

Crowd-in-the-Loop Reinforcement Learning

by

Zhefan Ye

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2026

Doctoral Committee:

Professor Benjamin Kuipers, Chair
Associate Professor Nikola Banovic
Professor Vineet R. Kamat
Professor Satinder Singh

Zhefan Ye

zhefanye@umich.edu

ORCID iD: 0000-0002-4706-5851

© Zhefan Ye 2026

DEDICATION

Dedicated to Chuyi, my parents, and my son.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Benjamin Kuipers, for his guidance and patience throughout my long and winding Ph.D. journey. His insights and encouragement were instrumental in helping me finish this dissertation. I am also grateful to my committee members, Professor Satinder Singh, Professor Nikola Banovic, and Professor Vineet Kamat for their support and feedback.

I owe a great deal to my colleagues. Specifically, I want to thank John Joon Young Chung, Anthony Liu, Jean Song, Jonathan Kummerfeld, Somayeh Molaei, Stephanie Shepard, and Zhiqiang Sui for countless whiteboard sessions, coding marathons, and discussions.

To my wife, Chuyi, thank you for putting up with me all these years. Your patience and caring for me during the long nights of writing and the stress of the defense were more important than I can express. I could not have reached this milestone without you. And to my parents, I owe everything to you.

Finally, a lighthearted but necessary thanks to my cat, Tie Tie, for the late-night company and for reminding me to take breaks, even if just to refill the food bowl.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	vii
LIST OF TABLES	xi
ABSTRACT	xii

CHAPTER

1 Introduction	1
1.1 Part I: Human-aided reward design	3
1.2 Part II: Relevance-driven efficient learning	4
1.3 Contributions	4
2 Background	6
2.1 Reinforcement Learning Fundamentals	6
2.1.1 Reward Function	7
2.2 Human-in-the-Loop Reinforcement Learning	7
2.2.1 Learning from Demonstration	7
2.2.2 Learning from Feedback	8
2.2.3 Learning from Outcomes	8
2.3 Relevance and Efficiency	9
2.3.1 Temporal Credit Assignment: Retrospective Relevance	9
2.3.2 Representation Learning: Prospective Relevance	9
2.4 Scaling Human Insight	9
2.4.1 Crowdsourcing	10
2.4.2 Weak Supervision	10
I Human-aided Reward Design	11
3 Rethinking Human-Elicited Reward Specification	12
3.1 Motivation	12
3.2 Limitations of Traditional Approaches	13

3.2.1	Trace Level Feedback	14
3.2.2	State Level Feedback	14
3.2.3	Sub-State Level Feedback	14
3.3	Three Phases of Reward Creation	15
3.4	Shifting the Perspective: From Needs to Knowledge	16
3.5	Conclusion	17
4	Improving Reward Specification through Identifying Causation of Rewards	18
4.1	Motivation	18
4.2	Approach	20
4.2.1	Constructed Reward Function via Pattern Sequences	20
4.3	Error Characterization Study	21
4.3.1	Setup	22
4.3.2	Methods and Results	22
4.3.3	Design Goals	27
4.4	RECEPT	27
4.4.1	Generating Reward Signals via Indicator Tracks	27
4.4.2	Querying for Indicator Patterns	28
4.4.3	Crowdsourcing Workflow	30
4.5	Experiments	34
4.5.1	Data Collection	34
4.5.2	Results	35
4.6	Discussion	39
4.7	Limitations and Future Work	40
4.8	Summary	41
II	Relevance-driven Efficient Learning	43
5	Improving Sample Efficiency by Identifying Relevance	48
5.1	Motivation	48
5.2	Related Work	49
5.2.1	Human-in-the-loop RL	50
5.2.2	Crowd-based Data Augmentation	50
5.3	Approach	51
5.3.1	Selective Memory Learning (SML)	51
5.3.2	Montage	53
5.4	Experimental Setup	54
5.4.1	Game Environment	54
5.4.2	Credit Assignment Framework	55
5.5	Results	55
5.6	Crowd Case Study	59
5.6.1	Protocol	60
5.6.2	Survey Questions	60
5.6.3	Results	61

5.6.4	Understanding Crowd Workers' Strategies	62
5.7	Discussion	63
5.7.1	Key Contributions and Performance Improvements	63
5.7.2	Human-powered Relevance Assessment Validation	63
5.7.3	Limitations and Practical Considerations	64
6	CARAFE: Crowd-Assembled Relevance Anticipation via (Estimated) Functional Equivalence	65
6.1	Motivation	65
6.2	Related Work	68
6.2.1	Representation Learning for Reinforcement Learning	68
6.2.2	Human-in-the-Loop Reinforcement Learning	69
6.2.3	Metric Learning from Human Feedback	69
6.2.4	Positioning CARAFE	69
6.3	Learning Representations from Relevance Priors	70
6.3.1	From Functional Similarity to Prospective Relevance	70
6.3.2	Eliciting Similarity via Triplet Comparison and Triplet Loss	71
6.3.3	A Crowdsourcing Workflow for Elicitation	72
6.4	Experimental Design and Research Questions	74
6.4.1	RQ1: Can Triplet-Induced Encoders Accelerate RL Training?	75
6.4.2	RQ2: Can Humans Provide Coherent Signals for Relevance-Based Similarity?	78
6.5	Results	79
6.5.1	Validation of Triplet-Induced Representation (RQ1)	79
6.5.2	Research Question 2 (RQ2)	83
6.6	Discussion	90
6.6.1	Characterizing Human Relevance Assessment	90
6.6.2	Limitations and Future Directions	91
6.7	Summary	92
7	Conclusion and Future Work	94
7.1	Summary of Contributions	94
7.1.1	The Unifying Theme: Leveraging Collective Intuition	95
7.2	The Intuition Frontier in the Age of Foundation Models	96
7.3	Limitations	97
7.4	Future Directions	97
7.5	Closing Thought	99
	BIBLIOGRAPHY	100

LIST OF FIGURES

FIGURE

1.1	This dissertation is composed of two parts. Part I addresses the challenge of reward specification by introducing a system to elicit reward indicator patterns from people to provide consistent feedback for RL agents during training. Part II tackles the challenge of the lack of relevance guidance by presenting a human-guided credit assignment system and a method for capturing similarity-induced representations through triplet state comparisons.	2
3.1	Comparison of reward creation approaches mapping to granularity levels, generalization capability, and the three phases of reward creation (design, recognize, and scale). Approaches are categorized by their trade-offs between specificity and adaptability across different problem domains.	13
4.1	Overview of RECEPT. RECEPT first collects state samples from the environment and elicits indicator patterns from the crowd using our CVR workflow. Then, RECEPT generates a reward function based on PRFG to detect the indicator patterns in order to produce reward signals as the agent interacts with the environment.	20
4.2	Learning curves of different false negative rate (FNR) values	23
4.3	Learning curves of different false positive rate (FPR) values	24
4.4	Learning curves of different duplicate values	25
4.5	Learning curves of different delay values	26
4.6	Tracking indicator patterns to form reward events. At each timestep, template matching detects patterns in the current state. The numbers below the images (1100, 1200) represent the cumulative game score , which remains constant after the initial reward event. Note that an indicator track can consist of visually distinct patterns (e.g., Track step 1 vs. Track step 2) as long as they are spatially and temporally contiguous. This allows RECEPT to track a semantic event (like an explosion) that changes appearance over time without triggering duplicate rewards.	28
4.7	The Collect-View-Refine workflow. The Collect step first elicits reward events that may contain indicator patterns from crowd workers and then requests workers to crop indicator patterns within the aggregated reward events. The View step subsequently uses the elicited patterns to search for examples of such patterns in the collected experience. Then, the Refine step asks workers to decide whether to keep indicator patterns using majority voting based on the result of the View step. The final output of the CVR workflow is a set of indicator patterns.	31

4.8	Our user interface for the Collect-View-Refine workflow. Left: a worker can click through a gameplay image sequence and create a reward event by clicking the “Set Start” and “Set End” button. The reward event is visualized below the scroll bar. If the worker does not observe any reward, she/he can check “There is no reward”. Center: a worker is first asked to select an image region from a reward event clip for indicator patterns. After selection, the worker needs to verify if the cropped image region contains an indicator pattern. Right: the Refine step asks a worker to determine whether the elicited pattern is correct based on the example found by the View step.	32
4.9	The blue curves represent the learning performance of A2C agents trained by the oracle reward functions. The green curves represent the learning performance of A2C agents trained by reward functions generated by RECEPT. Each curve is the averaged over three training instances, and the shaded area represents one standard deviation.	38
4.10	Learning performance of an A2C agent trained by RECEPT in Demon Attack. Each colored curve segment represents a round of training instance in which RECEPT elicits a set of indicator patterns from crowd workers to generate reward functions and subsequently produces reward signals to train the A2C agent. The left figure shows the learning curves plotted using the original reward value. The right figure shows the result of the same training instances but using the number of generated indicator tracks (reward events).	39
5.1	Montage: Human-powered, relevance-driven credit assignment via Selective Memory Learning (SML).	49
5.2	An RL agent first collects a trajectory from the environment. Human annotators then review each step in the trajectory and remove steps that are irrelevant to the eventual outcome. The updated trajectory is then used to update the policy.	52
5.3	DotShooter game environments. Left: One-target game. Right: Three-target game. The red dot on the left of each figure represent the agent, the white dot is the pellet moving from left to right, and the red dot on the right is the <i>true</i> target that has reward value 1. In three-target game, the green dots are the lesser targets that have reward value 0.5. The game terminates if one of the targets is hit.	54
5.4	Performance of the oracle credit assignment framework in six game environments. The x-axis represents the number of episodes in training and y-axis represents the reward received in an episode.	57
5.5	First episode to reach episode reward threshold. The x-axis represents the percentage of the optimal episode reward and the y-axis represents the first episode to reach a threshold. The vertical curves represent the agent never reach the threshold.	58
5.6	Relevance assessment user interface. Crowd workers can annotate the relevance of prior gameplay experience by clicking on the slider bar to view each step, followed by choosing Helpful! or Not helpful to determine whether the step of interest contributes to the reward signal.	59

5.7	We use the majority consensus of five workers’ responses per request to train an RL agent. The training process is stopped at episode 15 since the agent already reaches optimal performance. The shaded area is one standard deviation from the mean curves. The mean curve is the average of 10 evaluation rollouts. . . .	61
6.1	The CARAFE framework overview. (Left) Similarity Elicitation: We present triplets of states to crowd workers and elicit <i>prospective relevance</i> similarity assessment. In this illustrative driving example, workers identify that the daytime (state 1) and evening (state 3) scenes are functionally similar (both represent safe driving) compared to the snowy road scene (state 2), despite the lighting differences. (Middle) Representation Learning: We use these annotated triplets to train an encoder via triplet loss. This objective pulls the embeddings of functionally similar states (s_1, s_3) closer together while pushing dissimilar states (s_2) apart, encoding a human-derived relevance prior. (Right) RL in Target Env: The pre-trained encoder initializes the RL agent, allowing it to leverage the learned relevance prior to accelerate training in the downstream task.	66
6.2	Onboarding Interface: The entry screen introduces crowd workers to the specific Atari environment via a video demonstration and outlines the game mechanics, objectives, and scoring rules to establish a baseline of domain knowledge. . . .	74
6.3	Similarity Criteria Instructions: A dedicated instruction panel defines “functional similarity,” explicitly prompting workers to judge states based on potential future outcomes and available actions rather than superficial visual resemblance. . . .	75
6.4	Interactive Examples: Before beginning the task, the interface requires workers to review specific, interactive examples of functionally similar states to ensure they understand how to apply the criteria in practice.	76
6.5	The Triplet Comparison Interface: The main task view presents three animated state sequences. Workers are asked to select the pair that is most functionally similar. The interface includes a task counter and persistent access to instructions to maintain annotation quality.	77
6.6	Number of training states in the dataset	80
6.7	Number of annotated training triplet samples	81
6.8	Percentage of injected erroneous samples	82
6.9	Effect of dataset composition on performance	83
6.10	Triplet loss margins	84
6.11	Learning rates of the encoder during RL training	85
6.12	Absolute Agreement Gap throughout 50M training steps in Seaquest, with a zoomed-in view of the first 5 million steps (bottom).	86
6.13	Absolute Agreement Gap throughout 50M training steps in Demon Attack, with a zoomed-in view of the first 5 million steps (bottom).	87
6.14	Absolute Agreement Gap throughout 50M training steps in Battlezone, with a zoomed-in view of the first 5 million steps (bottom).	88
6.15	Normalized Agreement Gap across games. Seaquest (cyan) shows the strongest and most consistent alignment, while Battlezone (blue) remains the most volatile.	89

6.16	Comparison of filtering methods averaged across all games. The Representational Drift Filter (cyan) consistently yields higher normalized agreement than the unfiltered baseline (blue), particularly in later training stages.	90
------	---	----

LIST OF TABLES

TABLE

4.1	Performance of indicator pattern elicitation in Space Invaders	37
4.2	Performance of indicator pattern elicitation in Atlantis	37
5.1	Number of episodes required to reach maximum reward (N/A indicates the agent never reaches the maximum reward; 30x30 budgets are 10 times the 10x10 budgets).	56

ABSTRACT

This dissertation explores novel methods for leveraging human input to expand the scope and scale of interactive tasks that reinforcement learning (RL) is able to address. While RL provides a general framework for solving sequential decision-making problems, its generality comes at the expense of high training costs due to the fundamental difficulty of disentangling conflated signals from actions taken in an environment. RL requires both a well-defined reward function and a specification of the agent’s basic abilities to sense and act within an environment. However, turning these primitives into a model that captures cause and effect, particularly when the reward signal is temporally separated from the action by an unknown delay, requires a vast number of examples that scale dramatically with the complexity of the problem domain. This dissertation aims to leverage the pre-existing knowledge that humans possess across a diverse range of domains, analogous to how people learn from others via social learning, to make RL more practical in a broader spectrum of settings.

To make this guidance maximally scalable, available, and robust, this dissertation seeks to elicit feedback from dynamic groups of people (crowds) with minimal assumptions about their expertise. By employing crowds to provide otherwise missing signals, this work broadens the applicability of RL through two pathways: *reward specification* and *relevance guidance*. In Part I, we introduce a method for eliciting and constructing reward functions using patterns, demonstrating that non-experts can specify robust rewards via this approach. In Part II, we present methods for capturing general knowledge via the concept of *relevance*, utilizing human intuition to refine temporal credit assignment and construct state representations. These methods collectively demonstrate that human insight, when properly structured, can effectively supplement existing RL algorithms.

CHAPTER 1

Introduction

This dissertation explores how people and reinforcement learning (RL) can form a powerful combination to solve sequential decision-making problems in more scalable and extensible ways. It argues that leveraging crowds to provide RL agents with previously absent signals can expand the applicability of RL in real-world settings.

RL provides a general framework for addressing sequential decision-making problems. As a data-driven approach, RL enables agents to learn policies that maximize long-term rewards through environment interactions [98]. Recent advances in deep learning have enabled RL to tackle previously intractable problems, such as learning to play interactive and strategic games at human levels [67, 89, 91, 90]. However, RL still faces significant challenges when applied to real-world settings due to two primary issues: the difficulty of specifying accurate reward functions and a lack of task-relevant prior knowledge. Unlike games where scoring mechanisms are intrinsic to the environment, real-world tasks require humans to manually define objectives. Yet, specifying a reasonable, let alone accurate, reward function from the outset is often a non-trivial task. This difficulty in specification can lead to unwanted behaviors, such as reward hacking (i.e., exploiting misspecified objectives), and can hinder efficient policy training [5].

RL methods, particularly deep reinforcement learning (DRL), are known for their low sample efficiency, often requiring massive amounts of data to train functional agents [110]. Such data-driven approaches may not be viable in the real world due to practical and ethical constraints. For instance, in healthcare applications, acquiring large datasets through direct interaction with humans is costly and potentially dangerous. Furthermore, creating high-fidelity simulators for complex environments, such as human organs, is challenging. Therefore, improving sample efficiency can make RL agents more applicable to real-world tasks.

In contrast, humans possess diverse knowledge that enables them to perform tasks in various environments with minimal training or instruction. For example, people can quickly learn

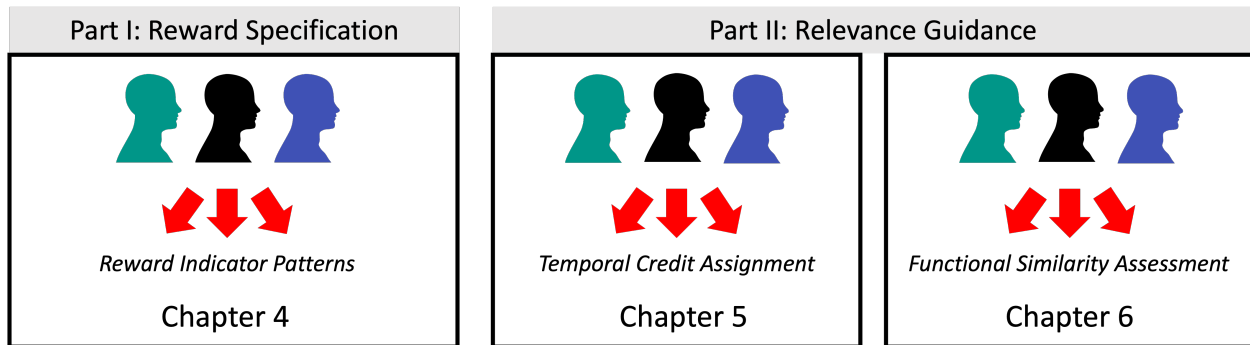


Figure 1.1: This dissertation is composed of two parts. Part I addresses the challenge of reward specification by introducing a system to elicit reward indicator patterns from people to provide consistent feedback for RL agents during training. Part II tackles the challenge of the lack of relevance guidance by presenting a human-guided credit assignment system and a method for capturing similarity-induced representations through triplet state comparisons.

to play games like bumper cars with little guidance, adapting their skills as needed. With the proliferation of platforms such as Amazon Mechanical Turk, crowdsourcing has been successfully applied in various annotation tasks (e.g., ImageNet [81] and VisualGenome [49]), which, in part, have contributed to the era of deep learning [50] as dynamic groups of people can provide scalability, diversity, and flexibility [74, 14, 95].

A critical limitation of current RL agents is their reliance on explicit, predefined structures, such as reward functions or relevance priors, to guide learning. People, however, excel at inferring implicit objectives (e.g., interpreting unstated goals from contextual cues or incomplete instructions), generalizing from sparse or noisy feedback (e.g., mastering a task from limited demonstrations or refining behavior based on ambiguous corrections), and adapting to dynamic contexts without explicit rules (e.g., adjusting strategies in response to unforeseen constraints or evolving environments). These capabilities stem from cognitive mechanisms like analogical reasoning and experiential learning, which enable humans to transfer knowledge across domains, interpret partial information, and iteratively refine strategies, all with minimal formal supervision. In contrast, RL agents struggle in such settings due to their dependence on dense reward signals and static optimization criteria.

To bridge this gap, this dissertation proposes two complementary pathways to leverage human insights: first, addressing the challenges of reward specification by eliciting patterns from people that indicate reward signals in Part I; and second, overcoming the lack of priors for relevance in RL by incorporating human guidance to shape representations in Part II. By decomposing the problem along these axes, this dissertation aims to improve the adaptability of RL agents using feedback from dynamic groups of people with minimal assumptions about

their abilities. This also reduces dependence on a limited number of experts, making the guidance more scalable, accessible, and reliable.

1.1 Part I: Human-aided reward design

Part I presents a unique perspective on human-aided reward design. Existing human-in-the-loop reward specification techniques, such as preference learning, reward shaping, and inverse reinforcement learning, often rely on humans to provide either coarse comparisons or complete demonstrations of desired behavior [5]. These methods can be limiting because they constrain how humans express their knowledge (e.g., by only allowing preference rankings or requiring near-optimal trajectories) and often assume that human-provided signals are perfect. As a result, these approaches may fail to capture subtle sub-task insights or place an excessive burden on humans to provide detailed and error-free feedback. Part I identifies these gaps in existing methods to motivate a more expressive, scalable, and consistent form of human feedback for reward generation.

Chapter 4 introduces RECEIPT (REwards from Crowd-Elicited PaTterns)—an approach that elicits pattern-based reward signals from non-expert users. Instead of requiring humans to specify a complete reward function or provide constant scalar feedback, RECEIPT identifies indicator patterns of rewards in the state space that correlate with progress toward the goal by tapping into the crowd’s ability to implicitly recognize causation and visual patterns in the task environment. With only a high-level understanding of the task goal, crowd workers can identify patterns that indicate desirable outcomes. These human-elicited patterns are then used to automatically provide scalar feedback to the RL agent as it interacts with the environment.

Importantly, RECEIPT is designed to produce consistent reward signals without continuously querying humans for input. In addition, this approach to reward design can be iterative: it does not assume that people know everything about the task environment upfront while allowing pattern collection over time as the RL agent explores previously unseen states.

Part I demonstrates that human-elicited indicator patterns can enable RL agents to learn effective policies even when the true environmental reward is unavailable. In fact, as shown in our experiments, an agent trained with RECEIPT-generated rewards can approach the performance of an agent trained with a ground-truth reward function, thus validating the efficacy of human-aided reward design.

1.2 Part II: Relevance-driven efficient learning

Part II addresses the second major challenge in applying RL to real-world tasks: the lack of task-relevant prior knowledge. Even with a well-defined reward function, an RL agent may struggle to learn if it cannot identify which portions of its experience are relevant to achieving the reward. Delayed rewards further complicate this issue by making it difficult for agents to attribute credit to the proper actions or states.

To tackle this, Part II explores how human-perceived relevance—human judgments about what factors or sub-state features affect the outcome—can be used to guide and accelerate the learning process.

Chapter 5 introduces Montage, a system based on the *Selective Memory Learning* (SML) algorithm, to improve an agent’s temporal credit assignment using human guidance. By focusing the agent’s learning on the most relevant parts of its experience through human feedback, Montage enables human-guided credit assignment. This allows the RL agent to consider only relevant states while ignoring irrelevant ones, mitigating the effects of delayed rewards. The results demonstrate that agents using Montage can learn faster and require fewer samples to achieve comparable performance, showing how human intuition about *what matters* in a task can enhance learning efficiency.

While Chapter 5 focuses on identifying which past events are critical for proper credit assignment within a single task, Chapter 6 extends the notion of relevance by introducing a human-guided representation learning approach that leverages collective intelligence to extract task-relevant knowledge from people.

Due to the nature of data-driven approaches, DRL agents may inadvertently learn representations that capture spurious details in the early stages of training, leading to learning inefficiency and suboptimal policy convergence. To address this, Chapter 6 introduces a novel framework that utilizes differential comparison—a structured approach where a dynamic group of human annotators identifies semantically similar states—to learn a more compact and generalizable representation that captures task-relevant features. The results show that the RL agents enhanced by this technique achieve better sample efficiency across various environments.

1.3 Contributions

This dissertation makes the following contributions to address the challenges of reward specification and the lack of prior knowledge in RL:

- Part I: We develop a crowd-in-the-loop method for reward function generation. Chap-

ter 4 presents the concept of pattern-based reward feedback and introduces RECEPT, a system that elicits and harnesses indicator patterns identified by people to provide a consistent reward signal to the agent. This approach provides an orthogonal method to prior work by allowing more expressive and iterative human input in reward specification.

- Part II: We introduce techniques to inject human knowledge of task relevance into the RL training loop for better efficiency and generalizability. Chapter 5 proposes Selective Memory Learning and its instantiation in the Montage system to improve temporal credit assignment. By leveraging human feedback to focus on outcome-critical states and actions, this approach demonstrates how RL can be accelerated in tasks with delayed rewards. Chapter 6 presents CARAFE, a method that enables RL agents to learn state representations that capture underlying task-relevant features of future outcomes, guided by human insight.

Together, these contributions demonstrate that human insight, provided at scale via crowdsourcing, can effectively supplement and guide reinforcement learning, making it more sample-efficient and applicable to real-world problems.

CHAPTER 2

Background

This chapter aims to establish the technical foundations for the two main parts of this dissertation: Human-aided Reward Design (Part I) and Relevance-driven Efficient Learning (Part II). We begin with the fundamental mathematical framework of RL, then discuss the specific challenges of incorporating human guidance, improving sample efficiency, and scaling supervision.

2.1 Reinforcement Learning Fundamentals

RL is a general paradigm for learning sequential decision-making problems. Unlike conventional machine learning approaches, such as supervised learning, RL studies how to act in dynamic environments via trial and error [44]. The task environment can be defined as a Markov decision process (MDP), which is specified by the tuple (S, A, T, R, γ) , where S and A are the sets of states and actions, respectively. T is a transition function that takes state s_t and action a_t at timestep t and results in the next state s_{t+1} . R is a reward function such that $R : S \times A \rightarrow \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers representing reward values. A reward signal at timestep t can be expressed as $r(s_t, a_t)$ [98]. The discount factor, γ , is used to weight reward signals according to their recency. An RL agent learns a policy, π , that maps a state s_t to an action a_t . If an agent acts according to an optimal policy, π_* , it will maximize the long-term cumulative reward in the environment [98].

In recent years, RL has undergone rapid development with the advent of deep neural networks (DNN). Since DNNs serve as powerful approximators for functions such as the policy π or the value functions V and Q , they enable RL agents to generalize across high-dimensional state spaces and tackle complex tasks that are otherwise computationally intractable in the standard MDP framework. The combination of deep learning and RL has been successfully applied in many fields, such as board games, robotic manipulation, and medical treatment [89, 91, 90, 84, 85, 60, 76, 61, 113, 20].

It is worth noting that the standard MDP formulation couples the environmental dy-

namics (S, A, T) with task-specific goals (R, γ) . While the dynamics often remain constant across tasks (e.g., the physics of a robot or the rules of traffic), the goals may vary. Part I of this dissertation primarily focuses on the latter, whereas Part II focuses on the former.

2.1.1 Reward Function

A reward function represents the goal of an RL agent [98]. The success of training RL algorithms relies on having reward functions that can provide proper oversight to the learning agents. However, as most real-world environments lack pre-defined reward functions, human intervention is often necessary to create them. Yet, designing proper reward functions can be challenging for a myriad of reasons, such as the complex dynamics of the environment, unexpected agent behaviors, and scalability. On the other hand, an ill-defined reward function can often be detrimental to RL agents as it may lead to problematic and even dangerous behaviors, posing a serious threat to real-world applications [5, 36].

2.2 Human-in-the-Loop Reinforcement Learning

Since manually specifying a mathematical reward function R is prone to error, researchers have developed methods to extract objectives directly from human input. In this section, we outline existing approaches to integrating human insight into the RL loop.

2.2.1 Learning from Demonstration

Demonstrations are a common way for humans to communicate their objectives to learning agents and have been widely applied in various applications, such as robotics [6, 99, 61, 27, 113, 40, 20]. Imitation learning uses supervised learning methods to learn a policy that mimics human demonstrations [8]. Inverse reinforcement learning instead derives a reward function from human demonstrations [71, 1]. These methods rely heavily on human expertise since sub-optimal demonstrations may result in sub-optimal policies and reward functions. In real-world settings, it is difficult to scale expertise because humans need to be able to demonstrate optimal or near-optimal policies, which are scarce and costly for complex tasks. Learning from human demonstration may also suffer from causal confusion [22], where an agent learns spurious correlations rather than true causal mechanisms. For example, in autonomous driving, an imitation learner might incorrectly associate the “brake light turning on” with the action of slowing down, rather than the true cause (the driver pressing the brake pedal in response to an obstacle).

2.2.2 Learning from Feedback

Apart from demonstration, humans can provide feedback on an agent’s policy to either create a reward function or modify the policy [70, 101, 47, 80, 32, 16, 63, 103]. Reward shaping is a technique that asks humans to evaluate an agent’s actions as the agent interacts with the environment in order to approximate a reward function [47, 103]. Policy shaping methods, on the other hand, request human evaluation of the agent’s actions to update the policy [32, 16]. Although shaping methods provide a means for humans to give direct feedback to the agent, they may cause the agent to learn unintended behaviors, as human feedback can be inconsistent over time [70, 48, 63, 77].

An alternative approach to creating reward functions from human feedback is by eliciting human preferences [3, 19, 42]. Instead of directly evaluating the agent’s actions, preference-based methods induce reward functions by asking humans to compare different trajectories of the agent [19]. However, these approaches suffer from one major challenge: humans need to understand the agent’s policy and intention. For tasks that demand complex reasoning, such as chess and Go, it can be particularly arduous for people to comprehend the context of the agent’s actions.

2.2.3 Learning from Outcomes

Prior work has attempted to automate reward generation by defining success outcomes. Daniel et al. devised a semi-supervised technique to learn a reward function by querying humans about a robot’s outcome trajectories [21]. Williams et al. used a semantic parser to create reward functions that reward the agent when it reaches the goal state [104]. While effective in controlled settings, these methods largely rely on success signals (e.g., did the robot achieve the final goal?). This sparse feedback is insufficient for long-horizon tasks where the agent requires intermediate guidance. Leike et al. addressed this by training a reward model from human feedback to produce reward signals in order to align the agent’s behaviors with the user’s intention [58]. Similarly, Singh et al. trained a convolutional neural network (CNN) to classify images of the final outcome of a robot manipulation task in order to provide positive/negative rewards to the robot [93].

The critical gap remains: how can we generate dense, intermediate reward signals without requiring constant, expert human supervision? Our method in Chapter 4 addresses this by shifting the human role from labeling actions to identifying *patterns*, which act as proxies for reward.

2.3 Relevance and Efficiency

While Sections 2.1 and 2.2 discuss how to define the reward R , having a reward function is often insufficient for practical learning due to the sample inefficiency of RL. To learn efficiently, an agent must be able to distinguish *relevant* signals from noise. This challenge manifests in two primary dimensions: establishing *retrospective relevance* (determining which past actions matter for the current outcome) and capturing *prospective relevance* (determining which current features matter for future outcomes).

2.3.1 Temporal Credit Assignment: Retrospective Relevance

In sequential decision-making, a reward obtained at time t may be the result of an action taken many steps earlier. The *temporal credit assignment* problem refers to the challenge of establishing *retrospective relevance*, i.e., identifying which past actions are responsible for a currently received reward [97]. As the horizon of the task increases, this association between relevant actions and consequences becomes increasingly tenuous, leading to high variance in gradient estimates and slow convergence. In environments with sparse rewards, this issue is exacerbated because the agent receives no feedback for the vast majority of its actions. We address this challenge in Chapter 5 (Part II) by proposing mechanisms that explicitly link past relevant events to current outcomes.

2.3.2 Representation Learning: Prospective Relevance

The efficiency of an RL agent is also heavily dependent on the state space S . In modern Deep RL, S often consists of high-dimensional observations such as images. Learning a policy directly from raw pixels requires the agent to simultaneously learn to perceive the environment (feature extraction) and to act within it (control) [59]. To address this, effective representation learning focuses on extracting features with *prospective relevance*. That is, the agent ought to identify features that are essential for predicting future outcomes, while discarding irrelevant noise. By mapping observations to a lower-dimensional latent space that captures only this essential information, agents can generalize better and learn faster. We explore this concept in depth in Chapter 6 (Part II).

2.4 Scaling Human Insight

To address the data requirements of modern RL (discussed in Section 2.3) and the difficulty of reward specification (Section 2.2), we need methods that can scale human insight effectively.

2.4.1 Crowdsourcing

Crowdsourcing has been widely used to scale human effort on tasks that would otherwise be considerably more difficult to solve. Specifically, crowdsourcing has been used to label visual, temporal, and text data points using online platforms, such as Amazon Mechanical Turk [81, 34, 94, 18]. For example, LegionAR uses an active learning approach to accelerate training a Hidden Markov Model to help crowd workers label real-time events [54]. Glance uses online crowds to massively parallelize video annotation using a conversational interaction paradigm [53]. Evorus employs machine learning techniques to learn from annotated data interactively, gradually reducing crowd oversight [41]. In this dissertation, we leverage crowdsourcing to access diverse, scalable, and readily available human insight, with minimal assumptions about worker expertise.

2.4.2 Weak Supervision

When scaling up data collection via crowdsourcing, the resulting labels are often noisy. *Weak supervision* offers a framework to handle this noise. In supervised learning, generating large numbers of clean labels is a bottleneck [81]. Prior work on weak supervision focuses on eliciting labeling functions that can noisily annotate large sets of data with relatively little human input [79, 78]. Snorkel, for instance, uses labeling functions to create generative models for assigning probabilistic labels to unlabeled data [78].

