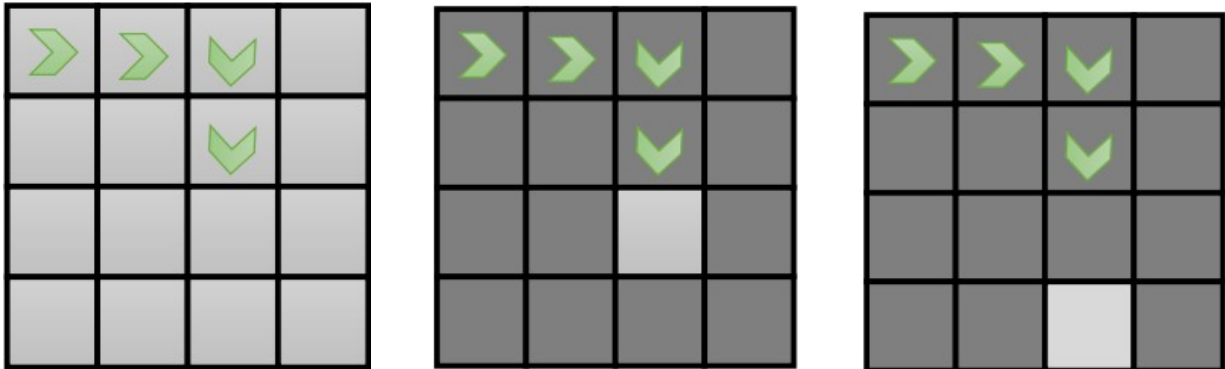


Figure 2.1: Different reward functions can induce the same optimal policy. Each grid shows a different assignment of rewards to states (lighter squares represent higher rewards). Despite these differences, the agent’s optimal behavior, following the arrows to reach the rewarding states, remains the same. This illustrates the fundamental ambiguity in IRL: multiple reward functions may explain the same observed behavior [48].

noise and suboptimal actions, which can make the optimization problem infeasible and cause the method to fail [79].

## 2.2.2 The Maximum Entropy Framework

To address the fragility of the strict optimality assumption, Ziebart et al. [79] introduced an approach based on the **principle of maximum entropy**. This framework models expert behavior probabilistically instead of deterministically. It assumes the likelihood of a demonstrated trajectory is exponentially proportional to its total reward, yielding a distribution analogous to the Boltzmann distribution in statistical mechanics [24]. While in principle one could consider many alternative probabilistic models, the exponential family distribution is uniquely appealing: it is the solution that maximizes entropy subject to feature-matching constraints, it is simple to normalize, and it remains differentiable with respect to the reward parameters. These properties make it mathematically natural and computationally convenient for inverse reinforcement learning. The key implication is that the agent’s behavior distribution is as unbiased as possible beyond the imposed constraints, avoiding arbitrary assumptions about unobserved preferences.

This probabilistic perspective allows MaxEnt IRL to handle noise and suboptimality gracefully. Suboptimal behaviors are not contradictions but are simply interpreted as

less likely outcomes. The framework resolves the ill-posed nature of [IRL](#) by selecting the probability distribution over trajectories that is maximally non-committal, or has the highest entropy, while still matching the expected feature counts observed in the expert’s demonstrations [79]. This approach provides a robust mathematical foundation that has become central to modern [Imitation Learning \(IL\)](#).

A primary computational bottleneck of MaxEnt [IRL](#) is the calculation of the partition function, which involves summing over all possible trajectories. This sum is intractable in large state spaces [79]. Despite this, the MaxEnt framework laid the groundwork for scalable, model-free techniques like [Generative Adversarial Imitation Learning \(GAIL\)](#), which approximate this process through adversarial training [37, 28].

## 2.3 Advances in Scalable [IL](#)

While the Maximum Entropy framework provided a principled way to handle suboptimal demonstrations, its computational demands limited its application. A typical MaxEnt [IRL](#) algorithm requires solving an [RL](#) problem in its inner loop, which is a resource-intensive task [6]. This bottleneck hindered its use in high-dimensional and continuous environments. The development of adversarial [IL](#) methods, inspired by connections to Generative Adversarial Networks (GANs), offered a breakthrough in scalability by bypassing the explicit reward learning step.

### 2.3.1 [GAIL](#)

Ho and Ermon [37] proposed [GAIL](#), which reframes [IL](#) as a game between two neural networks. A **generator** (the policy,  $\pi$ ) tries to produce trajectories that are indistinguishable from expert demonstrations, while a **discriminator** ( $D$ ) tries to tell them apart. This adversarial process forces the policy to match the expert’s behavior distribution.

The key insight is that this adversarial game is equivalent to minimizing the divergence between the state-action distributions (occupancy measures) of the policy and the expert [37]. The policy is updated using a policy gradient algorithm where the reward signal is provided directly by the discriminator. This model-free approach avoids the nested optimization loop of traditional [IRL](#), enabling significant performance gains in complex, high-dimensional tasks [37].

However, this scalability comes at a price. The discriminator in [GAIL](#) does not learn a meaningful or transferable reward function. At convergence, an optimal discriminator

outputs a constant value of 0.5 everywhere, providing no information about the expert’s underlying preferences [28]. **GAIL** learns to mimic behavior effectively, but it loses the core benefit of **IRL**: recovering a robust and generalizable reward function.

### 2.3.2 Adversarial Inverse Reinforcement Learning (**AIRL**)

To recover the reward function while maintaining the benefits of the adversarial approach, Fu et al. [28] developed **AIRL**. **AIRL** modifies the **GAIL** framework to ensure the discriminator learns a valid reward function.

The core problem **AIRL** solves is that the discriminator’s output in standard adversarial imitation is entangled with the environment’s dynamics. A reward that encourages visiting states that are easy to reach will not generalize to new environments with different dynamics [28]. **AIRL** addresses this by designing a special discriminator structure that disentangles a state-only reward function from the transition dynamics.

The progression from MaxEnt **IRL** to **GAIL** and then to **AIRL** represents a key advancement in **IL**. MaxEnt **IRL** provided a principled but computationally heavy method for reward inference. **GAIL** achieved scalability by forgoing explicit reward recovery. **AIRL** successfully combined the scalability of adversarial training with the original goal of **IRL**: learning a robust and transferable reward function [28, 6]. However, like all **LfD** methods, **AIRL**’s success depends on the assumption that the expert is competent and cooperative. This assumption breaks down when demonstrations are intentionally misleading.

## 2.4 Types of Imperfect Demonstrations

Most **IRL** methods operate under the assumption that expert demonstrations are optimal, or at worst, contain benign noise. In reality, demonstrations can suffer from various imperfections, ranging from unintentional mistakes to deliberate, malicious manipulation. While some forms of imperfection have been studied, the threat of an actively adversarial demonstrator remains a significant and underexplored problem in **IRL**.

### 2.4.1 Learning from Sub-optimal and Noisy Experts

A well-studied problem in **IRL** is learning from demonstrations that are flawed but not malicious. These imperfections can stem from sensor noise, environmental randomness, or human limitations like fatigue and lapses in attention [75, 13].

A substantial body of research aims to make **IRL** robust to these unintentional errors. Probabilistic frameworks like MaxEnt **IRL** inherently tolerate some suboptimality [79]. More recent methods explicitly model and learn from demonstrators with varying skill levels, which is useful when dealing with crowdsourced or heterogeneous data [18, 7]. These techniques often infer both the reward function and the expertise level of each demonstrator simultaneously [15]. Other approaches focus on filtering or re-weighting demonstrations to reduce the influence of low-quality examples [16]. The common theme in this research is reward signal recovery: the goal is to extract the expert’s true intent from a noisy dataset, assuming the demonstrator is fundamentally cooperative [51, 70].

### 2.4.2 Strategic and Deceptive Demonstrators

A more difficult challenge arises when imperfections in demonstrations are intentional. In this scenario, the demonstrator is a strategic agent with its own objectives, which may not align with the learner’s. This shifts the problem from statistical inference to game-theoretic reasoning [26].

One line of work addresses **strategic manipulation**. In the Multi-Principal Assistance Games framework, a learning agent interacts with multiple demonstrators (principals), each with a private reward function [26]. A rational principal may provide misleading demonstrations to bias the agent’s inferred reward function toward outcomes that serve the principal’s own interests. The intent is not necessarily to degrade the agent’s performance, but rather to strategically exploit the learning process for individual advantage.

A different form of strategic behavior involves active deception, where an expert is aware of being observed. This can be for **obfuscation**, to hide true intent, or for **projection**, to create a false belief in the observer. In Inverse-Inverse Reinforcement Learning (I-IRL), an expert (e.g., a cognitive radar) performs suboptimal actions to maximize the observer’s uncertainty about its true reward function, thus protecting sensitive information [60]. In contrast, research in Inverse Inverse Planning (IIP) explores how a demonstrator can manipulate its actions to project a specific, often false, objective. For example, an actor might carefully select actions to lead an audience to a desired conclusion [20, 21].

### 2.4.3 Malicious Intent: Data Poisoning in IRL

The most severe form of intentional imperfection is malicious sabotage. This threat is best understood as a **Data Poisoning attack**, a well-studied problem in adversarial machine learning [12, 11]. In a poisoning attack, an adversary injects carefully crafted

malicious samples into the training data to corrupt the learned model. These attacks can be **indiscriminate**, aiming to degrade the model’s overall performance, or **targeted**, causing the model to fail in specific, attacker-chosen situations while otherwise appearing normal [11, 33].

In the context of **IRL**, a malicious demonstrator acts as a data poisoner by providing harmful trajectories. The goal is no longer personal gain or secrecy, but the direct sabotage of the learned policy. For example, a targeted attack could train an autonomous vehicle to drive perfectly under most conditions but execute a dangerous maneuver when it encounters a specific, rare road sign.

Another related threat is state adversarial perturbation, where an adversary manipulates the states observed by the learner during training [17]. This creates a mismatch between the states used for reward inference and the true environment, leading to a non-robust policy. While the field has developed methods for handling noisy demonstrators, defending **IRL** algorithms against dedicated **Data Poisoning** and state perturbation attacks remains a critical, underexplored challenge.

## 2.5 Paradigms of Robustness in Learning from Demonstrations

The challenge of building robust learning agents is fundamentally about identifying where the adversary can inject data into the model. Most research in both Robust **RL** and Robust **IRL** assumes the adversary exists in the test environment. This external adversary attacks during deployment by perturbing the agent’s sensors, disrupting its actions, or altering the world’s dynamics after a policy has been learned.

This thesis, however, addresses an adversary who operates much earlier and from a more privileged position: an adversary disguised as a helpful expert who corrupts the training data itself. We investigate the threat of **Data Poisoning**, where a malicious demonstrator provides crafted examples to sabotage the learned reward function from within. We argue that defending against this internal adversary is a distinct and largely overlooked problem.

### 2.5.1 Paradigms of Robust **RL**

Robust **RL** aims to produce policies that perform well under worst-case conditions. The problem is typically framed as a zero-sum game between a learning agent and an adversary

who perturbs the environment or learning process. The specific form of these perturbations gives rise to different formulations of Robust RL [61].

- **Transition Robustness:** The adversary alters the environment’s transition dynamics,  $P(s'|s, a)$ , within a specified uncertainty set. The agent must learn a policy that is robust to the worst-case dynamics [61].
- **Disturbance Robustness:** Inspired by robust control, this paradigm models uncertainty as external forces that an adversary applies to the agent’s state transitions. The agent must learn a control policy that can withstand these destabilizing forces [61].
- **Action Robustness:** The adversary perturbs the agent’s actions before they are executed. This simulates effects like actuator noise or physical resistance [74].
- **Observation Robustness:** The adversary perturbs the agent’s state observations,  $s$ , without changing the underlying state of the world. This is especially relevant for agents that rely on high-dimensional sensory inputs like images [39].

## 2.5.2 Robustness Against Test-Time Perturbations in IRL

Recent work has started to apply robustness principles to IRL, primarily focusing on perturbations during policy execution. For example, the State-Adversarial Max-Margin IRL (SMM-IRL) algorithm learns a reward function in a State-Adversarial Markov Decision Process (SA-MDP), where an adversary can manipulate the agent’s state observations at *test time* [17, 77].

SMM-IRL is designed to defend against **test-time attacks**, where the environment itself is compromised during deployment. It assumes the expert demonstrations used for training are clean and reliable. This threat model is fundamentally different from **Data Poisoning**, which is a **training-time attack**. In a poisoning scenario, the adversary is the demonstrator who provides corrupted data, but the test environment is assumed to be clean. Therefore, existing robust IRL frameworks do not address the vulnerability of the learning algorithm to a malicious expert.

## 2.6 Synthesis and Identification of the Research Gap

This chapter has outlined the development of IRL, from early max-margin methods that assumed a perfect expert [57] to the probabilistic MaxEnt framework that could gracefully

handle noisy and suboptimal data [79]. This shift to a statistical inference perspective was critical, establishing a foundation that not only became a cornerstone of modern IRL but also enabled the development of scalable adversarial methods like GAIL and AIRL [37, 28].

At the same time, the field’s conception of the “expert” has matured. The naive model of a flawless oracle has been replaced by a more realistic spectrum of demonstrators. This includes methods for learning from well-intentioned but flawed experts [7, 18] and, more recently, from strategic agents who may be self-interested or deceptive [26, 60].

Despite this progress, a critical vulnerability persists. The majority of work on robustness, in both RL and IRL, focuses on defending against **test-time attacks** where an external adversary perturbs the agent during deployment [61, 17]. However, the threat of a **training-time attack**, where the adversary is the demonstrator who poisons the training data, has been largely unaddressed.

This reveals a crucial research gap: **the vulnerability of foundational IRL algorithms to training-time Data Poisoning attacks**. While modern IL has scaled to complex problems, these advances often inherit the principles, and thus the potential vulnerabilities, of the MaxEnt framework. The susceptibility of this core algorithm to a malicious demonstrator who intentionally provides corrupted data to sabotage the learned policy has not been systematically investigated. The precise nature of this fragility, and the methods to defend against it, remain open and pressing questions.

This thesis addresses this gap by investigating the following questions: How can an attacker optimally craft malicious trajectories to poison the MaxEnt IRL objective? What is the theoretical and empirical fragility of this algorithm to such attacks? And what defense mechanisms can be developed to ensure the integrity of the learning process in untrusted environments? Answering these questions is essential for the safe and reliable deployment of agents that learn from demonstration.

# Chapter 3

## Preliminaries and Problem Statement

This chapter builds the theoretical foundation for our analysis of vulnerabilities in [Inverse Reinforcement Learning \(IRL\)](#). We begin with the core framework of sequential decision-making, the Markov Decision Process (MDP). We then introduce [IRL](#) as the problem of inferring objectives from observed behavior and highlight its inherent ambiguities. As a solution, we provide a detailed look at Maximum Entropy [IRL](#), a principled probabilistic approach that is central to this thesis. Finally, we establish the security context by formally stating the core research problem of this work.

Readers already familiar with [Reinforcement Learning \(RL\)](#) and [IRL](#) may choose to skip directly to [Section 3.5](#), where the problem statement is introduced.

### 3.1 Sequential Decision-Making

Sequential decision-making is mathematically formalized by the Markov Decision Process (MDP), which provides the fundamental language for reinforcement learning [[71](#), [57](#)].

#### 3.1.1 The MDP Framework

An MDP is a mathematical framework for modeling decision-making under uncertainty, defined by a tuple  $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma, T)$ . [[62](#), [9](#)].

- **State Space ( $\mathcal{S}$ ):** A finite set of states describing the environment.
- **Action Space ( $\mathcal{A}$ ):** A finite set of actions available to the agent.
- **Transition Function ( $P$ ):**  $P(s' | s, a)$  is the probability of transitioning to state  $s'$  from state  $s$  after taking action  $a$ . It defines the environment’s dynamics and assumes the Markov Property, meaning the next state depends only on the current state and action [71].
- **Reward Function ( $R$ ):**  $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  specifies the immediate reward for taking an action in a state and implicitly defines the agent’s goal.
- **Discount Factor ( $\gamma$ ):**  $\gamma \in [0, 1]$  discounts future rewards, balancing immediate and long-term gains.
- **Horizon ( $T$ ):** Specifies the number of decision steps.

