UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science
Agents, Interaction and Complexity

Explaining the Future Context of Deep Reinforcement Learning Agents' Decision-Making

by

Mark Towers

B.Sc. in Computer Science

ORCiD: 0000-0002-2609-2041

A thesis for the degree of
Doctor of Philosophy

June 2025

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Doctor of Philosophy

Explaining the Future Context of Deep Reinforcement Learning Agents' Decision-Making

by Mark Towers

Deep reinforcement learning has achieved superhuman performance in numerous environments. Despite these advances, there are limited tools to understand why agents make decisions. A central issue is how specific actions enable agents to collect rewards or achieve goals far in the future. Understanding this future context for an agent's decision-making is critical to explaining their choices. To date, however, little research has explored such temporal explanations. Therefore, we investigate how to explain the future context of agents' decision-making for both pretrained agents, using a memory of past behaviour, and architecturally modified agents, explicitly outputting their next N expected rewards. We evaluate these explanations with user surveys, finding them preferred and more effective to baseline algorithms in Atari environments.

We develop three novel video-based explanations for pretrained agents. Two of these require no domain knowledge, a common feature of prior work, while the third incorporates limited domain knowledge. These approaches are the first local explanations that use a memory of how an agent acted in the past to explain their current decision-making. We collect similar decisions from past states or skills, showcasing them to users to help visualise an action's possible future outcomes.

We identify that deep reinforcement learning agents implicitly compute their beliefs about the future when predicting their rewards (i.e., Q-value or State-value). From this, we prove that an agent's Q-value can be transformed into computing the expected reward for each future timestep. This opens up the opportunity to explain an agent's confidence and decision-making for individual future timesteps. This innovation allows us to propose a novel training algorithm referred to as Temporal Reward Decomposition, where agents output their expected rewards for the next N timesteps. From this, we pioneer three novel explanations for users with a strong understanding of reinforcement learning. For non-technical users, we propose a fourth explanation using Large Language Models to summarise the future rewards in natural language.

We conduct two user surveys to evaluate our temporal explanations against two baseline algorithms. In the second, we propose a novel evaluation methodology inspired by debugging, where users must identify an unknown agent's goal from an explanation of its decision-making. We find that in both user surveys, our temporal explanations were preferred and, in the second, were significantly more effective for determining an agent's goal.

Contents

List of Figures **vii**

List of Tables **xi**

Declaration of Authorship **xiii**

Acknowledgements **xv**

Definitions and Abbreviations **xvii**

1 Introduction **1**
1.1 Thesis Contributions and Structure 4

2 Background **7**
2.1 Reinforcement Learning Mathematics 7
2.2 Deep Reinforcement Learning 9
2.3 Explanation Mechanism Properties 11
2.3.1 Explanation Audience 12
2.3.2 Model Interpretability 14
2.3.3 Explanation Type 15
2.3.4 Mechanism Categories 16
2.3.5 Explanation Medium 16

3 Literature Review **19**
3.1 Challenges in Explainable Reinforcement Learning 19
3.2 Explainable Reinforcement Learning Survey 21
3.2.1 Saliency Map Explanations 24
3.2.2 Natural Language Explanations 27
3.2.3 Temporal Explanations 29
3.2.4 Reward Explanations 33
3.3 Evaluating XRL Explanations 34
3.4 Discussion 36

4 Video-Based Explanations of Deep Reinforcement Learning Agents **39**
4.1 Explanation Mechanisms 41
4.1.1 Dataset Similarity Explanations 41
4.1.2 Skill Explanation 42
4.1.3 Plan Explanation 43
4.2 Skill Similarity Metrics 44

4.2.1 Skill Alignment Metric 44
4.2.2 Skill Distribution Metric 46
4.3 Quality of Explanations and Skills Extracted 46
4.3.1 Human Evaluation of Explanations 46
4.3.2 Skill Extraction Analysis 53
4.4 Discussion 54

5 Explaining an Agent's Future Beliefs through Temporally Decomposing Future Reward Estimators 57
5.1 Temporally Decomposed Future Reward Estimators 59
5.2 Retraining Pretrained Agents for TRD 63
5.3 Explaining an Agent's Future Beliefs and Decision-Making 65
5.3.1 What Rewards to Expect and When? 65
5.3.2 What Observation Features are Important? 68
5.3.3 What is the Impact of an Action Choice? 68
5.4 Discussion 70

6 A Comparative User Evaluation of Goal Identification using Explanations 73
6.1 Natural Language Summarisation of Future Rewards 75
6.2 Comparative User Evaluation Design 82
6.3 Analysing User Comprehension 87
6.3.1 Can Users Accurately Predict Agent Goals? 88
6.3.2 What do Users believe about the Explanation? 96
6.4 Discussion 102

7 Conclusion and Future Work 105

References 111

Appendix A Impact Assessment 123
Appendix A.1 The Motivation and Purpose of this Research 123
Appendix A.2 The Use of Human Studies for Impactful Research 125
Appendix A.3 Conducting Research with the Right Purpose and People . . . 126

Appendix B Saliency Map Hyperparameters 129

Appendix C Temporal Reward Decomposition for State-value function 133

Appendix D Code Implementation of Temporal Reward Decomposition 135

Appendix E LLM Prompts for Atari Games 137

Appendix F Screenshots of Comparative User Evaluation Survey 139

Appendix G More Analysis and Graphs from Comparative User Evaluation 151

List of Figures

1.1 a) Policy-based Explanation: An interpretable decision tree such that the decision-making paths can be explained. b) Feature-based Explanation: Piece colour represents their perceived importance to the agent's decision making Gupta et al. (2020). c) Temporal Explanation: Explains the agent's future decision-making with the numbered sequence of expected future actions. In practice, a video of these intended moves would be shown to the user. 2
1.2 Conceptual Diagram of the Chapters, Explanation Mechanisms, User Surveys completed in this Chapter. 6

2.1 Reinforcement Learning environment loop with an agent taking action for a given observation and receiving a reward and updated observation (along with termination status). 7
2.2 Visual representation of the definition of Explainable Artificial Intelligence 11
2.3 A graph of explanation properties with example XAI algorithms. Source: Belle and Papantonis (2021) 12
2.4 Outline of the motivations of different types of audiences in XAI. Source: Belle and Papantonis (2021) 14