However, most weak supervision work assumes data is available a priori. In RL, new experiences are generated dynamically as the agent interacts with the environment. In Chapter 4, we introduce a method that can provide the RL agent with feedback while the agent is interacting with the environment, utilizing patterns as labeling functions to annotate unlabeled experiences based on human insight.

Summary

Having established the foundations of RL and human-in-the-loop approaches, we now turn our focus to the central theme of Part I: reward specification. The background presented in this chapter highlights three key challenges: the difficulty in manually specifying rewards, the inefficiency of standard RL algorithms, and the need for scalable human feedback. In the following chapters, we present our framework for addressing these challenges.

Part I

Human-aided Reward Design

RL agents rely on reward functions to learn desired behaviors, yet many real-world environments lack well-defined reward signals. Instead, they depend on human input to specify rewards through limited interfaces such as preference comparisons [19] or demonstrations [1]. These methods often force humans into unnatural feedback modes—either reducing their input to isolated judgments or assuming perfect demonstrations—while ignoring the iterative, fine-grained way humans naturally understand tasks.

In Chapter 3, we formalize reward creation through three key aspects: (1) granularity levels, (2) generalizability, and (3) three phases of reward creation. We systematically analyze the limitations of current methods, arguing that they restrict human expressiveness by requiring either isolated judgments (preferences) or fully formed instances of ideal behavior (demonstrations), while assuming human inputs are perfect.

These gaps motivate our investigation of *pattern-based reward specification* in Chapter 4, where humans communicate via reusable, fine-grained visual signals (e.g., visual sprites, temporal sequences) that correlate with outcomes. Unlike traditional approaches, patterns scale feedback by matching across states so as to reduce the need for constant human intervention.

CHAPTER 3

Rethinking Human-Elicited Reward Specification

3.1 Motivation

In many real-world tasks, RL agents must rely on humans to guide the construction of reward functions. Without well-defined, environment-provided signals, agents depend on human input to understand what is desirable or undesirable, though such input may be incomplete or incorrect.

Traditionally, this human-elicited reward specification has taken limited forms. Humans may be asked to compare states (preference learning [19]), provide scalar feedback for particular states (DeepTAMER [103]), or demonstrate entire trajectories (inverse reinforcement learning, or IRL). While these approaches have seen success, they fail to capture the full depth and breadth of information that humans can provide.

This chapter challenges these conventional approaches. We argue that current methods restrict human expressiveness by forcing them to provide either isolated judgments (preferences) or fully-formed instances of ideal behavior (demonstrations), while assuming the human inputs are perfect. Consequently, these methods often fail to capture the explicit, sub-state level features that humans can readily identify.

Instead of relying on trace-level (trajectories of the state and actions) or state-level feedback, we propose a shift towards enabling humans to communicate finer-grained, pattern-based feedback that reflects sub-state level information. By focusing on *what humans know* rather than *what reward functions need*, Chapter 4 proposes a novel approach to reconstructing reward signals by eliciting reward indicator patterns. This approach assumes that people may not have complete knowledge at the outset and requires iterative improvement of the reward signals over time.

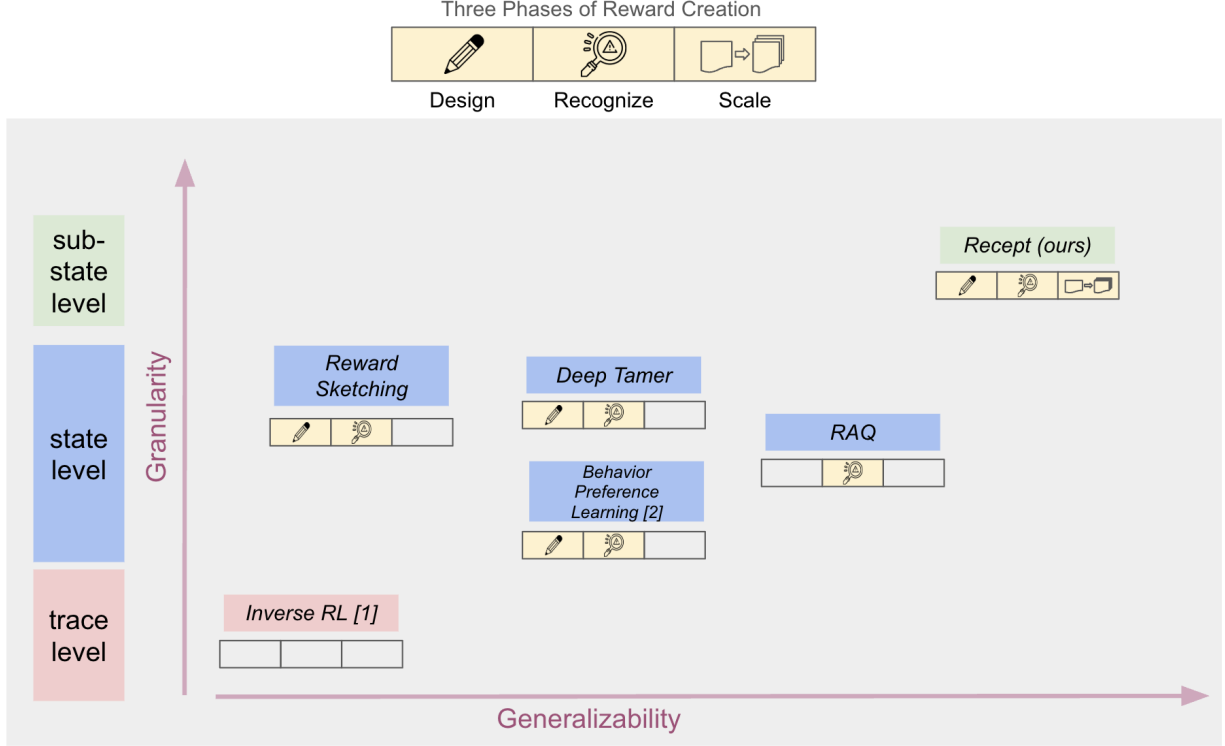


Figure 3.1: Comparison of reward creation approaches mapping to granularity levels, generalization capability, and the three phases of reward creation (design, recognize, and scale). Approaches are categorized by their trade-offs between specificity and adaptability across different problem domains.

3.2 Limitations of Traditional Approaches

We present two parallel perspectives for understanding the field of human-elicited reward creation: the three levels of reward representation (trace, state, and sub-state) and the three phases of reward creation (design, recognition, and scaling).

These perspectives highlight how reward specification evolves from high-level trajectory analysis (trace) to localized, pattern-specific insights (sub-state), while also covering the entire reward creation process, from defining desired behaviors to detecting and scaling appropriate feedback. Because human-provided reward signals are often noisy or inconsistent, analyzing them through these specific levels of granularity allows us to identify exactly where information is lost in traditional methods. By integrating these perspectives, our approach aims to create a more comprehensive and adaptive reward creation framework that better aligns with real-world complexities and human cognitive processes.

3.2.1 Trace Level Feedback

Trace-level approaches, such as inverse reinforcement learning (IRL), derive rewards by analyzing entire trajectories of agent behavior. By observing actions and outcomes, IRL attempts to infer the underlying reward function that results in the observed behavior [71, 1]. This method can be useful in domains where human demonstrations are abundant, optimal, and easily accessible.

However, trace-level methods face challenges in environments with incomplete or noisy data. While successful in controlled settings with expert pilots (e.g., autonomous helicopter flight [1]), previous work often assumes access to full oracle-like signals. This contrasts with real-world scenarios where human knowledge is frequently partial and uncertain. Because these approaches depend on full traces, they can struggle to generalize when faced with partial observations or suboptimal demonstrations [7, 111]. Moreover, the computational complexity of modeling entire trajectories can limit their scalability in large or dynamic environments.

3.2.2 State Level Feedback

State-level approaches focus on deriving reward signals through human feedback mechanisms like preference learning, reward sketching, and DeepTAMER [103]. These methods rely on direct input from users, who express preferences over different states or trajectories. For instance, preference learning algorithms ask participants to rank or choose between options, allowing the system to infer a reward function that aligns with these preferences. Similarly, reward sketching enables users to annotate trajectories or states with approximate reward values, and DeepTAMER refines reward learning by interacting with humans during training.

While these techniques are effective in capturing broad reward structures, they often fail to account for the fine-grained nuances of specific environments. State-level methods may overlook subtle contextual elements because users are evaluating the aggregate state rather than specific causal factors. Additionally, these methods can place a significant cognitive load on users, as they require consistent and accurate feedback across a potentially large state space [45, 15].

3.2.3 Sub-State Level Feedback

The sub-state level represents a novel approach that focuses on eliciting human knowledge about specific features or partial observations within an environment. Unlike state- or trace-level methods, this approach seeks to reconstruct rewards based on localized insights rather

than holistic preferences or trajectories. By narrowing the focus to sub-state elements, our method in Chapter 4 captures granular details that are often missed by broader frameworks.

To illustrate the expressive power of these localized insights, consider a domain expert who specifies: “Whenever the agent approaches a hazard without a safety mechanism, that is bad.” This specification transcends individual examples by assuming that the feature—in this case, the “hazard”—is causally connected to a negative reward regardless of the broader state context (e.g., whether the agent is in a residential hallway or a warehouse floor). While this assumption requires that features are robust, it allows for significantly better generalization.

By emphasizing feature-level knowledge, this approach reduces the cognitive demand on users while enhancing the interpretability and robustness of the reconstructed reward function. Our framework in Chapter 4 employs elicitation techniques that are designed to be user-friendly and efficient. These include methods for identifying key features, predicting likely outcomes, or providing localized feedback on partial states. By leveraging these techniques, we aim to create a reward reconstruction process that aligns closely with human insights while maintaining adaptability to diverse environments.

3.3 Three Phases of Reward Creation

We also introduce the concept of three phases of reward creation that are crucial for effective RL training:

1. **Design:** An expert often needs to specify the rewards based on their understanding of the environment and the desired agent behavior. This involves defining success criteria, specifying actions that should be encouraged or discouraged, and encoding these definitions into a reward function.
2. **Recognize:** Once reward signals are specified, there needs to be a system that recognizes desired reward patterns during RL training. This system identifies when the agent’s behavior aligns with the predefined criteria and triggers appropriate feedback.
3. **Scale:** Recognized rewards need to be scaled automatically as the agent encounters new states. While most existing methods achieve this through function approximation, our approach leverages built-in pattern matching, enabling scalable and efficient reward detection without additional approximation layers.

Our approach in Chapter 4 seamlessly integrates all three phases, providing a comprehensive reward creation pipeline that supports expert-driven design, automated recognition,

and scalable application through pattern-based methods. This allows us to deliver context-sensitive, adaptive feedback to RL agents, ensuring robust performance in dynamic environments.

3.4 Shifting the Perspective: From Needs to Knowledge

We propose a new framework: within our proposed framework, instead of constraining human input solely to state-level preferences or trace-level demonstrations, we will also enable humans to communicate pattern-level insights. Patterns can be thought of as sub-state conditions, relational structures, or situational cues that, when encountered by the agent, should prompt a particular reward signal.

Such patterns differ fundamentally from current elicitation approaches. Rather than relying on humans to produce either global judgments (preferences) or fully realized behaviors (demonstrations), patterns capture the partial, compositional knowledge humans naturally hold. They act as reusable, context-independent signals that can inform an agent’s reward function on the fly, regardless of how the environment evolves. The sub-state level pattern-based approach also offers the following advantages, including its ability to iteratively collect more complete and less noisy signals from people over time, akin to active learning:

- By focusing on localized knowledge, it can reduce the cognitive demand on crowd participants, allowing for more accurate and consistent feedback. This is particularly important in environments where users may lack the expertise or bandwidth to provide detailed trace-level or state-level annotations.
- The sub-state level approach enhances interpretability by explicitly mapping human insights to specific environmental features. This transparency not only improves trust in the system but also facilitates refinement of the reward model. For example, in collaborative robotic applications, sub-state feedback can be used to guide agents through novel settings by emphasizing critical features.
- By leveraging human knowledge at the sub-state level, the framework can identify and amplify signals that might otherwise be lost in broader analyses, as these signals are collected progressively, improving reliability and completeness over time.

The essence of this perspective lies in moving from a paradigm where humans must supply what the algorithm needs (choices between states or complete demonstrations) to one that elicits what humans already know. Humans understand rules, recognize warning signs, and can articulate meaningful conditions that transcend single examples. These insights are not

mere preferences or incremental feedback; they are portable, expressive structures that, once identified, can guide an agent’s behavior across a variety of circumstances. By unlocking the potential of pattern-level guidance, we tap into a richer source of human knowledge.

This shift promises more flexible, robust, and scalable reward specification strategies that are better aligned with human values and easier for humans to communicate. In addition, as large language models (LLMs) and foundation models continue to advance, they may serve as a natural bridge between human language descriptions of patterns and the formalized representations that agents require. With improved natural language understanding, these models can help translate human insights into codified patterns, enabling non-technical users to specify complex conditions intuitively.

3.5 Conclusion

Existing human-elicited reward specification methods, whether preference-based inputs or demonstration-driven IRL, often fail to capitalize on the reasoned logic and causal patterns that humans naturally apply. These methods typically assume perfect human input rather than accounting for its inherent imperfections and refining it iteratively. By enabling humans to describe reward patterns at a sub-state level, we open a path toward more expressive, generalizable, and meaningfully human-aligned reward functions. As foundation models improve in capturing and translating human intent, these pattern-based approaches will become even more accessible. Therefore, reframing the problem around what humans inherently know rather than forcing them into forms that machines require holds the key to more effective, human-aligned reinforcement learning systems.

CHAPTER 4

Improving Reward Specification through Identifying Causation of Rewards

4.1 Motivation

The goal of an RL agent is encoded in a reward function that provides numerical feedback to the agent so that it can learn an optimal strategy by maximizing expected long-term rewards [98]. Hence, the reward function is a critical component of the RL training pipeline that ultimately determines the behavior of the agent.

However, reward functions are often absent from many real-world environments. Asking humans to formally specify a reward function is challenging because it requires anticipating the changing context of an open-ended environment prior to agent deployment. If the reward function is missing or ill-defined, the agent may never learn to achieve its goal.

Humans, on the other hand, can learn to do new tasks in various environments throughout their lives without the need for a formally-specified set of reward outcomes. Thus, leveraging human insights about the causation of rewards in task environments is a promising means of constructing reward functions. Previous work has made progress towards integrating human insight into creating reward functions [1, 47, 19]. However, these approaches often require significant human effort and struggle to scale to the large volume of agent-environment interactions typically required to train RL systems.

In this chapter, we aim to use *patterns* to produce reward signals to train RL agents. In many environments, there exist patterns that indicate feedback should be given to the agent, such as an explosion sprite in Atari games and blinking brake lights in a driving scenario. We call such patterns *indicator patterns*. By leveraging pattern matching, a few indicator patterns could provide a large number of reward signals for an RL agent. Therefore, we introduce a pattern-based reward function generation (PRFG) approach that generates reward functions by adding indicator patterns as the agent interacts with the environment. To collect indicator patterns from the environment, we leverage human insight as people are

capable of identifying patterns in the environment [26, 64].

Extracting accurate indicator patterns is crucial to our approach since proper patterns can scale correct reward signals, but inaccurate patterns can scale erroneous signals as well due to the nature of pattern matching. To examine potential failure modes of using patterns to produce reward signals, we conduct an error characterization study in a controlled environment and find that four factors—*missing*, *false*, *duplicate*, and *delayed* rewards—can negatively affect RL training.

In order to avoid the pitfalls discovered in the characterization study, we use crowdsourcing to make pattern elicitation accurate. Identifying indicator patterns requires anticipating future states, i.e., whether the patterns collected now can be applied in the future agent interactions with the environment. However, it may be difficult for only a few people to foresee all possible future states. Hence, we use the crowd to overcome this because a group of annotators can collectively provide a more reliable and diverse set of patterns. To further make pattern elicitation robust, we present a new workflow, Collect-View-Refine (CVR), to enable crowd workers to collect patterns from the environment, view the potential outcomes of applying the collected patterns, and subsequently refine the patterns.

Based on PRFG, we implement *RECEPT*—a system that elicits visual indicator patterns using CVR and then detects these patterns in the environment to produce reward signals for RL agents. To test our system, we choose three Atari 2600 games from the Arcade Learning Environment [10], which is a well-established test suite for RL algorithms and provides built-in ground truth reward functions for evaluation. The experimental results show that RL agents trained by RECEPT perform 80% and 98% as well as those trained by the ground truth reward functions in Space Invaders and Atlantis, respectively. Furthermore, the result of progressive pattern elicitation and RL training in Demon Attack showcases RECEPT’s ability to adapt to an evolving environment.

In this chapter, our contributions are (1) PRFG, a human-in-the-loop approach that generates reward functions to provide scalable feedback for RL agents; (2) a characterization study that examines the potential failure modes of using patterns to produce rewards; (3) RECEPT, a system that implements PRFG and can produce reward signals to train RL agents in visual domains; (4) CVR, a new crowdsourcing workflow that guides crowd workers to generate accurate indicator patterns; and (5) the experimental results that demonstrate the efficacy of our approach and system.

Our PRFG approach has the potential to enable scalable human-in-the-loop RL systems to work in complex real-world environments. Meanwhile, the CVR workflow can facilitate robust pattern elicitation with crowdsourcing in numerous settings and domains.

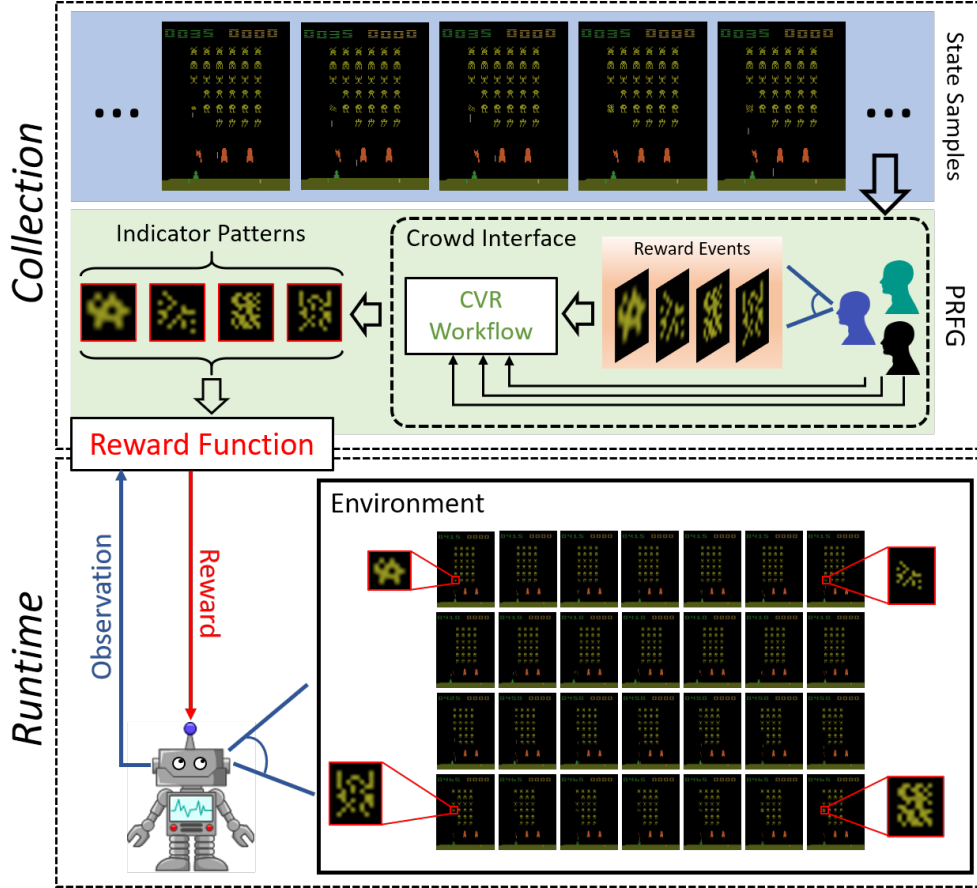


Figure 4.1: Overview of RECEPT. RECEPT first collects state samples from the environment and elicits indicator patterns from the crowd using our CVR workflow. Then, RECEPT generates a reward function based on PRFG to detect the indicator patterns in order to produce reward signals as the agent interacts with the environment.

4.2 Approach

In this section, we introduce our approach for producing reward signals using little human effort via *Pattern-based Reward Function Generation* (PRFG).

4.2.1 Constructed Reward Function via Pattern Sequences

Our goal is to create a reward function $\hat{r}$ that acts as a proxy for human intent. As the agent interacts with the environment, specific sensory content may co-occur with the desired outcome. We refer to these occurrences as *reward events*. For example, the visual sequence of an exploding sprite is a reward event in *Space Invaders*, as illustrated in the orange box in Figure 4.1. Since humans are capable of recognizing these events given past exposure to the environment [68, 72], we aim to elicit these events to construct a reward function.

Capturing reward events accurately is challenging because they are often temporal rather than static. In other words, they unfold over time. To address this, we seek to capture *indicator patterns* to serve as milestones within a reward event. An indicator pattern acts as a visual template for a specific segment of the event.

In this chapter, we define a reward event not as a single indicator pattern, but as an ordered *sequence* of patterns. This approach allows us to capture complex temporal dynamics; for instance, a “sprite explosion” event might consist of a sequence where the sprite transitions into a region of scattered pixels before eventually disappearing.

This reward function operates by tracking the completion of these sequences. At any timestep t , $\hat{r}(s_t)$ is calculated as the sum of the predetermined values of all reward events that have completed their specific pattern sequence at that moment. If no pattern sequences are completed at t , the reward is 0. This approach allows the reward function to scale naturally: if multiple events occur simultaneously, the agent receives their cumulative value. Algorithm 1 in Section 4.4.1 details the pattern matching and sequence tracking logic.

However, this reward function may initially be incomplete because novel indicator patterns may not be immediately visible. Some patterns may only emerge after the agent achieves intermediate goals, while others might be overlooked by humans due to the large volume of visual information. Consequently, maintaining a comprehensive reward function necessitates frequent human queries, which is difficult to sustain with a limited number of annotators. Furthermore, pattern matching is sensitive: a single inaccurate pattern definition can lead to numerous false positives. Expecting annotators to maintain high accuracy on such a complex task over extended periods is also unrealistic due to fatigue. Therefore, we leverage crowdsourcing to access a scalable workforce capable of addressing these pattern requests. To ensure robust annotation, we design a new crowdsourcing workflow that decomposes the pattern elicitation task into small, tractable steps (see Section 4.4.3).

In the following sections, we first conduct an error characterization study to examine potential failure modes associated with using patterns to produce reward signals. Subsequently, we implement a system to train RL agents based on PRFG and the findings from the characterization study.

4.3 Error Characterization Study

To better design our system, we conduct a study to investigate potential failure modes that might occur when using patterns to produce reward signals. We first introduce the experimental setup for the study, followed by the results and our findings.

4.3.1 Setup

Our study is based on two Atari 2600 games—Space Invaders and Atlantis. The test platform for the games is implemented by the Arcade Learning Environment (ALE) [10] with the OpenAI Gym interface [13]. ALE provides a built-in ground truth reward function for each game, which generates numerical feedback to the player based on the game state. We consider the ground truth reward functions as the *oracle* reward functions, $r^*(s)$, since the goal of our system is to reproduce the same signals generated by the ground truth reward function.

To evaluate the learning performance of an RL agent, we use two metrics—*learning curve* and its *area under the learning curve* (AULC). A learning curve illustrates the training progress of an RL agent. The x-axis of a learning curve plot represents the number of training timesteps, which is a step taken by the agent in the environment during training. And the y-axis represents the average episode reward, which is the average of total rewards in an episode. Note that any reward value in a learning curve is calculated based on the ground truth reward functions, and those values are different from reward signals that an agent receives during training. If an RL agent learns to perform the task better over time, average episode reward ought to increase as the number of training timesteps increases. By examining different learning curves, one can compare the performance of the corresponding RL agents at any given timestep. To quantify the overall performance of an RL agent, we use the AULC of a learning curve. AULC is the sum of the average episode reward at all available timesteps and measures how efficient an RL agent learns.

We chose A2C [66], a widely used RL algorithm [105, 69, 114], as our learning agent. The A2C agent is implemented by OpenAI baselines [23]—an open-source repository for RL algorithms. The hyperparameters of the A2C agent is set to the default values. During training, the agent receives clipped reward signals to facilitate learning, as described in [67]. Since the reward values in Space Invaders and Atlantis are all positive, the clipped reward signals are effectively either 0 or 1. When plotting a learning curve, the values of the average episode reward are calculated based on the ground truth reward functions for the purpose of comparison. For each condition, we run three training instances of the A2C agent and plot the average learning curve with the shaded area representing one standard deviation.

4.3.2 Methods and Results

We investigate four types of failure modes when identifying reward signals: missing reward, false reward, duplicate reward, and delayed reward. To quantify the impact of these errors on RL training, we conduct experiments where the agent is trained using a synthetic reward function $\hat{r}(s)$. This function is generated by systematically corrupting the ground truth

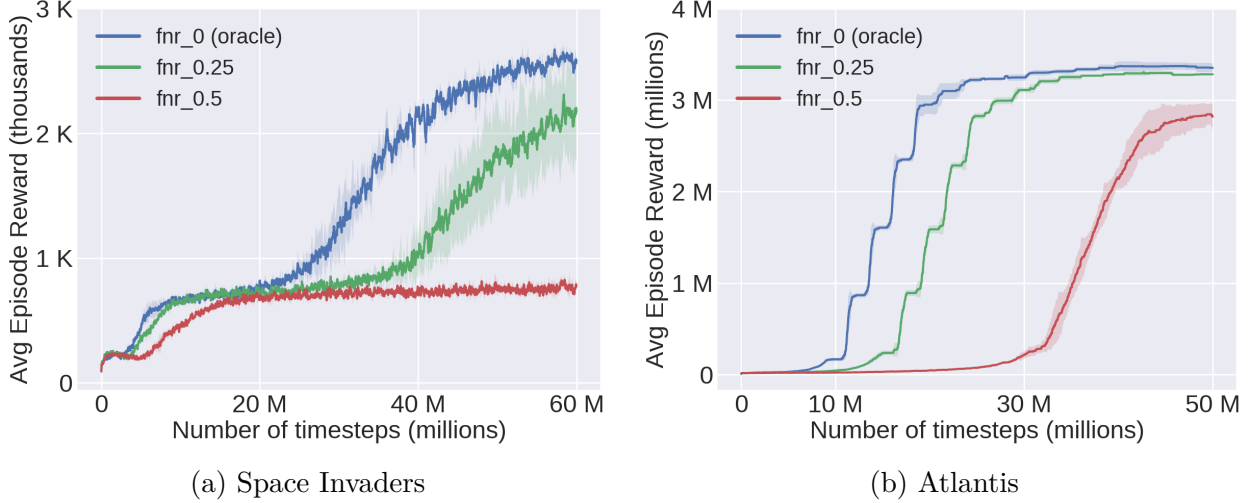


Figure 4.2: Learning curves of different false negative rate (FNR) values

reward function $r^*(s)$ according to specific error parameters. In each environment, the ideal learning curve is achieved by training an A2C agent with the ground truth reward function.

4.3.2.1 Missing Reward

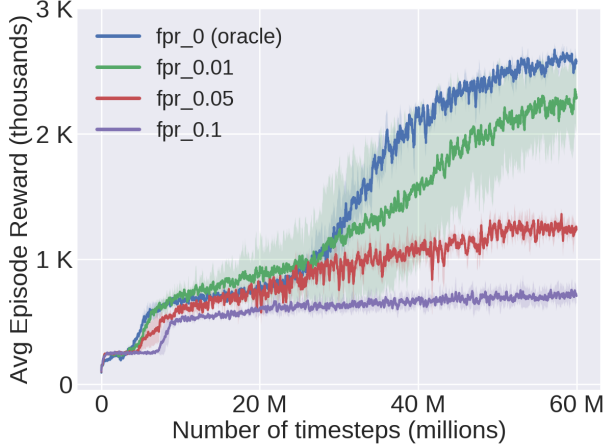
When detecting reward signals using patterns, it is possible that some reward events will be overlooked. We evaluate the agent’s *robustness* to missing information by manipulating the False Negative Rate (FNR), which is defined as:

$$\text{FNR} = \frac{\text{fn}}{\text{fn} + \text{tp}}$$

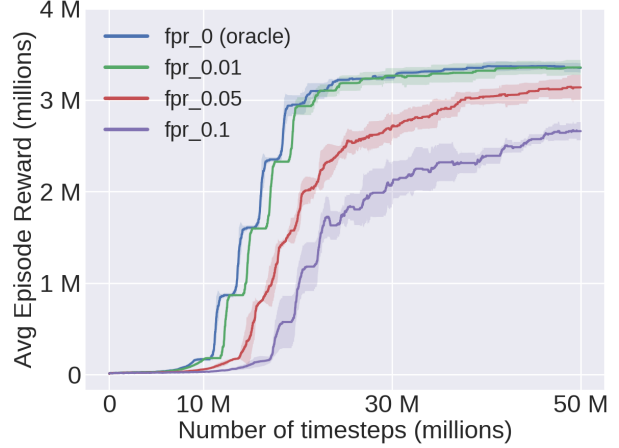
where fn (false negative) represents a ground truth reward signal that was omitted from $\hat{r}(s)$, and tp (true positive) represents a signal correctly retained.

At timestep t , if the oracle reward signal and the generated reward signal are both 1, we consider the generated reward signal to be true positive (tp). On the other hand, false negative (fn) occurs when the oracle reward signal is 1, and the generated reward signal is 0, which is a missed reward signal. By reducing positive oracle reward signals in $r^*(s_t)$, we can achieve different FNR values. In this setting, we test the A2C agent with two FNR values—0.25 and 0.5 (or missing 25% and 50% reward signals respectively).

As Figure 4.2 shows, missing rewards affect the training negatively for both games. In Space Invaders (Figure 4.2a), the differences of AULC between the oracle (FNR 0, blue curve) and other learning curves show that missing 25% of reward signals (FNR 0.25, green curve) causes 29% AULC drop from the oracle, and missing 50% of reward signals (FNR 0.5,



(a) Space Invaders



(b) Atlantis

Figure 4.3: Learning curves of different false positive rate (FPR) values

red curve) causes 56% AULC drop from the oracle. In Atlantis (Figure 4.2b), we observe a similar performance drop when the agent does not receive reward signals. However, the final performance of the agent with FNR 0.25 is close to that of the oracle. These results suggest that missing rewards have a negative impact on the learning efficiency of an RL agent, and its severity depends on the amount of missed reward signals.

4.3.2.2 False Reward

Inaccurate indicator patterns may lead to false rewards. Formally, at timestep t , the generated reward signal is a false reward if its value is positive, whereas the oracle reward signal is 0. To measure how false rewards may affect RL training, we use the false positive rate (FPR), which is defined such that

$$FPR = \frac{fp}{fp + tn}$$

True negative (tn) occurs if both the oracle reward signal and the generated reward signal are 0, and false positive (fp) represents a false reward. In this setting, we train A2C agents with three FPR values (0.01, 0.05 and 0.1).