Note that this work assumes an undiscounted finite-horizon MDP. In finite-horizon MDPs, the horizon length  $T$  implicitly plays a similar role by bounding the sum of rewards, even when  $\gamma = 1$ .

The agent’s behavior is defined by a **policy**,  $\pi(a|s)$ , which is a probability distribution over actions for a given state. The agent’s goal is to learn a policy that maximizes the **return**,  $G_t$ , which is the discounted sum of future rewards:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.1}$$

The core notation used throughout this thesis is summarized in Table 3.1.

Table 3.1: Core MDP and Reinforcement Learning Notation.

Symbol	Description
$\mathcal{S}$	A finite set of states (state space).
$s, s'$	Specific states, where $s, s' \in \mathcal{S}$ .
$\mathcal{A}$	A finite set of actions (action space).
$a$	A specific action, where $a \in \mathcal{A}$ .
$P(s'   s, a)$	Probability of transitioning to state $s'$ from state $s$ after taking action $a$ .

Table 3.1: Core MDP and Reinforcement Learning Notation (continued).

Symbol	Description
$R(s, a)$	Immediate reward for taking action $a$ in state $s$ .
$\gamma$	Discount factor, where $\gamma \in [0, 1)$ .
$\pi(a   s)$	Policy: the probability of taking action $a$ in state $s$ .
$\tau$	A trajectory (or episode): a sequence of state-action pairs $(s_0, a_0, s_1, a_1, \dots)$ .
$G_t$	Return: the discounted sum of rewards from time step $t$ .
$V^\pi(s)$	State-value function: the expected return starting from state $s$ and following policy $\pi$ .
$Q^\pi(s, a)$	Action-value function: the expected return after taking action $a$ in state $s$ and then following policy $\pi$ .
$V^*(s), Q^*(s, a)$	The optimal state-value and action-value functions, respectively.
$\pi^*$	An optimal policy that maximizes the expected return.

### 3.1.2 Value Functions and Optimal Policies

To evaluate a policy, we use **value functions** to estimate the expected return. The state-value function,  $V^\pi(s)$ , is the expected return from starting in state  $s$ , while the action-value function,  $Q^\pi(s, a)$ , is the expected return from taking action  $a$  in state  $s$  and then following policy  $\pi$ .

These functions adhere to a recursive relationship defined by the **Bellman equations** [9]. The ultimate goal is to find an **optimal policy**,  $\pi^*$ , which achieves the highest possible expected return from any state [71]. An optimal policy yields the optimal value functions,  $V^*$  and  $Q^*$ , which uniquely satisfy the **Bellman optimality equations**:

$$V^*(s) = \max_{a \in \mathcal{A}} \left( R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^*(s') \right) \quad (3.2)$$

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \max_{a' \in \mathcal{A}} Q^*(s', a') \quad (3.3)$$

Solving an MDP is equivalent to finding a value function that satisfies this optimality condition.

## 3.2 Solving Markov Decision Processes

When the model of a Markov Decision Process (MDP) is fully known, meaning both the transition probabilities  $P(s'|s, a)$  and reward function  $R(s, a)$  are available, finding an optimal policy becomes a planning problem. This problem can be solved using several methods, most notably Dynamic Programming (DP) and Linear Programming (LP).

### 3.2.1 Dynamic Programming

Dynamic Programming solves MDPs by converting the Bellman equations into practical, iterative algorithms [71, 10]. The two primary DP methods are Policy Iteration and Value Iteration.

**Policy Iteration (PI)** computes an optimal policy by alternating between two steps until the policy stabilizes [38]. Starting with an arbitrary policy  $\pi$ , the algorithm first performs **policy evaluation**, where the state-value function  $V^\pi$  is calculated by iteratively applying the Bellman expectation equation. Once the value function for the current policy has converged, the algorithm proceeds to **policy improvement**. In this step, a new, better policy is formed by acting greedily with respect to  $V^\pi$  for every state. This two-step cycle is guaranteed to converge to the optimal policy  $\pi^*$  in a finite number of iterations for a finite MDP [71].

**Value Iteration (VI)** combines the evaluation and improvement steps of PI into a single, more efficient update rule derived from the Bellman optimality equation [62]. Instead of iterating on the policy, VI iterates directly on the value function using the following update:

$$V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}} \left( R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V_k(s') \right) \quad (3.4)$$

This process repeats until the value function converges to the optimal value function  $V^*$ . The optimal policy  $\pi^*$  is then extracted by selecting the greedy action in each state with respect to  $V^*$ . While PI may converge in fewer iterations, each iteration is more computationally complex. The faster updates of VI often make it more efficient, particularly for problems with large state spaces [71].

### 3.2.2 Linear Programming

The problem of solving an MDP can also be formulated as a Linear Program (LP), offering a non-iterative approach to finding an optimal solution [62, 52]. This perspective is valuable because it formally establishes that solving an MDP is a convex optimization problem.

**The Primal Formulation.** The primal LP formulation solves for the optimal value function  $V^*$  directly. The objective is to find a value function that satisfies the Bellman optimality conditions while being as small as possible. The formulation is:

$$\begin{aligned} \min_V \quad & \sum_{s \in \mathcal{S}} \alpha(s) V(s) \\ \text{subject to} \quad & V(s) \geq R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s' | s, a) V(s'), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \end{aligned} \quad (3.5)$$

Here,  $\alpha(s)$  is a vector of positive weights (e.g., an initial state distribution) ensuring a well-posed problem, though the choice of  $\alpha(s)$  does not affect the optimal solution  $V^*$ . The constraints force the value of any state  $V(s)$  to be at least as large as the value that would be obtained by taking any action  $a$  and following the optimal policy thereafter. The minimization objective ensures that at the solution  $V^*$ , the inequality becomes an equality for at least one action in each state, thereby satisfying the Bellman optimality equation.

**The Dual Formulation and its Interpretation.** The dual of the primal LP provides a powerful and intuitive perspective by optimizing over a different set of variables. The dual formulation is:

$$\begin{aligned} \max_y \quad & \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} y(s, a) R(s, a) \\ \text{subject to} \quad & \sum_{a \in \mathcal{A}} y(s', a) = \alpha(s') + \gamma \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} y(s, a) P(s' | s, a), \quad \forall s' \in \mathcal{S} \\ & y(s, a) \geq 0, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \end{aligned} \quad (3.6)$$

The dual variables  $y(s, a)$  have a direct interpretation: they represent the discounted expected number of times action  $a$  is taken in state  $s$ . This quantity is known as the **state-action occupancy measure**. The objective is to find the occupancy measures that maximize the total expected discounted reward. The constraints ensure flow conservation, balancing the occupancy of a state with the probability of starting there and transitioning

into it from other state-action pairs. The power of the dual formulation lies in its direct link to policy execution. Once the optimal occupancy measures  $y^*(s, a)$  are found, the optimal stochastic policy  $\pi^*(a|s)$  can be recovered through simple normalization:

1

$$\pi^*(a|s) = \frac{y^*(s, a)}{\sum_{a' \in \mathcal{A}} y^*(s, a')} \quad (3.7)$$

This formulation yields a stationary policy because it assumes an infinite-horizon, discounted MDP. In Section 4.3, however, we consider a finite-horizon variant of the problem. In that case, the occupancy measure depends explicitly on time,  $y^t(s, a)$ , and the recovered policy  $\pi^t(a|s)$  becomes non-stationary (time-indexed). This distinction explains why the policy in Section 4.3 depends on  $t$ .

This dual perspective is central to our work, as it shifts the optimization problem from finding indirect values to finding direct measures of policy behavior.

### 3.3 Inverse Reinforcement Learning

The methods discussed in the previous section require a known and well-defined reward function  $R$ . In many real-world applications, this assumption is unrealistic. Manually designing a reward function to capture a desired behavior for a complex task, such as autonomous driving or robotic manipulation, is difficult and often impractical [1, 6]. A minor misspecification of the reward can lead to unintended and undesirable behavior.

**IRL** provides an alternative. Instead of using a reward function to generate a policy, **IRL** seeks to reverse this process. It infers the underlying reward function that best explains an expert’s observed behavior [57, 6].

#### 3.3.1 IRL Problem Formulation

The formal problem of **IRL** is to recover an agent’s reward function from its observed actions. An **IRL** algorithm takes two inputs: a set of expert-generated trajectories,

---

<sup>1</sup>The primal–dual LP written above corresponds to the infinite-horizon discounted case, where  $\gamma < 1$  ensures the expected return is finite and the dual variables  $y(s, a)$  represent discounted occupancy measures. For finite-horizon MDPs, the LP is modified to use time-indexed occupancies  $y^t(s, a)$  with explicit horizon  $T$ , which we adopt later in Section 4.3.

$\mathcal{D}_{\text{exp}} = \{\tau_1, \tau_2, \dots, \tau_M\}$ , where each trajectory  $\tau$  is a sequence of state-action pairs, and the environment’s dynamics,  $P$  [1, 79]. The objective is to find a reward function  $R^*$  under which the expert’s policy,  $\pi_E$ , is optimal or near-optimal.

To make this problem tractable, the reward function is commonly modeled as a linear combination of a set of  $d$ -dimensional features,  $\phi(s, a) \in \mathbb{R}^d$ :

$$R(s, a) = w^\top \phi(s, a)$$

In this model, the feature vector  $\phi(s, a)$  captures important properties of a state-action pair. For instance, in autonomous driving, features might include speed, distance to a leading vehicle, and lane centeredness. The unknown weight vector  $w \in \mathbb{R}^d$  represents the expert’s preferences for each feature [1, 79]. The IRL problem thus becomes the task of learning this finite-dimensional weight vector  $w$ .

A key concept in many IRL algorithms is **feature expectations**, which represent the expected discounted cumulative feature vector for a policy  $\pi$  [1]:

$$\mu(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t) \right]$$

The feature matching principle states that the goal is to find a policy  $\tilde{\pi}$  whose feature expectations  $\mu(\tilde{\pi})$  match the empirical feature expectations from the expert’s demonstrations,  $\mu_E$ . If  $\|\mu(\tilde{\pi}) - \mu_E\|_2 \leq \epsilon$  for a small  $\epsilon$ , the learned policy’s performance is guaranteed to be close to the expert’s performance with respect to the expert’s true, unknown reward function [1].

### 3.3.2 Ambiguity in Reward Inference: The Ill-Posed Nature of IRL

As discussed in Chapter 2, a fundamental challenge in IRL is that the problem is mathematically ill-posed [66, 57]. This means the reward function cannot be uniquely identified from the expert’s policy, even if that policy is perfectly optimal [4, 28]. Many different reward functions can explain the same observed behavior. This ambiguity arises from several sources:

1. **The Trivial Reward Function:** A constant reward function, such as  $R(s, a) = 0$  for all states and actions, makes every policy optimal. If all actions yield the same reward, no behavior is preferable to any other.

2. **Scaling Invariance:** If a policy  $\pi^*$  is optimal for a reward function  $R$ , it is also optimal for any positively scaled version  $\lambda R$ , where  $\lambda > 0$ . Uniformly scaling rewards does not change the relative preference between actions, so the optimal policy remains the same.
3. **Potential Shaping:** A more subtle source of ambiguity comes from potential-based reward shaping [56]. Any reward function can be altered with a specific “potential shaping” term without changing the optimal policy. For any function  $\Phi : \mathcal{S} \rightarrow \mathbb{R}$  that maps states to a scalar value, the transformed reward function:

$$R'(s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\Phi(s')] - \Phi(s)$$

produces the same set of optimal policies as the original reward function  $R$ . This implies that an infinite set of structurally different reward functions can generate the same optimal behavior, making them indistinguishable by observation alone.

This non-identifiability is the central hurdle that has driven the development of modern IRL algorithms. Early methods, like max-margin approaches, used heuristics to resolve the ambiguity, such as finding a reward function that makes the expert’s policy optimal by the largest possible margin [1]. The development of probabilistic frameworks, particularly Maximum Entropy IRL, was a direct response to this ambiguity, shifting the goal from finding a single correct reward function to characterizing a distribution over behaviors consistent with the expert’s demonstrations.

## 3.4 Maximum Entropy IRL

To address the core ambiguity in IRL, a probabilistic framework based on the principle of maximum entropy was introduced [79, 78, 76, 27]. This method infers the reward function that best explains the observed behavior while assuming as little as possible beyond the data.

### 3.4.1 The Maximum Entropy Principle

The principle of maximum entropy is a foundational idea in information theory, commonly applied in statistical inference [40, 79]. It states that, given certain known constraints (such as feature expectations from observed data), the best choice of a probability distribution

is the one with the highest entropy. This means selecting the most uniform or unbiased distribution that still satisfies the constraints. By avoiding unwarranted assumptions, this approach leads to simpler models that explain the data while reducing the risk of overfitting and improving generalization [79].

### 3.4.2 The MaxEnt IRL Framework

Maximum Entropy IRL (MaxEnt IRL) applies this principle by finding the probability distribution over trajectories,  $p(\tau)$ , that has the highest entropy while also being consistent with the expert’s observed behavior. This consistency is enforced by constraining the distribution’s expected feature counts to match the empirical feature counts from the demonstrations. This leads to the following constrained optimization problem:

$$\max_p \left[ - \sum_{\tau} p(\tau) \log p(\tau) \right] \quad \text{s.t.} \quad \sum_{\tau} p(\tau) \phi(\tau) = \mu_E \quad (3.8)$$

where  $\phi(\tau) = \sum_{(s,a) \in \tau} \phi(s, a)$  is the cumulative feature vector for a trajectory  $\tau$ , and  $\mu_E$  is the empirical feature count from the expert demonstrations. This formulation ensures that the learned distribution is maximally non-committal beyond the constraints imposed by the expert’s behavior.

The solution to this entropy maximization problem is a Gibbs distribution, where the probability of a trajectory  $\tau$  is exponentially proportional to the rewards accumulated along it [79, 27]:

$$P(\tau | w) = \frac{1}{Z(w)} \exp(w^\top \phi(\tau)), \quad (3.9)$$

Here, the reward weights  $w$  serve as the Lagrange multipliers for the feature matching constraints in the optimization problem.  $Z(w)$  is the partition function, which normalizes the distribution:

$$Z(w) = \sum_{\tau} \exp(w^\top \phi(\tau)).$$

The learning objective is to find the reward weights  $w$  that maximize the likelihood of the observed expert demonstrations  $\mathcal{D}_{\text{exp}}$ :

$$w^* = \arg \max_w \sum_{\tau \in \mathcal{D}_{\text{exp}}} \log P(\tau | w). \quad (3.10)$$

The gradient of the log-likelihood has an intuitive form [79]:

$$\nabla_w \mathcal{L}(w) = \mu_E - \mathbb{E}_{\tau \sim P(\cdot | w)}[\phi(\tau)], \quad (3.11)$$

where  $\mu_E$  is the empirical average of feature counts over the expert trajectories. This gradient reflects the difference between the expert’s feature expectations and those induced by the current model. Learning proceeds by adjusting  $w$  to minimize this gap. The optimization problem is convex and can be solved using gradient-based methods. For clarity and completeness, we summarize the standard Maximum Entropy IRL algorithm in Algorithm 1, following [79]. We include this detailed description because it provides the foundation for our later discussion of vulnerabilities and adversarial demonstrations.

The MaxEnt IRL framework offers a significant advantage over earlier deterministic approaches, as summarized in Table 3.2. By producing a full probability distribution over behaviors, it naturally accounts for noise and sub-optimality. This principled use of uncertainty helps resolve the ambiguity inherent in IRL and makes MaxEnt IRL more robust to imperfect demonstrations.

### 3.5 Problem Statement

The previous sections established that MaxEnt IRL provides a robust framework for learning from demonstrations, even when they are suboptimal. This robustness, however, relies on the fundamental assumption that the demonstrator acts in good faith [79, 7, 19]. This thesis challenges that assumption by investigating a scenario where a malicious adversary intentionally corrupts the demonstration data to manipulate the learning process. We model this as a data poisoning attack.