3.1 A saliency map where the purple areas show the points that most influence the policy of the agent. Source: Greydanus et al. (2018) 25
3.2 An example of LIME for an image of a dog playing the guitar in the image and the top 3 classes for the Google Inception v3 model. Source: Ribeiro et al. (2016a) 27
3.3 Shows how the rule-based and learner models use input data to generate explanations. Source: Wang et al. (2020) 28
3.4 Directed Acyclic graph for the causal model is used to generate explanations. Source: Madumal et al. (2020a) 30
3.5 a. MDP diagram: the relationship between states and actions. b. SMDP diagram: Edge colours represent different skills. c. AMDP diagram: colours represent the clusters of the aggregated states. d. SAMDP diagram: colour clusters found after transforming the state space with dashed arrows for explaining skills and red arrows showing the overall policy. Source: Zahavy et al. (2016) 31

4.1 Example skill network for Atari Breakout using hand-labelled skills. . . 44
4.2 Example individual question shown to the user. 48
4.3 Example contrastive question shown to the user. 49
4.4 Box plot of the user ratings of each explanation mechanism and question as part of the individual evaluation section in the user survey. 50

4.5 A boxplot of prior environment knowledge and per environment explanation ratings. 51
4.6 Heatmap for each statement of the user preference for each explanation mechanism. Grid elements are the percentage of users who explicitly prefer the row explanation mechanism over the column explanation mechanism. 52
4.7 A stacked histogram of per-user variance in the Explanation type ratings. 53
4.8 The skill alignment of two sets of skills for the hand-labelled and algorithmically-labelled trajectories. 54
4.9 A stacked histogram of skill lengths of hand-labelled and algorithmically discovered skills. The skill lengths were clipped to 70. 55

5.1 Example Gridworld with an agent and two paths (up and down) that contain different rewards. 58
5.2 Interquantile mean training curves for Atari TRD-DQN agents for three environments (Breakout, Space Invaders, and Ms Pacman) with three repeats, normalised by the teacher's score. Offline and Online indicate where training used the offline replay buffer and the online environment steps. 64
5.3 The Mean Squared Error between the student (TRD agent) and pretrained teacher agent averaged over three Atari environments with three repeats. Offline and Online indicate where training used the offline replay buffer and the online environment steps. 65
5.4 A Space Invaders observation (left) with the respective predicted next 40 future expected rewards (right) 66
5.5 A Breakout observation (left) with the respective predicted next 40 future expected rewards (right) 66
5.6 A Ms. Pac-man observation (left) with the respective predicted next 40 future expected rewards (right) 67
5.7 A Seaquest observation (left) with the respective predicted next 40 future expected rewards (right) 67
5.8 GradCAM saliency maps for the $t + 1$ and $t + 40$ expected reward along with their difference for a Breakout observation. 69
5.9 The difference of each future expected reward for taking Fire and Noop actions of the observation for the Atari Seaquest environment. 70
5.10 The difference of each future expected reward for taking Left and Right actions of the observation for the Atari Riverraid environment. 71

6.1 Flowchart of the Goal Identification Task 82
6.2 Ms Pacman with the different reward sources labelled 83
6.3 Flowchart of the comparative user evaluation. Blue boxes represent user questions, and the orange boxes represent information given to the user. 85
6.4 The number of users who got each possible number of correct answers for each explanation mechanism. 88
6.5 A bar chart for each explanation mechanism with the accuracy of the explanation mechanism across each agent goal. Each bar's accuracy is averaged across 100 answers from users. The dotted line is 25%, the expected accuracy for random guesses. 89

6.6 Confusion matrix for the number of times the predicted and true agent goal for each explanation mechanism occurs, where the rows are the actual agent goal and the column is the predicted. Correct answers are found on the top left to bottom right diagonal. Each row will sum to 100 as the 20 observations are each evaluated five times. 90
6.7 A histogram of the time taken for each explanation mechanism. The time taken is clipped at 60 seconds. 92
6.8 A scatter plot of each user's cumulative time taken against their accuracy across all questions, with the red line being a plot of the linear regression. Individual question time taken is clipped at 60 seconds. The dashed line is at 25%, representing the expected accuracy if users are selected randomly. 94
6.9 Barchart of the average accuracy (in blue) and a line plot of the time taken (in orange) for an ordinal question. 95
6.10 Bar chart of user accuracy with their selected confidence. Very unconfident and unconfident are grouped as (very) unconfident, and very confident and confident are grouped as (very) confident. The dashed line represents the expected accuracy with random guesses. 97
6.11 Scatter plot of each user's average accuracy and confidence for all explanation mechanisms. For the user average confidence, we discretise to Very Unconfident=1, Unconfident=2, Neutral=3, Confident=4, and Very Confident=5. 98
6.12 Histogram of the number of times on the left and the accuracy for each explanation mechanism, confidence, ease of identification, or understanding is selected by users. 100

Appendix A.1 The number of papers published each year from 2010 to 2024 that include "Explainable Artificial Intelligence" or "Explainable Reinforcement Learning". 124

Appendix B.1 A comparison of the saliency map explanations, Grad-CAM and Perturbation-based saliency map for DQN and Rainbow agents for the three testing environments: Breakout, Space Invaders and Seaquest. 129
Appendix B.2 A comparison of the target DQN neural network layer on the Grad-CAM explanation for the three testing environments: Breakout, Space Invaders and Seaquest. 130
Appendix B.3 A hyperparameter search for the perturbation spacing of 2, 4, 5, and 8 across three testing observations for the Breakout, Space Invaders and Seaquest environments. 131

Appendix D.1 Example implementation of the TRD Q-value loss function . . 135

Appendix F.1 Consent Page of the comparative user evaluation 139
Appendix F.2 Submission of User Prior Knowledge in Artificial Intelligence and Ms Pacman 140
Appendix F.3 Strategy Identification Outline to Users and Comprehension testing 141
Appendix F.4 Dataset Similarity Explanation description and comprehension testing. 142
Appendix F.5 Strategy Identification question for the Dataset Similarity Explanation 142