As Figure 4.3 illustrates, training A2C agents with $fpr = 0.01$ (green curves) can achieve comparable performance to the oracle (blue curves). But, if FPR is higher than 0.01, it would considerably decrease learning efficiency. When FPR is 0.1 (purple curves in Figure 4.3), the AULC is reduced by 59% and 42% from the oracle in Space Invaders and Atlantis,

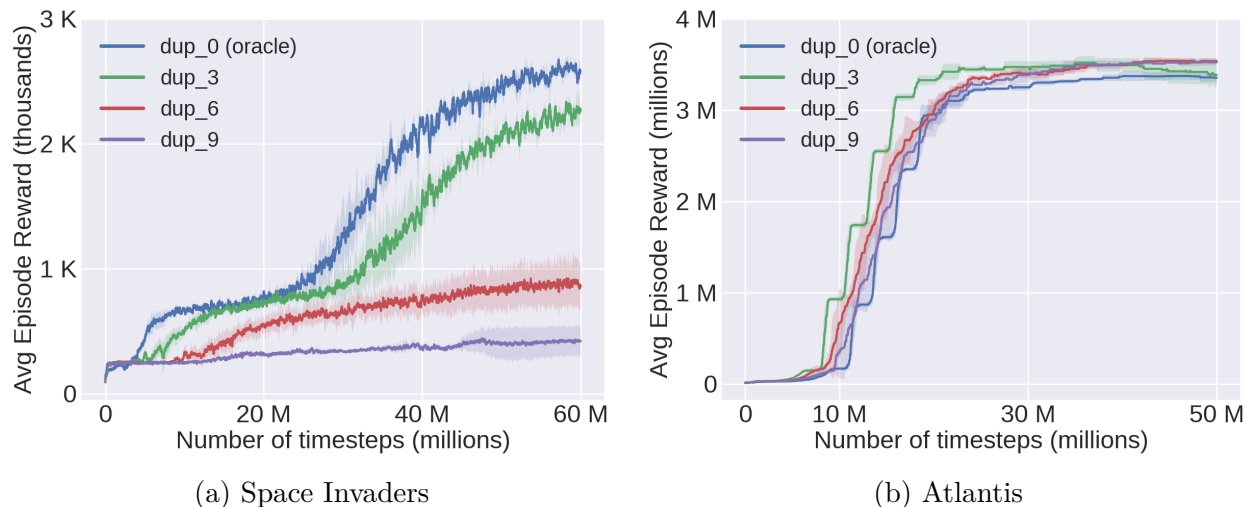


Figure 4.4: Learning curves of different duplicate values

respectively. This shows that an RL agent can be sensitive to false rewards during training, which poses a challenge to pattern-based methods since even a few inaccurate patterns may lead to many false matches.

4.3.2.3 Duplicate Reward

A reward event may last multiple timesteps in the environment and thus can trigger multiple indicator patterns. If we consider every detected indicator pattern as an individual reward signal, a multi-timestep reward event will cause duplicate rewards because any subsequent reward signal after the first one is the result of the same reward event. Yet, duplicate reward signals may not necessarily be false rewards because they correspond to a valid reward event. For instance, if a reward event occurs at timestep t and lasts four timesteps, the agent ought to receive only one reward signal. But if the agent also receives three extra reward signals at timestep $t + 1$, $t + 2$, and $t + 3$, we consider them as duplicate reward signals (*dup*) instead of false rewards, as they still occur within the duration of the reward event. To inspect the effect of duplicate rewards on RL training, we train the A2C agent with three *dup* values—3, 6, 9.

According to Figure 4.4, duplicate rewards have different effects on the agent in different game environments. In Space Invaders, 3 duplicate reward signals reduce AULC by 20% from the oracle, whereas 6 duplicates and more hurt the performance significantly—58% AULC drop from the oracle when *dup* = 6, and 77% AULC drop from the oracle when *dup* = 9 (see Figure 4.4a). However, in Atlantis, duplicate rewards do not seem to affect the performance negatively. In fact, *dup* = 3 even improves the AULC by 13% from the oracle

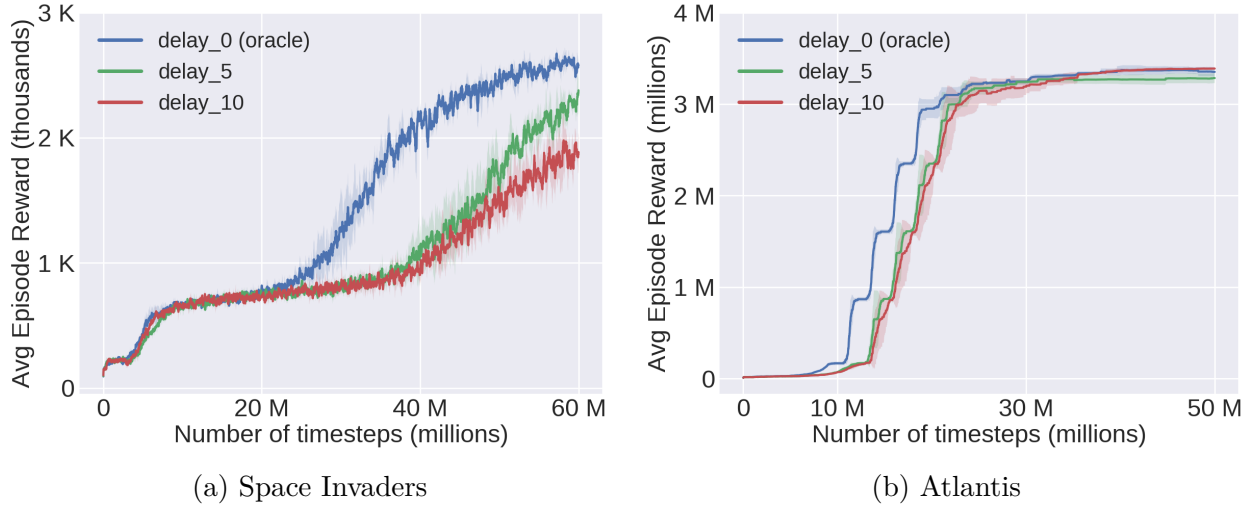


Figure 4.5: Learning curves of different delay values

(see Figure 4.4b). We suspect that the main reason is that multiple bullets are allowed to fire consecutively in Atlantis, as opposed to no bullet is allowed to fire until the previous one disappears (either by hitting a target or moving out of the screen) in Space Invaders. Thus, duplicate rewards in Atlantis essentially encourage the agent to fire more frequently since there is no penalty. These results suggest that the impact of duplicate rewards on RL agents is environment-specific, and excessive duplicate rewards can be harmful.

4.3.2.4 Delayed Reward

The problem of delayed rewards is one of the fundamental challenges in reinforcement learning [98] and occurs when an RL agent does not receive a reward signal after taking an action until a number of timesteps later. In our setting, we do not consider delay as the number of timesteps between an action and its corresponding reward signal because such delays are inherent in the dynamics of an environment. Instead, we only consider the “extra” delay between the start of a reward event and the moment when the reward event is detected. As mentioned in Section 4.3.2.3, a reward event may consist of multiple timesteps. If the reward event is not detected immediately, there will be a delay between the oracle reward signal and the generated reward signal. To examine the impact of delayed rewards, we train A2C agents with two delay values: 5 and 10 timesteps.

Figure 4.5 shows that delayed rewards reduce learning efficiency in both game environments. In Space Invaders (Figure 4.5a), a 5-timestep delay causes 28% AULC reduction from the oracle (green curve), and a 10-timestep delay causes 34% AULC reduction (red curve). In Atlantis, delayed rewards have a smaller negative effect on learning efficiency as

both delay values cause about 9% AULC reduction from the oracle. This is due to the fast travel speed of bullets in Atlantis, and thus the overall delay between a firing action and its reward signal is smaller, which mitigates the learning efficiency drop. Nevertheless, reducing delayed rewards can help train RL agents more efficiently.

4.3.3 Design Goals

According to the results of the error characterization study, we summarize that the design goals for our pattern-based reward function are to minimize: (1) missing rewards, (2) false rewards, (3) duplicate rewards, and (4) delayed rewards.

4.4 RECEPT

Based on the PRFG approach and the design goals derived from Section 4.3, we implement a system, REwards from Crowd-Elicited PaTterns (RECEPT), that provides reward signals to train RL agents. RECEPT consists of three components. The first component generates reward signals by detecting indicator patterns (Section 4.4.1); the second component determines when and how to query for acquiring new indicator patterns (Section 4.4.2); and the third component elicits indicator patterns from crowd workers using the Collect-View-Refine workflow (Section 4.4.3).

4.4.1 Generating Reward Signals via Indicator Tracks

Given a set of indicator patterns, we aim to generate reward signals using pattern matching. Since the state space of our test environments is composed of RGB pixels, we consider image regions as indicator patterns. Suppose that crowd workers produce a set of indicator patterns, $P = \{p_1, p_2, p_3, \dots\}$, we employ *template matching*, a pattern matching technique in computer vision, to determine if one of the patterns appears in state s_t . Template matching compares the pixel values of an indicator pattern with that of state s_t in a sliding window fashion and outputs the location of the best match.

However, since a reward event may last multiple timesteps, template matching alone cannot determine whether a detected indicator pattern belongs to an ongoing reward event. Therefore, in order to prevent duplicate rewards (Section 4.3.2.3), we use *indicator tracks*, inspired by multiple-object tracking in computer vision [106], to track the existing reward events while template matching is detecting indicator patterns. An indicator track is a spatio-temporal trajectory and represents one reward event, as illustrated in Figure 4.6. In the spatial domain, an indicator track is a series of locations of the detected indicator

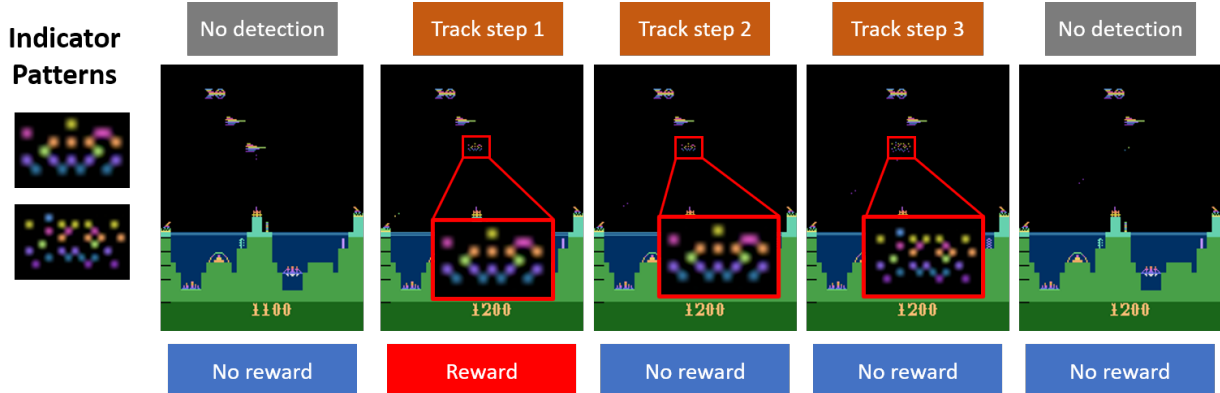


Figure 4.6: Tracking indicator patterns to form reward events. At each timestep, template matching detects patterns in the current state. The numbers below the images (1100, 1200) represent the cumulative **game score**, which remains constant after the initial reward event. Note that an indicator track can consist of visually distinct patterns (e.g., Track step 1 vs. Track step 2) as long as they are spatially and temporally contiguous. This allows RECEPT to track a semantic event (like an explosion) that changes appearance over time without triggering duplicate rewards.

patterns. In the temporal domain, it is a sequence that starts as the first indicator pattern of a reward event appears and ends at the timestep when the last indicator pattern of the event is detected.

When an indicator pattern is detected at timestep t , we first attempt to merge the pattern into one of the existing indicator tracks based on distance and temporal continuity. If successful, we update the location and timestep of the existing track without generating a reward to prevent duplicate rewards. If the detected pattern cannot be merged into any existing indicator track, we will form a new track and produce a positive reward. Therefore, the value of the reward signal corresponds to the number of *newly-formed* indicator tracks at timestep t . Algorithm 1 shows the details of this process.

Since our system is benchmarked against the ground truth reward functions, which clip all positive reward signals to 1 [67], we set the value of each reward event to 1 for the purpose of comparison. Yet, our system can allow users to specify a predetermined value for each reward event if required.

4.4.2 Querying for Indicator Patterns

In many environments, an RL agent may encounter only a fraction of the total number of indicator patterns in the beginning of the deployment, and many of them will not appear until some conditions are met. For instance, in Demon Attack, a player has to eliminate

Algorithm 1 Generating estimated reward signals via indicator tracks. Note: The matching condition allows an indicator track to persist even if the visual pattern evolves (e.g., an enemy sprite transitioning into an explosion sprite), provided the distinct patterns satisfy spatial and temporal continuity.

Require: A set of indicator patterns $P = \{p_1, p_2, p_3, \dots\}$, a max tracking distance threshold θ

```

1: Initialize an empty list of indicator tracks  $\mathcal{T} \leftarrow \{\}$ 
2: for timestep  $t = 1, 2, 3, \dots$  do
3:   Observe state  $s_t$ 
4:   Initialize an estimated reward signal  $\hat{r}_t \leftarrow 0$ 
5:   for each indicator pattern  $p_i \in P$  do
6:     Perform template matching given  $p_i$  and  $s_t$ 
7:     if  $p_i$  is detected at location  $loc_{p_i}$  in state  $s_t$  then
8:       tracked  $\leftarrow$  False
9:       for each indicator track  $\tau_j \in \mathcal{T}$  do
10:        Get the last location  $loc_{\tau_j}$  and timestep  $t_{\tau_j}$  of  $\tau_j$ 
11:         $\triangleright$  Check spatial and temporal continuity to allow pattern evolution
12:        if the distance between  $loc_{p_i}$  and  $loc_{\tau_j} \leq \theta$  and  $t == t_{\tau_j} + 1$  then
13:          Update  $\tau_j$  using  $\{loc_{p_i}, t, p_i\}$ 
14:          tracked  $\leftarrow$  True
15:        if tracked is False then  $\triangleright$  no existing track matches; indicates new reward
event
16:          Create a new indicator track  $\tau$  using  $\{loc_{p_i}, t, p_i\}$ 
17:           $\mathcal{T} \leftarrow \mathcal{T} \cup \tau$ 
18:           $\hat{r}_t \leftarrow \hat{r}_t + 1$   $\triangleright$  increment estimated reward
19:   Send estimated reward signal  $\hat{r}_t$  to the RL agent

```

four waves of demons in order for the split demons to appear. In order to prevent missing rewards, we ought to elicit indicator patterns progressively. Thus, we need to know *when to query* crowd workers to elicit new indicator patterns. Meanwhile, as the agent collects a vast amount of experiences while interacting with the environment, it is also crucial to know *how to sample* experience to avoid wasting human effort. This querying process is akin to active learning [87]. However, as opposed to traditional active learning methods [107, 62], our learning agent operates in the context of reinforcement learning instead of supervised learning.

When to query workers The objective of querying is to obtain new indicator patterns in the environment. In the beginning of agent deployment, we ask crowd workers to collect the first batch of indicator patterns in order to bootstrap the training process. Specifically, we let the agent act randomly (choosing random actions in each state) in the environment and collect state information for several episodes as it does not require any human effort.

The state information is then shown to crowd workers in order to elicit indicator patterns using the CVR workflow (see Section 4.4.3).

Given the first batch of indicator patterns, the RL agent will learn to maximize the total reward, i.e., to learn a policy that triggers as many reward events as possible. If the agent obtains the same amount of reward events consistently, we consider that the training instance has converged under the current batch of indicator patterns, and therefore, halt the training process. In practice, we record the total number of reward signals the agent receives per episode (episode reward), and then calculate the average value in the past 100 episodes. If the difference between the current average episode reward and the previous one is less than a predetermined threshold, ϵ (default: 1), we stop the training instance and launch the next round of elicitation to obtain a new batch of indicator patterns. It is worth noting that this query process could be repeated indefinitely until the user deems that the agent has completed a task or learned a skill. The interactive and progressive nature of our query approach allows a reward function to evolve over time so that humans do not need to specify an optimal reward function prior to agent deployment.

How to sample experience When querying workers for new indicator patterns, we ought to sample a subset of the collected experience since annotating all experience is too costly. In our case, we generate multiple episodes of gameplay with the currently trained agent once the training process halts, and store the gameplay images for elicitation. By applying the existing indicator patterns to the collected episodes, we find the episode with the *fewest* detected indicator patterns for crowd workers to annotate. Hence, we can avoid wasting human effort on annotating the reward events that were already discovered in previous rounds of elicitation.

Alternatively, we can sample individual gameplay images that have no indicator patterns instead of an entire episode, in which case it might be less costly for workers to annotate. However, individual gameplay images may not contain enough contextual information, such as temporal causality, for humans to understand the task or the environment. Therefore, it is necessary to utilize an entire episode for crowd workers to elicit indicator patterns.

4.4.3 Crowdsourcing Workflow

In order to elicit indicator patterns from collected experience, we introduce a new crowdsourcing workflow, Collect-View-Refine (CVR), which can further be generalized for other pattern elicitation tasks. Our workflow can assist crowd workers in obtaining an accurate set of patterns from raw data instances. In our setting, the goal of the workflow is to extract indicator patterns within reward events from gameplay images.

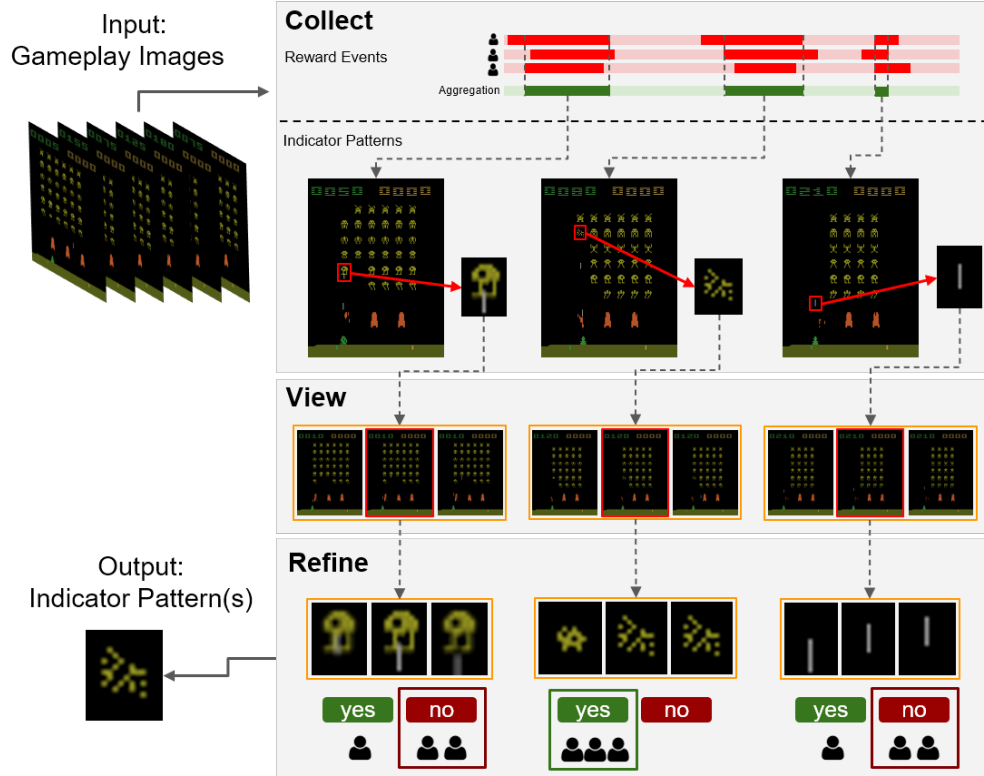


Figure 4.7: The Collect-View-Refine workflow. The Collect step first elicits reward events that may contain indicator patterns from crowd workers and then requests workers to crop indicator patterns within the aggregated reward events. The View step subsequently uses the elicited patterns to search for examples of such patterns in the collected experience. Then, the Refine step asks workers to decide whether to keep indicator patterns using majority voting based on the result of the View step. The final output of the CVR workflow is a set of indicator patterns.

Asking individual crowd workers to perform an entire pattern elicitation task can be cumbersome and error-prone, especially when the collected experience is lengthy. Instead, we split the elicitation task into three steps to ensure the quality of the elicited patterns while alleviating crowd workload. It is worth noting that our workflow can be generalized to tasks beyond visual domains (see Section 4.6).

As the name suggests, CVR is comprised of three steps. The *Collect* step aims to discover reward events and indicator patterns (Section 4.4.3.1). The *View* step searches for examples of the elicited indicator patterns and presents to workers (Section 4.4.3.2). Finally, the *Refine* step allows workers to remove indicator patterns based on the result from the View step (Section 4.4.3.3). Figure 4.7 illustrates each step of the CVR workflow.

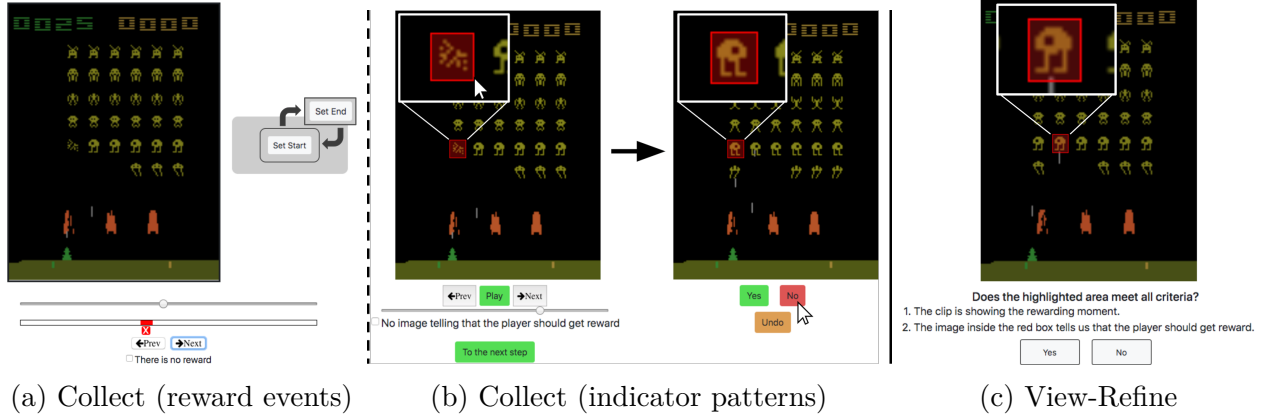


Figure 4.8: Our user interface for the Collect-View-Refine workflow. Left: a worker can click through a gameplay image sequence and create a reward event by clicking the “Set Start” and “Set End” button. The reward event is visualized below the scroll bar. If the worker does not observe any reward, she/he can check “There is no reward”. Center: a worker is first asked to select an image region from a reward event clip for indicator patterns. After selection, the worker needs to verify if the cropped image region contains an indicator pattern. Right: the Refine step asks a worker to determine whether the elicited pattern is correct based on the example found by the View step.

4.4.3.1 Collect

Given a sequence of gameplay images, the Collect step aims to produce a set of indicator patterns. Since there can be a large number of gameplay images in an episode, asking individual crowd workers to find indicator patterns at each timestep is a laborious task. Therefore, to reduce human effort, we first ask crowd workers to select reward events from gameplay images, followed by annotating indicator patterns within the selected reward events (see Collect in Figure 4.7).

The Collect step starts the elicitation task by showing crowd workers a short video clip about the environment. A worker is then given an episode of gameplay images. As Figure 4.8a illustrates, the worker can scan through the episode using a temporal bar. If a reward event is found, the worker can click the “Set Start” and “Set End” button to mark the start and the end of the reward event, respectively. To minimize missing reward, we instruct workers to select as many reward events as possible. Moreover, we recruit multiple workers to collect reward events for each episode and aggregate them by taking the majority vote on each timestep to avoid erroneous annotations.

Once the reward events are selected, the Collect step subsequently elicits indicator patterns from gameplay images within those events. We recruit a different set of crowd workers for this task. Each worker is assigned one reward event, and each event is annotated by

multiple workers. As Figure 4.8b shows, we provide an image cropping tool in our interface to enable the worker to draw a red box in order to crop the image region containing an indicator pattern. Then, we highlight the cropped region in all images within the current reward event and ask the worker to verify whether these regions are indicator patterns.

To aggregate the annotations, we first determine whether a gameplay image contains an indicator pattern by checking if the majority of workers chose to crop it. Then, we aggregate the cropped image regions for each selected gameplay image by taking the majority vote on each pixel. Finally, to remove duplicates after the majority voting, we compare each cropped region with the rest using template matching. If two cropped regions are matched, we remove the larger one. The output of the Collect step is a set of cropped image regions that can serve as indicator patterns.

4.4.3.2 View

Once an initial set of indicator patterns is acquired, the View step finds examples of the elicited patterns to present to crowd workers (see View in Figure 4.7). Due to the nature of pattern matching, an erroneous pattern could generate numerous false rewards. Thus, the View step allows the workers to see the potential outcomes of applying these patterns in the environment.

To find such examples, we employ the same template matching method from Section 4.4.1 to detect matched gameplay images within the collected experience. However, asking workers to view all matched gameplay images can be costly since some patterns may frequently appear in the environment. Instead, the View step looks for examples that suggest the collected indicator patterns are likely to be false. Since a reward event is often composed of multiple consecutive indicator patterns, a true indicator pattern is usually accompanied by other indicator patterns within the event. In other words, an indicator pattern is probably false if it appears by itself. Therefore, we aim to find one example of each indicator pattern that is most isolated from other patterns. To this end, the View step first finds all matched gameplay images in the collected experience for each indicator pattern, excluding the gameplay images from which the indicator patterns are cropped. Then, for each matched gameplay image, we form a three-timestep example of the indicator pattern using the matched image with two images in its adjacent timesteps from the collected experience. After performing template matching on this example using all indicator patterns, we count the total number of matched patterns in the example. Finally, by ranking all examples' numbers of matched patterns, the View step chooses the one with the minimum count as the example of the indicator pattern and subsequently presents to crowd workers.

4.4.3.3 Refine

Given the results of the View step, the Refine step asks crowd workers to determine whether the elicited patterns indeed indicate rewards. As Figure 4.8c shows, an image region is highlighted by the red box, and a worker can examine this region and decide if it is an indicator pattern. For each cropped image region, multiple workers are recruited, and the Refine step will remove the image region from the indicator pattern set if the majority of the workers choose “No” (see Refine in Figure 4.7). Since the View and Refine step are only meaningful when working jointly, we denote these two steps as the *View-Refine* step in the following sections.

Although we opt to use majority voting to refine indicator patterns, the Refine step can be extended to accommodate more elaborate operations, such as resizing and shifting, to make indicator patterns more accurate.

4.5 Experiments

In this section, we first present the details of crowdsourcing data collection and RL training experiments, followed by the evaluation metrics and the results of each component in RECEPT as well as the system as a whole.

4.5.1 Data Collection

We recruited unique workers from the U.S. with an acceptance rate of 95% for all data collection tasks. For reward event elicitation in the Collect step, we recruited five crowd workers for each episode, which is a total of 50 workers for ten episodes per game. Each worker was paid \$0.70 in Space Invaders and \$1.00 in Atlantis, as the lengths of episodes vary in different games. For indicator pattern elicitation in the Collect step, we assigned five workers for each reward event, and each worker was paid \$0.60. Finally, for the View-Refine step, we recruited another five workers to watch and verify each indicator pattern, and each of them was paid \$0.30. All payments were set to meet an hourly rate of \$10.00.

Demon Attack is a multi-level game such that the appearance of indicator patterns may vary at different levels. To elicit indicator patterns for Demon Attack, we ran multiple rounds of elicitation. In the first round, we adopted the same procedure used in Space Invaders and Atlantis. For each of the following rounds, we gathered ten episodes of gameplay images of the agent that was trained using the elicited patterns. Within the ten episodes, we chose the episode with the lowest number of detected indicator patterns. We paid each worker \$0.80 for reward event elicitation in the Collect step. For the rest of the steps in the workflow, we

paid the same amount as we did in Space Invaders and Atlantis.

In Space Invaders, crowd workers spent a total of 15.6 hours completing tasks, and the total cost was \$114.50. In Atlantis, crowd workers spent a total of 36.6 hours completing tasks, and the total cost was \$263.00. For all three rounds of elicitation in Demon Attack, crowd workers spent a total of 38.3 hours completing tasks, and the total cost was \$328.50.

Using the CVR workflow, we elicited 12 indicator patterns for Space Invaders and 43 patterns for Atlantis. As for Demon Attack, we applied the querying technique introduced in Section 4.4.2. We first elicited 17 indicator patterns to start RL training and stopped at the three millionth timestep as the agent ceased to receive more reward per episode. Then, we elicited another 12 indicator patterns given the trained agent’s gameplay images to update the reward function. Afterwards, the training process was resumed and stopped at the 9.6 millionth timestep. We obtained an additional eight indicator patterns for the final round of elicitation. As a result, a total of 37 indicator patterns were elicited for Demon Attack.

4.5.2 Results

To evaluate both the CVR workflow and the reward generation component in RECEPT, we first present the metrics for each component in the CVR, followed by the results of the workflow and finally the performance of RL training using RECEPT in Space Invaders, Atlantis, and Demon Attack.

4.5.2.1 Collect-View-Refine

We manually annotated ten episodes of a random agent’s gameplay in both Space Invaders and Atlantis in order to obtain the ground truth for evaluating reward event elicitation in the Collect step. The ground truth reward event starts at the timestep when a ground truth reward signal, produced by the oracle (ground truth) reward function, occurs and ends at the timestep when the last indicator pattern of the same reward appears.

We assess the performance of reward event elicitation in the Collect step using two metrics—precision and recall. Precision measures how many of the crowd-annotated timesteps are correct among all crowd-annotated timesteps, whereas recall measures how many of the ground truth timesteps are recovered by crowd annotations, such that

$$precision = \frac{tp}{tp + fp} \quad recall = \frac{tp}{tp + fn}$$

where tp , fp , and fn are defined in Section 4.3. Note that we calculate precision and recall on a *per timestep* basis for reward event elicitation because per-timestep comparison

allows us to better measure the temporal difference between ground truth events and the crowd-annotated events.

In Space Invaders, the precision is 0.73, and the recall is 0.74. In Atlantis, the precision is 0.81, and the recall is 0.87. The recall values show that crowd workers are able to find most of the ground truth reward events in the gameplay image sequences, and the precision values show that the majority of the elicited timesteps are within the ground truth reward events. This result suggests that reward event elicitation in the Collect step is able to provide adequate temporal reward cues for the following steps of the CVR workflow to elicit indicator patterns.

Next, we measure the performance of indicator pattern elicitation in the Collect step as well as the final output of the CVR workflow. Although it’s infeasible to find all indicator patterns in the game environments, we are able to use the product of indicator patterns to evaluate their quality. As mentioned in Section 4.4.1, indicator patterns can form indicator tracks to produce reward events. Hence, we can instead compare the generated reward events to the ground truth reward events.

In this setting, rather than using per timestep basis comparison, we use *per event* basis comparison for evaluating indicator pattern elicitation in the Collect step and the View-Refine step due to the fact that the timestep-wise difference is less important when comparing events. Thus, true positive (*tp*) occurs when a generated reward event has at least one timestep overlap with a ground truth reward event, whereas false positive (*fp*) occurs when a generated reward event has no overlap with any ground truth event. To obtain the ground truth, we annotated an additional 20 episodes of random gameplay in both Space Invaders and Atlantis. The metrics we use to evaluate the results are the same ones in Section 4.3.2—FNR, FPR, duplicate, and delay, as they can measure the impact of reward functions on RL training.

When inspecting the output of indicator pattern elicitation in the Collect step, we noticed that some crowd workers cropped image regions that are entirely background pixels, e.g., all black pixels. These background image regions are detrimental to template matching as they can trigger any gameplay image that has background pixels, which further underlines the importance of having the View-Refine step to remove the background image regions.

To investigate whether crowd workers are able to crop meaningful indicator patterns in the Collect step other than the few background image regions, we provisionally discard the background regions for the sole purpose of calculating the metrics. Note that these background regions are not removed from the indicator pattern set, which is the input to the View-Refine step. The first rows of Table 4.1 and 4.2 are the results of indicator pattern elicitation in the Collect step without background image regions in Space Invaders and

Space Invaders	FNR	FPR	duplicate	delay
Collect (w/o bg patterns)	0.02	0.17	3.41	0.95
View-Refine	0.04	0.0	0.85	2.32

Table 4.1: Performance of indicator pattern elicitation in Space Invaders

Atlantis	FNR	FPR	duplicate	delay
Collect (w/o bg patterns)	0.06	0.15	1.23	1.76
View-Refine	0.07	0.0	0.20	1.89

Table 4.2: Performance of indicator pattern elicitation in Atlantis

Atlantis. The low FNR values (0.02 and 0.06) show that the Collect step identifies most of the ground truth reward events, whereas the FPR values (0.17 and 0.15) suggest that some false indicator patterns exist in the indicator pattern set, and they need to be eliminated in the View-Refine step. The duplicate and delay values indicate that they may not have a significant impact on the learning performance of an RL agent, according to Section 4.3.2.3 and Section 4.3.2.4.