**Adversarial Demonstration Injection.** Consider a standard IRL setting where a learner aims to recover a reward function from a dataset of expert demonstrations,  $\mathcal{D}_{\text{exp}} = \{\tau_{\text{exp}}^i\}_{i=1}^M$ . The recovered reward function,  $\hat{R}$ , is then used to train a final deployed policy,  $\pi_{\text{deployed}}$ . Now, suppose an adversary generates a dataset of malicious demonstrations,  $\mathcal{D}_{\text{mal}} = \{\tau_{\text{mal}}^j\}_{j=1}^K$ , by following a malicious policy,  $\pi_{\text{mal}}$ . The adversary’s demonstrations are injected into the training set, creating a poisoned dataset  $\mathcal{D}_{\text{mod}}$ . This modified dataset is composed of a fraction of malicious data,  $\delta \in [0, 1]$ , and a fraction of expert data,  $1 - \delta$ . When the IRL algorithm is applied to  $\mathcal{D}_{\text{mod}}$ , it recovers a perturbed reward function,  $\tilde{R}$ , which in turn yields a corrupted deployed policy,  $\tilde{\pi}_{\text{deployed}}$ .

**Adversarial Objective.** The adversary’s goal is to design the malicious policy  $\pi_{\text{mal}}$  such that the performance of the resulting deployed policy,  $\tilde{\pi}_{\text{deployed}}$ , is minimized. Performance is measured by the expected return under the true (and unknown to the learner) expert reward function,  $R_{\text{exp}}$ .

---

**Algorithm 1** Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) [79]

---

- 1: **Input:** Expert trajectories  $\{\tau_i\}_{i=1}^N$ , feature map  $\phi(s, a)$ , learning rate  $\alpha$ , convergence threshold  $\epsilon$
- 2: **Initialize:** Reward weights  $w_0$  (e.g., zeros or small random values), iteration counter  $k = 0$
- 3: Compute empirical feature counts from expert demonstrations:

$$\mu_E = \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^T \phi(s_t^i, a_t^i)$$

- 4: **while**  $\|\mu_E - \mu_k\|_2 > \epsilon$  **do**
- 5:     Compute reward function:  $R_k(s, a) = w_k^\top \phi(s, a)$
- 6:     Compute soft value function  $V_k(s)$  via backward soft value iteration:

$$V_k(s) = \log \sum_a \exp \left( R_k(s, a) + \gamma \sum_{s'} P(s'|s, a) V_k(s') \right)$$

- 7:     Derive stochastic policy:

$$\pi_k(a|s) = \frac{\exp(R_k(s, a) + \gamma \sum_{s'} P(s'|s, a) V_k(s'))}{\sum_{a'} \exp(R_k(s, a') + \gamma \sum_{s'} P(s'|s, a') V_k(s'))}$$

- 8:     Estimate expected feature counts  $\mu_k$  under current policy  $\pi_k$  (e.g., via forward state visitation):

$$\mu_k = \mathbb{E}_{\tau \sim \pi_k} \left[ \sum_{t=0}^T \phi(s_t, a_t) \right]$$

- 9:     Update reward weights:

$$w_{k+1} \leftarrow w_k + \alpha(\mu_E - \mu_k)$$

- 10:      $k \leftarrow k + 1$
  - 11: **end while**
  - 12: **Return:** Final reward weights  $w_k$
-

**Similarity Constraint.** To be effective in a data poisoning attack, the adversarial demonstrations must closely resemble those of the expert in distribution. The adversarial demonstrations should be statistically similar to the expert demonstrations to avoid detection. We formalize this with a constraint:

$$d(\mathcal{D}_{\text{mal}}, \mathcal{D}_{\text{exp}}) \leq \epsilon, \quad (3.12)$$

where  $d(\cdot, \cdot)$  is a suitable divergence metric and  $\epsilon$  is a small tolerance threshold. Given the mechanics of MaxEnt IRL, a natural choice for  $d$  is a distance metric based on feature expectations.

Building upon these concepts, we formally investigate the following research problem:

**Problem 1.** *Can a malicious policy be designed to significantly impair the performance of a deployed policy learned from an IRL-recovered reward when a fraction  $\delta$  of expert demonstrations is substituted with adversarial data, while the adversarial data remains statistically indistinguishable from the expert’s? To what extent does the performance of this learned policy degrade?*

This problem gives rise to several fundamental questions that this thesis aims to answer:

1. What strategies can a malicious demonstrator employ to manipulate the learned policy most effectively?
2. How sensitive is MaxEnt IRL to adversarial corruption of its training data, and how much data manipulation is required for significant performance degradation?
3. In safety-critical environments, what are the practical implications of such adversarial corruption on agent behavior?

## Chapter Summary

This chapter established the theoretical groundwork and defined the central problem for the thesis. We began with the **Markov Decision Process (MDP)**, the formal framework for sequential decision-making. We then introduced **IRL** as the problem of inferring rewards from demonstrations, highlighting that it is fundamentally **ill-posed** due to the ambiguity of reward functions.

To address this challenge, we provided a detailed overview of **Maximum Entropy (MaxEnt) IRL**, a principled probabilistic method that resolves ambiguity by finding the

reward function that explains the expert’s behavior with the least possible commitment. This is achieved by finding a distribution over behaviors that matches the expert’s observed **feature expectations** while maximizing entropy.

Finally, we leveraged these concepts to formalize our research problem. We defined an **adversarial data poisoning** attack model where a malicious agent injects carefully crafted demonstrations into an expert dataset. The adversary’s goal is to corrupt the reward function inferred by a MaxEnt [IRL](#) algorithm, thereby degrading the performance of the final deployed policy, all while remaining undetected. With the foundational concepts established and the research problem defined, we now proceed to develop our attack methodology and analyze its effectiveness.

Table 3.2: A Comparison of IRL Paradigms: Max-Margin vs. Maximum Entropy.

Aspect	Max-Margin IRL [1]	Maximum Entropy IRL [79]
<b>Core Principle</b>	Seeks a reward function where the expert’s policy value exceeds that of all other policies by the largest margin.	Finds the reward function that makes the expert’s behavior maximally likely under a probabilistic model that is otherwise maximally non-committal.
<b>Optimization Problem</b>	A constrained optimization problem that maximizes the value margin between the expert’s policy and all alternatives.	An unconstrained convex optimization problem that maximizes the log-likelihood of the expert trajectories, which is equivalent to matching feature expectations.
<b>Ambiguity Resolution</b>	Assumes expert optimality and resolves ambiguity by finding the reward that makes the expert appear “most” optimal.	Resolves ambiguity by selecting the maximum entropy (most random) policy distribution consistent with the expert’s observed feature counts.
<b>Resulting Policy</b>	Typically deterministic, as the learned reward is used with a standard MDP solver (e.g., Value Iteration).	Inherently stochastic. The probability of a trajectory $\tau$ is exponentially proportional to its reward: $P(\tau) \propto \exp(R(\tau))$ .
<b>Robustness to Suboptimality</b>	Sensitive to suboptimal demonstrations. Can fail if no reward function renders the expert’s policy strictly optimal.	Naturally handles suboptimal and noisy demonstrations by modeling a distribution over all behaviors rather than assuming a single optimal path [78].

# Chapter 4

## Methodology

Building on the problem statement defined in Chapter 3, this chapter details the methodology for investigating adversarial attacks against Maximum Entropy [Inverse Reinforcement Learning \(IRL\)](#). We develop a formal, optimization-based framework to generate maximally harmful yet indistinguishable adversarial demonstrations. This allows us to precisely model the adversary’s capabilities and systematically evaluate the impact of their data poisoning attack on the learner’s final policy.

### 4.1 Threat Model and Methodological Assumptions

To formally analyze the research problem stated in Section 3.5, we must first define the specific threat model and methodological assumptions that ground our investigation. These choices allow us to establish a worst-case scenario to determine the system’s vulnerability.

- **Adversary’s Goal (Performance Degradation):** Consistent with our problem statement, the adversary’s primary objective is to manipulate the training data such that the final policy deployed by the learner achieves the lowest possible expected cumulative reward under the true, secret reward function.
- **Adversary’s Knowledge (White-Box Attack):** We assume a **white-box** attack model, where the adversary has complete knowledge of the system: the MDP’s transition function ( $P$ ), the expert’s true reward weights ( $w^*$ ), and access to the expert’s policy or demonstrations. This strong assumption is standard for foundational security analysis and allows us to establish the maximum potential threat.

- **Adversary’s Capability (Data Poisoning):** The adversary’s sole capability is to inject a limited number of maliciously crafted demonstration trajectories into the expert’s dataset, modeled by a contamination ratio  $\delta \in [0, 1]$ . The adversary cannot alter the IRL algorithm itself.
- **Learner and Environment Scope:** Our analysis focuses exclusively on **Maximum Entropy IRL** within finite-horizon, discrete-state, and discrete-action MDPs.
- **Similarity Metric:** We formalize the similarity constraint from Problem 1 using an  $\ell_2$ -norm bound on feature expectations, following the approach of [36]. This metric is computationally tractable and directly relevant to the feature-matching objective of MaxEnt IRL. The choice of the  $\ell_2$ -norm encourages the adversary to make many small, subtle changes across all features rather than a few large, obvious ones, making the attack more difficult to detect via simple observation. We also use the  $\ell_2$ -norm because it is a standard metric for measuring the distance between feature expectations in previous IRL work.

While the white-box assumption is a limitation, analyzing this worst-case scenario provides a crucial security baseline. Insights gained from this model can inform the development of defenses against more constrained, real-world attackers.

## 4.2 Modeling the Expert and the Adversary

To address this problem, we require formal models for the genuine expert and the malicious adversary. The expert provides demonstrations that the IRL agent aims to emulate, while the adversary attempts to corrupt this learning process. This section details the formal models for both agents, establishing the foundation for our analysis.

### 4.2.1 Expert Model: Soft Value Iteration

To align with the probabilistic nature of Maximum Entropy IRL, we model the expert as a *softly* rational agent rather than a perfectly optimal one [78, 79]. A softly rational agent balances the goal of maximizing cumulative reward with maintaining a degree of stochasticity, or entropy, in its policy. This behavior, which can capture more realistic forms of decision-making, is modeled using **Soft Value Iteration (SVI)**. SVI is a well-established modification of the classic Value Iteration algorithm [78, 35].

SVI incorporates an entropy term into the Bellman equation [79]. Whereas the standard Bellman operator uses a “hard” maximum to select the single best action, the **soft Bellman operator** uses a “soft” maximum. This operation, formally a log-sum-exp function, considers a weighted combination of all actions [78]. The resulting **soft Q-value**,  $Q_{\text{soft}}^*(s, a)$ , represents the value of a state-action pair adjusted for this policy entropy. The optimal soft value function is defined as [78]:

$$V_{\text{soft}}^*(s) = \alpha \log \sum_{a' \in \mathcal{A}} \exp \left( \frac{1}{\alpha} Q_{\text{soft}}^*(s, a') \right) \quad (4.1)$$

The optimal policy under this framework, which we use to model our expert,  $\pi_{\text{expert}}$ , is derived from a softmax distribution over the soft Q-values:

$$\pi_{\text{expert}}(a | s) = \frac{\exp \left( \frac{1}{\alpha} Q_{\text{soft}}^*(s, a) \right)}{\sum_{a' \in \mathcal{A}} \exp \left( \frac{1}{\alpha} Q_{\text{soft}}^*(s, a') \right)} = \exp \left( \frac{1}{\alpha} (Q_{\text{soft}}^*(s, a) - V_{\text{soft}}^*(s)) \right) \quad (4.2)$$

Here,  $\alpha > 0$  is a temperature parameter that controls the level of stochasticity. As  $\alpha \rightarrow 0$ , the policy approaches a deterministic optimal policy. Conversely, larger values of  $\alpha$  lead to more uniform, exploratory behavior. This model provides a principled method for generating the expert demonstrations,  $\mathcal{D}_{\text{expert}}$ , that serve as the ground truth for the **IRL** problem.

## 4.2.2 Adversarial Model: Constrained Value Minimization

The adversary’s goal is to find a policy that is maximally harmful yet stealthy. We model this as a constrained optimization problem: the adversary seeks a policy that **minimizes** the expected cumulative reward under the true reward function  $R^*$ , while ensuring its behavior stays close enough to that of the expert to avoid detection.

This stealth requirement is formalized through **feature matching**. Since MaxEnt **IRL** learns a reward function by matching the expert’s feature expectations, the adversary exploits this by generating trajectories whose feature counts are nearly identical to those of the expert. Specifically, the adversary’s expected feature vector must lie within an  $\epsilon$ -ball of the expert’s empirical feature vector  $\hat{\phi}_{\text{E}}$ , computed from the expert’s demonstration set  $\mathcal{D}_{\text{expert}}$ . We express this similarity constraint as:

$$\left\| \sum_{t,s,a} x(s, a)^t \phi(s) - \hat{\phi}_{\text{E}} \right\|_p \leq \epsilon \quad (4.3)$$

where  $x(s, a)^t$  is the occupancy measure of the adversarial policy and  $\phi(s)$  is the feature vector for state  $s$ . This constraint ensures that the adversary’s trajectories remain close to the expert’s, even as they encode a harmful objective.

The choice of norm  $\|\cdot\|_p$  defines the adversary’s stealth strategy. In this work, we use the  $\ell_2$ -**norm**, which encourages small, spread-out changes across all features, making the attack more subtle and harder to detect.

### 4.3 An Optimization Framework for Adversarial Attacks

We formulate the task of generating deceptive demonstrations as a convex optimization problem using occupancy measures and feature matching. This lets the malicious demonstrator find a policy that minimizes the true expected return while staying statistically close to the expert, and allows us to use efficient solvers to find the optimal deceptive policy under the given constraints.

#### 4.3.1 Occupancy Measures for Policy Representation

The key technique that makes our framework tractable is the use of **state-action occupancy measures**. An occupancy measure  $x(s, a)^t$  represents the expected number of times a policy visits state  $s$  and takes action  $a$  at time step  $t$ . There is a one-to-one correspondence between policies and the set of valid, feasible occupancy measures [72]. This is powerful because policy-dependent expectations, which are often complex, become linear functions of the occupancy measure. This allows us to transform the difficult problem of searching over the space of policies into an efficient search over the convex polytope of valid occupancy measures.

### 4.3.2 The Adversarial Optimization Problem

The adversary solves the following optimization problem to find the optimal deceptive occupancy measure,  $x^*$ :

$$\min_{x(s,a)^t} \sum_{s,a,t} \gamma^t x(s,a)^t \phi(s)^T w^* \quad (4.4a)$$

$$\text{subject to: } \left\| \sum_{s,a,t} x(s,a)^t \phi(s) - \hat{\phi}^E \right\|_2 \leq \epsilon, \quad (4.4b)$$

$$\sum_{a \in \mathcal{A}} x(s,a)^t = \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} P(s|s', a') x(s', a')^{t-1}, \quad \forall s, t, \quad (4.4c)$$

$$\sum_{s,a} x(s,a)^0 = p_0(s), \quad \forall s, \quad (4.4d)$$

$$x(s,a)^t \geq 0, \quad \forall s, a, t. \quad (4.4e)$$

Here, (4.4a) is the value-minimization objective, (4.4b) is the  $\ell_2$ -norm stealth constraint, and constraints (4.4c)-(4.4e) ensure that the resulting  $x$  is a dynamically feasible occupancy measure corresponding to a valid policy.

### 4.3.3 Policy Extraction from Occupancy Measures

The solution to the optimization problem is an occupancy measure,  $x^*(s,a)^t$ . To generate trajectories, this measure must be converted into an executable policy,  $\pi^t(a|s)$ .

Because we consider a finite-horizon MDP of length  $T$ , the extracted policy is non-stationary (time-indexed). This does not contradict the Markov property: the process remains Markovian, but optimal actions may vary across time steps since the remaining horizon length influences decision-making.