Appendix F.6 SARFA description and comprehension testing 143
Appendix F.7 Strategy Identification question for the SARFA Explanation . 144
Appendix F.8 TRD Summarisation description and comprehension testing . 145
Appendix F.9 Strategy Identification question for the TRD Summarisation Explanation 146
Appendix F.10 Optimal Action Description description and comprehension testing 147
Appendix F.11 Strategy Identification question for the Optimal Action Description Explanation 148
Appendix F.12 Overall User Rating of an Explanation Mechanism 148
Appendix F.13 Submission of User Information 150

Appendix G.1 An ordered bar chart of Figure 6.6 where the orange bars are the correct answers (diagonal data from the confusion matrix) and the blue bars are the incorrect answers. 151
Appendix G.2 Cumulative density function for each explanation mechanism and the time taken to answer each strategy identification question, split by whether users get the answer correct or not. 152
Appendix G.3 The average user accuracy for each observation, irrespective of the explanation mechanism or strategy. 152
Appendix G.4 A Swarm plot of each user characteristic with a dot for each user and their accuracy. 154
Appendix G.5 Scatterplot of the number of correct answers and user self-reported beliefs for each explanation mechanism with the sizes being proportional to the number of instances of each permutation. 155

List of Tables

3.1 Table of XRL techniques comparing their explanation classes, testing environment, and if they used a human study to evaluate their explanations. 22

6.1 Table of example observations and future expected rewards with the outputs for three different prompts: zero-knowledge, environment knowledge, and environment+observation knowledge 79
6.2 Table of Explanation Mechanisms used in the Survey with their explanatory medium and content. 86
6.3 Table of State Selection Algorithms with their average importance and embedding distance of the selected observations. 87
6.4 Each explanation mechanism's accuracy across all agent goals for the 100 survey participants and an expert user. 88
6.5 Table of p-values for each explanation mechanism of the time taken for correct and incorrect answers. 92
6.6 A Table of User Characteristics with the number of users that selected the characteristics, the average accuracy, and the time taken (in seconds) for each question. 95
6.7 Table of Explanation Mechanism and user confidence with the user accuracy and their count in brackets. 97
6.8 Average user rating for each explanation mechanism on the three overall rating questions with the Likert Ratings discretised (1.0 for the lowest rating and 5.0 for the highest rating). 99

Appendix G.1 A table of characteristics coefficients for Ordinary Least Squared Linear Regression to predict each user's overall accuracy. 153
Appendix G.2 Table of the number of users who answer each overall rating Likert rating with the average accuracy of users. 156
Appendix G.3 Optional additional thoughts from users for each explanation mechanism. 156
Appendix G.4 Optional final thoughts from users at the end of the survey. . 159

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:
   Mark Towers, Yali Du, Christopher Freeman, and Timothy J. Norman. Temporal explanations of deep reinforcement learning agents. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, pages 99–115. Springer, 2024b. doi: 10.3233/FAIA240814
   Mark Towers, Yali Du, Christopher Freeman, and Timothy J. Norman. Explaining an agent's future beliefs through temporally decomposing future reward estimators. In *ECAI 2024*, pages 2790–2797. IOS Press, 2024a

Signed:.. Date:..................

Acknowledgements

I want to express my profound gratitude to my supervisors, Prof. Timothy J. Norman, Dr. Yali Du, and Prof. Christopher Freeman, whose invaluable guidance, expertise, and encouragement have been pivotal throughout my PhD. I am also deeply appreciative of Mohammed Marikar and the Royal Bank of Canada for sponsoring me and showing a keen interest in my work.

I am incredibly grateful to my family, who have helped inspire me and have been constant supporters, particularly Maggie, our beloved cat.

To my friends in Southampton, thank you for your fun, encouragement, and support. A special thanks to my PhD colleagues, in particular, Olaf, Tyler, and John, who've sat next to me and been willing sounding boards, and to my housemates, whose companionship brought joy and balance to my life. Last but not least, thank you to Highfield Church, my small groups, my YTH group, and YTH team, as well as to my mentors, Niv and David, for all your joy, prayers, and thoughtful counsel over the last seven years.

Therefore, there is now no condemnation for those who are in Christ Jesus

Romans 8:1

Definitions and Abbreviations

$\mathbb{E}[\dots]$	The expected value operator
$q_\pi(s,a)$	Q-value for policy, π, in state, s, and action, a, (Eq. (2.2))
$s_\pi(s)$	State-value for policy, π, in state, s (Eq. (2.3))
S	The set of all possible states within an MDP, of which $s \in S$ is a particular state
S_t, A_t, R_t	The state, action and reward for timestep t, respectively
$\|\|x\|\|_2$	Euclidean Distance of x
$P\|\|Q$	Kullback–Leibler (KL) divergence of P to Q equal to $\sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
XRL	Explainable Reinforcement Learning
MDP	Markov Decision Process (Puterman, 2014)
DQN	Deep Q-learning (Mnih et al., 2015)
Rainbow	Combination of DQN extensions and improvements (Hessel et al., 2018)
ALE	Arcade Learning Environment, a.k.a., Atari (Bellemare et al., 2013)
Goal	The objective that an agent is attempting to complete to maximise its rewards
Belief	An state, reward or goal that the agent has confidence (can be low or high) will o
DSE	Dataset Similarity Explanation (Section 4.1.1)
SE	Skill Explanation (Section 4.1.2)
PE	Plan Explanation (Section 4.1.3)
t-SNE	t-distributed Stochastic Neighbour Embedding (Van der Maaten and Hinton, 2008)
KMeans	Algorithm proposed in MacQueen (1967)
Smith-Waterman	Algorithm proposed in Smith and Waterman (1981)
Likert Rating	Rating methodology proposed in Likert (1932)
GradCAM	Gradient-weighted Class Activation Mapping (Selvaraju et al., 2017)
PBSM	Perturbation-based Saliency Map (Greydanus et al., 2018)
TRD	Temporal Reward Decomposition

QDagger	Algorithm proposed in Agarwal et al. (2022)
N-Step	Multi-step update (Sutton and Barto, 2018)
LoE	Linearity of Expectation (Stirzaker, 2003, Page 166)
LLM	Large Language Model
OAD	Optimal Action Description
SARFA	Specific and Relevant Feature Attribution (Gupta et al., 2020)

Chapter 1

Introduction

In 2016, AlphaGo (Silver et al., 2016), an advanced neural network-based Reinforcement Learning (RL) agent, played against Lee Sedol, a former world champion, on the board game Go, and won. This was the first time a computer had defeated a professional Go player, winning four out of five games. However, interestingly, in the fourth game, AlphaGo lost decisively. As shown in a behind-the-scenes documentary (Kohs, 2017) on move 79, its internal evaluation determined it had a 70% chance of winning, but by move 141, AlphaGo resigned, believing it had less than a 20% chance of victory. Despite AlphaGo's strong performance, decisively winning the first three games and believing it had an advantageous position, AlphaGo was still flawed and lost.