The second rows of Table 4.1 and Table 4.2 are the results of the View-Refine step. Compared to the results of the Collect step, we found that the View-Refine step reduces the FPR values to 0.0 and causes the duplicate values to drop as well, which would improve the learning performance of an RL agent as shown in Figure 4.3 and Figure 4.4. This also suggests that all background image regions were eliminated by the View-Refine step. The FNR values are slightly higher than that of the Collect step because the View-Refine step may remove a small number of correct indicator patterns, which also explains the higher delay values. However, small increases in FNR and delay do not have a significant effect on RL training. These results show that the View-Refine step can effectively remove false indicator patterns after the Collect step and preserve most of the correct indicator patterns.

We have demonstrated that our CVR workflow is able to achieve minimal FNR and FPR values with acceptable duplicate and delay values, which lays a solid foundation for using elicited indicator patterns to generate reward functions so as to train RL agents.

4.5.2.2 RL Training using RECEPT

We use the learning curve plots and AULC, introduced in Section 4.3.1, to measure the performance of reward functions generated by RECEPT.

Figure 4.9 shows the learning curves of A2C agents trained by RECEPT and the oracle reward functions in Space Invaders and Atlantis. The green curves represent the agents

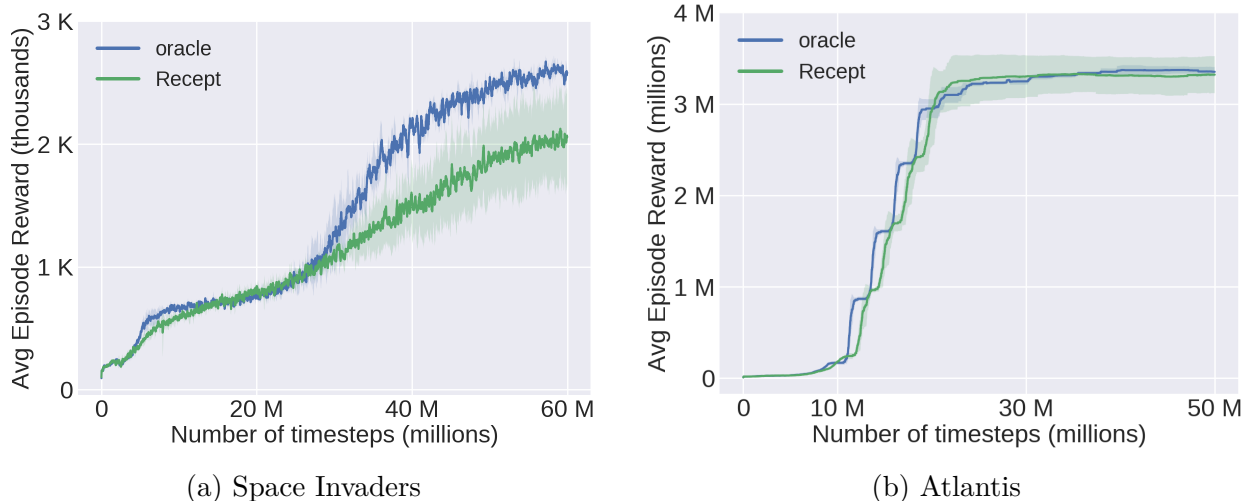


Figure 4.9: The blue curves represent the learning performance of A2C agents trained by the oracle reward functions. The green curves represent the learning performance of A2C agents trained by reward functions generated by RECEPT. Each curve is the averaged over three training instances, and the shaded area represents one standard deviation.

trained by RECEPT, whereas the blue curves represent the agents trained by the oracle reward functions. In Space Invaders (Figure 4.9a), the agent trained by RECEPT achieves 80% of the AULC of the agent trained by the oracle. The two agents perform similarly before 30 million timesteps, but the agent trained by the oracle learns more efficiently afterwards. We suspect one of the reasons is because sometimes an enemy sprite may disappear immediately after a target hit, instead of showing explosion patterns in the subsequent timesteps. RECEPT cannot produce reward signals without observing any visible pattern. This issue will increase in severity as the agent learns to hit targets more frequently and thus leads to reduced learning efficiency. In Atlantis (Figure 4.9b), the game environment displays explosion patterns consistently after target hits, so the agent trained by RECEPT achieves 98% of the AULC of that trained by the oracle, which indicates that RECEPT is able to provide sufficient reward signals to train an RL agent.

Figure 4.10 shows the result of progressive pattern elicitation and RL training using RECEPT in Demon Attack. Unlike Space Invaders and Atlantis, Demon Attack requires multiple rounds of indicator pattern elicitation since some indicator patterns will not appear until the agent reaches a certain level. Thus, we do not compare RECEPT with the oracle reward function in this setting. The colored curve segments in Figure 4.10 show different rounds of training instances in which RECEPT elicits indicator patterns to generate reward functions and subsequently trains the agent. Both Figure 4.10a and Figure 4.10b illustrate the performance of the same training instances but with different measures. In Figure 4.10a,

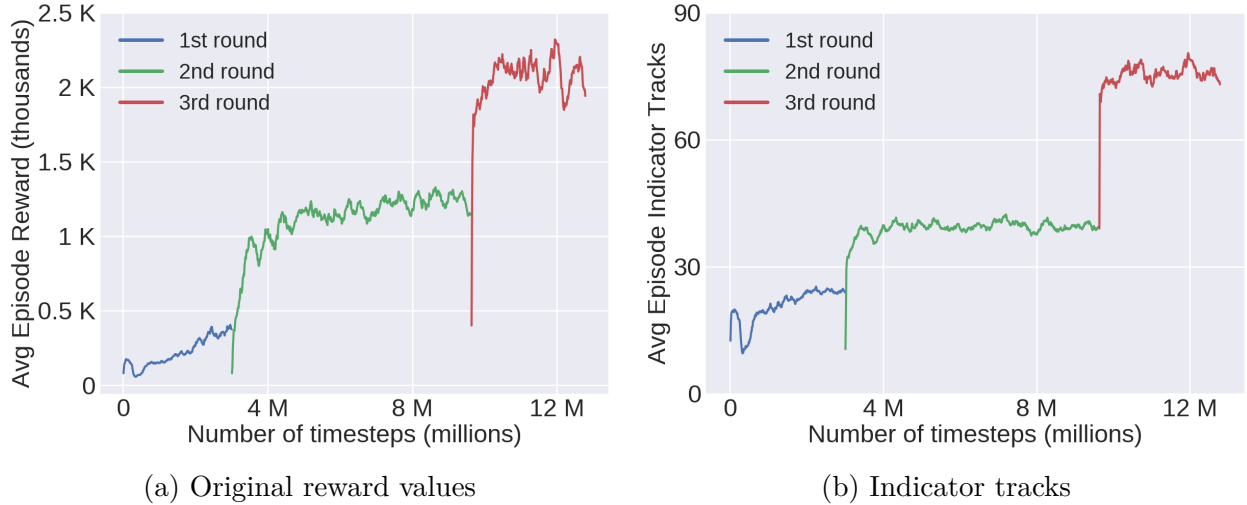


Figure 4.10: Learning performance of an A2C agent trained by RECEPT in Demon Attack. Each colored curve segment represents a round of training instance in which RECEPT elicits a set of indicator patterns from crowd workers to generate reward functions and subsequently produces reward signals to train the A2C agent. The left figure shows the learning curves plotted using the original reward value. The right figure shows the result of the same training instances but using the number of generated indicator tracks (reward events).

we plot the learning curves using the original reward value, whereas in Figure 4.10b, we instead use the number of generated indicator tracks, which are detected reward events.

As Figure 4.10b shows, the RL agent can learn to hit more targets after RECEPT updates the reward function with the newly-acquired indicator patterns. Correspondingly, the game score is also getting higher, as shown in Figure 4.10a. The sharp performance drop at the beginning of the second and the third round in Figure 4.10a is because when the agent resumes training in Demon Attack after each round of elicitation, the game starts at the first level where there are fewer enemy sprites and each sprite has lower reward values, compared to the level at which the agent was before the training instance was halted.

Figure 4.10 demonstrates that our RECEPT system, derived from PRFG and the design goals in Section 4.3, can update the reward function progressively as the agent gets better at playing the game. This property is crucial in many real-world task environments since it enables us to gradually train RL agents as they complete their intermediate goals.

4.6 Discussion

In this work, we demonstrated RECEPT using Atari 2600 games, which rely primarily on the visual channel for feedback. However, it is important to contextualize PRFG within

the broader scope of multi-modal environments. We define two properties—channel and mode—to better describe the scope of our approach and system.

- A *channel* is a source of sensory information from the environment, such as an RGB camera. If, for example, an autonomous vehicle is equipped with multiple cameras, the corresponding state space is *multichannel*.
- A *mode* describes a type of sensory information from the environment. Similarly, an autonomous vehicle can utilize multi-modal inputs, such as those from RGB cameras, LiDAR, and audio sensors.

Note that Atari 2600 game environments in OpenAI Gym do not have audio feedback.

- A *multi-modal state space must be multichannel* since the data of different modes are from different sensors. Case in point, an autonomous driving vehicle can have multiple different sensors, such as RGB cameras and sonars, installed to help the vehicle navigate. If the vehicle has two RGB cameras and one sonar, there are two different modes and three channels in total in the state space.

Reward events may exist in different channels. We call a channel an *event channel* if it contains at least one reward event. We call an event channel *complete* if it encodes an optimal reward function for a task. Multiple *incomplete channels* can be used in concert to capture more of the available reward signals.

We consider two event channels *isomorphic* if there exists a one-to-one relationship between the occurrences of their reward events. A perfect mapping would suggest that they have maximum mutual information, meaning that the second channel adds no new signal. For instance, Arcade game environments may have multiple event channels since reward events can be found in two channels of different modalities, i.e., visual and audio, and both channels are complete and isomorphic. We have demonstrated that our system, RECEPT, is able to elicit and detect indicator patterns in visual channels to produce reward signals.

4.7 Limitations and Future Work

Although PRFG can operate regardless of whether the environment is multi-modal or multichannel, it requires that the indicator patterns be *non-overlapping*. If a reward event can trigger multiple indicator patterns with *distinct* semantic meanings, such indicator patterns are *overlapping*. For instance, in many shooting games, a player is often rewarded extra points for hitting targets in a row. These targets have the exact same appearance, i.e., they share the same indicator pattern, yet they may yield different reward values (different semantic meanings) due to different preconditions. In this case, our approach cannot generate appropriate reward functions for such games as we assume indicator patterns are

non-overlapping so that the same indicator pattern would have the same semantic meaning in order to produce the same reward signal under any circumstance.

To deal with overlapping indicator patterns, we would need a new type of indicator patterns—*contextualized* indicator patterns—to enable the specification of conditions, dependencies, and other relationships between indicators. For example, a contextualized indicator pattern could have rules for temporal dependency such that it will not be detected unless some other events occur first. Or, an indicator pattern may depend on another pattern to trigger the reward signal. Therefore, to solve the overlapping challenges in our shooting game example, we ought to specify reward values for each indicator pattern based on the number of hits that have occurred before. Furthermore, contextualized indicator patterns can be used with existing work that enables the specification of temporal or spatial relationships, such as interval algebra [4] and scene graphs [43]. This would allow PRFG to go beyond the current limitations and adapt to more complex tasks and environments.

4.8 Summary

Designing proper reward functions poses a major challenge for enabling reinforcement learning in the real world. To this end, we introduce a new human-in-the-loop approach, PRFG, that generates reward functions via indicator patterns. To better understand the impact of reward functions on RL training, we conduct an error characterization study to set design goals for creating pattern-based reward functions. Based on PRFG and the characterization study, we implement RECEIPT, a system that elicits indicator patterns in visual domains from crowd workers and produces reward signals by detecting indicator patterns in the environment. To make pattern elicitation robust and accurate, we present a new workflow, Collect-View-Refine, to guide crowd workers. RL agents trained by RECEIPT in two Atari 2600 games perform 80% and 98% as well as those trained by the ground truth reward functions. The experiment of progressive pattern elicitation and RL training with Demon Attack further showcases RECEIPT’s ability to generate reward functions in an evolving environment, which is akin to many real-world task environments.

RECEIPT shows the potential for using a pattern-based approach to provide scalable feedback for RL agents, and we posit that our approach, PRFG, can be extended to other domains so long as reward events exist in human-perceivable channels.

In Part I, we established methods for constructing reward functions that provide RL agents with clear optimization objectives. However, the existence of reward signals alone solves only half of the problem: while rewards define what agents should achieve, they provide no inherent guidance on how to achieve them. Agents must still discover successful

strategies through exhaustive environment interactions—a process that remains computationally expensive and often impractical for real-world deployment.

This limitation stems from RL’s trial-and-error paradigm, where agents must explore vast state-action spaces to uncover meaningful reward pathways. The absence of structural guidance means agents may waste resources exploring irrelevant states or rediscovering known solutions. Even with well-designed rewards, this exploration bottleneck persists as a key barrier to efficient learning. In Part II, we address this limitation through the concept of *relevance*.

Part II

Relevance-driven Efficient Learning

Causality dictates the ultimate outcomes of an RL agent’s actions¹, yet RL agents bypass the immense complexity of learning the underlying causal rules. They pragmatically learn a mapping from states to actions that maximizes long-term reward. However, this outcome-centric approach comes at a steep cost: poor sample efficiency. Because the agent does not understand *why* things happen, it must explore its environment extensively through trial and error. Without prior knowledge of the environment’s causal mechanisms, the agent must stumble upon and contrast countless combinations of states and outcomes to isolate the factors that truly matter. This brute-force process of elimination is inherently slow and data-intensive. While RL guarantees an optimal policy with enough data, the staggering data requirements and associated training costs often make it impractical for real-world deployment.

Consider the challenge of training an autonomous vehicle to handle rare but critical scenarios. A pure data-driven, “tabula rasa” RL approach would be fundamentally constrained by data. It would require massive datasets capturing every possible variation of these rare events, which are prohibitively expensive and dangerous to collect. The agent would need to encounter and learn from these critical situations through exhaustive exploration of a vast state-action space, making the process slow, costly, and unsafe. This danger is exemplified by real-world failure modes like the 2023 Cruise autonomous vehicle (AV) incident, where a pedestrian was struck by a human-driven car and thrown into the path of an AV². The AV detected a collision but could not sense the pedestrian’s location, as she was trapped underneath the vehicle. Following its pre-programmed instructions to clear the traffic lane after a collision, the AV pulled over, dragging the pedestrian 20 feet. Because such a specific sequence of events—a third-party collision projecting a victim into a blind spot—is a statis-

¹The transition function in an MDP implicitly represents the causal relationships within the environment

²<https://www.sfchronicle.com/projects/2024/cruise-sf-collision-timeline/>

tical anomaly that almost never appears in training data, a data-driven RL agent effectively has zero experience to draw upon.

Faced with this data requirement problem, the alternative would be to build a complete causal model a priori by specifying every causal relationship: how siren volume indicates distance, how flashing lights signal priority, and how the vehicle’s approach direction (behind vs. oncoming) combined with road infrastructure dictates the correct response. A proper model would capture the underlying rationale, such as why yielding is necessary for a vehicle approaching from behind but not for an oncoming one separated by a median. Unfortunately, for any complex environment, constructing such a comprehensive model—essentially a complete “rulebook” of how the world works—is itself infeasible. The level of detail required to specify all causal mechanisms governing every situation is unattainable in practice.

We are thus caught between two impractical extremes. On one end, a data-driven RL agent that is flexible but requires a vast amount of experience. On the other, a meticulously engineered causal model that is data-efficient but requires an unattainable level of prior knowledge. To make RL practical, we need a combined approach that integrates the data efficiency of causal guidance with the flexibility of data-driven learning.

This is where human intuition offers a crucial clue. While people may not be able to articulate every causal rule, they can instantly identify which factors matter. We intuitively know that sirens, flashing lights, and the direction of approach are relevant signals for deciding whether to stop or yield. We do not need a complete causal model; instead, we learn how to act based on what we expect to be the most relevant information for the situation, given our understanding of the environment and objectives.

Therefore, we aim to tap into this intuition to provide a powerful, practical guide for RL agents to effectively narrow the “search space” of RL policies. Rather than exhaustively exploring all possibilities with equal likelihood, we can make RL systems more sample-efficient by guiding exploration toward states that could significantly impact outcomes while reducing emphasis on less relevant alternatives, thereby accelerating policy convergence and enabling practical real-world deployment.

In Part II, we introduce two methods for leveraging human intuition to extract early-stage relevance signals. We first formally define relevance as a new concept for improving RL. This concept manifests in two temporal forms: a retrospective form that focuses on credit assignment by identifying relevant past steps, and a prospective form that focuses on anticipating future scenarios. Chapters 5 and 6 explore these distinct perspectives in depth.

Definition of Relevance

We define relevance, $H(s)$, as *the magnitude of influence a state (or its components) is estimated to have on a specific outcome*. In other words, relevance describes the degree to which a state s shifts the likelihood of an outcome Ω from its baseline probability, weighted by the associated reward. To account for the fact that both large positive rewards and large negative rewards indicate high relevance, we define $H(s)$ using the absolute value:

$$H(s) = \left| \left[P(\Omega \mid s) - P(\Omega) \right] R(\Omega) \right|, \quad (4.1)$$

where:

- Ω is an outcome (a state or set of states with nonzero reward),
- $P(\Omega \mid s)$ is the conditional probability of Ω given s ,
- $P(\Omega)$ is the prior probability of Ω ,
- $R(\Omega)$ is the reward for Ω .

Interpretation of Relevance

The term $P(\Omega \mid s) - P(\Omega)$ captures the *marginal change* in probability due to state s : it measures how much observing state s increases (or decreases) the probability of Ω compared to the default expectation. Scaling this difference by $R(\Omega)$ ensures that relevance accounts for the practical significance of the outcome. For example, a state that marginally increases the probability of a high-reward outcome may be more relevant than one that significantly influences a low-reward outcome. By quantifying this influence, relevance enables RL agents to focus on high-impact states without needing to specify the causal rules of the environment.

Extending to State-Action Pairs

While Equation 4.1 focuses on states, many practical applications require assessing the relevance of specific *decisions* rather than just situations. This distinction is crucial: knowing *which states* are relevant is different from knowing *which actions* are relevant within those states. For the retrospective analysis in Chapter 5, where we evaluate entire trajectories of agent behavior, we need to assess the relevance of each state-action pair to determine which specific decisions were critical to the outcome.

To address this, we extend our relevance framework to state-action pairs by incorporating transition dynamics:

$$H(s, a) = p(s' \mid s, a)H(s') \quad (4.2)$$

where:

- $H(s')$ is the next state’s relevance as defined in Equation 4.1,
- $p(s' \mid s, a)$ is the transition probability.

This formulation evaluates an action’s relevance based on both its likelihood of leading to relevant future states and the relevance of those states. It allows us to distinguish between relevant actions that significantly impact outcomes and irrelevant actions that do not, even when taken in the same state.

Relevance in Retrospective Form

Temporal credit assignment remains one of the most critical challenges in RL. While RL traditionally employs recency weighting—assigning credit to past state-action pairs based on their proximity to rewards—this approach often suffers from slow convergence due to its reliance on iterative updates, which can lead to high sample complexity, particularly in sparse-reward tasks. To mitigate this, we use relevance *retrospectively* to integrate human insights into the credit assignment framework, thereby improving sample efficiency. Unlike RL agents, humans can leverage their prior knowledge about causality to identify relevant steps that are critical for achieving a desired outcome, even when these events are distant from the final reward. Moreover, they naturally direct their learning attention to the same type of outcome-determining factors that relevance captures. This ability allows us to shorten the effective length of an episode, enabling the RL agent to focus only on the most impactful steps.

In Chapter 5, we introduce *Montage*, a system that assigns relevance to each state-action pair in the agent’s experience such that a state-action pair is deemed irrelevant and thus excluded from policy updates if its contribution to an outcome Ω (as perceived by a human) is below an implicit threshold set by human annotators based on their understanding of the task environment.

Relevance in Prospective Form

Although retrospective relevance improves sample efficiency by pruning irrelevant steps, it faces two key limitations. First, it requires collecting agent experience before relevance

can be applied. This creates a chicken-and-egg problem where agents must first explore inefficiently. Second, its reliance on human annotations of past trajectories limits scalability. Consequently, Montage operates retrospectively on existing data without enabling proactive exploration guided by human insight.

We introduce *CARAFE* in Chapter 6, which addresses these limitations by shifting from retrospective analysis to *prospective* relevance prediction. To illustrate the necessity of this shift, consider again the Cruise AV incident. A human driver, utilizing prospective reasoning at the moment of collision, identifies that the unobserved location of the pedestrian is a critical unknown. They can anticipate that the action of moving the car would lead to a high-relevance, negative outcome. By capturing this anticipatory ability, CARAFE enables RL agents to: (1) estimate relevance in unseen states, and (2) reduce dependence on continuous human review. This approach allows agents to effectively skip irrelevant intermediate steps and focus on long-horizon implications. Unlike Montage, which applies relevance assessment to collected data, CARAFE learns a model that encodes human similarity judgments so as to create a representation space where proximity indicates similarity in potential outcomes.

Together, these two chapters demonstrate how both forms of relevance can improve sample efficiency in RL: retrospective analysis refines temporal credit assignment by identifying relevant past steps, while prospective relevance enables representation learning that highlights critical features in future states.

The work in Chapter 6 demonstrates the promise of prospective relevance while revealing the challenge of extracting clean signals from inherently noisy human input. Rather than concluding this approach, these findings shed light on the path forward, highlighting the need for more sophisticated signal extraction techniques. As a first attempt, this work establishes a foundation for future research in relevance-driven RL.

CHAPTER 5

Improving Sample Efficiency by Identifying Relevance

5.1 Motivation

The goal of RL algorithms is to learn a policy (actions to take in each state of the world) that maximizes the total long-term reward in the environment. However, the *sample efficiency* problem still poses a challenge to RL systems. This is particularly important in settings where large numbers of interactions during training are not possible, which is usually the case when a task cannot be simulated authentically. For most real-world tasks, such as medical treatment strategies in the healthcare domain, acquiring a large amount of data via direct interaction with the real environment can be costly, infeasible, and even dangerous. One of the reasons behind this sample efficiency problem is *temporal credit assignment*, i.e. assigning reward signals to past actions taken by an agent. This delay is generally detrimental to the learning process, but it is also ubiquitous for real-world tasks. Therefore, improving temporal credit assignment is a critical step towards achieving sample-efficient RL.

While people may not be adept at reasoning over causal relations explicitly, they possess intuitive insights into causality in many tasks. We aim to leverage this ability to enable better credit assignment, and thus improve sample efficiency for RL. Unlike RL agents that compute credit based on every step, humans can intuitively identify key steps (state-action pairs) that are essential for an outcome within a long sequence of agent experience. This allows RL agents to effectively shorten the credit assignment trace, focusing on the most influential state-action pairs ($H(s, a)$ in Eq.4.2).

In this chapter, we propose *Selective Memory Learning* (SML), an algorithm to enable relevance-driven credit assignment that alleviates the effects of delayed reward in RL to achieve better sample efficiency. Our framework is based on the Monte Carlo (MC) on-policy control method, which collects an episode of the agent interacting with the environment before updating its policy, allowing humans to annotate the collected experience by eliminating irrelevant steps.

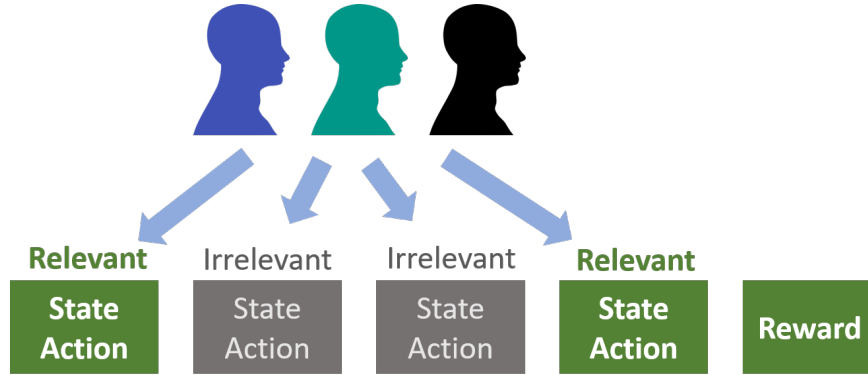


Figure 5.1: Montage: Human-powered, relevance-driven credit assignment via Selective Memory Learning (SML).

We validate the efficacy of our approach through a series of controlled simulations that allow us to vary the setting and budget available to the agent. Our results demonstrate that our approach can significantly accelerate training even with a limited budget. We then create a web interface that enables crowd workers to perform our credit assignment task, and run a case study showing that not only are people able to perform this task well in our toy domain, but also that any remaining errors do not eliminate the benefit of our approach.

Our contributions are three-fold:

- We design a reinforcement learning framework that incorporates relevance-driven credit assignment as part of the learning process (as shown in Figure 5.1).
- We validate our approach using a suite of simulation studies that vary environment settings and training budgets in a controlled manner.
- We conduct a case study with crowd workers that demonstrates people are able to successfully accelerate the training of an RL agent.

More broadly, this work demonstrates a novel approach for improving temporal credit assignment through collective intelligence. Our results show that Montage enables efficient training of RL agents, paving the way for human-in-the-loop RL methods that can support real-world deployment.

5.2 Related Work

In this section, we introduce previous work on human-in-the-loop RL, followed by how crowd intelligence can be used to provide scalability and enhance human-AI integration.

5.2.1 Human-in-the-loop RL

Integrating RL with human intelligence to create better RL agents has been the focus of much work. Imitation learning uses supervised learning to learn a policy that mimics human demonstration [8]. Apprenticeship learning derives a reward function from observed behavior, such as expert demonstrations [1, 71]. Ross et al. introduce Dataset Aggregation (DAgger) to allow imitation learning agents to be iteratively trained from human demonstrations [80]. Hester et al. proposed Deep Q-learning from Demonstration (DQfD) to combine temporal difference updates with supervised classification of the demonstrator’s actions [40]. Those methods rely on human demonstration to train an agent. However, in many real-world tasks, acquiring a large amount of human demonstration may be infeasible due to cost.

Another type of approaches to utilize human feedback is by eliciting human preferences from comparing the agent’s behaviors to approximate reward functions. [19, 42, 3]. Those techniques require humans to rank RL agents’ learned policies. For tasks that demand complex reasoning and expertise, it may be challenging to provide consistent and scalable human feedback.

Humans can also provide feedback to an agent’s behavior to modify, or shape, either reward functions or policies [70, 47, 103, 32]. They are often used to tackle the *sparse reward* problem and provide additional rewards to guide the agent’s learning process. TAMER enables humans to evaluate the agent’s actions while the agent is interacting with the environment in order to learn a predictive model of human reward [47]. Warnell et al. further extend this framework to deep reinforcement learning by employing a deep autoencoder to extract features from visual state [103]. Policy shaping instead asks human trainers to evaluate and modify the agent’s behavior [32]. Our proposed method, in contrast, does not modify reward signals from the environment or inject extra information into the policy during RL training. SML instead removes irrelevant state-action pairs with respect to reward signals from the agent’s experience before it performs policy updates, which allows SML to be able to integrate with aforementioned approaches.

5.2.2 Crowd-based Data Augmentation

The need for exploration in RL settings combined with the temporal credit assignment problem makes it challenging for RL agents to be trained in feasible time. For some domains, such as medicine, a “complete” dataset can never be acquired due to the controlled nature of exploration. Prior work in non-RL settings has applied crowd-based methods to reduce the data needs, and augment existing data to provide a more complete picture. For example, ARchitect [55] relies on crowd workers to capture dependencies and relationships between the

actions being taken in an activity recognition setting. Via this crowd-based approach, they are able to collect seven times the number of examples over purely observational methods when training for activity recognition. Further augmenting crowd-based responses, Pujara et al. [75] propose a method for labeling Twitter data that first uses existing feature labels to collapse relevant categories, and then has crowd workers supervise the annotation of these categories or regrouping as necessary.

More generally, there exist crowd-based approaches towards augmenting data in domains with data sparsity issues. These include crisis informatics (e.g., [65, 96]), human activity recognition (e.g., [12, 29, 46]), real-time audio and video transcribing (e.g., [38, 53]), taxonomy generation (e.g., [18]), etc. Here, we propose applying similar approaches for augmenting and filtering data, but in the context of RL training setting. RL training data is particularly sparse, and temporal credit assignment is not always a feasible way for training RL agents for complex domains in real time. We see crowd-based approaches being relevant towards not only creating reward functions (as done by prior work in RL), but also for honing in on the relevant causal training data to use for a given sparse event, as we propose here.

5.3 Approach

In this section, we first introduce our selective memory learning (SML) algorithm. SML provides the underlying technical component that enables our relevance-driven credit assignment system—Montage.

5.3.1 Selective Memory Learning (SML)

We argue that not all steps in the trajectory are beneficial to the learning process. Often, credit is attributed to steps that are not related to the final outcome [97].

SML is based on an episodic RL agent, where an episode in RL is a sequence of state-action pairs from agent-environment interaction. An episodic RL agent updates its policy, π , at the end of an episode. The experience of an episode is represented as a trajectory $\tau_{R(\Omega)}$, which consists of a sequence of state-action pairs, $[s_t, a_t]_{t \in [t_0, t_T-1]}$, where T is the terminal timestep in the episode and $R(\Omega)$ is the reward outcome at the end of the episode. We assume a sparse reward setting where rewards are only obtained at the end of an episode; therefore, per-step rewards are omitted. Yet, it is trivial to extend SML to work with per-step reward.

At each timestep t , if a state-action pair (s_t, a_t) , produced by the agent, is deemed *relevant* by crowd workers given outcome $R(\Omega)$, it will be appended to a new trajectory $\tau'_{R(\Omega)}$. The resulting trajectory $\tau'_{R(\Omega)}$ is then used to update the agent’s policy. Relevance, in this case,

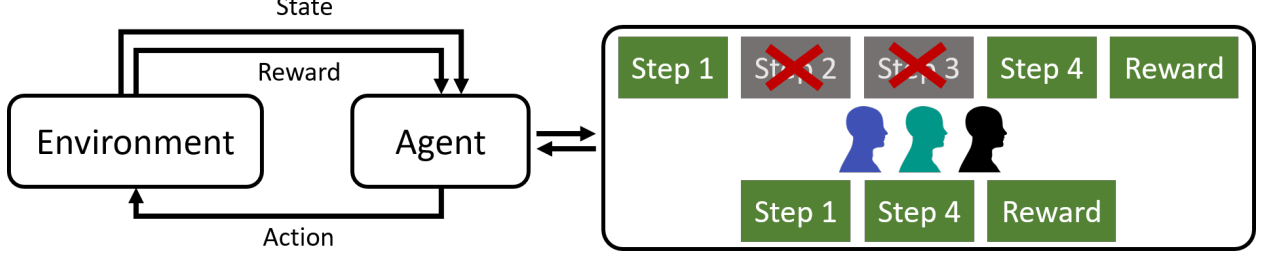


Figure 5.2: An RL agent first collects a trajectory from the environment. Human annotators then review each step in the trajectory and remove steps that are irrelevant to the eventual outcome. The updated trajectory is then used to update the policy.

serves as a gating mechanism: “Does the state-action pair (s_t, a_t) lead to at least ϵ ($\epsilon \in [0, 1]$) portion of the outcome $R(\Omega)$?” Therefore, a state-action pair is relevant to outcome $R(\Omega)$ if it satisfies the following condition:

$$H(s_t, a_t) \geq \epsilon |R(\Omega)| \quad (5.1)$$

Algorithm 2 shows how relevance is used to enable policy update in conjunction with Equation 5.1, as illustrated in Figure 5.2.

Algorithm 2 Selective Memory Learning

Require: Relevance model $H(s, a)$ and a threshold ϵ from human annotators

- 1: Initialize policy π
 - 2: **for each** episode **do**
 - 3: Generate a trajectory $\tau_{R(\Omega)} = [(s_0, a_0), (s_1, a_1) \dots, (s_T, a_T)]$ by following policy π and ending with reward $R(\Omega)$.
 - 4: Initialize $\tau'_{R(\Omega)} \leftarrow$ empty list
 - 5: **for each** timestep $t \in [0, T - 1]$ **do**
 - 6: $(s_t, a_t) \leftarrow \tau(t)_{R(\Omega)}$
 - 7: **if** $H(s_t, a_t) \geq \epsilon R(\Omega)$ **then**
 - 8: $\tau'_{R(\Omega)} \leftarrow \tau'_{R(\Omega)} \cup (s_t, a_t)$
 - 9: Update policy π according to the updated trajectory $\tau'_{R(\Omega)}$
-

Since SML does not modify any signal from the environment, it can complement other human-in-the-loop RL approaches, such as shaping [70, 47, 103, 32] and inverse reinforcement learning [71].