First, we compute the time-dependent state occupancy measure,  $\tilde{x}^t(s)$ , by marginalizing over actions:

$$\tilde{x}^t(s) := \sum_{a \in \mathcal{A}} x^*(s,a)^t \quad \forall s \in \mathcal{S}, t = 0, 1, \dots, T. \quad (4.5)$$

The time-indexed policy  $\pi^t(a|s)$  is then recovered by normalizing the state-action occupancy by the state occupancy at each time step [30]:

$$\pi^t(a|s) = \begin{cases} \frac{x^*(s,a)^t}{\tilde{x}^t(s)}, & \text{if } \tilde{x}^t(s) > 0, \\ \pi_0^t(a), & \text{otherwise,} \end{cases} \quad (4.6)$$

where  $\pi_0^t(a)$  is a default policy (e.g., a uniform random policy) for states not visited under the optimal adversarial plan. By construction, this ensures that for each time  $t$  and state  $s$ , the action probabilities sum to one:  $\sum_{a \in \mathcal{A}} \pi^t(a | s) = 1$ . This policy is then used to generate the set of adversarial demonstrations,  $\mathcal{D}_{\text{adv}}$ , which can be used to evaluate the vulnerability of an [IRL](#) agent.

### 4.3.4 Algorithmic Implementation

The problem in (4.4) is a Quadratically Constrained Linear Program (QCLP), which can be solved efficiently using standard convex optimization software (e.g., CVXPY with a solver like MOSEK or Gurobi). The overall process for generating adversarial data is summarized in Algorithm 2.

---

#### Algorithm 2 Adversarial Trajectory Generation

---

- 1: **Input:** Environment  $(P, \gamma, p_0)$ , true reward weights  $w^*$ , expert policy model (SVI with parameter  $\alpha$ ), stealth budget  $\epsilon$ .
  - 2: Generate a set of expert demonstrations  $\mathcal{D}_{\text{expert}}$  by rolling out  $\pi_{\text{expert}}$ .
  - 3: Compute the empirical expert feature expectations  $\hat{\phi}^E$  from  $\mathcal{D}_{\text{expert}}$ .
  - 4: Construct the optimization problem defined in Equation (4.4).
  - 5: Solve the QCLP to obtain the optimal adversarial occupancy measure  $x^*(s, a)^t$ .
  - 6: Extract the time-indexed adversarial policy  $\pi_{\text{adv}}^t(a|s)$  from  $x^*(s, a)^t$ .
  - 7: Generate a set of adversarial demonstrations  $\mathcal{D}_{\text{adv}}$  by rolling out  $\pi_{\text{adv}}^t$ .
  - 8: **Output:** Adversarial dataset  $\mathcal{D}_{\text{adv}}$ .
- 

### 4.3.5 Baseline for Comparison: The Constrained Randomized Policy

To evaluate the strategic nature of our adversarial policy, we must compare it against a meaningful baseline. A naive attack could simply inject random noise, which might also degrade the [IRL](#) agent’s performance. To demonstrate that our optimization-based attack is more effective than random corruption, we establish a baseline that is subject to the same statistical constraint as our adversary.

This baseline is a randomized policy whose occupancy measure is also constrained to satisfy Equation 4.4b. In states where the feature-matching constraint is not active, this policy defaults to uniform random action selection, as described by the ‘otherwise’

case in Equation 4.6. This baseline is more rigorous than unconstrained noise because it maintains the same statistical similarity to the expert as the optimized adversary. By comparing the performance degradation caused by our strategic adversary to that caused by this constrained random policy, we can isolate the impact of the value-minimization objective, separating strategic harm from the effects of mere statistical deviation.

## 4.4 Justification of the Adversarial Objective

This section provides the formal mathematical reasoning that underpins our choice of adversarial objective. We demonstrate that for an adversary conducting a data poisoning attack, the goal of maximally degrading the performance of the final policy learned by a feature-matching IRL agent is provably equivalent to minimizing the expected value of the adversarial demonstrations themselves. This equivalence is critical, as it validates the specific optimization problem formulated in our methodology.

We begin by formally defining the components of the adversarial interaction.

**Definition 1** (Component Definitions). *Let the following be defined for an MDP with true reward weights  $w^* \in \mathbb{R}^d$  and feature map  $\phi$ :*

- **Feature Expectations**  $\mu(\pi) \triangleq \mathbb{E}_\pi[\sum_t \phi(s_t)]$ : *The expected feature vector for a policy  $\pi$ .*
- **Policy Value**  $V(\pi) \triangleq w^{*\top} \mu(\pi)$ : *The true expected return of policy  $\pi$ .*
- **Expert and Adversarial Features**:  $\mu_E \triangleq \mu(\pi_E)$  and  $\mu_A \triangleq \mu(\pi_A)$  *are the feature expectations of the expert and adversarial policies, respectively.*
- **Mixed Features**  $\mu_{mix} = (1 - \delta)\mu_E + \delta\mu_A$ : *The feature expectations of the contaminated dataset, where  $\delta \in (0, 1)$  is the mixing (contamination) ratio.*
- **Relearned Policy**  $\pi_{relearned}$ : *The policy learned by the agent, which, by the principle of feature matching, satisfies  $\mu(\pi_{relearned}) = \mu_{mix}$ .*

The adversary’s ultimate goal is to select their policy  $\pi_A$  (and thus its feature vector  $\mu_A$ ) to minimize the value of the final policy deployed by the learner,  $V(\pi_{relearned})$ , subject to a stealth constraint  $\|\mu_A - \mu_E\|_2 \leq \epsilon$ . The following theorem proves that this goal is equivalent to the more direct objective of minimizing  $V(\pi_A)$ .

**Theorem 1** (Equivalence of Adversarial Objectives). *Minimizing the value of the final learned policy,  $V(\pi_{\text{relearned}})$ , with respect to the adversary’s choice of  $\mu_A$  is mathematically equivalent to minimizing the value of the adversarial policy,  $V(\pi_A)$ , under the same constraints. Formally:*

$$\arg \min_{\mu_A} V(\pi_{\text{relearned}}) = \arg \min_{\mu_A} V(\pi_A)$$

*Proof.* The adversary’s primary objective is to solve the following optimization problem:

$$\min_{\mu_A \text{ s.t. } \|\mu_A - \mu_E\|_2 \leq \epsilon} V(\pi_{\text{relearned}})$$

We begin by expanding the objective function,  $V(\pi_{\text{relearned}})$ , using the definitions provided in 1. By the principle of feature matching, the learned policy’s features are equal to the mixed features.

$$\begin{aligned} V(\pi_{\text{relearned}}) &= w^{*\top} \mu(\pi_{\text{relearned}}) \\ &= w^{*\top} \mu_{\text{mix}} \end{aligned}$$

Substituting the definition of  $\mu_{\text{mix}}$ :

$$\begin{aligned} V(\pi_{\text{relearned}}) &= w^{*\top} ((1 - \delta)\mu_E + \delta\mu_A) \\ &= (1 - \delta)w^{*\top} \mu_E + \delta w^{*\top} \mu_A && \text{(by linearity of the dot product)} \\ &= (1 - \delta)V(\pi_E) + \delta V(\pi_A) && \text{(by definition of policy value)} \end{aligned}$$

The adversary’s optimization problem can therefore be expressed as finding the  $\mu_A$  that solves:

$$\min_{\mu_A} [(1 - \delta)V(\pi_E) + \delta V(\pi_A)]$$

In this expression, the term  $(1 - \delta)V(\pi_E)$  is a constant with respect to the optimization variable  $\mu_A$ , since the expert’s policy  $\pi_E$ , the true reward weights  $w^*$ , and the contamination ratio  $\delta$  are all fixed from the adversary’s perspective. Furthermore, since  $\delta$  is a positive constant, minimizing an affine function of the form  $C_1 + \delta \cdot g(x)$  (where  $C_1$  is a constant) is equivalent to minimizing the function  $g(x)$  itself.

Here,  $C_1 = (1 - \delta)V(\pi_E)$  and our function is  $g(\mu_A) = V(\pi_A)$ . Therefore, the argument  $\mu_A$  that minimizes the full expression is the same as the argument that minimizes  $V(\pi_A)$ . This shows the equivalence of the problems and completes the proof.  $\square$

The result of Theorem 1 confirms that our chosen adversarial objective, minimizing the value of the crafted demonstrations, is sufficient to produce demonstrations that most effectively degrade the agent’s final performance.

## 4.5 Chapter Summary

This chapter detailed the formal methodology for investigating adversarial attacks against Maximum Entropy IRL. We began by establishing a precise white-box threat model, where an adversary’s goal is to degrade the learner’s policy performance by injecting a limited number of poisoned demonstrations. The adversary’s stealth is formally defined by an  $\ell_2$ -norm similarity constraint on the demonstrations’ feature expectations.

The core of our methodology is an optimization framework that models the generation of adversarial data as a convex optimization problem. By representing policies with state-action occupancy measures, we transformed the complex search over policy space into a tractable Quadratically Constrained Linear Program (QCLP). This allows the adversary to efficiently find a policy that minimizes the expected cumulative reward under the true reward function while remaining statistically indistinguishable from the expert. We also outlined the complete algorithmic process for generating these adversarial trajectories and established a constrained random policy as a baseline for comparison.

Critically, we provided a formal justification for our adversarial objective. We proved through Theorem 1 that minimizing the value of the adversarial demonstrations is mathematically equivalent to the adversary’s ultimate goal: minimizing the performance of the policy ultimately learned by the agent from the poisoned data. This theoretical validation confirms the soundness of our approach. With this principled methodology in place, we are equipped to empirically evaluate the threat of such adversarial attacks.

# Chapter 5

## Simulation Studies

This chapter presents a series of simulation studies designed to empirically validate the concepts developed in this thesis. We conduct controlled experiments to assess the impact of mixed-policy demonstration data on [Inverse Reinforcement Learning \(IRL\)](#) and the performance of policies trained on the inferred rewards. The primary objective is to quantify how the presence of non-expert trajectories, specifically those generated by malicious and random policies, affects the ability of a standard [IRL](#) algorithm to recover an expert’s underlying reward function.

Our investigation is guided by two central research questions:

1. How sensitive are standard [IRL](#) algorithms, such as Maximum Entropy [IRL](#), to demonstration datasets that contain a mixture of expert and non-expert behaviors?
2. How do different types of non-expert data, specifically, systematically adversarial versus purely random trajectories, differ in their impact on the reward inference process?

To address these questions, we perform experiments in two well-understood grid-world environments: **CliffWorld** and **Four Rooms**. These environments are computationally tractable yet represent distinct challenges common in [Reinforcement Learning \(RL\)](#), namely safety-critical navigation and complex exploration. By systematically varying the proportion of non-expert data, denoted by  $\delta$ , and a feature similarity constraint, denoted by  $\epsilon$ , we map the vulnerability surface of the [IRL](#) process.

This chapter is structured as follows. First, we detail the simulation setup, including the methods for generating and mixing trajectories, the process for training the learning agent,

and the specific characteristics of the experimental environments. Next, we present and analyze the results from our two case studies, dedicating a section to each. In these sections, we evaluate the performance of the learned policies and interpret the graphical results to understand the vulnerabilities observed. We conclude with a comparative analysis and a discussion of the key findings, highlighting their broader implications for applying IRL in real-world scenarios where data sources may not be perfectly reliable.

## 5.1 Simulation Setup

We built a simulation pipeline to test how the composition of demonstration data affects reward inference and policy learning. The pipeline’s core components are: the policies used to generate trajectories, the data mixing procedure, the final policy training stage, and the evaluation environments.

### 5.1.1 Trajectory Generation and Mixing

The foundation of our experiments is a dataset that mimics a scenario where expert demonstrations are contaminated with data from other sources. This is a realistic assumption in many applications where data is collected from users with varying skill levels or intentions.

#### Policy Definitions

We begin by defining three distinct policies for each environment:

- **Expert Policy ( $\pi_{\text{expert}}$ ):** To align with the probabilistic framework of our model, as discussed in 4.2, the expert is not assumed to be perfectly optimal, but rather *softly rational* [79]. This expert policy, derived using Soft Value Iteration (SVI), represents a highly proficient behavior that balances maximizing cumulative reward with maintaining a degree of stochasticity. It serves as the target behavior for the learning agent to imitate. We generate a set of 1000 expert trajectories by executing this policy.
- **Malicious Policy ( $\pi_{\text{malicious}}$ ):** An adversarial policy designed to find worst-case demonstrations that mislead the reward learning process. As explained in 4.3 rather than being random, this policy algorithmically generates behaviors that are subtly

yet maximally detrimental. These behaviors are constrained to remain plausible enough to be confused with expert demonstrations, making them effective for discovering vulnerabilities in the reward model. A corresponding set of 1000 malicious trajectories is generated.

- **Random Policy ( $\pi_{\text{random}}$ ):** A policy that selects an action uniformly at random from the set of available actions in each state subject to the similarity constraint. This policy serves as a baseline for unstructured, non-expert behavior. We generate 1000 trajectories by executing this policy.

## Dataset Composition

With these three sets of trajectories, we create our final training datasets. Each dataset is a mixture of expert trajectories and one of the two non-expert types (malicious or random). A **mixing ratio**, denoted by  $\delta \in [0, 1]$ , controls the composition by defining the proportion of non-expert trajectories. For a total dataset size of  $N = 1000$  trajectories, the final set contains  $(1 - \delta) \cdot N$  expert trajectories and  $\delta \cdot N$  non-expert trajectories.

For instance, a mixing ratio of  $\delta = 0.1$  results in a dataset composed of 900 expert trajectories and 100 malicious (or random) trajectories. This process allows us to systematically study how the level and type of data contamination affect the outcome. The final mixed dataset serves as the input to the Maximum Entropy IRL algorithm, for which we use the standard implementation provided by the `imitation` library [31]. Our central hypothesis is that the algorithm will be misled by the non-expert data, and by observing how the inferred reward function changes with different levels of contamination, we can quantify the robustness of the method.

### 5.1.2 Reward Inference and Policy Training

The ultimate test of an inferred reward function is the quality of the policy it produces. A reward function that appears plausible but leads to a suboptimal or unsafe policy is not practically useful. Therefore, our evaluation process involves two key steps.

First, the MaxEnt IRL algorithm infers a reward function from the mixed-policy dataset. Second, we use this inferred reward function to train a new policy from scratch. We refer to this policy as the **deployed policy** ( $\pi_{\text{deployed}}$ ). The training of  $\pi_{\text{deployed}}$  is a standard RL problem where the agent’s objective is to find a policy that maximizes the cumulative sum of the *inferred* rewards.

The performance of this deployed policy is our primary evaluation metric. We measure its effectiveness by computing the **empirical average return**, which is the average total *true* reward the policy accumulates over many evaluation episodes. Using the ground-truth reward for evaluation allows us to objectively measure how well the deployed policy performs the intended task, independent of the potentially flawed inferred reward. A significant drop in the average true return of  $\pi_{\text{deployed}}$  indicates that data contamination has successfully poisoned the reward inference process, resulting in an ineffective or dangerous policy.

### 5.1.3 Simulation Environments

The choice of environment is crucial in any RL study. To ensure our findings are generalizable, we selected two environments that present distinct challenges: the classic **CliffWorld** and a modified **Four Rooms** environment. Both are visualized in Figure 5.1.

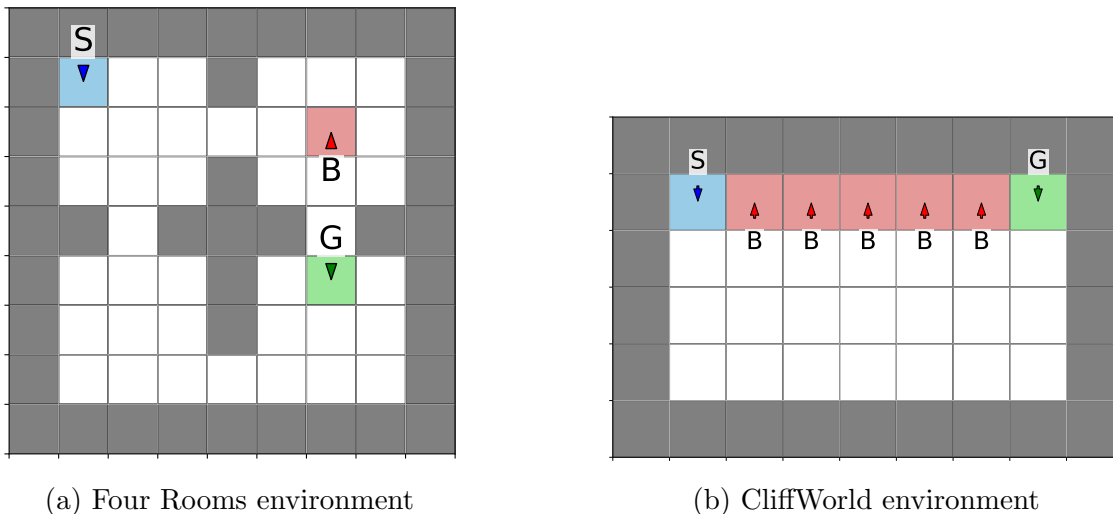


Figure 5.1: A visualization of the two grid-world environments used in our simulation studies. The goal state (G) is shown in green, the hazardous state (B) is in red, and the agent’s starting state (S) is in blue. These environments provide structured testbeds for safety and exploration challenges.