Although the loss was relatively trivial, with little at stake other than the player's pride and prize money, the existence of such flaws in agents is a critical problem that researchers, industry, and regulators must address if we do not want AI to go wrong in the real world with possibly more dire consequences. With RL having been proven in video games and simulations, i.e., virtual world (Mnih et al., 2015; Schulman et al., 2017), academia and industry have begun working on applying agents to the real world (Degrave et al., 2022; Ahn et al., 2024). Importantly, this shift in the deployment domain will result in agents possibly needing to interact with humans, operate in high-risk settings, etc (Dulac-Arnold et al., 2021) has significantly raised the importance and interest of research minimising or even eliminating delusions, like in AlphaGo, from agent behaviour.

This recognition that delusions, also called hallucinations, can occur in critical settings for a variety of complex problems has promoted a surge in research around safe and explainable artificial intelligence (AI). Deep neural networks have been the main subject of research, being the state-of-the-art solution for a large number of problems: image recognition (He et al., 2016), text generation (Brown et al., 2020), and playing games (Mnih et al., 2015). Both these subfields (safe and explainable AI) have similar goals of building trust with humans through preventing catastrophic failures and providing the reasons for a particular output, respectively.

In this thesis, we explore explainable artificial intelligence (XAI) in temporal games, as although safe AI aims to prevent dangerous decisions, providing initial trust for a system, this may not be enough for humans to *fully* trust an agent if they cannot understand how particular decisions are reached even if they know an action will be safe. This is specifically critical for neural networks, where their behaviour cannot be verified for all inputs due to their parametric nature.

For supervised learning, e.g., image recognition and text generation, a large body of research exists investigating how and why machine learning models generate particular outputs (Belle and Papantonis, 2021). However, a significantly more limited body of research has extended explanations to Reinforcement Learning (Sutton and Barto, 2018), a subfield of machine learning in which an agent takes sequential actions over time, learning to maximise its reward over time. This subfield of Explainable Reinforcement Learning (XRL) is important as (Agogino et al., 2019) found that while applying a range of supervised learning XAI algorithms to RL agents could provide "simple insights", they struggled to find "good intuitive explanations". This highlights the need for specialised XAI algorithms for RL agents.

Within the existing XRL literature, algorithms may be categorised into several groups (Qing et al., 2022); for example, but not limited to feature-based, policy-based, and contrastive explanations. Each attempt to explain different elements of an agent, e.g., feature-based explanations focus on the importance of each object in an observation; policy-based explanations consider the agent's decision-making as symbolic logic; and those that are contrastive explain why one action was taken over another. A more comprehensive survey of XRL research is provided in Section 3.2.

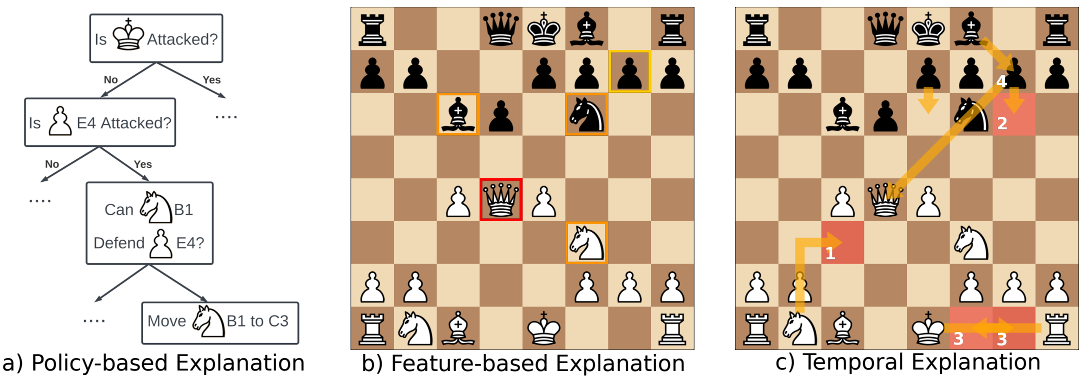


FIGURE 1.1: a) Policy-based Explanation: An interpretable decision tree such that the decision-making paths can be explained. b) Feature-based Explanation: Piece colour represents their perceived importance to the agent's decision making Gupta et al. (2020). c) Temporal Explanation: Explains the agent's future decision-making with the numbered sequence of expected future actions. In practice, a video of these intended moves would be shown to the user.

In this thesis, within XRL, we investigate temporal explanations, a distinct type of explanation that has received far less attention to date. Temporal explanations focus on the *future* intentions/purposes of the agent policy, i.e., the future goals, rewards, or actions that the agent wishes to achieve or see. Figure 1.1 illustrates an example temporal explanation, along with policy and feature-based explanations for a given chess position. The reason we focus on explaining an agent's future considerations is twofold.

First, the nature of sequential decision-making in RL means that agents take a sequence of actions to achieve a goal or objective with a reward for each decision, which implies that agents think temporally. This differs significantly from supervised learning, where models learn a correct answer in a single decision.[1] RL agents learn policies that maximise their total reward over a whole episode. So, when selecting an action, despite this being for an isolated timestep, an agent's decision-making must implicitly consider its future impact, most obviously, on future rewards. Therefore, for explanation mechanisms to fully explain an agent's decision-making, they should explain these future considerations, i.e., potential (sub)goals, future rewards, or actions. As a result, we argue that temporal explanation offers a more complete picture of an agent's decision-making, uncovering not just the what and the why of the agent's current decision (for which feature and policy-based explanations focus) but also the anticipated implications of an action for future observations, actions, rewards, or goals.

Secondly, psychological surveys of human explanatory mechanisms have found that temporal explanations are one of the most common types, along with feature and action-based explanations (Miller, 2019; Madumal et al., 2020a). Therefore, as explanations are ultimately for human consumption, we should mirror how people explain decisions for RL agents. This ensures that generated explanations are scientifically accurate and intuitively relatable to the human user.