5.3.1.1 Interpretation of Relevance

Given Equation 4.2 and Equation 5.1, we can derive the following:

$$\begin{aligned}
p(s_{t+1} \mid s_t, a_t) H(s_{t+1}) &\geq \epsilon |R(\Omega)| \quad (\text{Equation 4.2}) \\
p(s_{t+1} \mid s_t, a_t) \left| \left[P(\Omega \mid s_{t+1}) - P(\Omega) \right] R(\Omega) \right| &\geq \epsilon |R(\Omega)| \quad (\text{Equation 4.1}) \\
p(s_{t+1} \mid s_t, a_t) \left| \left[P(\Omega \mid s_{t+1}) - P(\Omega) \right] \right| &\geq \epsilon
\end{aligned}$$

Since transitions are observed in the collected trajectory, thus $p(s_{t+1} \mid s_t, a_t) = 1$. Therefore,

$$|P(\Omega \mid s_{t+1}) - P(\Omega)| \geq \epsilon \quad (5.2)$$

Therefore, human annotators act as the relevance model $H(s_t, a_t)$ to examine each state-action pair and determine whether the difference ($|P(\Omega \mid s_{t+1}) - P(\Omega)|$) between the outcome prior and the conditional probability of the outcome given the next state (by observing the transition) is above or equal to a threshold ϵ , based on their knowledge. This formulation intuitively explains how relevance is used to prune irrelevant steps by leveraging people’s ability to recognize relevance.

5.3.2 Montage

Montage is a system that implements SML by integrating an episodic RL agent with the selective memory component. The agent updates its policy using only a curated subset of its experiences—essentially a “highlight reel” of the most impactful steps for a given outcome (hence the name Montage). This is achieved through *relevance assessment*, where human annotators provide binary relevance labels to filter the learning experience.

5.3.2.1 Implementation

MC is a tabular RL method that uses a numerical look-up table to represent its policy [98]. We use this classic RL algorithm as a proof-of-concept implementation for our SML approach. MC employs an action-value table (i.e., Q function) to decide which action to take in the environment at each timestep. The Q function is updated by sampling and averaging the cumulative reward for each state-action pair in the trajectory after the SML component updates the trajectory.

We set the γ value to 1 to disable the discount factor. In each episode, the original trajectory is saved with RGB images of the game screen. Humans can review each step of the trajectory and decide whether the step is helpful to the reward the agent receives in the

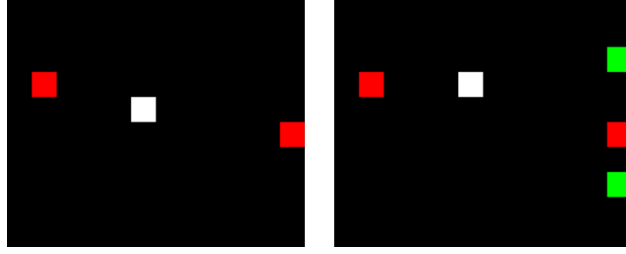


Figure 5.3: DotShooter game environments. Left: One-target game. Right: Three-target game. The red dot on the left of each figure represent the agent, the white dot is the pellet moving from left to right, and the red dot on the right is the *true* target that has reward value 1. In three-target game, the green dots are the lesser targets that have reward value 0.5. The game terminates if one of the targets is hit.

end. Afterwards, the modified trajectory is then used for policy improvement.

5.4 Experimental Setup

We first introduce our customized game environment and its variations, then present our experiment protocol and results.

5.4.1 Game Environment

Our game environment, “DotShooter”, contains two dots, where one is controlled by an agent and the other is the target. The game allows the agent to move up, down, shoot, or perform no operation (noop). The agent can fire one pellet at a time and does not receive reward until the pellet hits the target. Once a target is hit, the agent will receive points. The state space, $\mathcal{S}$, of the environment consists of the vertical locations of the agent and the target, $\{y_a, y_g\}$. The action space $\mathcal{A}$ is {up, down, shoot, noop}.

In order to extensively test the adaptability of our framework, we create four game modes:

- One Target One Hit (1T1H)
- One Target One Shot (1T1S)
- Three Targets One Hit (3T1H)
- Three Targets One Shot (3T1S)

Here, a *hit* indicates that target can only be hit once and the game will terminate. Similarly, one *shot* means that the agent is only allowed to shoot once and the game will terminate if the pellet either hits a target or goes out of the bound. In three-target mode, the game has one primary target, which is worth 1 point and two “distractions”, i.e. targets with smaller

rewards (0.5 points). If one of the targets is hit, the game will terminate. Figure 5.3 shows one-target and three-target game modes. For one-target mode, we further create two extra games with different dimensions (10x10 and 30x30) so as to test whether our framework can handle games with larger state-space. Therefore, there are six versions of “DotShooter” game in total: 10x10 1T1H, 10x10 1T1S, 10x10 3T1H, 10x10 3T1S, 30x30 1T1H, and 30x30 1T1S.

5.4.2 Credit Assignment Framework

The goal of our credit assignment framework is to remove unnecessary state-action pairs in trajectories during training. We create two types of credit assignment frameworks: oracle framework and human-powered framework. For both frameworks, we create a *budget* system to determine how many times the human or oracle can perform the credit assignment task. The framework will only be activated when the agent receives a reward. The budget will decrease until it is zero. Afterwards, the framework will stop the training process.

5.4.2.1 Oracle Credit Assignment

We devise an *oracle* to simulate human response. Our oracle Montage is based on the optimal policy π^* . Essentially, knowing where the target is, actions following π^* will align the agent with the target first and shoot a pellet. Hence, the oracle reviews trajectories of collected experience and locates all the state-action pairs that do not comply with the optimal policy, π^* . Then, the oracle will remove those pairs from the trajectory. Oracle is only activated when the agent receives a reward. Otherwise, the entire trajectory will be used for policy improvement.

5.5 Results

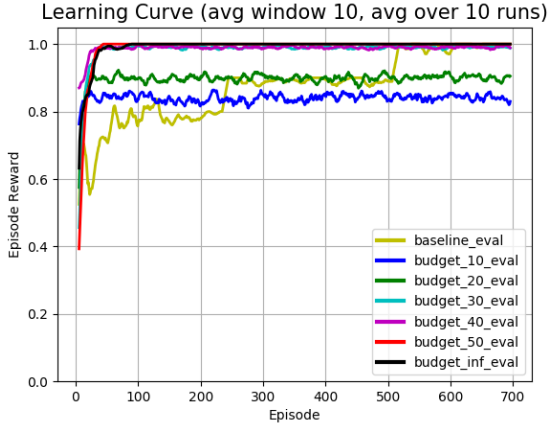
To understand how our oracle credit assignment framework improves performance, we compare our framework with the baseline MC method. Figure 5.4 shows the performance in six game environments. The yellow curves represent the baseline MC method and the rest of curves represent oracles with different budgets. Each curve is the mean curve of 10 trial runs during evaluation rollouts and each evaluation rollout evaluates the policy for 10 episodes and calculate average episode reward. The maximum episode reward in Figure 5.4 is 1.0 since the environment only allows one target hit per episode. The goal is to reach the maximum episode reward using as few episodes as possible. Our oracle with infinite budget (black curves) is the best-performing curves across all six game setups. This demonstrates that our

	10x10				30x30 (10×budget)	
Mode	1T1H	1T1S	3T1H	3T1S	1T1H	1T1S
baseline	513	200	1692	892	649	691
budget 10	N/A	N/A	N/A	N/A	92	N/A
budget 20	N/A	933	N/A	N/A	81	N/A
budget 30	38	62	50	N/A	103	461
budget 40	27	166	96	72	80	411
budget 50	32	88	54	66	88	455
budget ∞	40	79	54	81	120	573

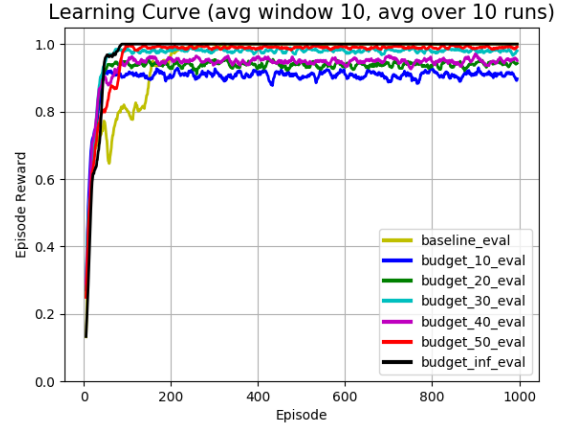
Table 5.1: Number of episodes required to reach maximum reward (N/A indicates the agent never reaches the maximum reward; 30x30 budgets are 10 times the 10x10 budgets).

credit assignment framework are robust regardless of game environments. If we reduce the size of the budget, the performance of the oracle decreases accordingly. The blue curves in Figure 5.4 are the oracles with the lowest budgets, it is consistently the worst agent across all the environments. The size of the budget is too small to fully train an agent. As the budget size increases, the agent’s performance improves as well until its policy becomes optimal. Afterwards, the budget size becomes irrelevant. In Figure 5.4a, budget 30 is sufficient to fully train the agent (cyan curve); extra budget will not further improve the performance of the agent. We argue that for every environment, there exists a budget threshold, τ , that would allow the agent to be fully trained. We observe the same phenomenon in all game variations (Figure 5.4).

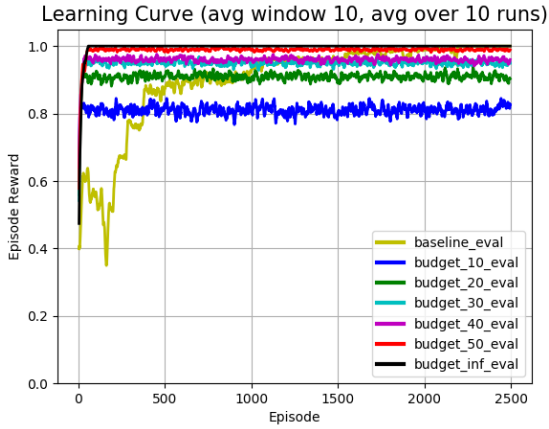
In Figure 5.5, we further demonstrate the relationship between the oracle budget and agent performance. The x-axis in Figure 5.5 represents the percentage of the optimal episode reward whereas the y-axis is the first episode to reach this reward threshold. For instance, the blue curve in Figure 5.5a (MC baseline) reach the maximum reward threshold 1.0 at episode 513 and oracles with budgets that are ≥ 30 are able to reach optimal within 50 episodes. This further illustrates the performance improvement brought by our SML approach. Table 5.1 lists the number of episodes needed to achieve maximum episode reward, which further shows our SML oracle with sufficient budget is able to outperform the baseline MC in every game variations.



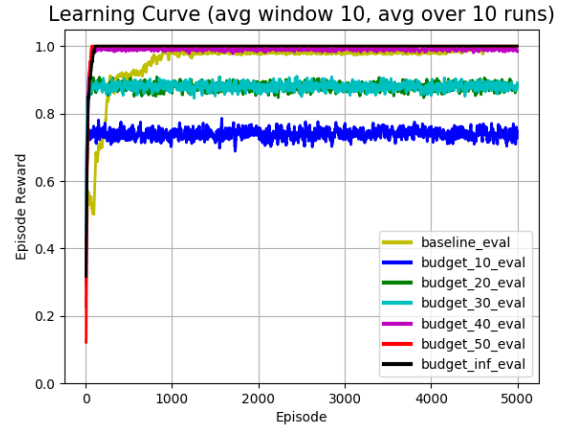
(a) 10x10 1T1H



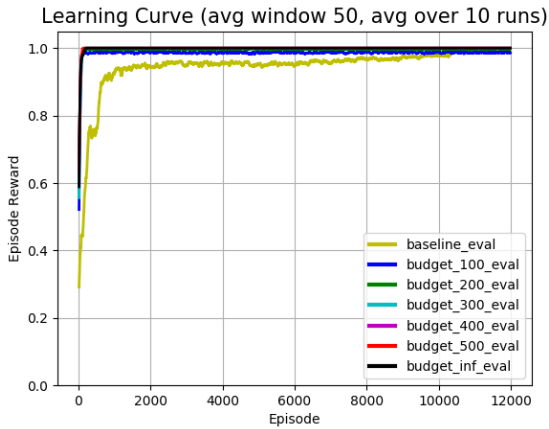
(b) 10x10 1T1S



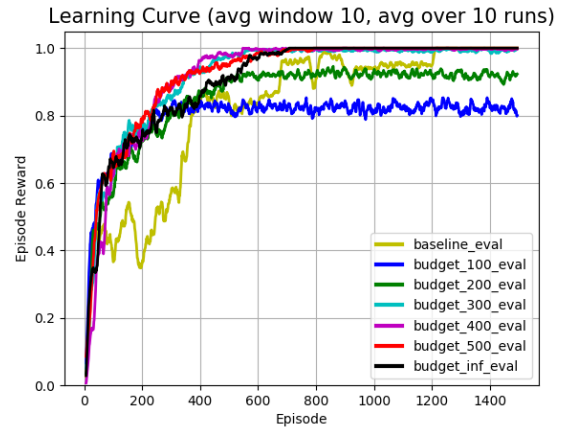
(c) 10x10 3T1H



(d) 10x10 3T1S

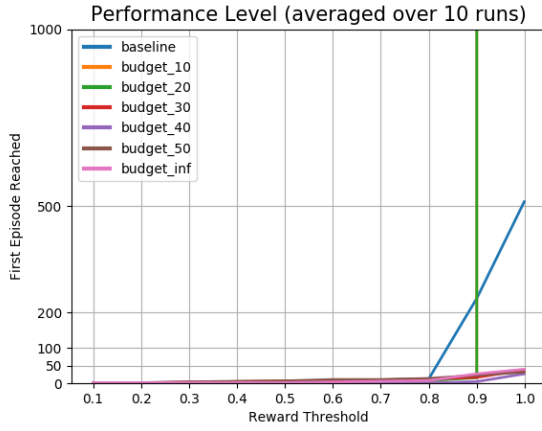


(e) 30x30 1T1H

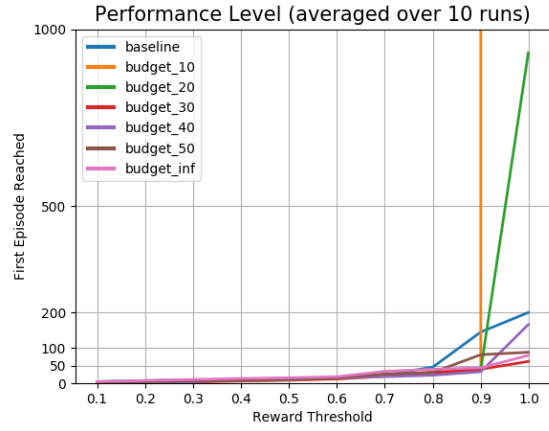


(f) 30x30 1T1S

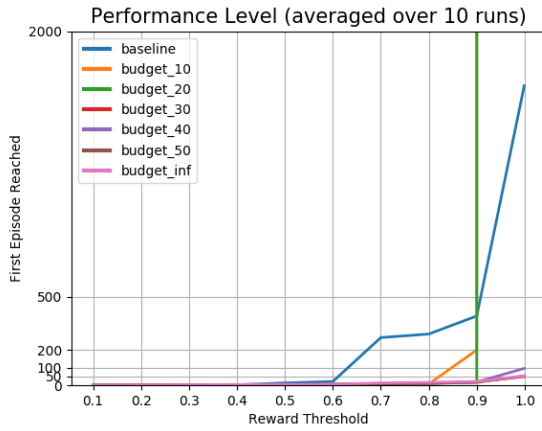
Figure 5.4: Performance of the oracle credit assignment framework in six game environments. The x-axis represents the number of episodes in training and y-axis represents the reward received in an episode.



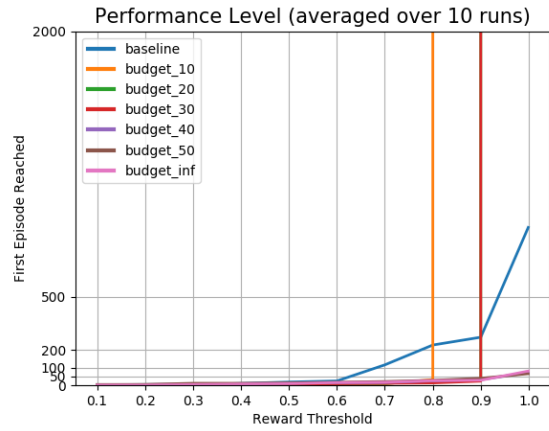
(a) 10x10 One Target One Hit



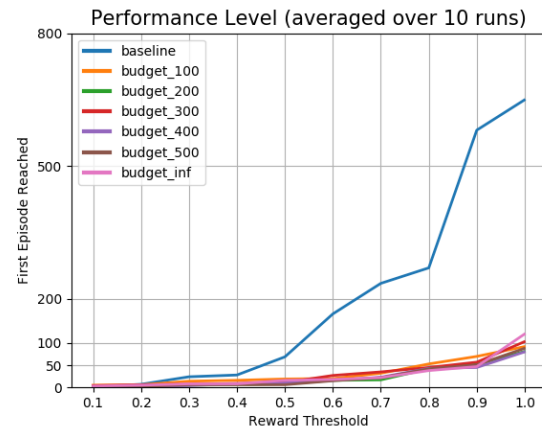
(b) 10x10 One Target One Shot



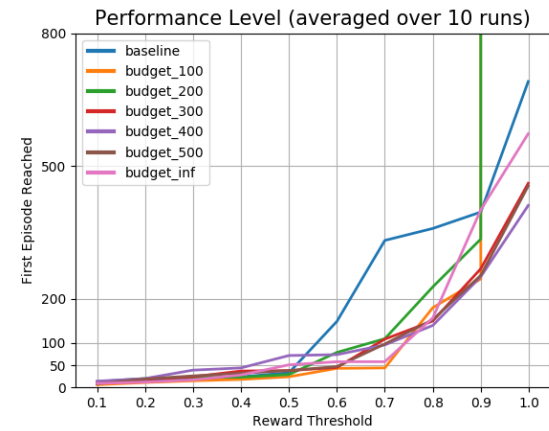
(c) 10x10 Three Targets One Hit



(d) 10x10 Three Targets One Shot



(e) 30x30 One Target One Hit



(f) 30x30 One Target One Shot

Figure 5.5: First episode to reach episode reward threshold. The x-axis represents the percentage of the optimal episode reward and the y-axis represents the first episode to reach a threshold. The vertical curves represent the agent never reach the threshold.

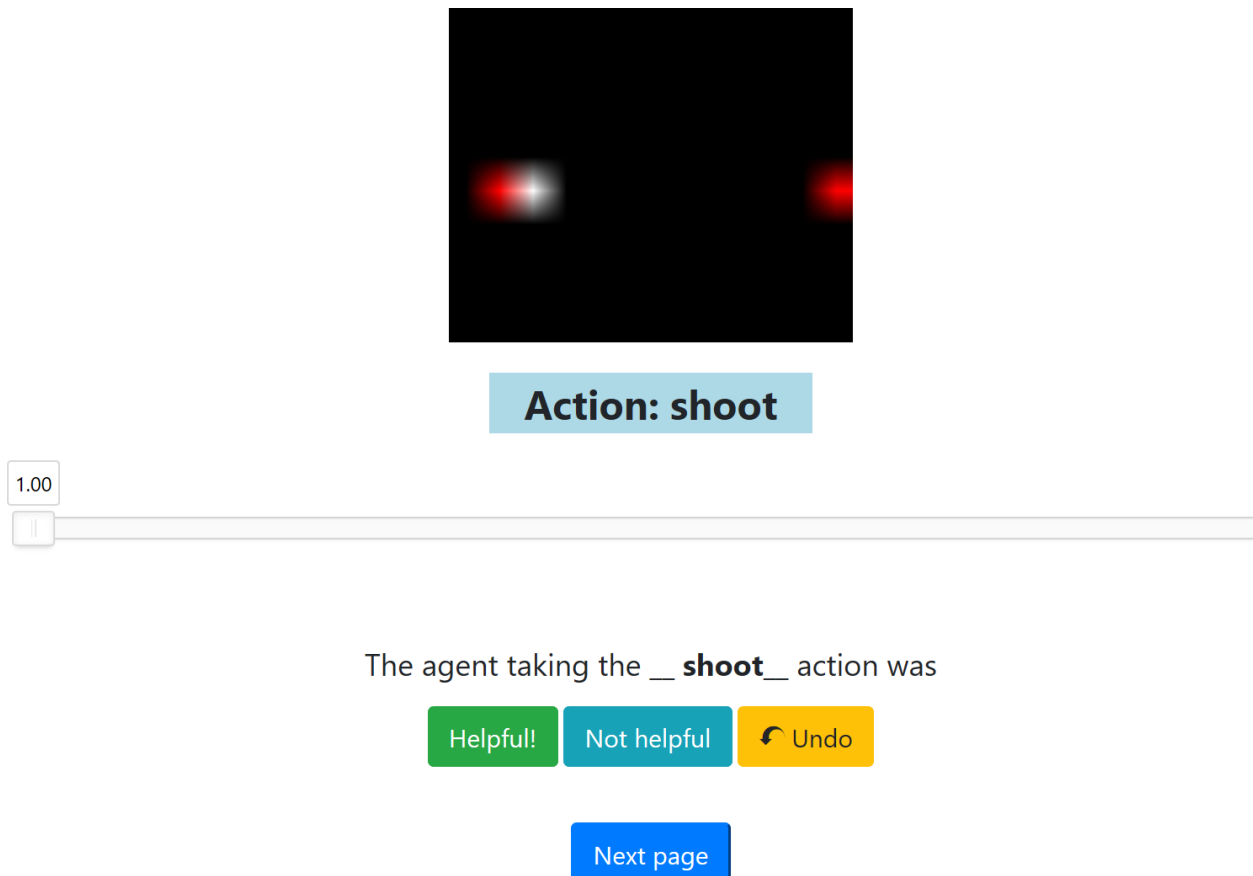


Figure 5.6: Relevance assessment user interface. Crowd workers can annotate the relevance of prior gameplay experience by clicking on the slider bar to view each step, followed by choosing Helpful! or Not helpful to determine whether the step of interest contributes to the reward signal.

5.6 Crowd Case Study

Our Montage system demonstrates the efficacy of the relevance-driven credit assignment approach through oracle-based simulations. To validate that this approach can be deployed in real-world settings, we utilize crowd intelligence to provide relevance assessment for collected agent experience.

In this setup, crowd workers are asked to review trajectories of an agent’s experience and decide which state-action pairs are relevant with regard to the reward received at the end of the episode. The web UI we designed for relevance assessment is shown in Figure 5.6. The remaining state-action pairs are then saved and sent back to the agent for policy improvement, following Algorithm 2. If the agent does not receive a reward signal, the system will not make any request.

We train a Monte Carlo control RL agent with crowd workers’ responses in a 10x10 1H1T game environment to demonstrate the efficacy of human-powered relevance assessment. Our system sends a request to crowd workers every time the agent receives a reward signal. For each request, we ask 5 workers to annotate the episode and use majority consensus to determine which state-action pairs should be removed from the episode trajectory. Afterwards, the agent uses the remaining state-action pairs to update its policy.

5.6.1 Protocol

We recruited human subjects from Amazon Mechanical Turk¹. For each episode, only unique workers were allowed to perform the task. We designed a web interface that enables workers to view and modify the trajectory of an episode. The interface uses a slider bar to display images based on the worker’s selection. The image is an RGB gameplay frame from our game and represents a state s_i . The interface also displays the action text, a_i , corresponding to s_i . Workers can then decide whether this state-action pair, (s_i, a_i) , is *helpful* or *not helpful* to the final outcome. Once workers finish this task, we direct them to a survey regarding their decisions and overall strategy for the task and the game.

5.6.2 Survey Questions

Our survey is divided into four parts. The first part randomly samples one state-action pair for each action that the workers deem *helpful*, and the question is “You selected this frame as a frame that makes progress towards the reward. Can you explain why?”. The second part randomly samples one state-action pair that the workers deem *not helpful* and the question is “You selected this frame as a frame that does not make progress towards the reward. Can you explain why?”. From the first two parts, our goal is to understand how workers make decisions when they view the frame in the context of the entire gameplay trajectory.

The third part asks the crowd worker to identify an overall strategy for the agent, and the fourth part asks them about any strategies they would personally adopt for the game (not for the agent in particular). The questions are “In this setup, what is the overall strategy the AI player should follow to achieve / obtain the reward?” and “If you were playing the game, what strategy would you follow to achieve the end-goal in this setup?” From these two questions, our goal is to understand if workers are able to comprehend the policy the agent is utilizing and what the optimal policy is.

¹<https://www.mturk.com/>

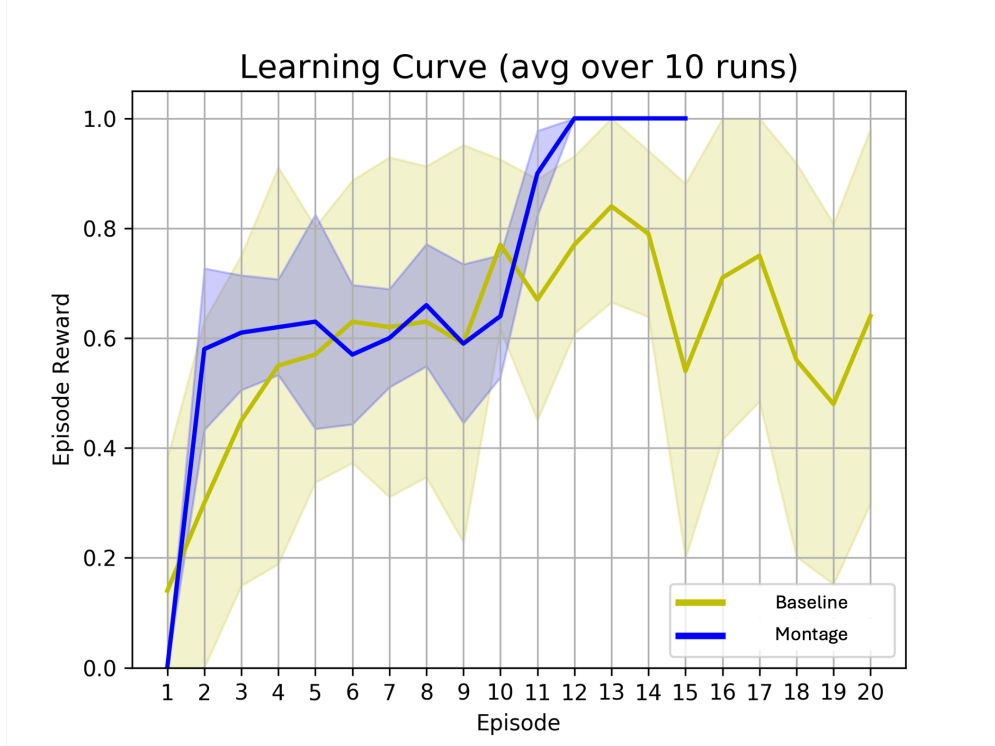


Figure 5.7: We use the majority consensus of five workers’ responses per request to train an RL agent. The training process is stopped at episode 15 since the agent already reaches optimal performance. The shaded area is one standard deviation from the mean curves. The mean curve is the average of 10 evaluation rollouts.

5.6.3 Results

We compare crowd workers’ responses with the oracle output (the ground truth) to examine how robust worker responses are in our game environment. To do this, we collect a total of 10 annotations from workers in one episode request, from which we obtain all combinations of 10 responses taken five at a time. This results in a total of 252 combinations. We then take the majority consensus of those five responses and compare that with the oracle response. Compared to the oracle (i.e., the ground truth), our accuracy is 85.6%, with a precision of 0.70 and a recall of 1.0. This shows that humans are able to identify all critical steps (recall = 1.0). However, they tend to include some “noop” actions after the agent shoots a pellet, which are unnecessary but not incorrect.

Figure 5.7 compares human-powered SML with the MC control baseline. This result demonstrates that the agent trained with crowd responses reaches maximum reward within 12 episodes, significantly outperforming the baseline. It should be noted that while the baseline curve averages 10 independent training runs to account for the stochastic nature

of RL exploration, the Montage curve represents a single training run averaged over 10 evaluation rollouts. This distinction arises from the resource constraints inherent in crowd-sourcing. However, our robustness analysis supports the representativeness of this single run by demonstrating that crowd inputs remain consistent across different worker subsets. Thus, we posit that the low variance in the Montage curve reflects the stability of the crowd-guided policy, in contrast to the high variance inherent in unguided exploration.

5.6.4 Understanding Crowd Workers’ Strategies

From analyzing our survey responses, we find that most crowd workers (87%) agree with the general strategy that needs to be adopted to win the DotShooter game: align the firing dot (i.e., the agent-controlled dot on the left) to the target dot (i.e., the dot on the right), and then shoot. Crowd workers understand the underlying need for minimizing the training data and identifying only the relevant state-action pairs as a means of increasing training efficiency.

“My overall strategy is to line up the dot to hit the target. To achieve this goal, I would try and hit the target with as minimal steps as possible.” (P49)

“There is no need to [take an action to change state], the white ball has already been shot. So it does not need any [more] progress” (P82)

While we did see lower precision than recall because some crowd workers included *noop* steps as helpful, there were others who understood exactly why the *noop* steps were unnecessary. P23 noted that “The shot was already in progress, doing nothing does not get you any closer.”

Even though we do not provide any specific instructions about the game or how it should be played, crowd workers add several qualifiers to this general strategy. When asked about the strategy an agent or they themselves would apply to this game in general, several participants provided answers assuming that the game was a two-player setup, and tried to find strategies based on anticipating a “return shot”. They noted that they would try to “stay more in the middle until it [the agent] sees the trajectory the return shot is likely to take” (P11) or “Find weakness in the opponent’s actions by studying them” (P44).

All participants agreed that the strategy they would follow is what they think the agent should learn as well, and were able to identify the primary state-action pairs for relevance attribution. Overall, this highlights that human-powered approaches are a feasible way of improving upon the existing training baselines in RL domains.

5.7 Discussion

This chapter has introduced Montage, a novel framework that leverages relevance retrospectively to address the fundamental challenge of temporal credit assignment in RL. By incorporating human insights to identify essential state-action pairs within collected trajectories, Montage enables more efficient policy updates and accelerates the learning process.

5.7.1 Key Contributions and Performance Improvements

Our experimental results demonstrate that Montage achieves substantial performance gains across multiple DotShooter game configurations. The framework consistently outperforms baseline Monte Carlo methods, with agents reaching optimal performance in significantly fewer episodes. For instance, in the 10x10 1T1H configuration, Montage with sufficient budget achieved optimal performance within 40 episodes, compared to 513 episodes required by the baseline approach—representing an order-of-magnitude improvement in sample efficiency.

The robustness of our approach is evidenced by its effectiveness across different game variations, including environments with larger state spaces (30x30) and more complex reward structures (three-target configurations). The budget-based analysis further reveals that even limited human interventions can produce meaningful improvements, with performance scaling predictably with available budget until reaching a saturation point where optimal policies are achieved.

5.7.2 Human-powered Relevance Assessment Validation

Our crowd study provides compelling evidence that human workers can effectively perform the relevance assessment task for Montage. With 85.6% accuracy compared to oracle ground truth and perfect recall in identifying critical steps, crowd workers demonstrated the ability to distinguish between relevant and irrelevant state-action pairs. The high recall is particularly significant, as it ensures that no essential learning steps are omitted from policy updates.

Survey responses showed that workers intuitively understood the underlying game strategy and the purpose of relevance annotation, with 87% of participants correctly identifying the optimal policy of aligning and shooting. While some workers included unnecessary *noop* actions (leading to lower precision), this did not negatively impact overall learning performance, as demonstrated by the human-powered framework outperforming both baseline and oracle approaches in our experiments.

5.7.3 Limitations and Practical Considerations

Despite these promising results, Montage faces several practical limitations. The requirement for human annotation of collected trajectories creates scalability challenges in data-intensive environments. The episodic nature of SML restricts applicability to settings where complete trajectories can be collected before policy updates. In addition, the binary nature of relevance assessment, while effective, may oversimplify the continuous spectrum of state-action importance in more complex domains.