## CliffWorld

As depicted in Figure 5.1b, CliffWorld is a grid-based navigation task widely used to study the trade-off between exploration and safety [71]. The environment consists of a grid where an agent starts at a designated location (S) and must navigate to a goal (G). A large section of the grid is a “cliff”, which represents a failure state.

The reward structure reflects the high stakes of a safety-critical application:

- **Goal:** Reaching the goal state provides a reward of +10.
- **Cliff:** Stepping into any state in the cliff region results in a large penalty of  $-10$ .
- **Step Penalty:** Each action incurs a small penalty of  $-1$  to encourage finding an efficient path.

The optimal policy in CliffWorld involves taking a longer, safer path along the edge of the grid. Any policy that attempts a shortcut risks falling into the cliff. This makes CliffWorld an excellent environment for studying how adversarial data can manipulate a learned policy into taking unsafe actions. The planning horizon is set to  $T = 10$ , emphasizing the need for both safe and efficient navigation.

## Four Rooms

The Four Rooms environment, shown in Figure 5.1a, presents an exploration challenge. Based on the MiniGrid framework [22], this environment consists of four interconnected rooms with narrow passageways. The agent starts in one room and must navigate to a goal located in another.

For our experiments, we modified the standard Four Rooms environment to ensure **full observability**. This means the agent perceives the entire state of the environment at each timestep. This modification simplifies the problem by removing the challenge of partial observability, allowing us to focus purely on planning and decision-making in a larger state space.

The reward structure is defined as follows:

- **Goal:** Reaching the goal state yields a reward of +1.
- **Hazardous State:** Entering a specific “hazardous” state (B) results in a penalty of  $-3$ .

- **Step Penalty:** A penalty of  $-1$  is applied for each step to encourage efficiency.

Compared to CliffWorld, Four Rooms has a significantly larger state space and a longer planning horizon of  $T = 50$ . The main difficulty is discovering the narrow passageways to construct a path to the goal. This makes it a suitable test case for understanding how adversarial data might affect a learned policy’s ability to explore and navigate in a more complex environment.

## 5.2 Case Study 1: CliffWorld

The CliffWorld environment serves as our primary testbed for evaluating the impact of adversarial data in a safety-critical context. The severe penalty for falling off the cliff makes any deviation from the optimal safe path highly consequential. In this section, we analyze how the performance of the deployed policy degrades in the presence of malicious and random non-expert trajectories.

### 5.2.1 Impact of Malicious Trajectories

Our first experiments investigate the effect of the mixing ratio,  $\delta$ , which controls the percentage of non-expert trajectories in the demonstration dataset. We do this by fixing a value for  $\epsilon$  and changing  $\delta$  gradually. The results, presented in Figure 5.2, are clear. The most critical observation is that **malicious trajectories have a disproportionately larger negative impact than random trajectories**. For example, a small contamination of just 10% malicious data causes a drop in the deployed policy’s average return that is comparable to that from a 50% contamination with random data. This finding highlights a crucial vulnerability: MaxEnt IRL is particularly susceptible to structured, adversarial noise. An adversary does not need to overwhelm the dataset with bad examples; a few carefully chosen trajectories can be sufficient to compromise the learned reward function and, consequently, the safety of the resulting policy.

This heightened sensitivity is a direct result of how the malicious policy is generated. It actively produces trajectories that differ from the expert’s in terms of feature counts (e.g., visiting different states) while remaining plausible enough to mislead the learning algorithm. In CliffWorld, this often means generating trajectories that move closer to the cliff’s edge or hit the cliff. When MaxEnt IRL processes this contaminated data, it may incorrectly infer that states near the cliff are not as dangerous as they truly are, leading to a deployed policy that takes unsafe shortcuts.

*Although our method does not explicitly target particular actions, in practice the resulting malicious trajectories tend to concentrate their errors around the most safety-critical states (e.g., near the cliff edge). This emergent behavior mirrors what one might expect from an adversary who deliberately manipulates only high-stakes actions, even though the attack here is trajectory-level rather than action-level. As a result, the observed vulnerability already approximates the effect of a more targeted adversary, highlighting how easily structured noise can undermine safety.*

## 5.2.2 Sensitivity to the Similarity Constraint

The second parameter we investigate is the similarity bound,  $\epsilon$ , which constrains how much the feature counts of non-expert trajectories can deviate from those of the expert. A smaller  $\epsilon$  forces non-expert trajectories to be very similar to the expert’s, while a larger  $\epsilon$  allows for greater deviation.

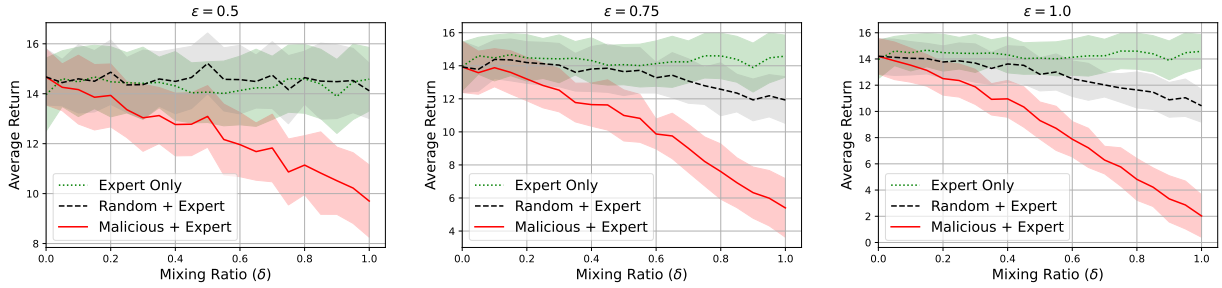
The plots in Figure 5.3 show a consistent trend: as  $\epsilon$  increases, the average return of the deployed policy steadily decreases. This is intuitive, as a larger  $\epsilon$  gives the adversary more freedom to generate misleading trajectories. At very low values of  $\epsilon$ , the constraints are tight, so neither the malicious nor the random policy can deviate significantly from the expert, and the learned policy’s performance remains high.

However, an interesting phenomenon occurs at higher values of  $\epsilon$ . The impact of the random baseline begins to plateau because the random policy, being undirected, cannot effectively exploit the additional freedom to degrade performance further. In contrast, the malicious policy continues to drive down the average return. This demonstrates that the malicious policy is more efficient at using its deviation budget to cause harm.

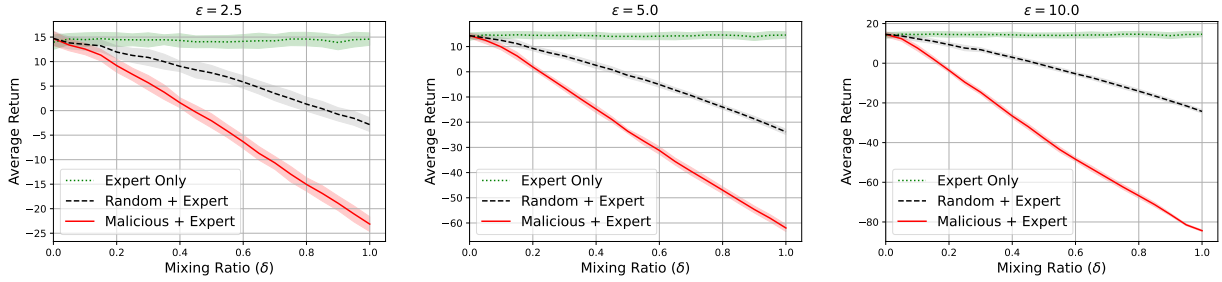
## 5.2.3 Summary of Findings in CliffWorld

The short planning horizon of CliffWorld ( $T = 10$ ) likely exacerbates the performance degradation, as the agent has little opportunity to recover from a poor decision. A single wrong step can lead to a fall into the cliff. In summary, the CliffWorld case study underscores a critical vulnerability for IRL applications in high-risk domains. The results demonstrate that:

1. Adversarially generated trajectories are significantly more damaging than random noise.

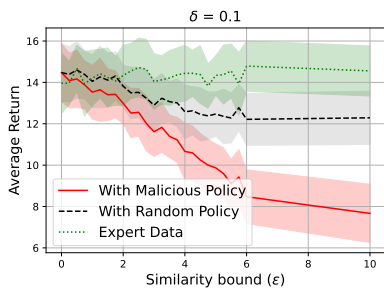


(a) Similarity bound ( $\epsilon$ ) of 0.5    (b) Similarity bound ( $\epsilon$ ) of 0.75    (c) Similarity bound ( $\epsilon$ ) of 1.0

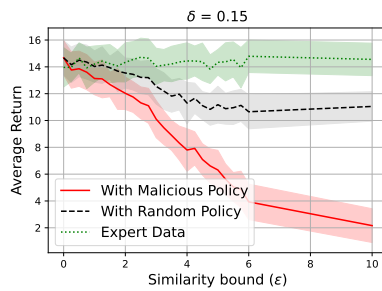


(d) Similarity bound ( $\epsilon$ ) of 2.5    (e) Similarity bound ( $\epsilon$ ) of 5.0    (f) Similarity bound ( $\epsilon$ ) of 10.0

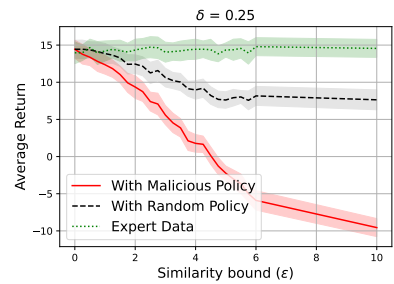
Figure 5.2: Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in CliffWorld as the mixing ratio ( $\delta$ ) varies for different fixed similarity bounds ( $\epsilon$ ). Each plot shows the average return of the policy trained on the reward inferred from a dataset contaminated with either malicious or random trajectories. Results are compared against expert-only data ( $\delta = 0$ ). Curves are averaged over 30 seeds, with shaded regions indicating standard deviation.



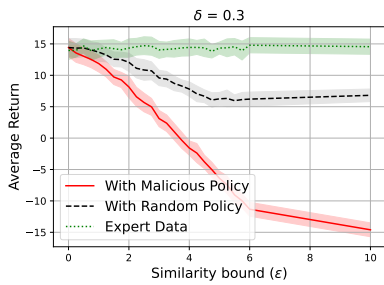
(a) Mixing ratio ( $\delta$ ) of 10%



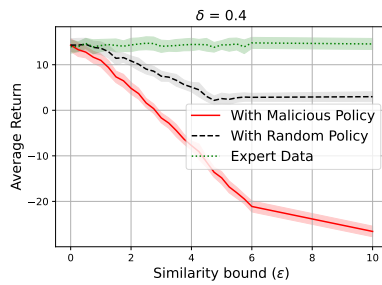
(b) Mixing ratio ( $\delta$ ) of 15%



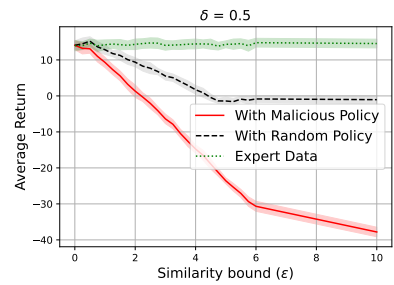
(c) Mixing ratio ( $\delta$ ) of 25%



(d) Mixing ratio ( $\delta$ ) of 30%



(e) Mixing ratio ( $\delta$ ) of 40%



(f) Mixing ratio ( $\delta$ ) of 50%

Figure 5.3: Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in CliffWorld as the similarity bound ( $\epsilon$ ) varies for different fixed mixing ratios ( $\delta$ ). Each plot shows the average return of the policy trained on the inferred reward. Curves are averaged over 30 seeds, with shaded regions indicating standard deviation.

2. Even a small fraction of adversarial data can severely compromise the performance and safety of the learned policy.
3. The short horizon and severe penalties characteristic of safety-critical tasks can amplify the effects of data poisoning.

These findings strongly suggest that for any real-world application of IRL where safety is a primary concern, developers must actively consider and defend against the possibility of systematic, adversarial manipulation of the demonstration data.

## 5.3 Case Study 2: Four Rooms

Moving from the safety-critical nature of CliffWorld, the Four Rooms environment allows us to study the impact of adversarial data on a task defined by exploration in a large state space. The primary challenge is not avoiding catastrophic penalty but efficiently navigating a complex world to reach a goal. The longer planning horizon of  $T = 50$  also provides more opportunity for an agent to recover from suboptimal moves.

### 5.3.1 Analysis of Adversarial Impact

The most immediate observation from the results in Four Rooms (Figure 5.4 and 5.5) is that the overall impact of adversarial trajectories is less severe than in CliffWorld. The deployed policy’s performance degrades more gracefully as the mixing ratio  $\delta$  and similarity bound  $\epsilon$  increase. This reduced sensitivity can be attributed to several factors:

- **Larger State Space:** The environment and its corresponding feature space are much larger. In this high-dimensional space, it is more difficult for an adversary to find specific feature deviations that can effectively and consistently mislead the IRL algorithm.
- **Longer Planning Horizon:** With a horizon of  $T = 50$ , the cumulative feature counts are less sensitive to single-state deviations. The impact of a few “bad” state visits can be diluted over the course of a long trajectory.
- **Milder Penalties:** The task is one of reachability, not strict safety. The penalty for entering the hazardous state ( $-3$ ) is mild compared to the catastrophic cliff penalty.

Despite this increased resilience, the adversarial threat remains. The malicious policy is still consistently more effective at degrading performance than the random baseline, although the gap between them is smaller than in CliffWorld. This confirms that the vulnerability of MaxEnt IRL to directed attacks is a general phenomenon, even if its severity is modulated by the environment.

### 5.3.2 Parameter Sensitivity in a Complex Environment

The effective scale of the similarity bound  $\epsilon$  also differs in this environment. Due to the higher-dimensional feature space and longer trajectories, the numerical values of feature count differences are naturally larger. This is why the experiments in Four Rooms explore a higher range for  $\epsilon$  (up to 30.0) compared to CliffWorld (up to 10.0). This observation highlights how the environment’s characteristics, such as its size and horizon, directly influence the dynamics of the feature-space adversarial attack.

### 5.3.3 Summary of Findings in Four Rooms

The Four Rooms case study provides a crucial counterpoint to the severe failures observed in CliffWorld. It demonstrates that the severity of the adversarial impact depends heavily on the structure of the environment and the task. Key takeaways include:

1. **Complexity as a Partial Defense:** Larger state spaces and higher-dimensional features can offer a degree of natural robustness against feature-space attacks, making it harder for an adversary to craft a maximally damaging trajectory.
2. **General Vulnerability Persists:** While the impact is less severe, the performance degradation from malicious data is still consistent and more significant than that caused by random data, reaffirming the fundamental nature of the vulnerability.
3. **Environment-Specific Attack Parameters:** The effective range and impact of hyperparameters like  $\epsilon$  are tied to the environment’s properties, emphasizing that adversarial robustness is not a one-size-fits-all problem.

Overall, while the Four Rooms environment is less susceptible to catastrophic failure, the results still clearly show that adversarial manipulation can lead to a quantifiable decrease in policy efficiency and goal-achievement.