Taken together, the implicit decision-making of RL agents and common explanation structures utilised by humans provide strong justification for investigating temporal explanations for RL. Despite this, surveying the XRL literature, we find that temporal explanations are a significantly underdeveloped type of mechanism (Section 3.2). Therefore, for the rest of this thesis, we developed several novel temporal explanation mechanisms that explain an agent's future considerations (Sections 4.1, 5.3, and 6.1) and evaluate if users find these explanations effective (Sections 4.3 and 6.3).

[1] Text generation is a possible exception where models sequentially generate words (tokens) to answer an input in Pretrain. Though the more recent paradigm of RLVR (reinforcement learning through verifiable rewards) for post-training encourages more temporally aware thinking (DeepSeek-AI et al., 2025). See Chapter 7 for a discussion on applying temporal explanations to large language models.

1.1 Thesis Contributions and Structure

Across the next six chapters of this thesis, we outline the background knowledge of RL and XAI, a literature survey of XRL approaches, followed by three research chapters developing novel temporal explanation mechanisms and evaluation methodologies, ending with future work that can build upon the work presented. We summarise their content and contributions below. Two chapters have been peer-reviewed and published.

Chapter 2 sets out background information for the rest of the thesis, including the mathematical notation for describing RL, training algorithms for RL agents with neural networks, and descriptions of explainability attributes used in the literature review and the explanation mechanisms we develop.

In Chapter 3, we first outline the unique set of challenges in XRL that XAI algorithms for supervised learning problems do not address (Section 3.1). Using the properties of the explanation mechanism in Section 2.3, we review the literature on XRL, summarising the surveyed algorithms in Table 3.1 with their respective mechanism properties, testing environment, and whether a user survey was conducted. In addition, we provide a detailed overview for a subset of these mechanisms, in particular, saliency map approaches, natural language explanations, temporal explanations, and reward explanations in Sections 3.2.1 to 3.2.4. Following this, Section 3.3 reviews the evaluation approaches of the different user surveys listed in Table 6.2. Finally, we discuss the recurring strengths and weaknesses of the surveyed XRL algorithms, particularly temporal explanations (Section 3.4).

Surveying prior temporal explanations found that most required extensive domain knowledge. Therefore, in Chapter 4, we research temporal explanations with no or limited domain knowledge of the environment[2] through video-based explanations (one of the first to use as the primary explanation medium). We propose three novel explanation mechanisms; the first two use a memory of prior agent behaviour to extract similar contexts or skills to explain an agent's possible future decision paths (Sections 4.1.1 and 4.1.2 referred to as Dataset Similarity Explanation and Skill Explanation). The third novel explanation utilises minimal domain knowledge of an agent's skills to annotate the Skill Explanation's video (Section 4.1.3 referred to as the Plan Explanation). In Section 4.3.1, we conduct and analyse a human survey with our three novel video-based temporal explanations along with two feature-based explanation mechanisms (Selvaraju et al., 2017; Greydanus et al., 2018), asking users about their understanding, details, satisfaction with the explanations, and predictability of the agent's next action. We found that our explanations were rated on average 1 point

[2]We don't assume any knowledge of an environment, including environment dynamics, reward function structures, etc, commonly used in utilised-based MDP-style environment. Rather, we assume the environment is a POMDP (Partially Observable Markov Decision Process) with a human-comprehensible rendering of its current state.

higher (rated from 1 to 5) than the feature-based explanations when individually rated. When compared directly to each other, our explanations were preferred 80.7% of the time compared to the feature-based explanations. Furthermore, we propose two novel metrics for evaluating agent skills in Section 4.2 and evaluate them in Section 4.3.2. In summary, we show that our simplistic video-based temporal explanations are effective compared to feature-based algorithms, even without specialist domain knowledge. Chapter 4 was accepted for presentation at the AAMAS EXTRAAMAS workshop 6th edition and published as part of Springer Lecture Notes of Artificial Intelligence conference proceedings under the title "Temporal Explanations of Deep Reinforcement Learning Agents" (Towers et al., 2024b).

In Chapter 5, motivated by the need to extract implicit temporal beliefs from pre-trained models, which can be challenging to validate. We explore an alternative direction within temporal explanations to investigate how an agent's training can be modified to learn explicit temporal features that can be explained. We novelly identify that the Q-value, a sum of the expected future rewards, implicitly encodes the agent's future expectations; however, as a scalar value, it isn't interpretable. Therefore, in Section 5.1, for the first time, we show that the Q-value can be decomposed as a vector of the next N expected reward (Eq. (5.6)), referred to as Temporal Reward Decomposition (TRD). We prove Eq. (5.6) equivalence to the standard scalar Q-value (Theorem 5.1 and Eq. (5.7)) and propose a necessary novel loss function (Eq. (5.14)) to learn it. To test the scalability of TRD, in Section 5.2, we demonstrate that pre-trained Atari agents can be efficiently retrained to incorporate TRD, matching a standard DQN agent's performance in two million steps for a variety of TRD hyperparameters. To explain an agent's next N expected rewards, we propose three methods: visualising the future expected rewards, understanding the temporal importance of observation features, and a contrastive visualisation of the impact of actions on future rewards (Sections 5.3.1, 5.3.2, and 5.3.3). Each of these explainability methods highlights novel information about the agent's decision-making process, inaccessible through prior approaches. Chapter 5 was accepted for presentation and publication at ECAI (European Conference on Artificial Intelligence) 2024, under the same chapter title, (Towers et al., 2024a). In addition, we gave an invited talk at the TSDO (Trustworthy Sequential Decision-Making and Optimization) workshop and completed an outreach activity for the general public and academics at the conference.