These limitations motivate the need for a complementary approach that can anticipate relevance rather than analyze it post-hoc. In Chapter 6, we introduce CARAFE, a framework that shifts from retrospective to prospective form—predicting relevance for unseen states rather than analyzing past trajectories. This forward-looking approach eliminates the dependency on collected trajectories and continuous human intervention, while extending the benefits of relevance-guided learning to general RL algorithms beyond episodic settings. By learning to anticipate which states and actions will be relevant to desired outcomes, CARAFE represents the next evolution in making relevance-driven RL practically deployable in complex, real-world environments.

CHAPTER 6

CARAFE: Crowd-Assembled Relevance Anticipation via (Estimated) Functional Equivalence

6.1 Motivation

In Chapter 5, we demonstrated that *retrospective relevance*—the identification of critical state-action pairs from past trajectories—can refine temporal credit assignment and improve sample efficiency. However, this approach introduces a fundamental structural limitation. Standard RL systems integrate reward *immediately* into their value function or policy. Our retrospective review approach, while effective, *gates* this learning loop: the agent receives a reward but cannot update its policy until additional information is acquired. This creates a dependency on an external signal (human review of outcomes) that inherently adds a second layer of sparsity beyond the reward signal itself. *Montage* (Chapter 5) thus trades immediate (though naive) learning for a sparser (but more accurate) signal. Even more critically, the retrospective approach leaves the agent unable to learn from new experiences on the fly.

This limitation motivates a paradigm shift. We want to retain the high-quality signal from human intuition but overcome the dependency on a per-update review. The goal is to move from *retrospective analysis* (identifying what *was* relevant) to *prospective approximation* (anticipating what *will be* relevant). These approaches represent two sides of the same coin—leveraging human insight for guidance—but the latter seeks to internalize this insight by building a model that *approximates* human judgment. If an agent can learn a representation of what likely matters, it regains the ability to immediately integrate new rewards in a human-informed way.

However, the rich, generalizable knowledge people possess about tasks—the intuitive “gut feeling” that one situation differs from another—is largely implicit. Directly asking humans, “How relevant is this state for achieving a future outcome?” is infeasible, as people cannot reliably quantify the abstract causal influence of a single state on distant futures. The core challenge, therefore, is to acquire prospective relevance from signals humans *can* reliably

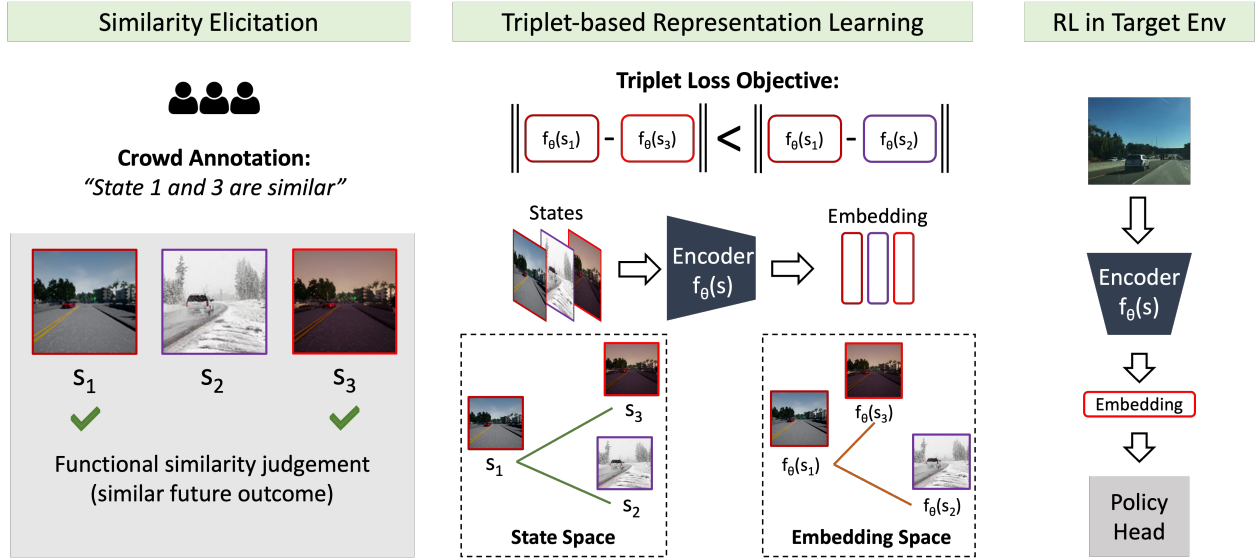


Figure 6.1: The CARAFE framework overview. (Left) Similarity Elicitation: We present triplets of states to crowd workers and elicit *prospective relevance* similarity assessment. In this illustrative driving example, workers identify that the daytime (state 1) and evening (state 3) scenes are functionally similar (both represent safe driving) compared to the snowy road scene (state 2), despite the lighting differences. (Middle) Representation Learning: We use these annotated triplets to train an encoder via triplet loss. This objective pulls the embeddings of functionally similar states (s_1, s_3) closer together while pushing dissimilar states (s_2) apart, encoding a human-derived relevance prior. (Right) RL in Target Env: The pre-trained encoder initializes the RL agent, allowing it to leverage the learned relevance prior to accelerate training in the downstream task.

provide.

We reframe this challenge as a *representation learning* problem. Our goal is to learn a state representation where proximity encodes similarity in outcome potential. In other words, if state A is known to be highly relevant, and state B is embedded close to A in this representation space, then state B can be inferred to also be highly relevant. The question becomes: how can we elicit the supervisory signal for this relevance-informed representation from human intuition? Our insight is that while people cannot explicitly score relevance, they can reliably perform *comparative assessments*, i.e., intuitively judging whether two states are similar in a way that matters for outcomes.

We leverage this capability through *differential comparison*, specifically triplet comparisons: by presenting crowd workers with three states and asking them to select the pair most similar in terms of future potential, we collect data that reflects a human prior over state relevance. This comparative approach serves as a practical and reliable proxy for capturing their implicit ranking of relative relevance. We then use a triplet loss function [83] to train

an encoder that maps raw states into a low-dimensional embedding space where this human-understood notion of relevance is preserved, as illustrated in Figure 6.1. This approach is analogous to the reward-predictive representations in Chapter 4, but with an important difference: instead of predicting reward from pixels, we learn a representation that directly encodes a human-derived *relevance prior*.

Although our simulation results demonstrate the power of this signal, cognitive science suggests that human expertise is rarely uniform and often context-dependent [25, 17, 31]. However, our user study reveals that despite this variability, the raw human feedback provides a robust signal. We find that unfiltered similarity judgments yield a $\sim 30\%$ gain in representational alignment compared to random baselines. This indicates that while the raw human relevance prior is inherently useful, it becomes even more effective when filtered for consequential contexts.

This chapter makes four key contributions:

- **A Framework for Relevance-Based Representation Learning:** We present CARAFE, a novel framework that learns state representations from human similarity judgments. By formally linking functional similarity to prospective relevance and implementing this connection through triplet loss, CARAFE enables agents to learn a relevance-informed, low-dimensional representation that can accelerate RL training.
- **Viability of Relevance-Based Representations:** Through extensive simulation studies using trained agents as oracles, we demonstrate that representations learned from similarity judgments can significantly accelerate downstream RL training. These studies establish the theoretical viability of our approach and characterize its robustness across various conditions, including dataset size, annotation quantity, and error tolerance, demonstrating that a relevance-informed representation is a powerful mechanism for improving sample efficiency.
- **Quantification of the Human Relevance Prior:** We provide empirical evidence that raw human similarity assessment naturally align with expert representations. Across all tested environments, unfiltered human annotations achieved an approximate 30% Normalized Agreement Gap compared to the baseline. This indicates that human intuition provides a representational signal that is significantly closer to an optimal policy’s encoding than a random initialization, effectively reducing the search space for the agent.
- **Context-Dependency of Human Intuition:** Our user study reveals that human judgments are not uniformly reliable, but their coherence is strongly context-

dependent. We identify a filtering condition—based on agreement between trained and untrained agents—that isolates states where human judgments are most meaningful. When focused on these “consequential” states, human annotations align significantly better with expert policies, demonstrating that the key to leveraging human intuition lies in understanding *when* it is most applicable.

Together, these contributions establish CARAFE as a foundation framework for human-guided RL. While a full end-to-end demonstration using human data for pretraining was prohibitively expensive, our work provides: (1) proof that the mechanism works in principle with ideal data, and (2) evidence that humans can provide the necessary signal. By showing both the theoretical potential and practical considerations of learning relevance priors from human feedback, this work advances toward more sample-efficient RL.

6.2 Related Work

Our work on prospective relevance sits at the intersection of several research areas: representation learning for reinforcement learning, human-in-the-loop guidance, and sample-efficient exploration. We review the most relevant literature in these domains and position our contribution accordingly.

6.2.1 Representation Learning for Reinforcement Learning

A long-standing challenge in RL is learning effective representations from high-dimensional inputs. Methods range from autoencoders [108, 35] and contrastive learning approaches like CURL [56] and CPC [73], to models that predict future states or rewards [37, 30].

Our approach is most aligned with methods that learn representations predictive of long-term outcomes. For instance, Deep Successor Representations (SR) [51, 9] learn a feature space that combines immediate reward prediction with expected future state occupancy. Similarly, Deep Bisimulation [112] attempts to learn representations where states are indistinguishable if they share identical reward and transition dynamics.

The work in Chapter 4 learns reward-predictive representations from pixels. CARAFE differs by not learning from environmental rewards or dynamics directly. Instead, we learn a representation from human judgments that is *pre-trained* to reflect a prior understanding of which states are consequentially similar.

6.2.2 Human-in-the-Loop Reinforcement Learning

Leveraging human input to guide RL agents is a well-established strategy for improving sample efficiency and aligning agent behavior. Common approaches include:

- **Learning from Demonstrations (LfD):** Agents learn by imitating expert trajectories, exemplified by algorithms like DQfD [40] and Behavior Cloning. While powerful, LfD requires the human to be a proficient actor in the environment, which is a significant barrier.
- **Learning from Preferences:** Agents learn a reward function by asking humans to compare trajectory snippets [19, 57, 82]. This is closely related to our use of comparisons. However, a key distinction is that preference-based RL typically focuses on evaluating the quality of *behavior* or *outcomes*, whereas CARAFE focuses on the *predicted potential* of raw states, which decouples the feedback from specific agent actions.

6.2.3 Metric Learning from Human Feedback

Our technical approach of using triplet loss to learn from human similarity judgments is grounded in metric learning. This technique has been highly successful in computer vision for tasks like face recognition [83] and state similarities based on temporal proximity (Time-Contrastive Networks) [86, 24].

Most relevant to our work is SIRL (Similarity-based Interactive Representation Learning) [11], which uses pairwise comparisons to align robot representations with human concepts. While methodologically similar, the objective differs fundamentally. SIRL aims for *semantic alignment* to ensure the agent focuses on features humans deem concept-defining (e.g., distinguishing “spilling” from “pouring”). In contrast, CARAFE targets *functional alignment* via prospective relevance. We do not aim to capture human semantic categories, but rather the human’s causal intuition about potential outcomes. For example, visually distinct states (e.g., a fire pit vs. a spike trap) might be semantically distant in SIRL but would be embedded closely in CARAFE because they share the same prospective outcome (termination).

6.2.4 Positioning CARAFE

It is critical to clarify CARAFE’s position relative to representation learning. While CARAFE *leverages* representation learning techniques, specifically triplet loss, it is not

reducible to them. Standard representation learning is typically an unsupervised or self-supervised process aimed at compressing state information based on pixel-level or temporal statistics.

CARAFE, in contrast, is a framework for *eliciting and integrating human relevance priors*. We use representation learning as a mechanism to encode these priors, but the core contribution lies in: (1) the upstream design of the comparative assessment task that captures prospective relevance, and (2) the downstream application of this relevance-informed representation to bootstrap sample-efficient RL.

The novelty of CARAFE lies in the source of supervision, i.e., human comparative assessment about state similarity based on outcome potential, rather than the architectural choice. This distinction is crucial: while we build upon existing metric learning techniques, CARAFE’s value is in transforming human intuition about prospective relevance into a form that can directly accelerate RL.

6.3 Learning Representations from Relevance Priors

The CARAFE framework is designed to translate intuitive human assessment about state similarity into a structured, low-dimensional state representation that accelerates RL training. Rather than asking humans to directly estimate relevance, which requires explicit probability estimation beyond human capability, we leverage their ability to assess *functional similarity*. This approach establishes a formal link between functional similarity and prospective relevance, then operationalizes this connection through an ML pipeline based on triplet loss and scalable crowdsourcing.

6.3.1 From Functional Similarity to Prospective Relevance

The concept of relevance, as defined in Equation 4.1, quantifies how essential a state is for achieving specific outcomes. However, directly eliciting relevance assessment from humans is challenging, as people cannot reliably estimate precise probability distributions over outcomes. Instead, we leverage human ability to assess *functional similarity*—the degree to which two states lead to comparable futures in terms of both potential outcomes and their likelihoods.

Functional similarity serves as a practical proxy for relevance assessment. When humans compare states, they intuitively consider whether being in one state versus another meaningfully changes what the agent can achieve. For example, in autonomous navigation, humans judge going left versus right around a tree as functionally similar because both

lead to continued travel with positive outcomes, while crashing into the tree leads to mission failure. By mapping these to states where s_i and s_j represent successfully bypassing the tree and s_k represents an imminent collision, this similarity assessment directly implies $|H(\Delta(s_i, s_j))| < |H(\Delta(s_i, s_k))|$. In other words, states with similar future outcomes have smaller relevance differences than those with divergent futures.

This intuition can be formalized by relating functional similarity to the distribution of possible outcomes $\Omega \in \mathcal{O}$:

$$H(\Delta(s_i, s_j)) \approx \sum_{\Omega \in \mathcal{O}} \left| [P(\Omega|s_i) - P(\Omega|s_j)] R(\Omega) \right| \quad (6.1)$$

where $\mathcal{O}$ represents the set of all potential outcomes that humans consider when making similarity judgments.

6.3.2 Eliciting Similarity via Triplet Comparison and Triplet Loss

To translate functional similarity assessment into a useful representation, CARAFE employs the triplet loss function, a technique widely used in metric learning to structure an embedding space based on similarity [83]. The objective is to train an encoder, denoted f_θ , that maps high-dimensional state observations into a low-dimensional embedding space where geometric distance corresponds to human-perceived functional similarity.

The training process is structured around triplets of data points:

- Anchor (s_a): A reference state.
- Positive (s_p): A state that a human annotator has labeled as being more functionally similar to the anchor than the negative.
- Negative (s_n): A state that the same annotator has labeled as being less functionally similar to the anchor than the positive.

The triplet loss function is formulated to enforce a margin-based constraint on the distances between these points in the embedding space produced by f_θ :

$$L(s_a, s_p, s_n) = \max(\|f_\theta(s_a) - f_\theta(s_p)\|^2 - \|f_\theta(s_a) - f_\theta(s_n)\|^2 + \alpha, 0) \quad (6.2)$$

The function computes the squared Euclidean distance between the anchor and the positive, $d(a, p) = \|f_\theta(s_a) - f_\theta(s_p)\|^2$, and the distance between the anchor and the negative, $d(a, n) = \|f_\theta(s_a) - f_\theta(s_n)\|^2$. The loss is zero if the negative sample is farther from the anchor than the positive sample by at least a predefined margin, α . Otherwise, the loss is positive,

and the optimization process will adjust the encoder’s parameters, θ , to push the positive closer to the anchor and/or push the negative farther away.

This loss function directly instantiates the ordinal relationships in functional similarity assessments. When humans indicate that s_p is more functionally similar to s_a than s_n is, they provide an implicit ranking of relevance differences: $|H(\Delta(s_a, s_p))| < |H(\Delta(s_a, s_n))|$. The triplet loss enforces this ordinal structure in the embedding space to create a representation where Euclidean distance $d(s_i, s_j)$ serves as a proxy for the relevance difference $|H(\Delta(s_i, s_j))|$.

By minimizing the loss, the training process forces the encoder f_θ to learn a mapping where geometric proximity corresponds to similarity in prospective relevance. This transformation is visualized in the middle panel of Figure 6.1: while the raw states (s_1, s_2, s_3) may be visually distinct, the encoder projects them into an embedding space where functionally similar states (s_1, s_3) are clustered together, effectively encoding the human relevance prior.

This approach differs subtly but importantly from other metric learning techniques, such as contrastive loss. While contrastive loss aims to pull all positive pairs together and push all negative pairs apart by a fixed margin, triplet loss focuses only on the relative ordering of distances. This is particularly well-suited for learning from functional similarity, as the goal is not to force all “similar” states into one tight cluster but rather to learn a representation that preserves the ordinal relationships of their future outcomes.

6.3.3 A Crowdsourcing Workflow for Elicitation

To gather triplet annotations from people, CARAFE utilizes a scalable crowdsourcing workflow developed through an iterative design process.

6.3.3.1 Design Rationale and Process

The workflow design emerged from addressing a core challenge: how to transform human intuition about prospective relevance into concrete, reliable annotations at scale. We initially experimented with various approaches before converging on the current design:

Comparison Format: We began with *pairwise* comparisons, asking workers whether two states were functionally similar. However, this binary framing was confusing—workers struggled to apply a uniform similarity threshold. We then tested *triplet* comparisons (“select the most similar pair”), which provided a clearer relative judgment. Early pilot studies with larger comparison sets (4-6 states) showed that workers became overwhelmed and consistency dropped, which confirmed triplets as the optimal balance between information gain and cognitive load.

State Representation: We evaluated static screenshots versus animated GIFs. Static

frames often omitted critical temporal information (e.g., object motion, impending threats), leading workers to focus on superficial visual features. Short animated sequences (4-5 frames) provided minimal temporal context about dynamics. This also aligns better with how agents perceive states [67] while maintaining quick load times.

Instruction and Onboarding: The wording of the prompt underwent several iterations. Initial minimal instructions, such as "Which two states are most similar?", were abstract and led to inconsistent interpretations. We developed multi-stage onboarding with game-specific tutorials, explicit similarity criteria, and interactive examples. This scaffolding was essential for helping workers internalize the underlying functional context of game states, ensuring they could distinguish between superficial visual patterns and meaningful state semantics.

The final design represents an optimized solution to these challenges, balancing annotation quality, worker comprehension, and practical feasibility.

6.3.3.2 User Interface and Task Design

The crowdsourcing workflow utilizes a multi-stage user interface to onboard, instruct, and query workers.

- **Onboarding** (Figure 6.2): Workers first see an entry screen that introduces the task, such as categorizing scenes from the ATARI game Seaquest. This page includes a video demonstrating the game’s mechanics. Before starting, workers can access detailed, expandable instructions that explain the game’s objectives, rules, and scoring (e.g., rescuing divers, managing oxygen, shooting enemies).
- **Similarity Criteria** (Figure 6.3): A dedicated instruction screen explicitly defines the criteria for similarity. Workers are asked to consider functional equivalence rather than visual similarity. The instructions prompt them to imagine: “What situations the player could potentially end up in,” “what the player is able to do (or not do) from the setting,” and the “chance these situations occur” and whether they lead to “similar outcomes (e.g., points awarded, changes made to the environment, effects on the player).”
- **Examples** (Figure 6.4): We also provide three examples of similarity based on the criteria above. Workers need to click the “Reveal” button to advance to the next triplet until they see all three examples.
- **Main Task** (Figure 6.5): The primary task interface presents a triplet of states. As noted in the original design, each state is a short animated GIF (composed of consecutive game frames) to provide minimal but temporal context about the environment’s

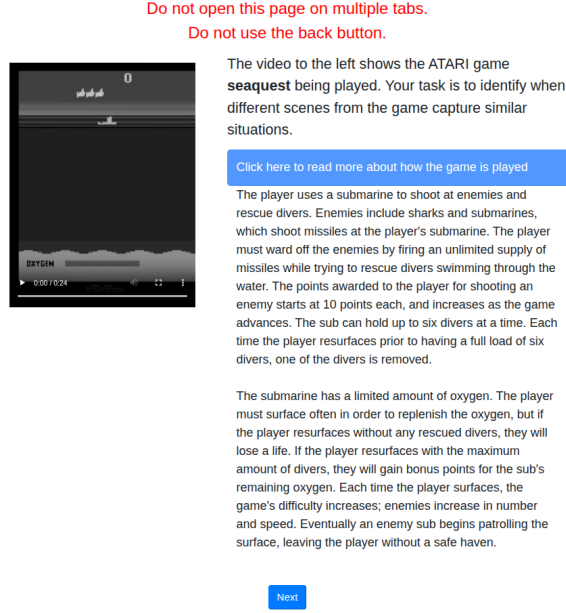


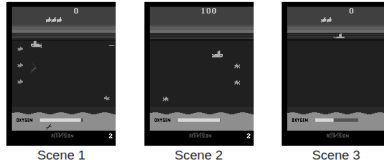
Figure 6.2: Onboarding Interface: The entry screen introduces crowd workers to the specific Atari environment via a video demonstration and outlines the game mechanics, objectives, and scoring rules to establish a baseline of domain knowledge.

dynamics, following the input structure used by deep RL agents [67]. The worker’s task is to select the two scenes from the three that they judge to be “most similar” based on the provided functional criteria, and then click “Submit”. The interface also displays a task counter (e.g., “Number of tasks left”) and provides a “View Instructions” button for workers to review the similarity criteria at any time.

6.4 Experimental Design and Research Questions

To validate the CARAFE framework, a series of experiments were designed to answer two central research questions (RQ). The first investigates the theoretical viability of using triplet-induced representations to improve RL sample efficiency under idealized conditions. The second examines the practical feasibility of using noisy, non-expert human annotations as the source for these representations.

Help a player categorize game scenes



When considering **which two scenes are most similar to one another**, consider:

1. What situations the player could potentially end up in starting from each scene above. More similar scenes would lead to more similar sets of situations in the future.
Try to imagine:
 - *what the player is able to do (or not do) from the setting they start from in each scene?*
 - *how the rest of the environment may change?*
2. Of the situations that *could* occur starting from each scene above, is the chance these situations occur similar? Do situations with similar chances of occurring also have similar outcomes (e.g., points awarded, changes made to the environment, effects on the player)?
More similar scenes would have similar chances of ending up in similar situations with similarly impactful outcomes.

Figure 6.3: Similarity Criteria Instructions: A dedicated instruction panel defines “functional similarity,” explicitly prompting workers to judge states based on potential future outcomes and available actions rather than superficial visual resemblance.

6.4.1 RQ1: Can Triplet-Induced Encoders Accelerate RL Training?

Before assessing the quality of annotations from human crowd workers, it is essential to establish a proof-of-concept: if provided with a perfect or near-perfect source of similarity assessment, can a representation learned via triplet loss confer a tangible benefit to a downstream RL agent? This question addresses the fundamental soundness of the CARAFE approach.

To investigate this, we conducted a series of simulation studies using an “oracle” to generate similarity labels. This approach first establishes a proof-of-concept using perfect, noise-free labels to isolate the potential value of the relevance signal. We then systematically introduce varying levels of synthetic noise into these oracle labels to study the framework’s tolerance to annotation errors, which mimics the imperfections expected from human judgments.

Methodology

- **The Oracle:** The oracle for similarity was constructed from the feature encoder of extensively-trained MDQN [102] or DQN agent¹. This design choice is grounded in the

¹<https://github.com/floringogianu/atari-agents>

Click [Reveal](#) to show which two scenes are most similar. Then, click [Next](#) to start the tasks.

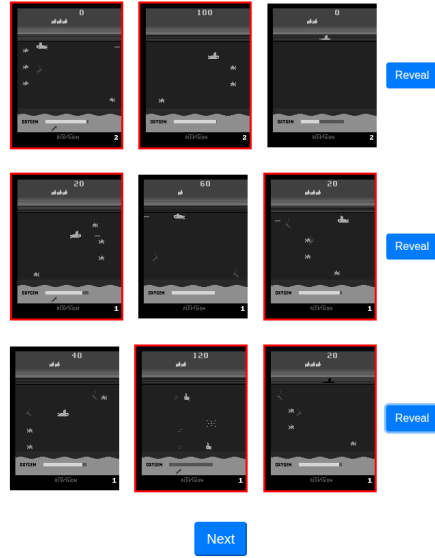


Figure 6.4: Interactive Examples: Before beginning the task, the interface requires workers to review specific, interactive examples of functionally similar states to ensure they understand how to apply the criteria in practice.

premise that an agent extensively trained on a task has learned a state representation where states with similar Q-values or optimal policies are mapped to nearby points in the embedding space. We selected the trained agent’s encoder because it directly encodes the relationship between states and their expected long-term value, which is a proxy for prospective relevance. For any given triplet of states, similarity labels were generated by computing the Euclidean distances between their vector representations as produced by the oracle encoders. The state pair with the smallest distance was labeled as the positive pair.

- **The Baseline Agent:** A Rainbow-DQN [39] agent was trained from scratch with randomly initialized weights, representing the standard deep RL approach without any relevance-based prior.
- **The CARAFE Agent:** A separate CNN encoder was pretrained exclusively on the oracle-generated triplet labels using the triplet loss function. The weights of this pre-trained encoder were then used to initialize the corresponding layers of a Rainbow-DQN agent, which was then trained on the downstream task.
- **Environment and Metrics:** We use the Atari 100k Benchmark, which contains 26 Atari games, to evaluate RL agents as it has been widely used to measure sample

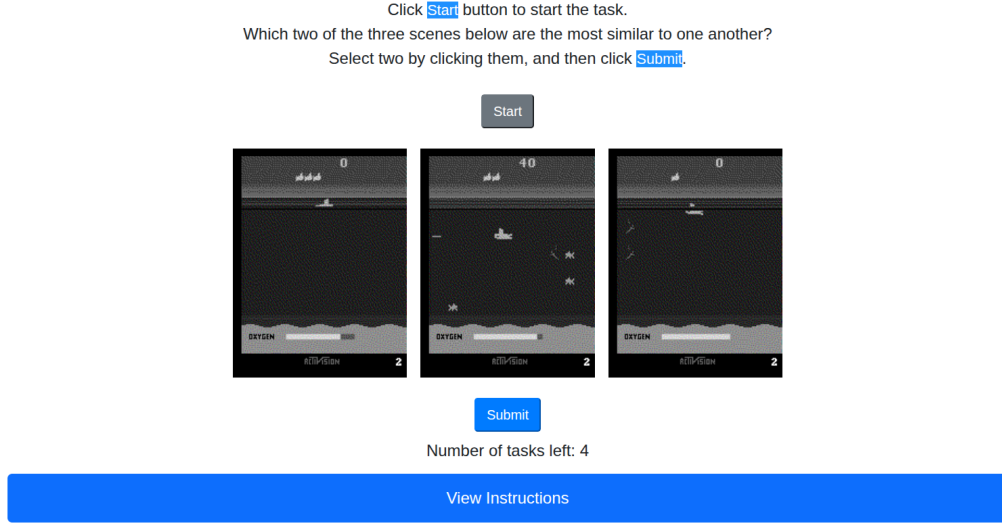


Figure 6.5: The Triplet Comparison Interface: The main task view presents three animated state sequences. Workers are asked to select the pair that is most functionally similar. The interface includes a task counter and persistent access to instructions to maintain annotation quality.

efficiency [2]. For each game, we run five independent training instances with different initial seeds to account for the stochasticity of RL training. To measure the performance of the triplet-induced encoder, we utilize the sample-efficiency curve, plotted over 100,000 training steps, to evaluate the downstream RL agents powered by the triplet-induced encoders. Sample-efficiency curves, or learning curves, describe the relationship between the number of training steps and their corresponding human-normalized score (agent’s episodic reward normalized by the performance of an average human player), i.e., given the same number of steps, the higher the score, the better the performance. To examine the overall performance of an RL agent, we use interquartile mean (IQM)—the mean of the middle 50% of scores—to aggregate scores at every 10,000 training steps for all runs and all games due to its robustness against outliers [2].

6.4.2 RQ2: Can Humans Provide Coherent Signals for Relevance-Based Similarity?

This research question transitions from the idealized oracle to the real-world scenario of using crowd workers. The primary objective is to determine whether the similarity judgments provided by non-expert humans contain a coherent and useful signal that aligns with the notion of prospective relevance, and to characterize the nature of this signal, including its potential noise profile.

6.4.2.1 Task Environments

To test the robustness of human perception across varying levels of environmental complexity, three games were selected from the Atari benchmark: Seaquest, Demon Attack, and Battlezone. These games represent a spectrum of “motion estimability”, i.e., the ease with which an observer can predict the movement and interaction of objects. Seaquest has predictable, patterned enemy movements, while Battlezone features a first-person 3D perspective with a limited field of view, making it significantly harder to anticipate outcomes. This variation allows us to test whether human judgments remain reliable as environmental complexity increases.

6.4.2.2 Data Collection

Human annotations were collected using the Amazon Mechanical Turk platform. For each of the Atari games, triplets were formed by sampling states from the experience of an agent following a random policy. Crowd workers were then asked to provide similarity judgments for these triplets. As detailed in Section 6.3.3, the user interface provided comprehensive onboarding, including game-specific introduction and explicit instructions on the criteria for “functional similarity” to guide their selections.

6.4.2.3 The Agreement Metric and Rationale

Training a full RL agent on the collected human data would be prohibitively expensive, which requires tens of thousands of annotations per game. Therefore, we developed an intermediate proxy metric to evaluate the quality of the human signal. “Agreement” is defined as the fraction of triplets for which the majority vote of the human annotators matches the similarity judgment produced by a reference model (the agent’s encoder). This metric serves as a cost-effective way to assess whether human judgments align with task-relevant representations.

We track this metric longitudinally across the agent’s training checkpoints to visualize the emergence of alignment. To allow for cross-environment comparison and isolate the effect of learning from random chance, we compute two derived metrics: the *Absolute Agreement Gap* (the difference between the trained and untrained agent’s agreement) and the *Normalized Agreement Gap* (the absolute gap scaled by the total possible improvement). Finally, we analyze the signal’s robustness by applying filtering strategies based on crowd consensus levels and “Representational Drift,” which isolates triplets where the agent’s representation actively changes between checkpoints.

6.5 Results

The experiments show a comprehensive set of results that both validate the core mechanics of the CARAFE framework and reveal the layered characteristics of human-provided relevance signal.

6.5.1 Validation of Triplet-Induced Representation (RQ1)

The simulation studies using an oracle-generated dataset provided strong evidence that a representation learned from similarity judgments can significantly accelerate RL training.

To answer RQ1, we demonstrate that there exist triplet-induced encoders that possess the representational knowledge to accelerate downstream RL training. To do so, we hypothesize that extensively trained RL agents have such representational knowledge. Thus, we elicit triplet comparisons using the feature encoding component of such an agent—a series of convolutional layers followed by a fully-connected layer. The similarity between two states is determined by the Euclidean distance of the extracted feature vectors using the encoder of extensively trained RL agents. Given such triplets, we can pretrain an encoder using triplet loss. This triplet-induced encoder will then be used to initialize the feature encoding component of an RL agent for downstream RL training.

6.5.1.1 Dataset Size

As Figure 6.6 shows, the agents with triplet-induced encoder outperform the baseline agent (blue), which suggests that the pretrained encoders have the representational knowledge for downstream RL training.

Although our method is tested in Atari games, which are simulated environments without data limits, oftentimes real-world tasks may not allow unlimited data collection due to cost

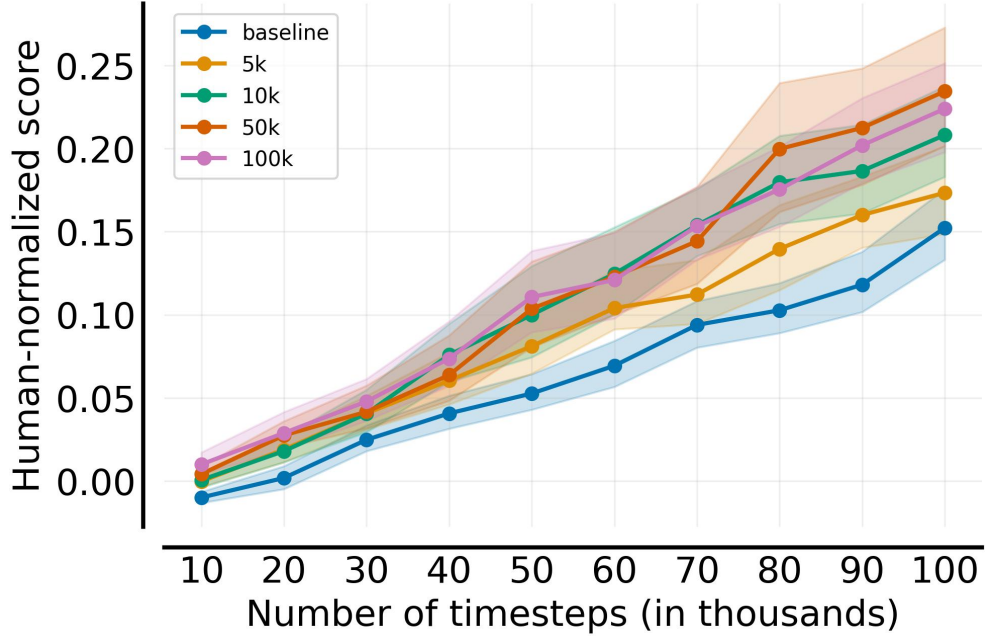


Figure 6.6: Number of training states in the dataset

or safety. Thus, we vary the number of states in a dataset to measure whether our method can deal with data scarcity.