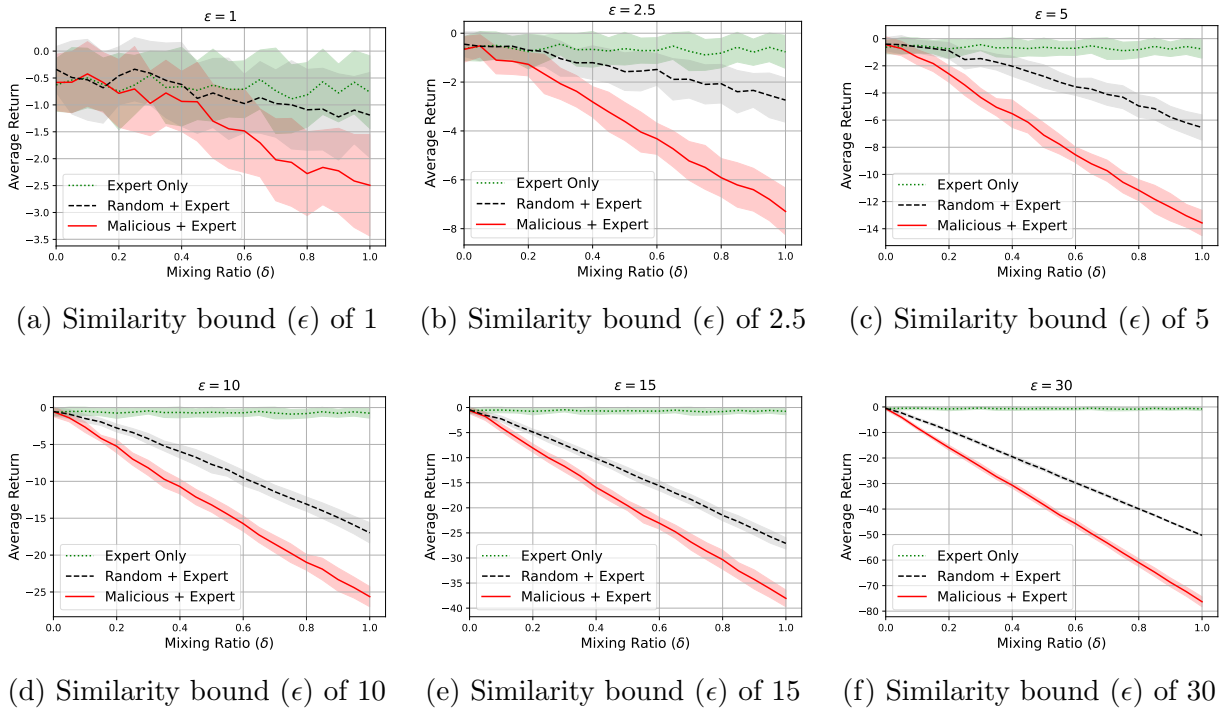
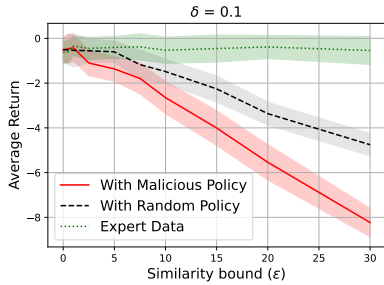
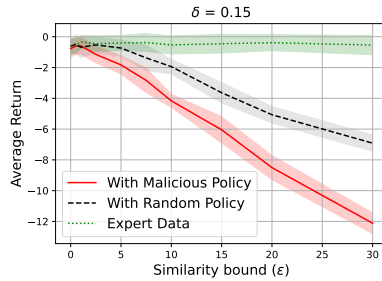


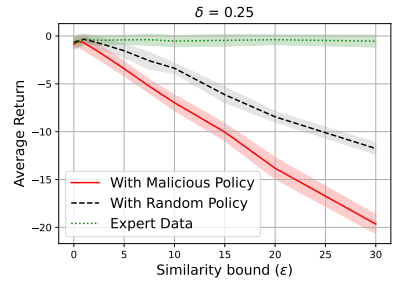
Figure 5.4: Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in Four Rooms as the mixing ratio ( $\delta$ ) varies for different fixed similarity bounds ( $\epsilon$ ). Each plot shows the average return of the policy trained on the reward inferred from a dataset contaminated with either malicious or random trajectories. Results are compared against expert-only data ( $\delta = 0$ ). Curves are averaged over 24 seeds, with shaded regions indicating standard deviation.



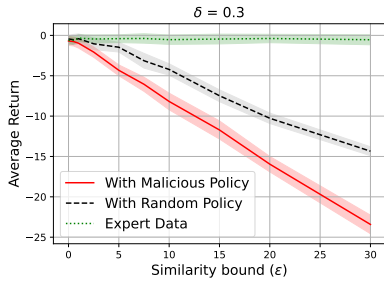
(a) Mixing ratio ( $\delta$ ) of 10%



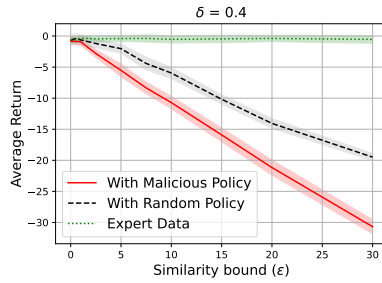
(b) Mixing ratio ( $\delta$ ) of 15%



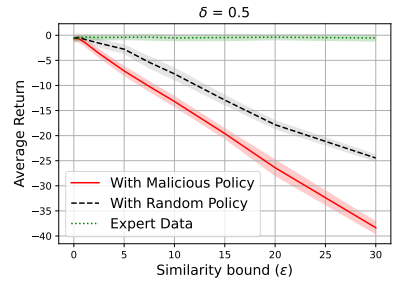
(c) Mixing ratio ( $\delta$ ) of 25%



(d) Mixing ratio ( $\delta$ ) of 30%



(e) Mixing ratio ( $\delta$ ) of 40%



(f) Mixing ratio ( $\delta$ ) of 50%

Figure 5.5: Performance of the deployed policy ( $\pi_{\text{deployed}}$ ) in Four Rooms as the similarity bound ( $\epsilon$ ) varies for different fixed mixing ratios ( $\delta$ ). Each plot shows the average return of the policy trained on the inferred reward. Results for malicious data are compared against a random data baseline and expert-only data. Curves are averaged over 24 seeds, with shaded regions indicating standard deviation.

## 5.4 Comparative Analysis and Discussion

A direct comparison of the CliffWorld and Four Rooms case studies yields a broader understanding of how mixed-policy data affects reward learning. This section synthesizes the findings from both environments and discusses the overarching principles they reveal about the vulnerabilities of IRL.

### 5.4.1 Synthesis of Environmental Effects

The most noticeable conclusion from our experiments is that while the vulnerability of MaxEnt IRL to adversarial data is a general phenomenon, its manifestation is highly **environment-dependent**. The contrasting results can be attributed to a few key properties:

- **Risk Profile:** CliffWorld is a high-risk environment with a brittle reward structure, where a single mistake leads to catastrophic penalty. Four Rooms is a lower-risk environment where mistakes lead primarily to inefficiency. Adversarial attacks are far more potent in high-risk settings.
- **State Space Complexity:** The small, simple state space of CliffWorld makes it easier for an adversary to manipulate feature counts in a targeted and impactful way. In the larger feature space of Four Rooms, the effect of any single adversarial trajectory is diluted, providing a form of natural robustness.
- **Planning Horizon:** The short horizon in CliffWorld ( $T = 10$ ) amplifies the impact of poor decisions, as there is little time for recovery. The longer horizon in Four Rooms ( $T = 50$ ) allows for more robust, averaged behavior to emerge over time.

The comparative heatmaps in Figure 5.6 provide a clear visual summary of the performance differences across environments. In CliffWorld (left), average return drops sharply for higher mixing ratios and larger similarity bounds, with a broad region of poor performance shown in darker shades. In contrast, the Four Rooms environment (right) maintains higher returns over a larger parameter range, with performance degradation appearing only at the upper extremes of both parameters. These patterns suggest that the impact of a feature-based adversarial attack is closely tied to the structure and dynamics of the environment.

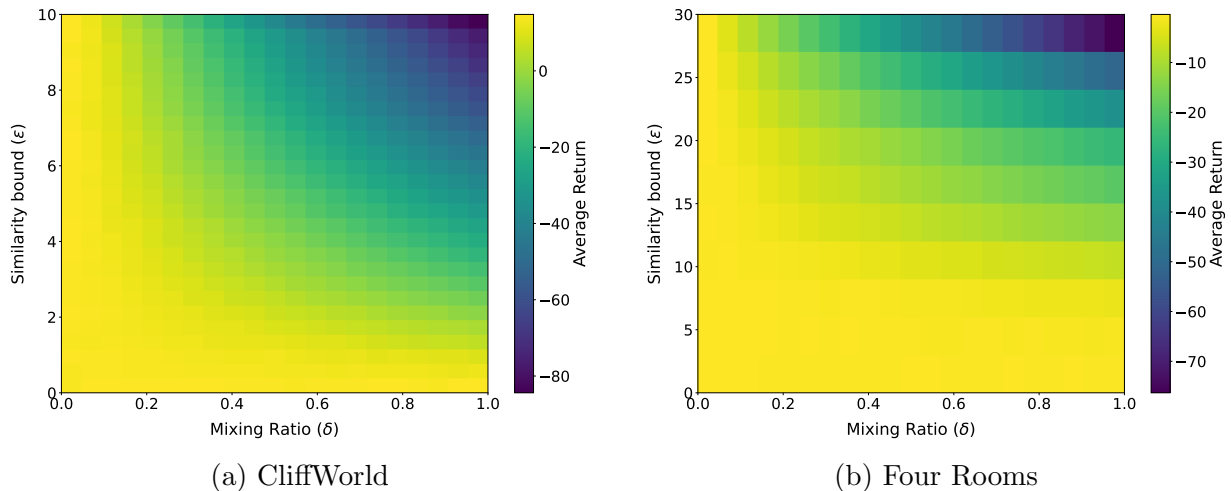


Figure 5.6: Side-by-side comparison of heatmaps showing the average return of the deployed policy as a function of the mixing ratio  $\delta$  (x-axis) and the similarity bound  $\epsilon$  (y-axis). The color scale represents policy performance, with yellow indicating high return and dark blue indicating low return. CliffWorld (a) exhibits substantial performance drops over a wide parameter range, while Four Rooms (b) remains robust except at the highest  $\delta$  and  $\epsilon$  values.

### 5.4.2 The Disparate Impact of Malicious and Random Data

Across both environments, a consistent pattern emerged: **adversarial trajectories are fundamentally more damaging than random trajectories**. Randomness introduces directionless noise, which a statistical method like MaxEnt IRL can often average out, especially at lower contamination levels. Malicious trajectories, however, represent a directed attack. They are optimized to find weaknesses in the learning algorithm’s assumptions and to push the inferred reward function in a specific, undesirable direction. Our results empirically confirm that this directed pressure is far more effective at corrupting the learning process. This implies that defending against such attacks requires more than simple data filtering; it requires methods that can identify and mitigate systematically misleading examples.

### 5.4.3 Chapter Summary and Key Findings

Our simulation studies have empirically characterized the vulnerabilities of Maximum Entropy IRL to mixed-policy data. The key findings of this chapter are:

1. **Fundamental Vulnerability:** MaxEnt IRL is susceptible to data poisoning from non-expert trajectories. This vulnerability can lead to flawed reward functions and, in turn, to suboptimal or unsafe deployed policies.
2. **Adversarial Magnification:** The negative impact is significantly magnified when the non-expert data is generated by a malicious adversary rather than a random process. A small fraction of adversarial data can cause a disproportionately large degradation in policy performance.
3. **Environment as a Modulator:** The degree of vulnerability is strongly modulated by the environment’s characteristics. High-risk environments with simple state spaces and short horizons, like CliffWorld, are extremely fragile. More complex environments with lower risk profiles, like Four Rooms, exhibit greater resilience but are not immune.
4. **Predictable Degradation:** The performance of the policy learned from the inferred reward tends to degrade in a predictable, often linear, fashion with increases in the proportion of adversarial data ( $\delta$ ) and the allowed feature deviation ( $\epsilon$ ).

In conclusion, this chapter has empirically validated the theoretical concerns that motivate this thesis. Through controlled experiments, we have shown that learning from demonstrations is not an inherently robust process when the demonstration data source is untrustworthy. These findings establish the practical need for developing new, more robust imitation learning techniques capable of defending against the vulnerabilities we have characterized.

# Chapter 6

## Conclusion and Future Work

This thesis investigated the adversarial vulnerability of Maximum Entropy [Inverse Reinforcement Learning \(IRL\)](#). Our goal was to understand how malicious demonstrations could corrupt a learned reward function and the resulting policy. This chapter consolidates our findings, discusses the study’s limitations, presents an extension of our work into adversarial detection, and outlines promising directions for future research.

### 6.1 Summary of Contributions

Our research provides a focused analysis of adversarial attacks on the MaxEnt [IRL](#) framework, demonstrated through experiments in the *Four Rooms* and *CliffWorld* gridworlds. The central contribution of this thesis is demonstrating that a small number of strategically crafted adversarial trajectories can severely degrade an agent’s learned policy. The magnitude of this degradation depends on the environment’s complexity and the nature of its risks.

Our key findings are summarized as follows:

- **High Impact in Critical Environments:** In *CliffWorld*, an environment with high-risk penalties, poisoning the expert data with just 10% adversarial trajectories caused a catastrophic decline in policy performance. The impact was comparable to that of a much larger (50%) set of random trajectories, which highlights the threat of targeted attacks in safety-critical systems.

- **Influence of Environmental Complexity:** In the *Four Rooms* environment, which has a larger state space and requires longer-horizon planning, performance degraded more gradually. The scale of this environment makes it more difficult for an adversary to craft demonstrations that are both deceptive and significantly alter the expert’s feature expectations. Nevertheless, the results confirm that the policy is still compromised.
- **Superiority of Strategic Adversaries:** We showed that our adversarial generation method, which directly targets the mechanics of MaxEnt IRL, causes a more severe performance drop than introducing random noise. By generating trajectories that are intentionally misleading yet remain within a specified similarity bound ( $\epsilon$ ) of the expert’s feature expectations, the adversary effectively poisons the learning process.

Collectively, these findings emphasize the vulnerability of MaxEnt IRL to strategic manipulation and highlight the need for robust learning algorithms, particularly for agents deployed in real-world systems.

## 6.2 Limitations of the Current Study

While our findings illuminate a critical vulnerability, it is essential to acknowledge the limitations that frame our study. These boundaries define the scope of our conclusions and provide a clear context for interpreting the results.

### 6.2.1 Assumption of a Fully-Informed Adversary

A significant limitation is the assumption that the adversary has full knowledge of the expert’s demonstration set and the environment’s dynamics. This “white-box” attack model represents a worst-case scenario and may not be realistic in many applications. A “black-box” adversary with only partial information would face a more difficult task. Our study thus establishes a baseline for vulnerability, but future work should explore more constrained threat models.

### 6.2.2 Fixed Planning Horizon

Our experiments assume that all trajectories have a fixed, finite length. While many real-world tasks can have variable durations, this was a deliberate choice to ensure a rigorous

evaluation.

Variable-horizon environments can provide an implicit informational side-channel about the task’s reward function. Episode termination, whether from success or failure, can offer a strong signal that allows an agent to achieve high performance by simply learning to manipulate the episode duration. This phenomenon does not generalize to complex problems lacking such clear terminal cues [46]. To ensure our results reflect the agent’s ability to learn from the demonstrations rather than from confounding environmental cues, we adopt the recommended practice of using fixed-horizon settings [23, 46].

### 6.2.3 Computational Complexity and Scalability

The optimization process used to generate adversarial trajectories is computationally intensive and scales with the size of the state space. This cost limited our analysis to environments with discrete and relatively small state-action spaces. Applying our methodology directly to problems with large or continuous state spaces would be computationally infeasible with current methods.