With our emphasis on the importance of user evaluations throughout this thesis, in Chapter 6, we propose a novel evaluation methodology based on the user's debugging agent strategies. Before the evaluation, to improve TRD's perceived interpretability for end-users, we propose a natural language summarisation technique using large language models to help explain TRD's visualisation graphs, reducing the need for technical knowledge to interpret Chapter 5's explanations (Section 6.1). We explore various prompts, from just technical information on what the future expected rewards represent to multi-modal prompts containing the agent's observation and a reward

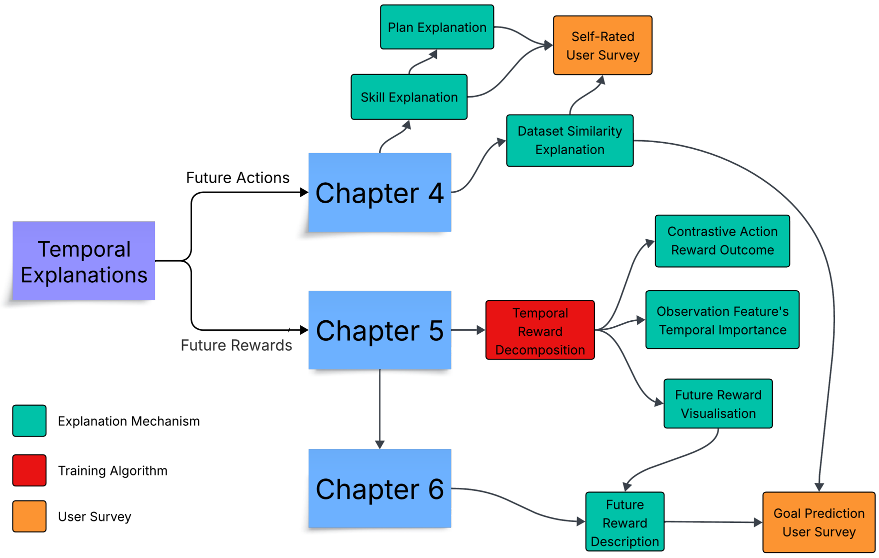


FIGURE 1.2: Conceptual Diagram of the Chapters, Explanation Mechanisms, User Surveys completed in this Chapter.

function summary, finding that each could accurately summarise an agent's next rewards. For the evaluation, we propose a novel methodology where users identify an agent's goal given an observation, explanation, and several options. This provides an objective measure of an explanation mechanism's actionability on a task closely related to debugging an agent's policy. We additionally ask users for their confidence in each selection, the explanation mechanisms' overall ease of use, understanding, and confidence using a five-point Likert rating (Likert, 1932). This provides a mixture of objective and subjective measures of each explanation mechanism that can be compared and analysed. For this survey, we compare four algorithms: Dataset Similarity Explanation (Section 4.1.1), TRD Summarisation (Section 6.1), Specific and Relevant Feature Attribution (Gupta et al., 2020), and a description of the optimal action. We find our temporal explanations had the highest average accuracy of 53.0% and 34.9% for Dataset Similarity Explanation and TRD Summarisation, respectively. At the same time, the two baselines achieved accuracy very close to random (25%), 28.7% and 22.5% for Optimal Action Description and SARFA, respectively. We additionally investigate correlations between the explanation's accuracy and the user's confidence, time taken to answer, and characteristics, finding weak or no correlation between most of them.

Finally, in Chapter 7, we summarise this thesis, reflecting on the research conducted. For future work, we highlight three novel directions for Temporal Reward Decomposition, presented in Chapter 5 and expanded on in Chapter 6, addressing limitations and future algorithmic applications. Finally, we identify further research required in XRL to capitalise on the work presented in this thesis.

Chapter 2

Background

This chapter outlines background information utilised throughout the thesis with Section 2.1 discussing reinforcement learning's (RL) mathematical notation, Section 2.2 introducing training RL agents using neural networks, and Section 2.3 summarising terminology for describing explanation mechanisms.

2.1 Reinforcement Learning Mathematics

Sequential decision-making problems can often be described using Figure 2.1 (Sutton and Barto, 2018); given an environment's state S_t at timestep t, an agent selects an action, A_t, with the environment providing the resultant reward, R_{t+1} and subsequent state S_{t+1} at timestep $t+1$. This process loops either forever or until a termination state is reached. Importantly, the agent's goal is to select actions that maximise its cumulative reward over time, $\sum_{t=0} R_t$. In this section, we outline the mathematical notation used to describe environments and how optimal agents can be learnt through trial and error, interacting with the environment.

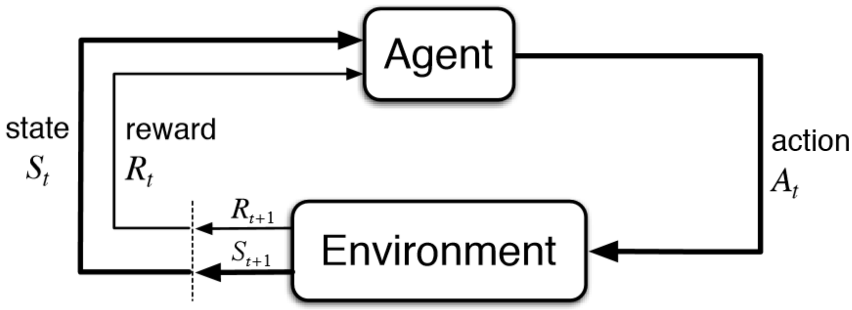


FIGURE 2.1: Reinforcement Learning environment loop with an agent taking action for a given observation and receiving a reward and updated observation (along with termination status).

For reinforcement learning environments, Markov Decision Process (MDP, Puterman (2014)) provides a mathematical description using two variables, S and A, and three functions, R, P, and T. The variable S denotes the set of all possible environment states on which the agent makes decisions, and A denotes the set of all possible actions the agent could take in the environment. The function R is the environment's reward function, $R(s,a) \rightarrow \mathbb{R}$, such that for a given state-action pair (s,a), the function returns the resultant scalar reward. When taking an action, the next state is not always deterministic; $P(s'|s,a)$ is the transition function computing the probability of the next state, s', given the prior state and action (s,a). Finally, environments are often designed for agents to reach a given goal, where the environment loop will end, referred to as a goal state. Not shown in Figure 2.1, $T(s)$ prescribes the termination condition on which the agent has reached a goal (or failure) state, and the environment loop ends.

For simplicity, following Sutton and Barto (2018), we denote S_t, A_t and R_t as the state, action and reward received for timestep t. Additionally, to practically implement the environment, train and evaluate agents on MDP environments, we utilise Gymnasium (Towers et al., 2024c), a maintained version of the OpenAI Gym library (Brockman et al., 2016) which implements a number of popular benchmark environments such as CartPole and supports the Arcade Learning Environments (Bellemare et al., 2013), more commonly known as the Atari environments.