As shown in Figure 6.6, the triplet-induced encoders, trained with datasets containing 10,000 states and above, enable RL agents to outperform the baseline. This demonstrates that our method is capable of leveraging a small amount of data. Furthermore, the encoders pretrained with 50,000 states and 100,000 states produce similar performance, which suggests that our method is insensitive to dataset size change once it reaches a threshold.

6.5.1.2 Annotation Quantity

The other aspect of dataset size is the number of similarity labels we elicit from crowd workers. Therefore, we investigate how reliant our method is on the annotation quantity by varying the number of annotations while fixing the number of training steps for the representation encoder. According to Figure 6.7, the performance shows no improvement over the baseline at 100,000 steps when the number of similarity labels drops to 10,000. Thus, we augment existing data by jittering annotated triplet images as introduced in [109] to mitigate annotation scarcity, as demonstrated by the purple curve.

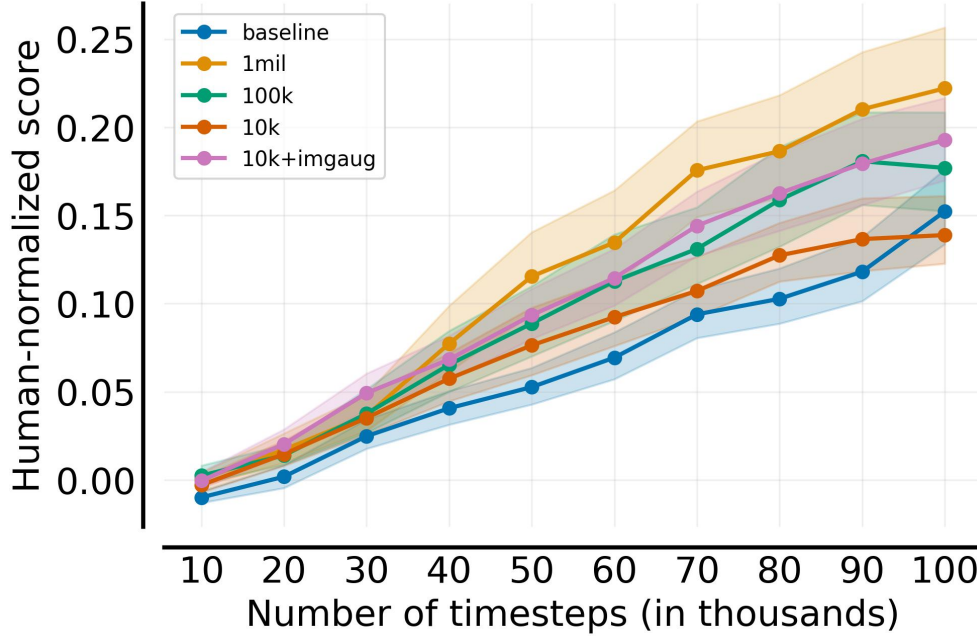


Figure 6.7: Number of annotated training triplet samples

6.5.1.3 Error Tolerance

Our method relies on crowd workers to provide similarity labels, and thus the annotation process may inadvertently introduce errors. To estimate how robust our method is against annotation errors, we deliberately inject 5%, 10%, and 20% erroneous labels into M-DQN annotated dataset. Figure 6.8 shows our method can tolerate up to 10% error without significantly affecting the performance.

6.5.1.4 Dataset Composition

Figure 6.9 also demonstrates the effect of different kinds of training data (*random*, *trained*, and *mixed*) on triplet-induced encoders. The mixed dataset condition slightly outperforms the rest. This is likely because the mixed dataset better approximates the distribution shift an RL agent experiences. An agent begins by exploring randomly and gradually converges toward an optimal policy; a mixed dataset containing both random and expert states captures this full spectrum of experience.

6.5.1.5 Triplet Loss Margin

As we use the triplet loss function to train our representation encoder, one of the most important parameters is the loss margin, which determines whether the loss value is greater

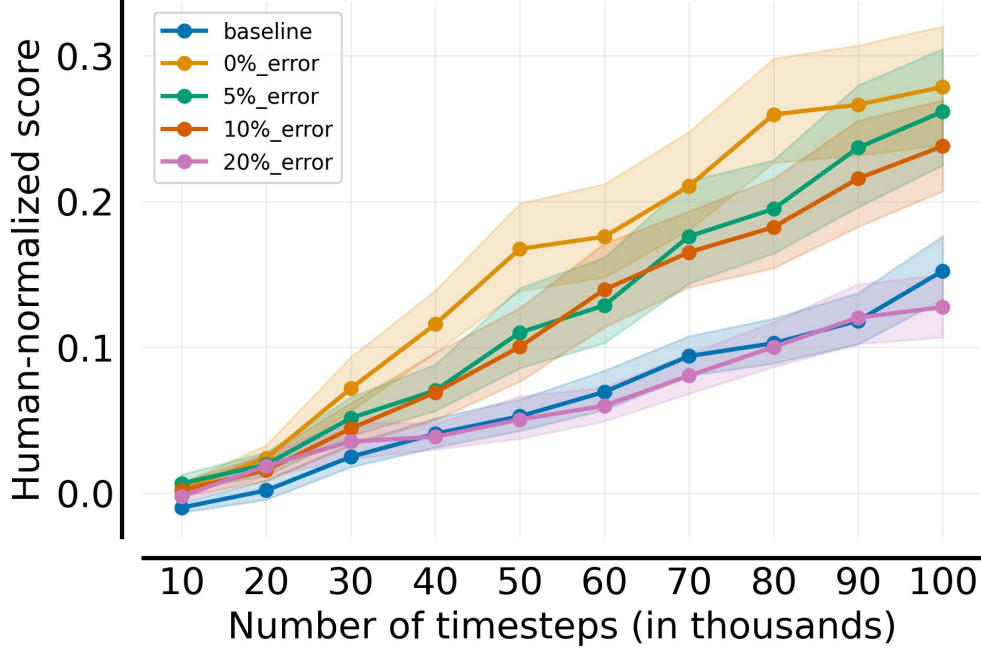


Figure 6.8: Percentage of injected erroneous samples

than zero, i.e., the loss is zero if the difference between the anchor-positive (positive pair) distance and anchor-negative (negative pair) distance is below the margin. Therefore, a larger loss margin penalizes the triplets where the positive pairs are not far enough from the negative pairs.

Figure 6.10 shows that the loss margin plays a key role in the quality of the triplet-induced encoder. If the margin value is too high (0.2 and 0.1) or too low (0.025), we observe marked differences between the performances of the downstream RL agents. This shows that a proper loss margin is critical in enabling our method and also indicates that comparing and distinguishing states is a challenging task due to subtlety.

In practice, the loss margin acts as a hyperparameter that requires tuning similar to learning rates. It effectively sets the “resolution” of the similarity. To find the optimal margin, one can perform a hyperparameter sweep (e.g., grid search) on a held-out validation set of triplets.

6.5.1.6 Learning Rate

The parameters of the feature encoding component can be modified according to learning rates of the downstream RL agent. If the learning rate is 0, then the parameters are fixed, whereas learning rate 1×10^{-4} is the same as that of other layers of the networks. Interestingly, a medium learning rate of 1×10^{-6} achieves the best performance as shown in

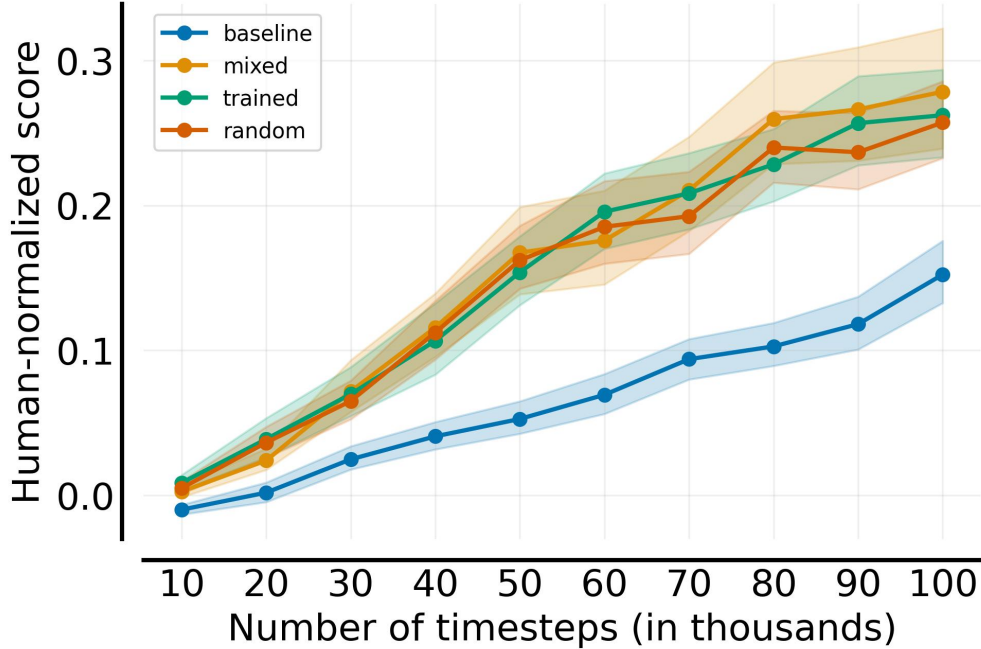


Figure 6.9: Effect of dataset composition on performance

Figure 6.11. We conjecture that a medium learning rate allows the agent to take advantage of representational knowledge from the triplet-induced encoder while being able to adjust the encoder’s parameter according to its learning progression, but a fixed encoder will not allow such adjustment, and high learning rate may not preserve the knowledge.

Based on the simulated studies, we argue that the dataset size, annotation quantity, and error tolerance study answer RQ1 that triplet-induced encoders possess representational knowledge of the task environment, and the more states or annotated triplets directly lead to better downstream performance, while less noise injected into the encoder would also result in better learning efficiency. Whereas dataset type, triplet loss margin, and learning rate suggest the best practice to leverage the triplet-induced encoder when used in downstream RL training.

6.5.2 Research Question 2 (RQ2)

To answer RQ2, we utilized AMT to collect human similarity judgments and compared them against the latent representations of our RL agents. As detailed in the methodology, we define “Agreement” as the fraction of triplets for which the majority vote of human annotators matches the similarity judgment produced by the agent’s encoder.

We evaluated the alignment between human perception and the agent’s representation

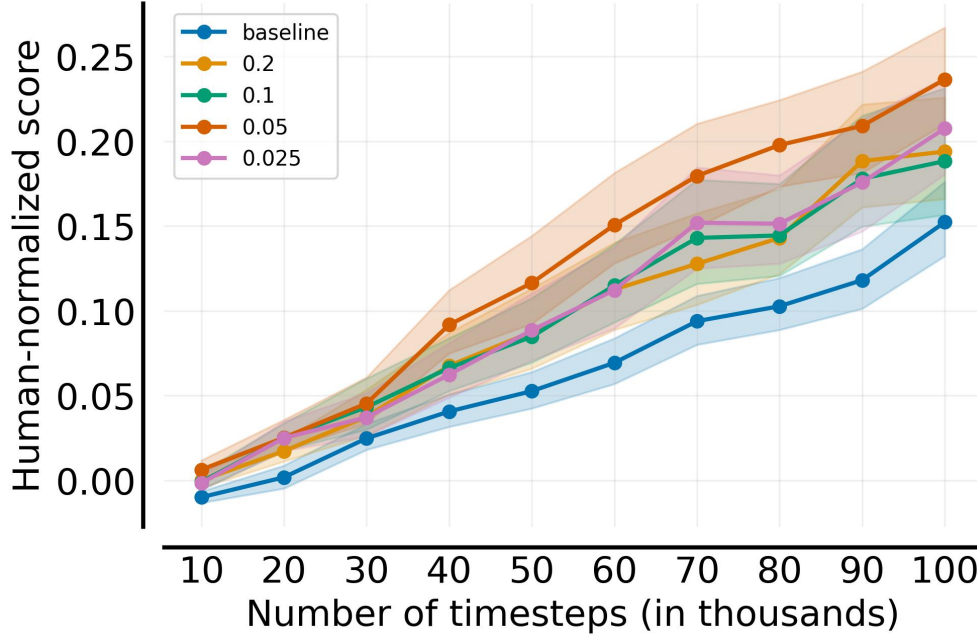


Figure 6.10: Triplet loss margins

across the training lifecycle of a Deep Q-Network. We compared the human agreement levels against two distinct baselines: an *Untrained* model (random initialization, step 0) and a *Training* model.

6.5.2.1 Crowd vs. Machine over Training

To further characterize this alignment and isolate the effect of representation learning from random chance, we tracked the *Absolute Agreement Gap* throughout the training process. This metric is defined as the element-wise difference between the agreement rate of the evolving agent at each training checkpoint and the static agreement rate of the randomly initialized untrained baseline in order to quantify the net increase in human-agent alignment attributable specifically to the training process.

Figures 6.12, 6.13, and 6.14 illustrate this evolution across three environments: *Seaquest*, *Demon Attack*, and *Battlezone*. To better visualize the dynamics of the early-stage learning phase, each figure includes a zoomed-in view of the first 5 million training steps.

Observing the trajectory of alignment across all three games shows a consistent three-stage developmental pattern, although the timing and magnitude vary by environment.

First, as highlighted by the zoomed-in views, we observe a rapid initial rise in the agreement gap across all environments. In *Seaquest*, the gap peaks sharply at roughly 0.4 within the first 500k steps. *Demon Attack* shows a similar trend, where the gap rapidly climbs

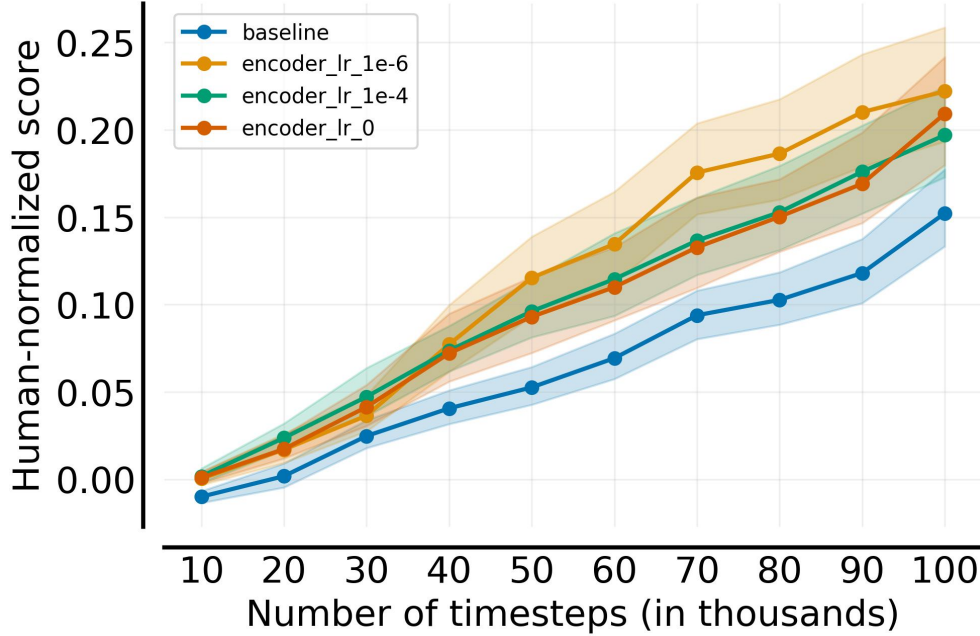


Figure 6.11: Learning rates of the encoder during RL training

to peak at approximately 0.1 within the first 300k steps before fluctuating. In *Battlezone*, despite its complexity, the gap rises to over 0.25 within the first 3 million steps. This high-agreement early-stage phase indicates that the agent’s initial acquisition of basic task competencies—such as survival and identifying salient objects—naturally aligns with human visual intuition. This finding validates the premise that human insight is particularly valuable for bootstrapping early-stage RL, as the agent is naturally predisposed to learn features that humans already recognize.

Second, following this initial spike, we observe a “dip” or divergence where the agreement gap declines. This represents an “adolescent phase” where the agent begins to optimize for specific, often narrow or counter-intuitive strategies to maximize reward (e.g., exploiting non-visual game mechanics). During this phase, the agent’s internal representation may overfit to these strategies, drifting away from the visual semantics humans rely on.

In *Battlezone* (Figure 6.14), this dip is particularly pronounced and delayed, occurring deeply around the 15M step mark. Unlike the 2D environments where visual features map directly to game dynamics, *Battlezone* requires a two-stage learning process due to its 3D, first-person perspective and partial observability. The agent must first master 3D navigation and perspective-taking (Phase 1) before it can effectively optimize for combat (Phase 2). This complexity delays the onset of the divergence. The deep dip at 15M likely marks the point where the agent discovers a high-reward strategy that is functionally effective but

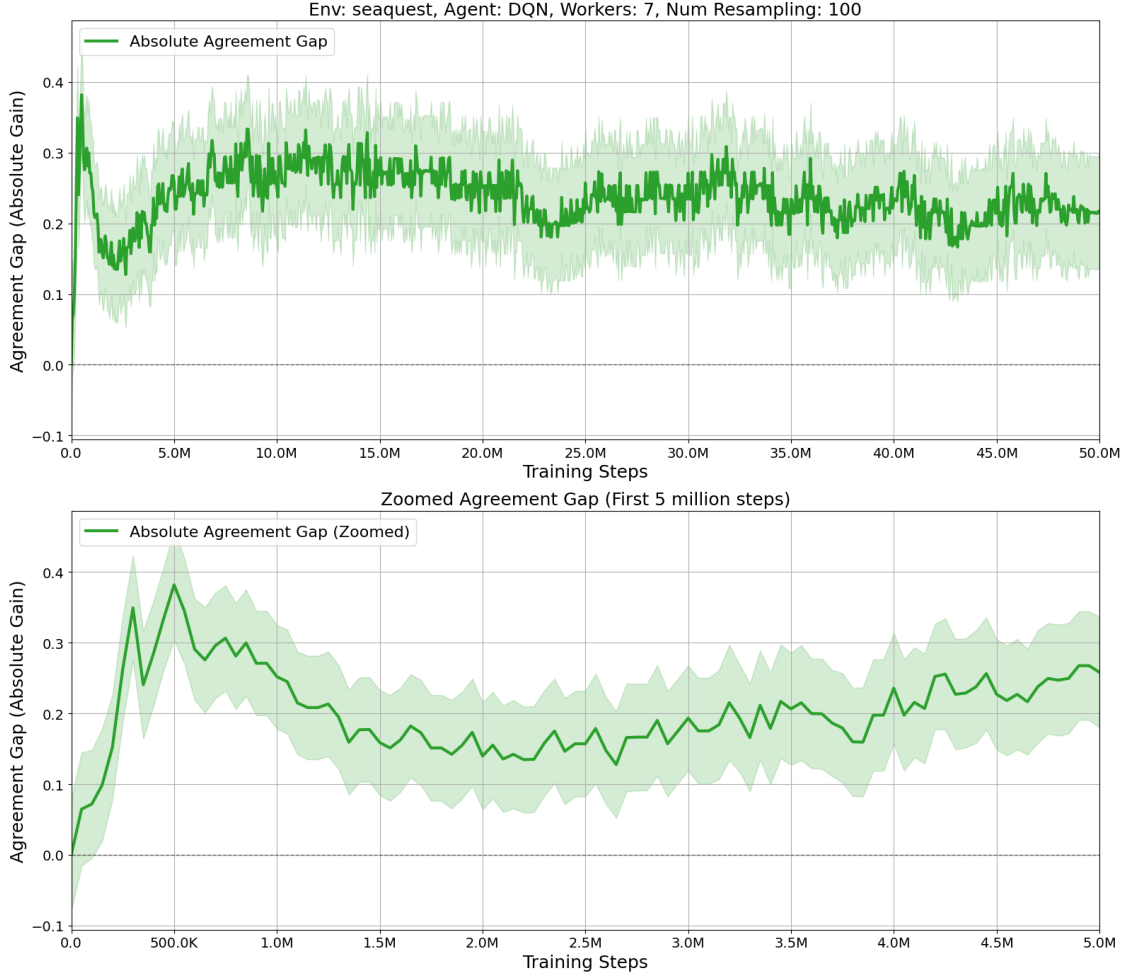


Figure 6.12: Absolute Agreement Gap throughout 50M training steps in Seaquest, with a zoomed-in view of the first 5 million steps (bottom).

visually opaque to human observers (e.g., navigating via hidden states rather than visible enemies).

Finally, the subsequent increase (the right side of the V-shape) suggests that as the agent’s policy recovers from exploitative strategies to more robust, generalized behaviors, its internal representation re-aligns with human visual intuition. The agent effectively learns that long-term survival requires attending to the same broad visual cues that humans prioritize.

6.5.2.2 Cross-Game Comparison

To compare performance across environments with varying baselines, we computed the *Normalized Agreement Gap*, which scales the absolute gain relative to the maximum possible improvement (i.e., $(\text{Training} - \text{Untrained}) / (1 - \text{Untrained})$).

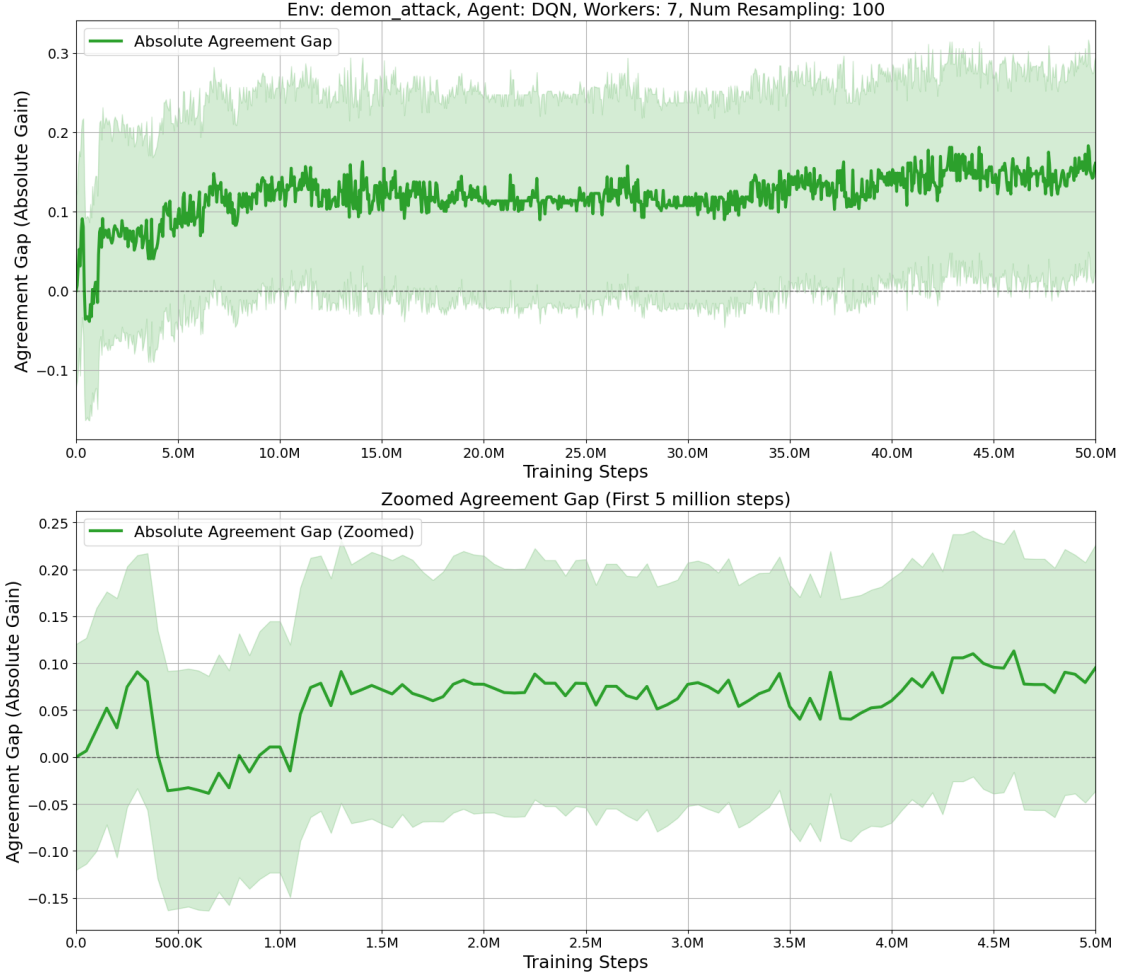


Figure 6.13: Absolute Agreement Gap throughout 50M training steps in Demon Attack, with a zoomed-in view of the first 5 million steps (bottom).

As shown in Figure 6.15, *Seaquest* (cyan curve) consistently demonstrates the highest alignment. It maintains a normalized gap between 30% and 50%. *Demon Attack* (brown curve) shows a moderate but stable alignment around 20%. In contrast, *Battlezone* (blue curve) fluctuates significantly, which is consistent with the finding that human intuitive judgment is context-dependent—it is most coherent in environments where outcomes are visually predictable and clearly observable.

6.5.2.3 Representational Drift Filtering

Finally, we investigated whether we could improve the quality of the signal by filtering for states where the agent’s representation is actively evolving. We compared the agreement rates when using a “Representational Drift Filter”—which selects triplets where the agent’s

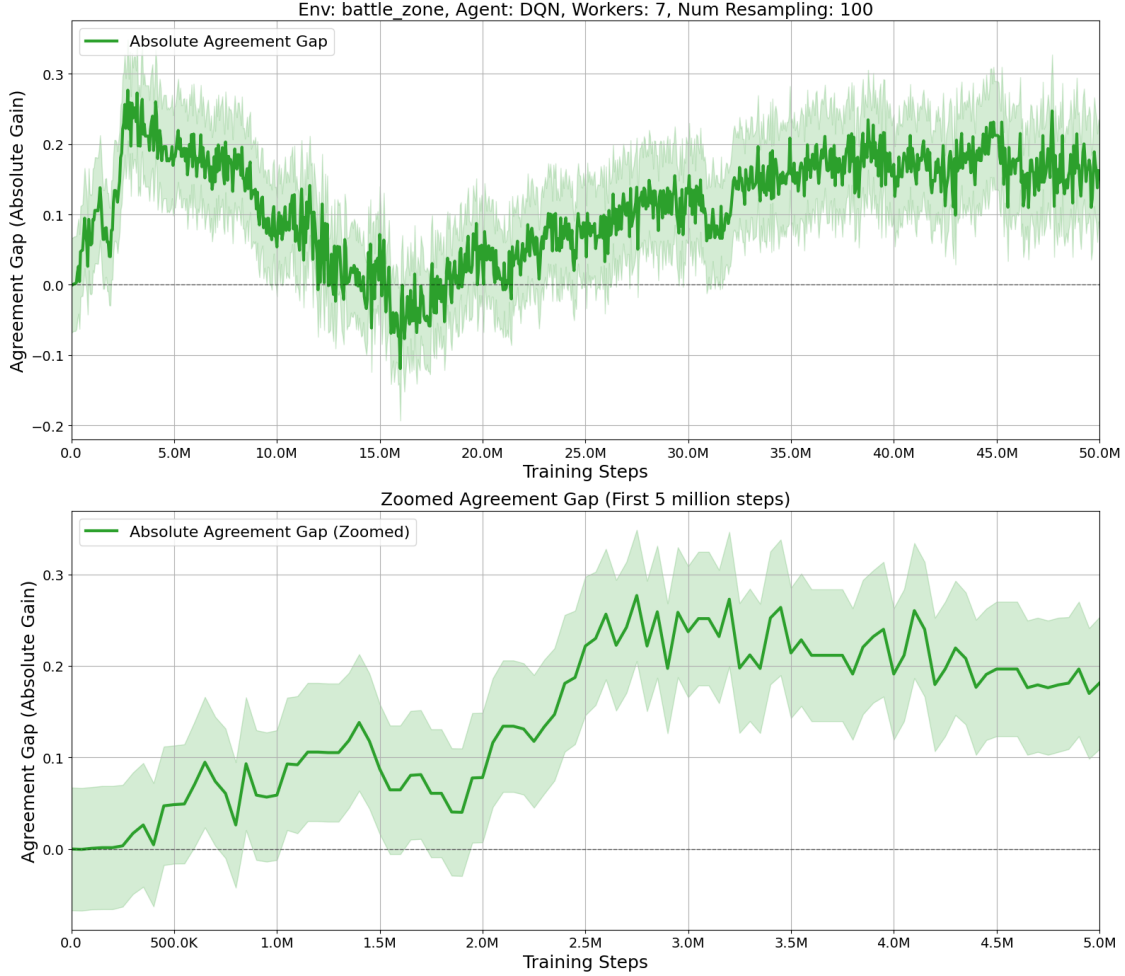


Figure 6.14: Absolute Agreement Gap throughout 50M training steps in Battlezone, with a zoomed-in view of the first 5 million steps (bottom).

representation changes between checkpoints—against a “No Filter” baseline. Specifically, we implemented this filter by identifying the subset of triplets where the agent’s preferred pair (the “top choice”) shifted during training. For a given triplet, we computed the pair of states with the minimum Euclidean distance in the latent space of both the untrained model and the current training model. The Representational Drift Filter retains only those triplets where this top-1 similarity choice differs between the two models. By isolating instances where the agent “changes its mind,” this method targets the active learning frontier so as to distinguish consequential representational shifts from trivial visual patterns.

Figure 6.16 presents the normalized agreement gap averaged across all three games. It is crucial to highlight the performance of the “No Filter” baseline (blue curve). Even without targeting specific states, the raw crowd consensus maintains a Normalized Agreement Gap of approximately 30% throughout training. This suggests that the “average” human judgment

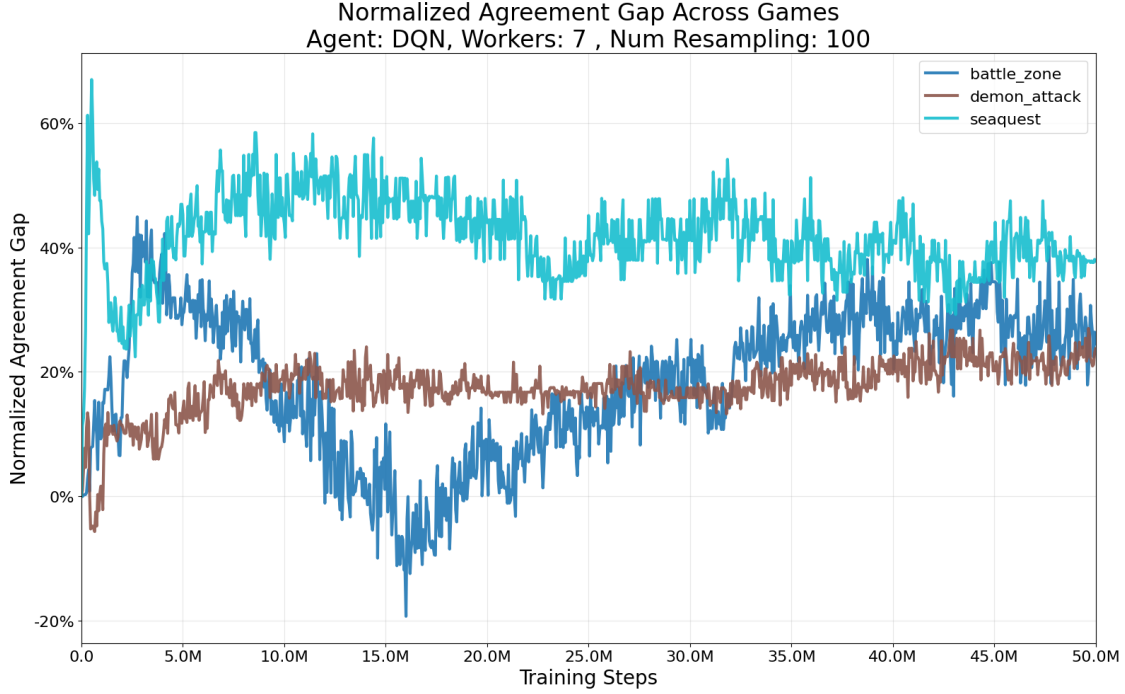


Figure 6.15: Normalized Agreement Gap across games. Seaquest (cyan) shows the strongest and most consistent alignment, while Battlezone (blue) remains the most volatile.

contains a coherent relevance signal that is structurally superior to the random representation of an untrained agent. The human prior, therefore, acts as a warm-start for the representation space. The improvement seen in the “Representational Drift Filter” (cyan curve) reveals that while the baseline human signal is strong, it is more effective (40%) when targeted at states undergoing representational shifts, rather than being applied uniformly.