### 6.2.4 Focus on the Maximum Entropy IRL Framework

This study has focused exclusively on the MaxEnt IRL framework. Other IRL paradigms, such as Bayesian IRL or Generative Adversarial Imitation Learning (GAIL), may possess different robustness characteristics. Our findings of vulnerability cannot be automatically generalized to these other methods.

### 6.2.5 Discrete State and Action Spaces

Our analysis was conducted entirely within discrete state and action spaces. Many real-world applications, particularly in robotics, involve continuous domains. Extending this work to continuous spaces would require function approximators, such as neural networks, which could introduce different dynamics and vulnerabilities.

## 6.3 Extension: Unsupervised Detection of Adversarial Trajectories

Given the demonstrated vulnerability of MaxEnt IRL, a natural line of defense is to detect and filter malicious data. This section details our investigation into whether unsupervised learning methods can distinguish between expert and adversarial trajectories. The central hypothesis is that malicious and expert trajectories possess structurally different latent features that can be identified through representation learning.

### 6.3.1 Learning Trajectory Embeddings with a VAE

We frame detection as an anomaly detection problem. The core idea is to learn a low-dimensional vector representation, or embedding, for each trajectory using a Variational Autoencoder (VAE) [44]. In this learned latent space, we expect trajectories from different sources to form distinct clusters.

Our model is a  $\beta$ -VAE with an LSTM-based architecture. The encoder, a two-layer LSTM, processes a sequence of state-action pairs and maps the entire trajectory to a latent distribution (mean  $\mu$  and log-variance  $\log \sigma^2$ ). The decoder, an LSTMCell, then attempts to reconstruct the original action sequence given a sample from this latent distribution and the true state sequence.

The model was trained for 1000 epochs for *Four Rooms* and 100 for *CliffWorld* using the  $\beta$ -VAE objective function with a  $\beta$  value of 0.1. This encourages the model to learn a more disentangled latent space. For the *Four Rooms* environment, we used a hidden dimension of 128 and a latent dimension of 64. For the *CliffWorld* environment, the latent dimension was increased to 128 to better capture its dynamics. The convergence of the training process for both environments is shown in Figure 6.1.

### 6.3.2 Visualization and Cluster Analysis

To interpret the learned embeddings, we used two dimensionality reduction techniques: Principal Component Analysis (PCA) [50] for a linear projection and t-Distributed Stochastic Neighbor Embedding (t-SNE) [49] for a non-linear visualization. These methods help us see if the expert (red) and adversarial (blue shades) trajectories are separable in the latent space.

### 6.3.3 Preliminary Findings and Discussion

Our results show that learning trajectory embeddings is a promising direction for adversarial detection, though its effectiveness varies by environment.

In the *Four Rooms* environment, the separation is remarkably clear (Figure 6.2 and 6.3). The PCA plot shows the expert trajectories forming a distinct cluster on the left, well-separated from the adversarial data. Furthermore, the adversarial cluster exhibits a clear gradient, with lighter blue points (lower  $\epsilon$ ) positioned closer to the expert cluster and darker points (higher  $\epsilon$ ) moving progressively farther away. The t-SNE visualization makes this separation even more stark, revealing tight, well-defined clusters for the expert data and for different levels of adversarial attack. This strong result suggests that in environments with clear goal-directed structures, the VAE can effectively learn to distinguish legitimate and malicious behavior.

In the *CliffWorld* environment, detection is more challenging (Figure 6.4 and 6.5). The PCA plot shows a general separation, but with significant overlap between expert and adversarial points. The t-SNE plot improves the visualization by forming more defined clusters, but the boundary is less clear than in *Four Rooms*. The expert trajectories themselves are fragmented into several smaller groups, and adversarial points are interspersed nearby. This suggests that in environments where optimal and suboptimal behaviors are more nuanced, the latent representations are less easily separable.

This investigation reinforces a central conclusion of our thesis: defending against adversarial demonstrations is a difficult problem. While detection offers a potential layer of security, its reliability is not guaranteed and depends heavily on the task. This motivates the pursuit of IRL algorithms that are intrinsically robust, rather than reliant on a separate, potentially fallible, filtering step.

## 6.4 Future Research Directions

The insights and limitations of this thesis point to several important avenues for future work.

### 6.4.1 Developing Inherently Robust IRL Algorithms

The most critical research direction is the development of IRL algorithms that are inherently robust to adversarial data. Rather than relying on data filtering, these methods

would be designed to be resilient. Promising approaches include:

- **Robust Optimization:** Reformulating the MaxEnt IRL objective within a minimax framework, where the agent optimizes for a reward function that performs well even under a worst-case choice of demonstrations.
- **Data-Centric Regularization:** Incorporating techniques from robust statistics to automatically down-weight the influence of outlier trajectories on the learned reward function.
- **Ensemble Methods:** Learning an ensemble of reward functions from different subsets of the demonstration data, where the final policy is derived from a consensus that could mitigate the impact of a few malicious examples.

## 6.4.2 Improving Unsupervised Detection Methods

Our preliminary work on unsupervised detection can be extended significantly. Future efforts could focus on developing more sophisticated representation learning techniques tailored for detecting adversarial intent. This includes exploring contrastive learning methods to more forcefully separate expert and non-expert behaviors in the embedding space, or designing architectures that explicitly model trajectory sub-goals to better identify anomalous behavior.

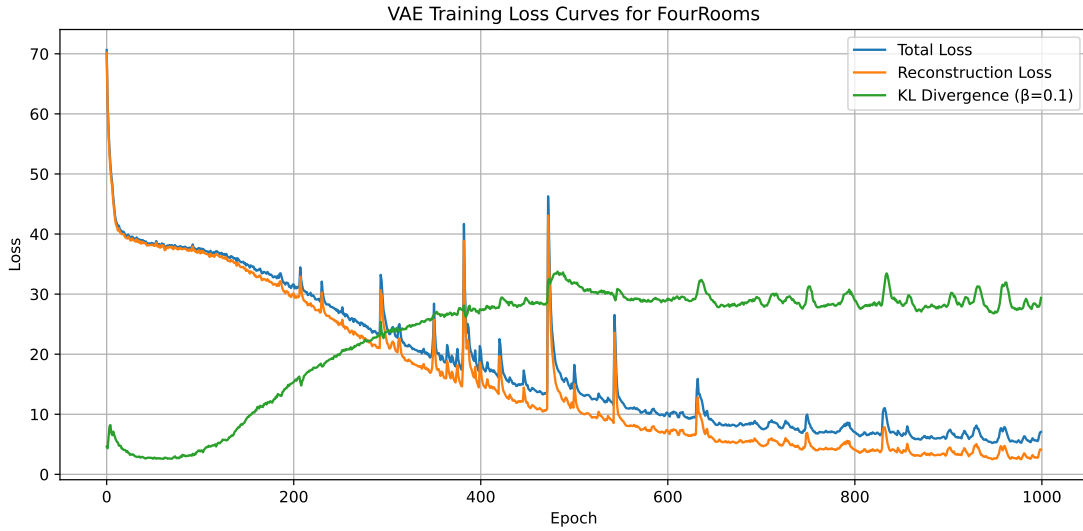
## 6.4.3 Extension to Multi-Agent Scenarios

Applying these concepts to Multi-Agent IRL (MA-IRL) opens up a rich problem space. In a cooperative setting, one agent could be a “traitor,” providing demonstrations that sabotage the team goal. In a competitive setting, an adversary might “help” an opponent learn a suboptimal strategy that is easily exploitable. Investigating these threats is foundational for building secure and reliable multi-agent systems.

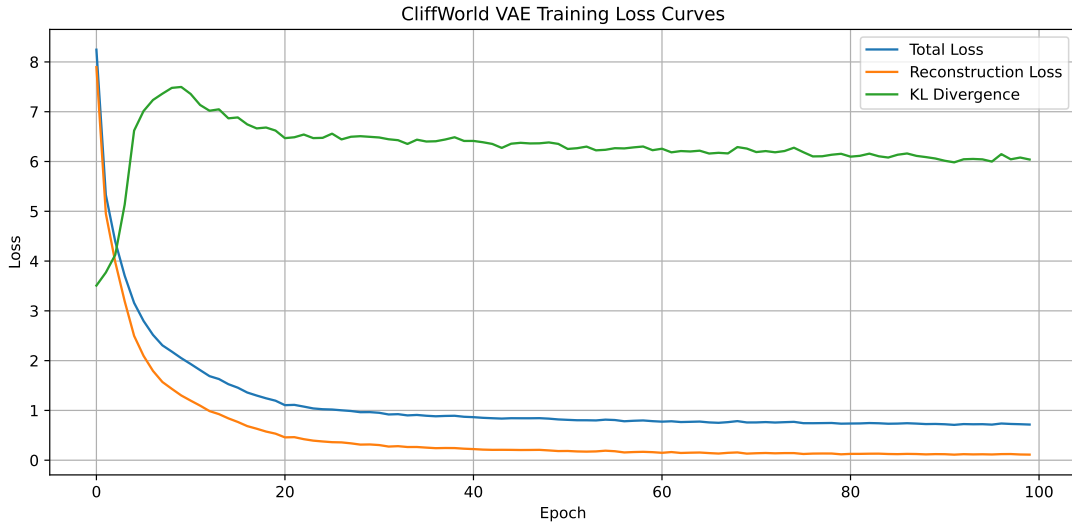
## 6.5 Concluding Remarks

This thesis has explored the critical domain of adversarial security in IRL. We have demonstrated that the Maximum Entropy IRL algorithm is vulnerable to purposeful manipulation. By crafting plausible yet deceptive demonstrations, an adversary can corrupt the

learned reward function and degrade an agent’s policy, with severe consequences in safety-critical contexts. We extended this work to explore unsupervised detection of such attacks, and our findings show that while this is a promising approach, its success is not universal and highlights the inherent difficulty of the problem. As autonomous systems are deployed in increasingly high-stakes environments, the assumption of benign training data is no longer tenable. This work is a step toward a more security-conscious paradigm for imitation learning, one that acknowledges potential threats and actively seeks to build systems that are not just intelligent, but also trustworthy and resilient.



(a) *Four Rooms* Environment



(b) *CliffWorld* Environment

Figure 6.1: VAE training loss curves for both environments, showing the total loss, reconstruction loss, and KL divergence over 1000 epochs (Four Rooms) and 100 epochs (CliffWorld). The stable convergence indicates that the model successfully learned to compress and reconstruct the trajectories.

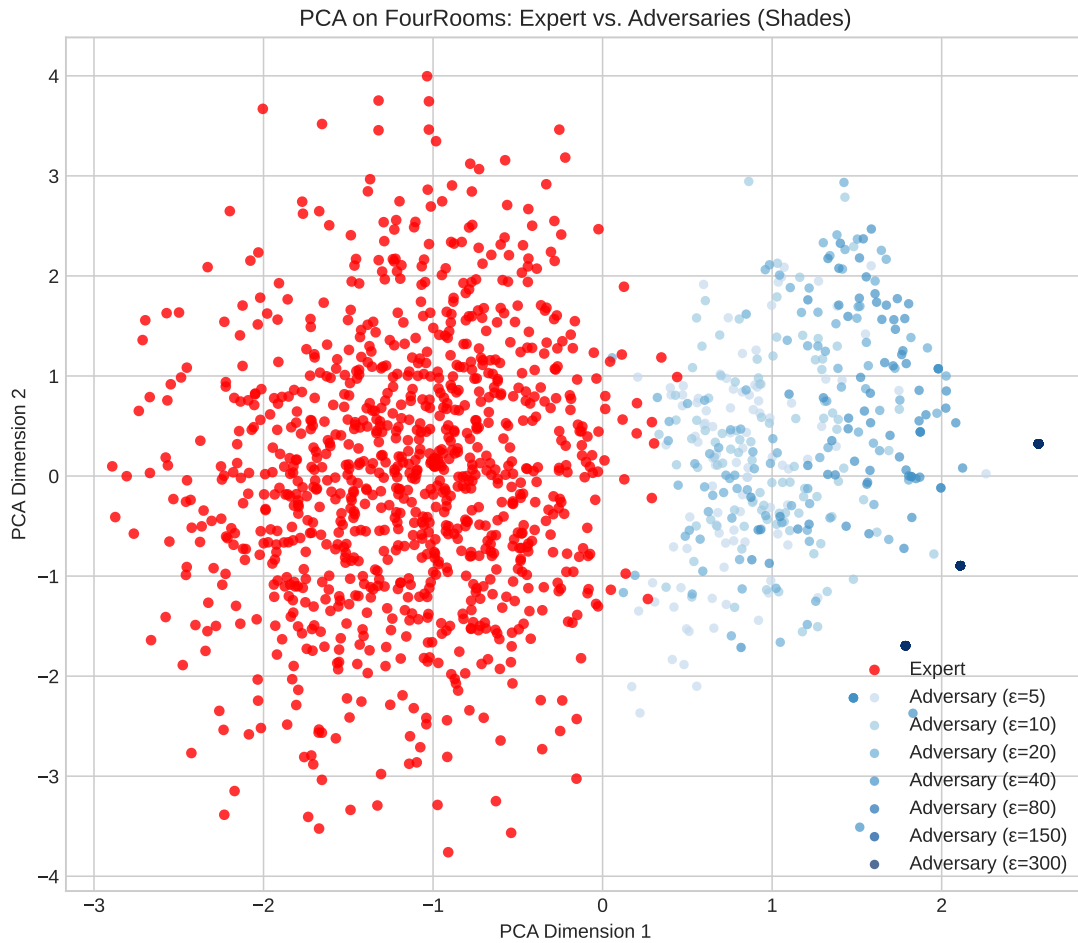


Figure 6.2: PCA Visualization of trajectory embeddings for the *Four Rooms* environment. Expert trajectories are shown in red, while adversarial trajectories are shown in shades of blue, with darker shades corresponding to a higher attack budget  $\epsilon$ .

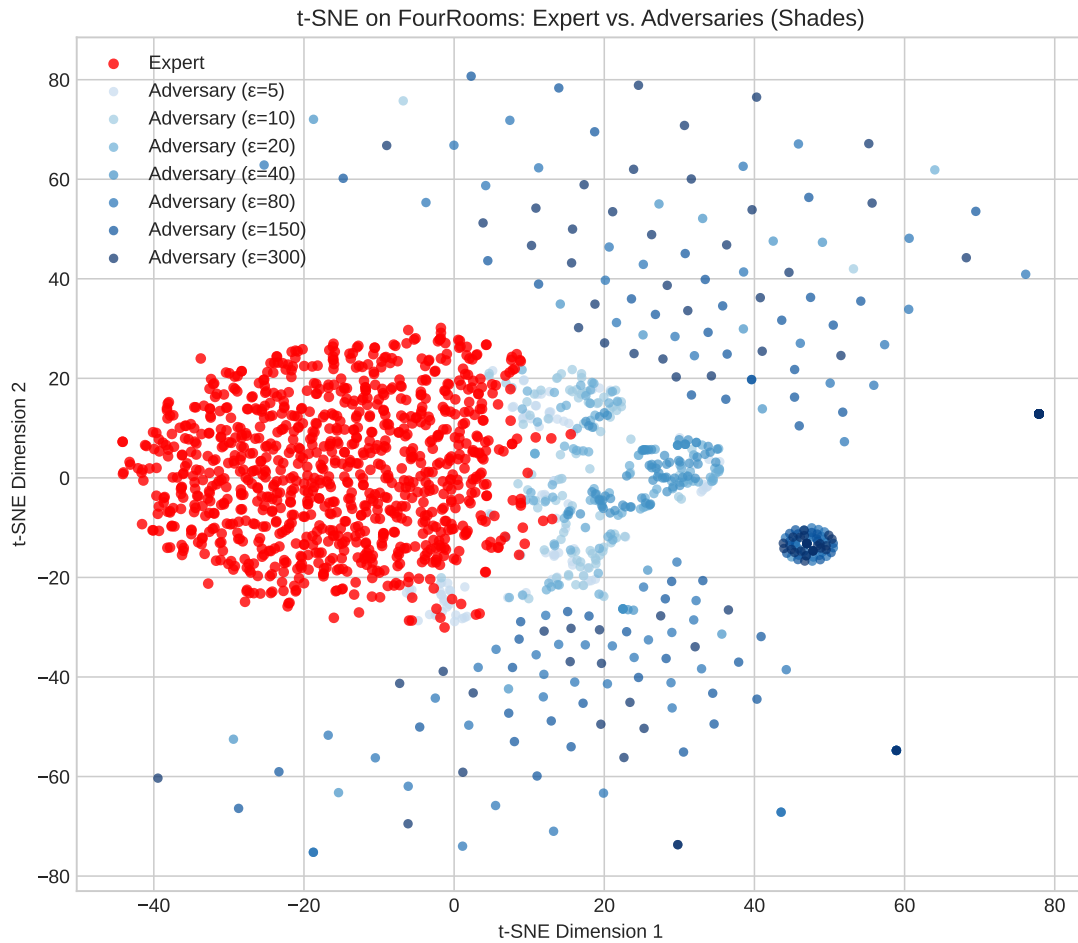


Figure 6.3: t-SNE Visualization of trajectory embeddings for the *Four Rooms* environment. Expert trajectories are shown in red, while adversarial trajectories are shown in shades of blue, with darker shades corresponding to a higher attack budget  $\epsilon$ .