For an agent, we wish them to learn a policy, $\pi(S) \rightarrow A$, mapping observations to actions that maximise its cumulative rewards over an episode. Furthermore, to incentivise the policy to collect rewards sooner, we apply an exponential discount factor ($\gamma \in [0,1)$). For a policy π, this expected sum of future rewards may be defined in terms of the q-values, $q_\pi(s,a)$ or the state-value, $v_\pi(s)$ that predicts for a given state-action (s,a) or given state s only (Eqs. (2.2) and (2.3) respectively). These functions model the expected (mean) sum of discounted future reward as the environment and/or policy might be stochastic.

$$q_\pi(s,a) = \mathbb{E}_\pi\Big[R_t + \gamma R_{t+1} + \cdots + \gamma^n R_{t+n} | S_t = s, A_t = a\Big] \tag{2.1}$$

$$= \mathbb{E}_\pi\Big[\sum_{n=0}^{\infty} \gamma^n R_{t+n} | S_t = s, A_t = a\Big] \tag{2.2}$$

$$v_\pi(s) = \mathbb{E}_\pi\Big[\sum_{n=0}^{\infty} \gamma^n R_{t+n} | S_t = s\Big] \tag{2.3}$$

Importantly, the q-value and state-value can be reformulated as recursive functions using the discounted next state (Eqs. (2.4) and (2.5)). For the Q-value, we assume that the optimal action is taken, which is the maximum Q-value for the set of possible actions in the next state.

$$q_\pi(s,a) = \mathbb{E}_\pi[R_t + \gamma \max_{a' \in A} q_\pi(S_{t+1}, a') | S_t = s, A_t = a] \tag{2.4}$$

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma v_\pi(S_{t+1}) | S_t = s] \tag{2.5}$$

Using the recursive nature of the functions, a bootstrapping-based approach is commonly utilised to learn the optimal policy minimising the error between predicted future rewards, $q_\pi(s,a)$ for a given state-action (s,a) and the actual future reward, r, plus the maximum expected future rewards of the next state, $max_{a' \in A} q_\pi(s', a')$ (Eq. (2.6)). This recursive function is called the Bellman Equation, where α is a learning rate to control how much q_π is adjusted for each update (Bellman, 1954).

$$q_\pi(s,a) = q_\pi(s,a) + \alpha(r + \gamma \max_{a' \in A} q_\pi(s', a') - q_\pi(s,a)) \tag{2.6}$$

It was later proved, using the MDP notation and under certain initial conditions, that Eq. (2.6) will converge to the optimal policy, π^*, given an infinite number of steps (Watkins and Dayan, 1992). To learn q_π required a large table of state-action pairs, with the table being iteratively updated until it converged. However, for practical applications, this posed significant limitations for the observation and action spaces, such that observation and actions are discretised, and the environment is tractable (all state-actions are known). As a result, neural networks have become the de facto method to encode q_π in modern RL.

2.2 Deep Reinforcement Learning

With the limitations of tabular RL approaches outlined above, researchers investigated numerous alternative methods, with neural networks becoming the dominant approach. Mnih et al. (2015) was the first general RL algorithm, called Deep Q Learning (DQN), that achieved superhuman results without changing any hyperparameters in the Arcade Learning Environment (Bellemare et al., 2013), 60+ ROMs for the Atari 2600 console colloquially referred to as Atari environments. DQN forms the basis of all agents explained in this thesis.

This was achieved with several important changes to the training setup compared to Q-learning. The agent policy, previously a table of every state-action Q-value, was replaced with a single neural network, θ, where the input is the state and the output is the Q-values for all actions. This allowed efficient look-up for all Q-values of a state requiring a single forward pass of a neural network and was scalable to high-dimensional and continuous inputs, e.g., images and robotic actuators. Another advantage of DQN over Q-learning was the ability for the environment to be intractably large (without the need to visit every possible state), as the neural network can extrapolate to unseen states. Utilised an experienced replay buffer, D, originally proposed by Lin (1992), a circular queue of a fixed length, the observations, actions, next observations, rewards, and terminations from an environment step are stored, saving the last n instances. For Mnih et al. (2015), this equated to 1 million samples. For each training step, a batch of b samples is uniformly selected from this buffer to learn from. In the case of Mnih et al. (2015), $b = 32$.

$$L_Q(D) = \mathbb{E}_{(s,a,R,s')\sim D}\left(q_\theta(s,a) - (R + \gamma \max_{a' \in A} q_{\theta'}(s',a'))\right)^2 \tag{2.7}$$

Finally, Mnih et al. (2015) proposed a novel loss function, Eq. (2.7) with a minor change to the Bellman equation for stability, where θ' represents a target network. As the bootstrap error can cause runaway values in the network, caused by a divergence in the values of s_t and s_{t+1}, Mnih et al. (2015) proposed an online network, θ, that is actively updated and a target update that is periodically updated. In the case of Mnih et al. (2015), every 1000 training updates the target network weights are replaced by the active network's weights. This prevents the online network from diverging too quickly, causing policy instability and possible collapse (van Hasselt et al., 2018).

To help minimise the training time of deep RL agents, Agarwal et al. (2022) proposed a workflow utilising learnt policies to reuse or transfer their knowledge to new agents called QDagger. We utilise this algorithm in Chapters 5 to show that our algorithm, TRD, could be efficiently incorporated into previously trained agents. QDagger proposes two changes to the training schemes of any neural network-based algorithm. First, adding a distillation loss to the agent's policy loss that minimises the KL divergence between a student's (the new agent) and a teacher's (a pretrained agent) policy (Eq. (2.8) where $L_Q(D)$ represents the standard loss function of the agent):

$$L_{\text{QDagger}}(D) = L_Q(D) + \lambda_t \mathbb{E}_{s\sim D}\left[\sum_a \pi_T(a|s) \log \pi(a|s)\right] \tag{2.8}$$

This encourages the new student to mimic the teacher's expected future rewards, regulated by λ_t, equal to the ratio of the teacher's performance to the student's performance. This means that if the student improves upon the teacher, the student's policy is not constrained by the teacher. Secondly, QDagger proposes an offline training stage using a teacher's (pretrained agent) replay buffer. Before allowing the agent to explore the environment for itself, Agarwal et al. (2022) rolls out the teacher's model, generating high-quality actions from across the observation space. Using the teacher's replay buffer, like a supervised learning dataset, enables the agent to approximate the teacher's policy, providing a head start in training. Agarwal et al. (2022) showed that

QDagger allows student agents to match the teacher's performance for Atari environments with 20x fewer observations than training from scratch.