It is important to note, however, that this Representational Drift Filter serves as an idealized, retrospective analysis rather than a practically deployable selection mechanism. In a real-world application, the “current” training model is unavailable during the initial data collection or pre-training phase. Nevertheless, this result serves as an existence proof for signal quality to illustrate that if we can successfully identify and target these regions of active representational change, the alignment between human intuition and agent learning can be improved.

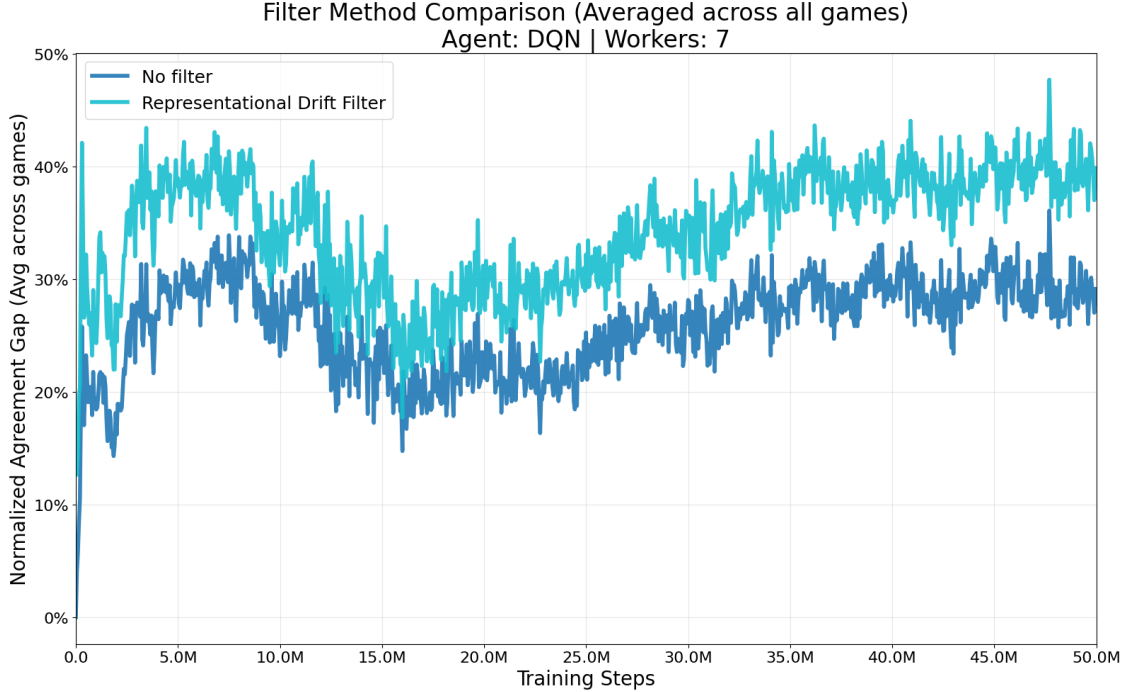


Figure 6.16: Comparison of filtering methods averaged across all games. The Representational Drift Filter (cyan) consistently yields higher normalized agreement than the unfiltered baseline (blue), particularly in later training stages.

6.6 Discussion

6.6.1 Characterizing Human Relevance Assessment

The oracle studies (RQ1) demonstrated that similarity-induced representations lead to faster and more efficient learning. This establishes a powerful new paradigm for human-guided RL: instead of providing feedback at the level of scalar rewards or trajectory-level credit assignment, CARAFE allows humans to contribute at the more abstract, foundational level of state representation.

Our results in RQ2 show that the utility of human similarity assessment exists on a spectrum rather than being binary. In environments with high visual clarity and predictable dynamics, such as *Seaquest*, we observed strong alignment (30 – 50% normalized gap) even with naive crowdsourcing, which indicates that human relevance priors are naturally coherent enough to drive learning in such tasks. However, in environments like *Battlezone*, the alignment drops to 20% due to partial observability and 3D complexity. This implies that while human intuition is a viable proxy for relevance, its reliability degrades when the visual modality disconnects from the underlying state dynamics.

Crucially, our analysis of the agreement gap trajectory shows that this alignment is not static but developmental. The rapid initial rise in agreement across all environments—visible in the zoomed analysis of the first 5 million steps—suggests that human intuition is most strongly aligned with the agent’s early-stage learning of basic competencies, such as survival. The subsequent “dip,” most prominent in *Battlezone* at 15M steps, likely marks a divergence where the agent transitions from these shared visual basics to optimizing for exploitative strategies. The eventual recovery indicates a return to robustness. This temporal pattern suggests that human insight is particularly valuable for *bootstrapping* agents through the initial “cold start” phase, effectively providing a general prior.

Our analysis of the Normalized Agreement Gap (Figure 6.16) indicates that the “gut feeling” of people effectively provides a “30% head start” in differentiating states compared to a tabula rasa agent. This baseline validity implies that the primary challenge in human-in-the-loop RL is not necessarily correcting human errors, but rather identifying the “consequential” moments (as shown by our filtering results) where this human prior can provide the highest information gain.

It is important to qualify our use of the “Agreement Gap.” We utilized an extensively trained agent as an oracle—a proxy for “ground truth” relevance. A low agreement score suggests that human judgments diverge from the representation of a converged, reward-maximizing agent. While this divergence typically implies noise in human assessment, we acknowledge a secondary possibility: humans may be identifying valid functional similarities that the agent has not yet learned or that rely on features distinct from the agent’s policy. Thus, our reported Agreement Gap should be viewed as a conservative estimate of utility: it measures the *safety* of the signal (alignment with known optimality) rather than its absolute potential to uncover novel strategies.

6.6.2 Limitations and Future Directions

The primary limitation remains the cost of collecting sufficient human annotations. However, our findings suggest a promising direction: rather than collecting massive amounts of undifferentiated human feedback, future systems could identify and prioritize triplets where human assessment provides the most value. Our temporal analysis suggests that this value is highest in the early training stages to establish a visual foundation, and later when targeted at states undergoing representational drift.

By encoding this intuitive anticipation of functional equivalence into the agent’s representation space, CARAFE effectively transfers human intuition into a format the machine can optimize. This suggests that the future of human-in-the-loop systems lies not just in cor-

recting agent behavior (actions), but in shaping agent perception (representation) to bridge the gap between human and machine cognition.

6.7 Summary

This chapter introduces CARAFE, a framework that addresses a fundamental limitation of retrospective relevance methods by learning a *prospective relevance* prior from human intuition. We shift the paradigm from gated, post-hoc review to internalizing human insight through a learned state representation that anticipates which states are critical for future outcomes.

The core innovation of CARAFE is its method for eliciting this prior. Recognizing that humans cannot reliably quantify abstract relevance, we instead leverage their capacity for comparative assessment. By collecting triplet similarity assessment from crowd workers—asking which pair of states is most similar in terms of future potential—we obtain a practical proxy signal. This signal trains an encoder, via triplet loss, to produce a low-dimensional embedding where geometric distance reflects human-perceived functional similarity so as to encode a relevance prior.

Our simulation studies establish the theoretical viability of this approach. Using an expert agent as an oracle to generate ideal similarity labels, we demonstrate that a representation learned via triplet loss can significantly accelerate downstream RL training. This confirms that a relevance-informed representation is a powerful mechanism for improving sample efficiency.

Our user study shows that human intuition is not uniformly reliable but follows a distinct developmental trajectory relative to agent learning. We identify a strong natural alignment during the early “bootstrapping” phase, where agents acquire basic visual competencies, followed by a divergence phase. This divergence is particularly pronounced and delayed in complex 3D environments like Battlezone. However, alignment recovers as agent strategies mature and generalize. This finding reframes the challenge from seeking a perfect human prior to strategically leveraging human insight where it is most informative: as a powerful accelerator for early-stage learning.

The study further quantifies the intrinsic value of human intuition. We find that raw, unfiltered human similarity judgments maintain a Normalized Agreement Gap of approximately 30% across tested environments. This implies that human relevance priors provide a signal that is structurally closer to an expert representation than a random initialization, which offers a “head start” for learning.

In summary, CARAFE provides a novel framework for human-guided RL by: (1) for-

malizing the learning of prospective relevance priors from human similarity assessment; (2) demonstrating the strong sample efficiency gains enabled by such representations under ideal conditions; and (3) quantifying the signal quality of human relevance priors, demonstrating that raw human feedback provides a $\sim 30\%$ improvement in representational alignment over an untrained baseline. This establishes human insight not merely as a corrective mechanism, but as a critical, quantifiable resource for bootstrapping early-stage learning.

While the prohibitive cost of large-scale crowdsourcing precluded a full end-to-end training run with human data, this work shows how to encode human insight into a reusable representation and thus moves a step closer to agents that can learn effectively with human prior.

CHAPTER 7

Conclusion and Future Work

This dissertation has explored a fundamental challenge in applying RL to real-world problems: the reliance on explicit, well-defined structures such as reward functions and relevance priors. These structures are often absent or difficult to specify, yet they are essential for guiding efficient learning. To bridge this gap, we proposed a paradigm shift: instead of requiring humans to provide what the algorithm needs—dense, formal specifications—we developed methods to elicit and utilize what humans naturally possess—intuitive, *fuzzy* signals about patterns and relevance.

By leveraging the collective intelligence of dynamic groups (crowds) with minimal assumptions about their expertise, we demonstrated two complementary pathways to augment RL systems. Part I addressed the absence of reward functions by introducing a method for pattern-based reward specification. Part II tackled the lack of relevance priors by developing techniques that inject human insight into both credit assignment and state representation. Together, these contributions provide a novel framework for making RL more accessible, adaptable, and applicable to a broader range of domains.

7.1 Summary of Contributions

This dissertation makes the following key contributions:

Part I: Human-aided Reward Design. We introduced a new perspective on reward specification that moves beyond traditional methods like preference learning or demonstration, which can constrain human expressiveness and assume ideal input. Chapter 3 formalized this perspective, arguing for the elicitation of reusable, fine-grained patterns that correlate with outcomes. This led to the development of **RECEPT** (Chapter 4), a system that elicits visual indicator patterns from crowds. Through a novel *Collect-View-Refine* (CVR) workflow, RECEPT robustly gathers patterns that serve as “labeling functions” to enable the automatic generation of consistent reward signals via pattern matching. Experiments in Atari environments showed that agents trained with RECEPT-generated rewards

could achieve up to 98% of the performance of agents trained with ground-truth reward functions. This validates pattern-based reward design as a viable and scalable alternative.

Part II: Relevance-driven Efficient Learning. We addressed the second major challenge in RL—the lack of prior knowledge about what is relevant for learning—by formalizing the concept of *relevance* and demonstrating its utility in two forms. **Montage** (Chapter 5) implemented *retrospective relevance* through Selective Memory Learning (SML), a framework that uses human feedback to identify and focus policy updates on the state-action pairs that are critical to an outcome so as to improve temporal credit assignment. In controlled simulations and a crowd study, Montage led to a significant acceleration in learning, allowing the agent to reach optimal performance with an order of magnitude fewer episodes.

To overcome the inherent limitations of retrospective analysis, we introduced **CARAFE** (Chapter 6), which shifts to *prospective relevance*. Chapter 6 presented an approach that learns a state representation to encode a human-derived prior about which states are functionally similar in their potential outcomes. By eliciting similarity assessments from crowds and training an encoder with triplet loss, CARAFE produces a low-dimensional embedding where proximity reflects human intuition about relevance. Simulation studies confirmed that such representations can dramatically accelerate downstream RL training. Furthermore, our user study demonstrated that unfiltered crowd feedback provides a 30% improvement in representational alignment over an untrained baseline and revealed that human intuition is not uniformly reliable but context-dependent.

7.1.1 The Unifying Theme: Leveraging Collective Intuition

A core theme unifying these contributions is the strategic use of *non-expert crowds* as a source of scalable, generalizable insight. We intentionally imposed minimal assumptions about human contributors, i.e., requiring neither domain expertise nor the ability to provide perfect demonstrations or consistent scalar feedback. This approach is predicated on three principles:

- **Generality through Minimal Assumptions:** Crowds represent the most general form of human input. By designing methods that work under these relaxed constraints, we aim to solve the hardest, most general version of the human-in-the-loop RL problem so as to maximize the potential scope of applications.
- **Precise Elicitation Enables Scalable Feedback:** The key to harnessing the “wisdom of the crowd” lies not in direct annotation of massive datasets, but in eliciting compact, reusable rules or judgments. RECEPT’s CVR workflow and CARAFE’s

triplet comparison are both designed to aggregate noisy crowd input into precise, high-leverage signals (patterns and similarity priors) that can be automatically applied at scale.

- **Capturing Intuition to Guide Learning:** The fundamental value extracted from people is their intuition—the *ability to recognize causation, identify what matters, and judge functional similarity*. This dissertation demonstrates that these fuzzy signals, when properly captured and structured, can provide the otherwise missing guidance RL agents need to learn efficiently and avoid mistakes that seem obvious to humans.

7.2 The Intuition Frontier in the Age of Foundation Models

The field of AI is undergoing a rapid transformation driven by foundation models. A prominent trend, exemplified by techniques like Reinforcement Learning with Verifiable Rewards (RLVR) [28, 52, 33, 100], seeks to minimize reliance on expensive human annotation in favor of automated verification, self-supervision, or feedback from powerful models themselves. For tasks with objective, verifiable correctness (e.g., code generation, mathematical reasoning), this shift is logical and powerful, making human input a potential bottleneck.

However, this trend primarily addresses the domain of *objective truth*. It leaves a critical gap in scenarios governed by *subjective, implicit, or ambiguous dynamics*—precisely the domain where human intuition excels. Navigating an open-ended world, interpreting unstated social norms, or making trade-offs under uncertainty often relies on quick, reasonable “guesses”. This is the *Intuition Frontier*.

This dissertation positions itself at this frontier. Crucially, human intuition is not only an engine for handling ambiguity, but also the aggregated output of a vast, lived experience, i.e., a powerful, subconscious synthesis of a lifetime of interaction, experimentation, and tangential learning [92]. For example, an AI may master a simulated Rubik’s cube, but it misses the broader cognitive lessons in manipulation, reasoning, and algorithm design that a person internalizes through play. Intuition represents this irreplaceable repository of heterogeneous, grounded knowledge that is absent from static datasets.

The methods developed in this dissertation—RECEPT’s pattern elicitation (Chapter 4), Montage’s relevance filtering (Chapter 5), and CARAFE’s functional similarity judgments (Chapter 6)—are mechanisms for codifying this uniquely human ability. They translate the subjective, hard-to-articulate nature of intuition into structured signals (i.e., reward patterns, relevance labels, similarity constraints) that machines can optimize. In an era moving toward automated correctness, our work argues for the unique value of human intuition as

a fundamental guide for learning in ambiguous, real-world contexts. It is a resource fundamentally different from, and complementary to, the optimization powers of current AI.

7.3 Limitations

While this work demonstrates the promise of crowd-in-the-loop RL, several limitations must be acknowledged:

Context-Dependent Performance of Human Intuition. Our findings, particularly in Chapter 6, reveal that the alignment between human intuition and an optimal agent’s representation is not uniform. In environments with clear visual causality (e.g., *Seaquest*), human judgments showed strong coherence. In contrast, in environments with partial observability or complex abstractions (e.g., *Battlezone*), alignment was weaker and more volatile. This does not necessarily mean human intuition is “wrong,” but it highlights that its utility is context-dependent. The reliability of the crowd as an oracle degrades when the perceptual channel (e.g., a single first-person view) disconnects from the underlying functional state.

Scalability vs. Depth of Insight. There exists a fundamental tension between the *depth* of human insight, which requires focused cognitive effort, and the *massive scale* of data needed to train modern RL systems or foundation models. While our methods are designed to be more efficient than per-sample annotation, eliciting high-quality patterns or similarity judgments for complex, high-dimensional state spaces remains a non-trivial investment. The cost of comprehensive human feedback at the scale required to train a meaningfully capable model is currently prohibitive.

7.4 Future Directions

The limitations and trends discussed above illuminate promising avenues for future research:

Relaxing Constraints to Incorporate Expertise. This dissertation tackled the problem under the most general constraints (non-expert crowds). A natural direction is to relax this constraint and investigate how *specialized expertise* could enhance these methods. Domain experts could provide more accurate patterns, finer-grained relevance assessments, or more nuanced similarity criteria, potentially leading to even greater sample efficiency and robustness without degrading the generality of the approach. The frameworks established here can serve as a foundation for integrating tiered human input, from crowd commonsense to expert insight.

Foundation Models as “Intuition Scalers.” The most compelling future direction lies in viewing foundation models not as replacements for human intuition, but as tools

to scale the specific intuitive priors that humans provide. Here, human intuition serves as the essential “seed” or high-level semantic rule. For instance, a small set of crowd-elicited indicator patterns (from RECEIPT) or triplet comparisons (from CARAFE) could be leveraged with a Vision-Language Model (VLM) to recognize these relevant patterns or assess functional similarity across millions of unlabeled frames, effectively generalizing the human-experiential prior at machine scale. This creates a powerful hybrid pipeline that directly addresses the experience gap: humans provide the compressed “what matters” from a lifetime of experience, and foundation models instantiate and generalize that prior at machine scale. This approach combines human semantic understanding and experiential aggregation with automated scalability.

Hybrid Training Regimes and Process Supervision. Future foundation model post-training pipelines could strategically combine different forms of feedback. Automated methods (such as RLVR or self-play) could handle verifiable, logical skill acquisition [88]. Our human-relevance priors could then be injected to guide high-level goal formulation, resolve ambiguous trade-offs, or provide the “common sense” oversight that prevents degenerate or unintended behaviors.

A potential application of these mechanisms lies in addressing a critical weakness in current large language models (LLMs): their inability to handle uncertainty in chain-of-thought (CoT) reasoning, particularly for open-ended domains like social science, economics, or strategic planning. In such contexts, each reasoning step carries epistemic (knowledge-based) or aleatoric (inherently ambiguous) uncertainty that current CoT implementations ignore. This leads to overconfident or poorly calibrated reasoning. Montage’s stepwise relevance assessment framework can be adapted to curate uncertainty-aware CoT data. By having human annotators—either crowd workers or domain experts—rate each reasoning step not just for binary relevance but also for confidence and ambiguity type with respect to the final outcome, we can generate training data that encodes refined human judgment about uncertainty. This curated data can then train foundation models to generate CoT traces with calibrated confidence estimates at each step, enabling more transparent and reliable reasoning.

Beyond retrospective filtering, CARAFE’s prospective relevance framework offers a mechanism to address the challenge of coherence in latent space reasoning. While latent reasoning improves efficiency by performing inference within hidden states, it inherently lacks interpretability [115]. CARAFE can mitigate this by serving as a grounding mechanism for these hidden states. By constraining the reasoning model to trajectories that remain consistent with the semantic structure of CARAFE’s relevance space, one can promote alignment of the internal reasoning process with human preferences, even in the absence of intermediate

textual supervision.

In this hybrid training regime, reasoning models would thus learn to integrate: (1) *verifiable skills* acquired through automated feedback for objective tasks (e.g., code generation, mathematical proof), and (2) *uncertainty-calibrated reasoning* guided by human intuition for subjective, open-ended problems. This approach would allow foundation models to not only solve problems but also express when and why they are uncertain, which is a critical capability for building trustworthy AI in complex real-world domains.

7.5 Closing Thought

The path to creating robust, generalist AI requires more than immense compute and automated verification of objective correctness. It demands mechanisms for agents to acquire and align with the *common sense intuition*—the aggregated wealth of lived experience—that allows humans to navigate a complex, open-ended, and often ambiguous world. This dissertation has taken steps in that direction, demonstrating that the collective intuition of people can be systematically elicited and transformed into actionable guidance for learning agents. By bridging the gap between human intuition and machine precision, and by providing a framework to use human insight as a scalable prior, we move closer toward creating AI systems that are not only powerful but also adaptable, sample-efficient, and aligned with the subtle ways humans understand and interact with the world.

BIBLIOGRAPHY

- [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. ACM, 2004.
- [2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. Advances in neural information processing systems, 34:29304–29320, 2021.
- [3] Riad Akrou, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 116–131. Springer, 2012.
- [4] James F Allen. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11):832–843, 1983.
- [5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. CoRR, abs/1606.06565, 2016.
- [6] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. Robotics and autonomous systems, 57(5):469–483, 2009.
- [7] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. Artificial Intelligence, 297:103500, 2021.
- [8] Michael Bain and Claude Sammut. A framework for behavioural cloning. In Machine intelligence 15, pages 103–129.
- [9] A Barreto, W Dabney, R Munos, JJ Hunt, T Schaul, H van Hasselt, and D Silver. Successor features for transfer in reinforcement learning. arxiv. arXiv preprint arXiv:1606.05312, 2016.
- [10] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, 2013.
- [11] Andreea Bobu, Yi Liu, Rohin Shah, Daniel S Brown, and Anca D Dragan. Sirl: Similarity-based implicit representation learning. In Proceedings of the 2023

- ACM/IEEE International Conference on Human-Robot Interaction, pages 565–574, 2023.
- [12] Ramon F Brena and Enrique Garcia-Ceja. A crowdsourcing approach for personalization in human activities recognition. Intelligent Data Analysis, 21(3):721–738, 2017.
 - [13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.
 - [14] Ricardo Buettner. A systematic literature review of crowdsourcing research from a human resource management perspective. In 2015 48th Hawaii International Conference on System Sciences, pages 4609–4618. IEEE, 2015.
 - [15] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. arXiv preprint arXiv:2307.15217, 2023.
 - [16] Thomas Cederborg, Ishaan Grover, Charles L Isbell, and Andrea L Thomaz. Policy shaping with human teachers. In Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
 - [17] William G. Chase and Herbert A. Simon. Perception in chess. Cognitive Psychology, 4(1):55–81, 1973.
 - [18] Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. Cascade: Crowdsourcing taxonomy creation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 1999–2008. ACM, 2013.
 - [19] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4299–4307. Curran Associates, Inc., 2017.
 - [20] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–9. IEEE, 2018.
 - [21] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active reward learning. In Proceedings of Robotics: Science and Systems (RSS), January 2014.
 - [22] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In Advances in Neural Information Processing Systems, pages 11698–11709, 2019.

- [23] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. <https://github.com/openai/baselines>, 2017.
- [24] Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. In 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 1577–1584. IEEE, 2018.
- [25] K Anders Ericsson, Robert R Hoffman, Aaron Kozbelt, and A Mark Williams. The Cambridge Handbook of Expertise and Expert Performance. Cambridge Handbooks in Psychology. Cambridge University Press, 2 edition, 2018.
- [26] Manfred Fahle. Human pattern recognition: parallel processing and perceptual learning. Perception, 23(4):411–427, 1994.
- [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, 2016.
- [28] Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. arXiv preprint arXiv:2410.15115, 2024.
- [29] Enrique Garcia-Ceja and Ramon Brena. Activity recognition using community data to complement small amounts of labeled instances. Sensors, 16(6):877, 2016.
- [30] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In International Conference on Machine Learning, pages 2170–2179. PMLR, 2019.
- [31] Fernand Gobet and Herbert A Simon. Recall of rapidly presented random chess positions is a function of skill. Psychonomic bulletin & review, 3(2):159–163, 1996.
- [32] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In Advances in neural information processing systems, pages 2625–2633, 2013.
- [33] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.
- [34] Danna Gurari, Yinan Zhao, Suyog Dutt Jain, Margrit Betke, and Kristen Grauman. Predicting how to distribute work between algorithms and humans to segment an image batch. Int. J. Comput. Vis., 127(9):1198–1216, 2019.
- [35] David Ha and Jürgen Schmidhuber. World models. arXiv preprint arXiv:1803.10122, 2018.

- [36] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In Advances in neural information processing systems, pages 6765–6774, 2017.
- [37] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In International Conference on Learning Representations (ICLR), 2020.
- [38] Simone Hantke, Zixing Zhang, and Björn W Schuller. Towards intelligent crowd-sourcing for audio data annotation: Integrating active learning in the real world. In INTERSPEECH, pages 3951–3955, 2017.
- [39] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.
- [40] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
- [41] Ting-Hao Kenneth Huang, Joseph Chee Chang, and Jeffrey P Bigham. Evorus: A crowd-powered conversational assistant built to automate itself over time. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, page 295. ACM, 2018.
- [42] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In Advances in Neural Information Processing Systems, pages 8011–8023, 2018.
- [43] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3668–3678, 2015.
- [44] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. Journal of artificial intelligence research, 4:237–285, 1996.
- [45] Timo Kaufmann, Sarah Ball, Jacob Beck, Eyke Hüllermeier, and Frauke Kreuter. On the challenges and practices of reinforcement learning from real human feedback. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 276–294. Springer, 2023.
- [46] Reuben Kirkham, Carlton Shepherd, and Thomas Plötz. Blobsnake: gamification of feature extraction for ‘plug and play’ human activity recognition. In Proceedings of the 2015 British HCI Conference, pages 74–81. ACM, 2015.
- [47] W. Bradley Knox and Peter Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In IEEE 7th International Conference on Development and Learning, August 2008.

- [48] William Bradley Knox. Learning from human-generated reward. PhD thesis, The University of Texas at Austin, 2012.
- [49] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.
- [50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
- [51] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. arXiv preprint arXiv:1606.02396, 2016.
- [52] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. arXiv preprint arXiv:2411.15124, 2024.
- [53] Walter S Lasecki, Mitchell Gordon, Danai Koutra, Malte F Jung, Steven P Dow, and Jeffrey P Bigham. Glance: Rapidly coding behavioral video with the crowd. In Proceedings of the 27th annual ACM symposium on User interface software and technology, pages 551–562. ACM, 2014.
- [54] Walter S Lasecki, Young Chol Song, Henry Kautz, and Jeffrey P Bigham. Real-time crowd labeling for deployable activity recognition. In Proceedings of the 2013 conference on Computer supported cooperative work, pages 1203–1212. ACM, 2013.
- [55] Walter S Lasecki, Leon Weingard, George Ferguson, and Jeffrey P Bigham. Finding dependencies between actions using the crowd. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 3095–3098. ACM, 2014.
- [56] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. International Conference on Machine Learning (ICML), 2020.
- [57] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In International Conference on Machine Learning (ICML), 2021.
- [58] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. CoRR, abs/1811.07871, 2018.
- [59] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Francois Goudou, and David Filliat. State representation learning for control: An overview. Neural Networks, 108:379–392, 2018.

- [60] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research, 17(1):1334–1373, 2016.
- [61] Sergey Levine and Vladlen Koltun. Guided policy search. In International Conference on Machine Learning, pages 1–9, 2013.
- [62] Edwin Lughofer. Hybrid active learning for reducing the annotation effort of operators in classification systems. Pattern Recognition, 45(2):884–896, 2012.
- [63] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. Interactive learning from policy-dependent human feedback. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2285–2294. JMLR. org, 2017.
- [64] Mark P Mattson. Superior pattern processing is the essence of the evolved human brain. Frontiers in neuroscience, 8:265, 2014.
- [65] Sarah E Minson, Benjamin A Brooks, Craig L Glennie, Jessica R Murray, John O Langbein, Susan E Owen, Thomas H Heaton, Robert A Iannucci, and Darren L Hauser. Crowdsourced earthquake early warning. Science advances, 1(3):e1500036, 2015.
- [66] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928–1937, 2016.
- [67] 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, 2015.
- [68] P Read Montague, Peter Dayan, and Terrence J Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. Journal of neuroscience, 16(5):1936–1947, 1996.
- [69] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In Advances in Neural Information Processing Systems, pages 2775–2785, 2017.
- [70] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In In Proceedings of the Sixteenth International Conference on Machine Learning, pages 278–287, 1999.
- [71] Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In Proc. 17th International Conf. on Machine Learning, 2000.
- [72] Yael Niv. Reinforcement learning in the brain. Journal of Mathematical Psychology, 53(3):139–154, 2009.

- [73] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.
- [74] John Prpić, Araz Taeihagh, and James Melton. The fundamentals of policy crowdsourcing. Policy & Internet, 7(3):340–361, 2015.
- [75] Jay Pujara, Ben London, and Lise Getoor. Reducing label cost by combining feature labels and crowdsourcing. In ICML workshop on Combining learning strategies to reduce label cost, 2011.
- [76] Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo A. Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. CoRR, abs/1711.09602, 2017.
- [77] Divya Ramesh, Anthony Liu, Jean Song, Andres Echeverria, Nicholas Waytowich, and Walter S Lasecki. Yesterday’s reward is today’s punishment: Contrast effects in human feedback to reinforcement learning agents. In The 19th International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2020.
- [78] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason A. Fries, Sen Wu, and Christopher Ré. Snorkel: rapid training data creation with weak supervision. VLDB J., 29(2-3):709–730, 2020.
- [79] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In Advances in neural information processing systems, pages 3567–3575, 2016.
- [80] 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, 2011.
- [81] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3):211–252, 2015.
- [82] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In Robotics: Science and Systems, 2017.
- [83] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 815–823, 2015.
- [84] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International conference on machine learning, pages 1889–1897, 2015.

- [85] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017.
- [86] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In ICRA, 2018.
- [87] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.
- [88] Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. arXiv preprint arXiv:2511.22570, 2025.
- [89] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484, 2016.
- [90] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362(6419):1140–1144, 2018.
- [91] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. Nature, 550(7676):354, 2017.
- [92] David Silver and Richard S Sutton. Welcome to the era of experience.
- [93] Avi Singh, Larry Yang, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019, 2019.
- [94] Rion Snow, Brendan O’Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In Proceedings of the conference on empirical methods in natural language processing, pages 254–263. Association for Computational Linguistics, 2008.
- [95] Jean Y. Song. Eliciting and Leveraging Input Diversity in Crowd-Powered Intelligent Systems. PhD thesis, The University of Michigan, 2019.
- [96] Shay Sosko and Sagi Dalyot. Crowdsourcing user-generated mobile sensor weather data for densifying static geosensor networks. ISPRS International Journal of Geo-Information, 6(3):61, 2017.
- [97] Richard S Sutton. Temporal credit assignment in reinforcement learning. 1984.

- [98] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.
- [99] Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 617–624. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- [100] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. arXiv preprint arXiv:2501.12599, 2025.
- [101] Andrea Lockerd Thomaz, Cynthia Breazeal, et al. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1. AAAI Press, 2006.
- [102] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. Advances in Neural Information Processing Systems, 33:4235–4246, 2020.
- [103] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
- [104] Edward C Williams, Nakul Gopalan, Mine Rhee, and Stefanie Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–7. IEEE, 2018.
- [105] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In Advances in neural information processing systems, pages 5279–5288, 2017.
- [106] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In Proceedings of the IEEE international conference on computer vision, pages 4705–4713, 2015.
- [107] Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. Effective multi-label active learning for text classification. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 917–926. ACM, 2009.
- [108] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Improving sample efficiency in model-free reinforcement learning with linearly augmented image representations. In AAAI, 2021.
- [109] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In International Conference on Learning Representations, 2021.

- [110] Yang Yu. Towards sample efficient reinforcement learning. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 5739–5743. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
- [111] Maryam Zare, Parham M Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. IEEE Transactions on Cybernetics, 2024.
- [112] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. arXiv preprint arXiv:2006.10742, 2020.
- [113] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–8. IEEE, 2018.
- [114] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. In Advances in Neural Information Processing Systems, pages 4644–4654, 2018.
- [115] Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. arXiv preprint arXiv:2507.06203, 2025.