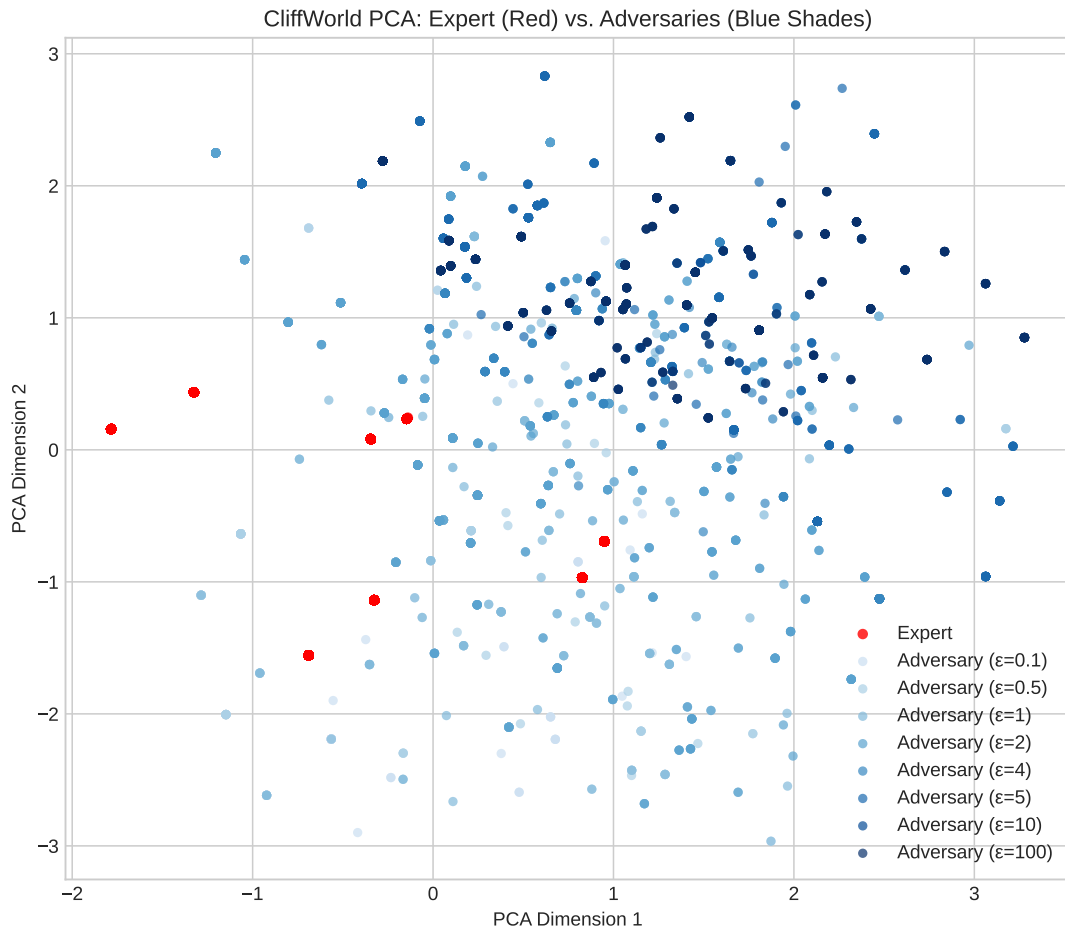


Figure 6.4: PCA Visualization of trajectory embeddings for the *CliffWorld* environment. Separation between expert (red) and adversarial (blue) trajectories is visible but less distinct than in *Four Rooms*, highlighting the increased difficulty of detection in this safety-critical setting.

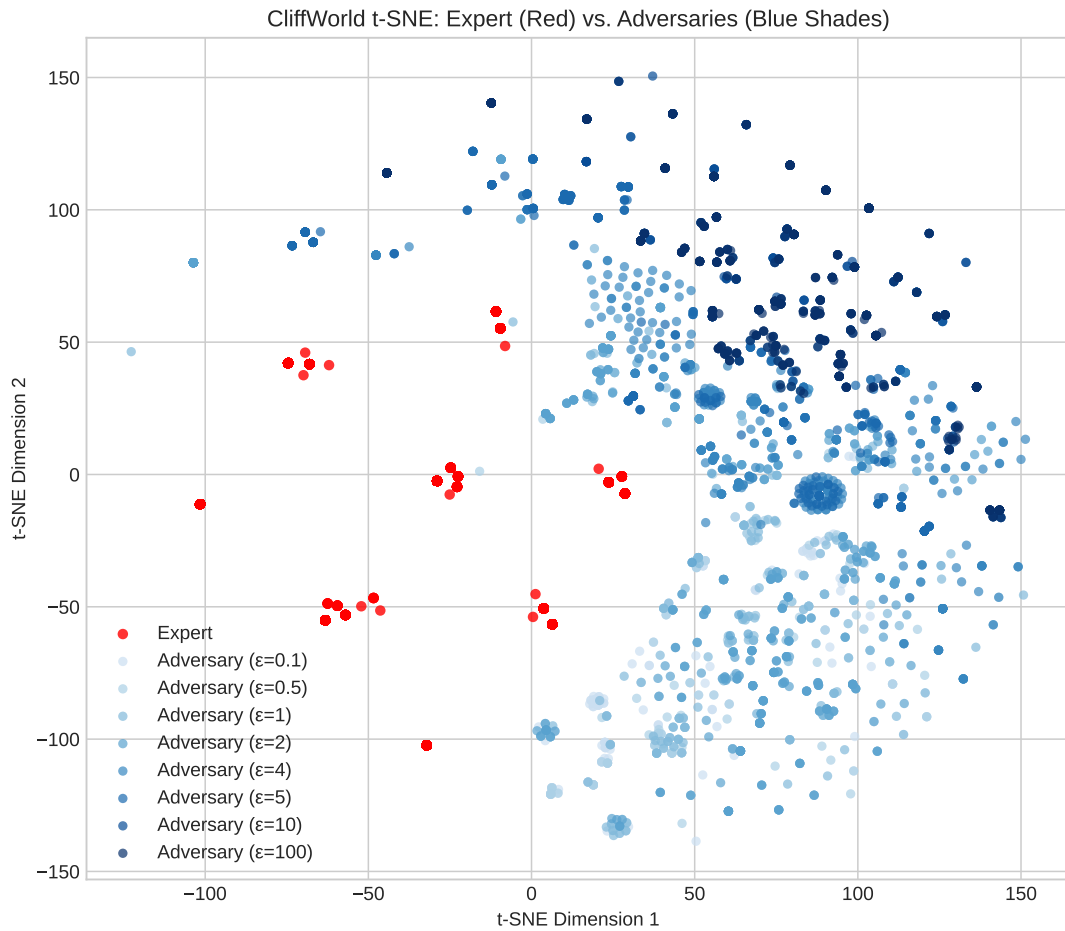


Figure 6.5: t-SNE Visualization of trajectory embeddings for the *CliffWorld* environment. Separation between expert (red) and adversarial (blue) trajectories is visible but less distinct than in *Four Rooms*, highlighting the increased difficulty of detection in this safety-critical setting.

# References

- [1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.
- [2] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1948.
- [3] Arezoo Alipanah and Yash Pant. Generating Malicious Demonstration Policies to Exploit Vulnerabilities in Inverse Reinforcement Learning. *Proceedings of the Canadian Conference on Artificial Intelligence*, May 19 2025. <https://caiac.pubpub.org/pub/kh0yxczc>.
- [4] Kareem Amin, Nan Jiang, and Satinder Singh. Repeated inverse reinforcement learning. *Advances in neural information processing systems*, 30, 2017.
- [5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- [6] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.
- [7] Mark Beliaev and Ramtin Pedarsani. Inverse reinforcement learning by estimating expertise of demonstrators. *arXiv preprint arXiv:2402.01886*, 2024.
- [8] Mark Beliaev, Andy Shih, Stefano Ermon, Dorsa Sadigh, and Ramtin Pedarsani. Imitation learning by estimating expertise of demonstrators. In *International Conference on Machine Learning*, pages 1732–1748. PMLR, 2022.
- [9] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

- [10] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.
- [11] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML’12, page 1467–1474, Madison, WI, USA, 2012. Omnipress.
- [12] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2154–2156, 2018.
- [13] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.
- [14] Daniel Brown and Scott Niekum. Efficient probabilistic performance bounds for inverse reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [15] Daniel S Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. In *Conference on Robot Learning*, pages 362–372. PMLR, 2018.
- [16] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.
- [17] Mine Melodi Caliskan, Saeed Ghoorchian, and Setareh Maghsudi. Robust inverse reinforcement learning under state adversarial perturbations.
- [18] Pamela Carreno-Medrano, Stephen L Smith, and Dana Kulić. Joint estimation of expertise and reward preferences from human demonstrations. *IEEE Transactions on Robotics*, 39(1):681–698, 2022.
- [19] Pablo Samuel Castro, Shijian Li, and Daqing Zhang. Inverse reinforcement learning with multiple ranked experts. *arXiv preprint arXiv:1907.13411*, 2019.
- [20] Kartik Chandra, Tzu-Mao Li, Joshua Tenenbaum, and Jonathan Ragan-Kelley. Acting as inverse inverse planning. In *Acm siggraph 2023 conference proceedings*, pages 1–12, 2023.

- [21] Kartik Chandra, Tzu-Mao Li, Joshua B Tenenbaum, and Jonathan Ragan-Kelley. Storytelling as inverse inverse planning. *Topics in Cognitive Science*, 16(1):54–70, 2024.
- [22] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36:73383–73394, 2023.
- [23] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- [24] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2006.
- [25] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.
- [26] Arnaud Fickinger, Simon Zhuang, Dylan Hadfield-Menell, and Stuart Russell. Multi-principal assistance games. *arXiv preprint arXiv:2007.09540*, 2020.
- [27] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.
- [28] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.
- [29] Zichang Ge, Changyu Chen, Arunesh Sinha, and Pradeep Varakantham. On learning informative trajectory embeddings for imitation, classification and regression. *arXiv preprint arXiv:2501.09327*, 2025.
- [30] Mahsa Ghasemi, Abolfazl Hashemi, Haris Vikalo, and Ufuk Topcu. No-regret learning with high-probability in adversarial markov decision processes. In *Uncertainty in Artificial Intelligence*, pages 992–1001. PMLR, 2021.

- [31] Adam Gleave, Mohammad Taufeque, Juan Rocamonde, Erik Jenner, Steven H Wang, Sam Toyer, Maximilian Ernestus, Nora Belrose, Scott Emmons, and Stuart Russell. imitation: Clean imitation learning implementations. *arXiv preprint arXiv:2211.11972*, 2022.
- [32] Adam Gleave and Sam Toyer. A primer on maximum causal entropy inverse reinforcement learning. *arXiv preprint arXiv:2203.11409*, 2022.
- [33] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- [34] Ziwei Guan, Tengyu Xu, and Yingbin Liang. When will generative adversarial imitation learning algorithms attain global convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1117–1125. PMLR, 2021.
- [35] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.
- [36] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29, 2016.
- [37] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.
- [38] Ronald A Howard. Dynamic programming and markov processes. 1960.
- [39] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.
- [40] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.
- [41] Yiding Jiang, Evan Liu, Benjamin Eysenbach, J Zico Kolter, and Chelsea Finn. Learning options via compression. *Advances in Neural Information Processing Systems*, 35:21184–21199, 2022.
- [42] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

- [43] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International conference on machine learning*, pages 2469–2478. PMLR, 2018.
- [44] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [45] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- [46] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations (ICLR)*, 2019.
- [47] John Lafferty, Andrew McCallum, Fernando Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Icml*, volume 1, page 3. Williamstown, MA, 2001.
- [48] Sergey Levine. Cs285 deep reinforcement learning, lecture 20. <https://rail.eecs.berkeley.edu/deeprlcourse/deeprlcourse/static/slides/lec-20.pdf>, 2022. Accessed: 2025-07-06.
- [49] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [50] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.
- [51] Tien Mai, Quoc Phong Nguyen, Kian Hsiang Low, and Patrick Jaillet. Inverse reinforcement learning with missing data. *arXiv preprint arXiv:1911.06930*, 2019.
- [52] Alan S Manne. On the job-shop scheduling problem. *Operations research*, 8(2):219–223, 1960.
- [53] Lev McKinney, Yawen Duan, David Krueger, and Adam Gleave. On the fragility of learned reward functions. In *NeurIPS 2022 Workshop on ML Safety*, 2022. arXiv:2301.03652.
- [54] Alberto Maria Metelli, Filippo Lazzati, and Marcello Restelli. Towards theoretical understanding of inverse reinforcement learning. In *International Conference on Machine Learning*, pages 24555–24591. PMLR, 2023.

- [55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [56] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.
- [57] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.
- [58] Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.
- [59] Ritesh Noothigattu, Tom Yan, and Ariel D Procaccia. Inverse reinforcement learning from like-minded teachers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9197–9204, 2021.
- [60] Kunal Pattanayak, Vikram Krishnamurthy, and Christopher Berry. Inverse-inverse reinforcement learning. how to hide strategy from an adversarial inverse reinforcement learner. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3631–3636. IEEE, 2022.
- [61] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pages 2817–2826. PMLR, 2017.
- [62] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [63] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.
- [64] Siddharth Reddy, Anca Dragan, Sergey Levine, Shane Legg, and Jan Leike. Learning human objectives by evaluating hypothetical behavior. In *International Conference on Machine Learning*, pages 8020–8029. PMLR, 2020.
- [65] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

- [66] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.
- [67] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.
- [68] Avi Singh, Eric Jang, Alexander Irpan, Daniel Kappler, Murtaza Dalal, Sergey Levine, Mohi Khansari, and Chelsea Finn. Scalable multi-task imitation learning with autonomous improvement. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2167–2173. IEEE, 2020.
- [69] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.
- [70] Prasanth Sengadu Suresh and Prashant Doshi. Marginal map estimation for inverse rl under occlusion with observer noise. In *Uncertainty in Artificial Intelligence*, pages 1907–1916. PMLR, 2022.
- [71] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.
- [72] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.
- [73] Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.
- [74] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.
- [75] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4950–4957, 2018.
- [76] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.
- [77] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in neural information processing systems*, 33:21024–21037, 2020.

- [78] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.
- [79] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

# Glossary

**Covariate Shift** A distributional mismatch between training and deployment, where small errors during imitation cause the agent to visit unseen states, leading to error accumulation [7](#)

**Data Poisoning** An adversarial attack where malicious samples are injected into the training data to corrupt the learned model; can be targeted or indiscriminate [ix](#), [11–14](#)

**MDP** Markov Decision Process, a mathematical framework for modeling sequential decision-making, defined by the tuple  $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ , where  $\mathcal{S}$  is the set of states,  $\mathcal{A}$  the set of actions,  $P$  the state transition probability function,  $R$  the reward function, and  $\gamma \in [0, 1)$  the discount factor [4](#)