2.3 Explanation Mechanism Properties

Within the literature, researchers must select several important properties when developing an explanation mechanism. These properties are explanation audience, model interpretability, explanation type, explanation medium, and mechanism category (Sections 2.3.1 to 2.3.5 respectively). Figure 2.3 links these explanation properties together with example algorithms. Before discussing these properties, we examine the definition of explainability that will influence our approach for the rest of this Section.

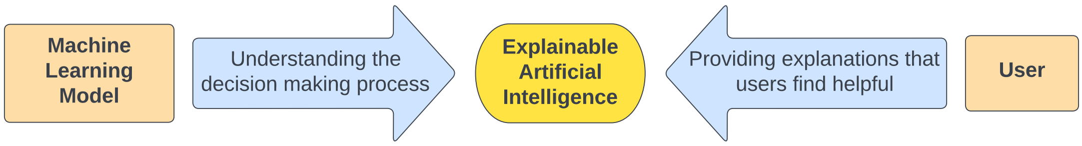


FIGURE 2.2: Visual representation of the definition of Explainable Artificial Intelligence

Within XAI, there is no agreed-upon definition of explainability or interpretability (Belle and Papantonis, 2021; Puiutta and Veith, 2020), meaning that researchers can have different aims and criteria for an effective XAI algorithm. In this work, we follow Puiutta and Veith (2020) who defines explainability as "the ability to not only extract or generate explanations for the decisions of the model but also to present this information in a way that is understandable by human (non-expert) users to, ultimately, enable them to predict a model's behaviour". This definition brings together two problems: first, a model's decision can have an explanation generated from it, and second, the explanation must be understandable to humans, allowing users to predict future behaviour. We believe that these two parts of the definitions are critical for explainable and interpretable AI algorithms.

A consequence of this definition is that no "optimal" or "ideal" explanation will exist, as for a particular agent, observation, environment, or user could cause a different explanation to be subtly preferred or viewed as optimal. Furthermore, the way explanations are evaluated could cause different explanations to be ranked as the most effective. Despite the challenges that no optimal explanations exist for all situations, we don't believe this differs significantly from other areas of machine learning research. For different datasets or environments in RL, training algorithms will vary in optimality, and their effectiveness depends on the testing criteria. Therefore, like the rest of machine learning, testing XAI across a range of users, settings, and models is an important component to ensure that researchers can understand the general effectiveness of an explanation.

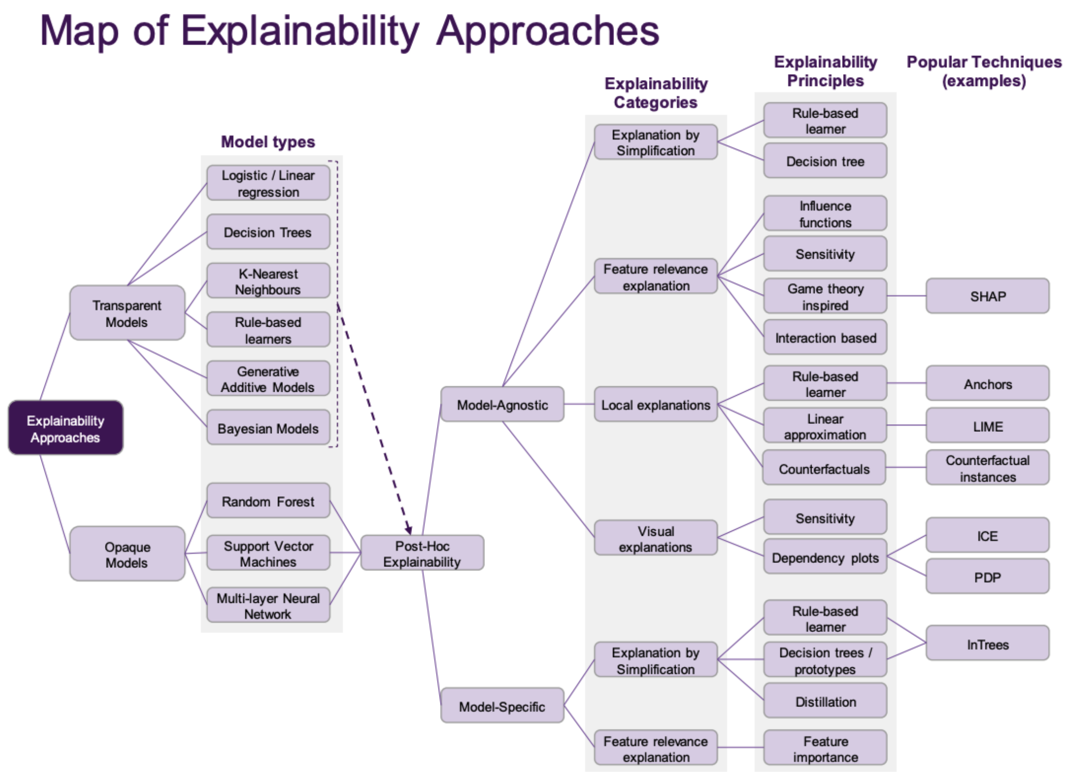


FIGURE 2.3: A graph of explanation properties with example XAI algorithms. Source: Belle and Papantonis (2021)

Although this definition of explainability is logical and obvious, we show in Section 3.2 that a large number of papers in XRL have ignored the second part of this definition, not evaluating the effectiveness of their explanations with user surveys. This is important as Miller et al. (2017) has argued that if XAI researchers ignore user surveys, then it is like "inmates running the asylum". That is, if explanations are designed by and *for* researchers, who often are not the target audience, the proposed algorithms may not be effective for the target audience, i.e., end-users and regulators. It should be noted that some XAI algorithms are designed for researchers to use in debugging models. However, even in this case, we are unaware of user surveys utilising researchers to show the effectiveness of a proposed explanation method.

2.3.1 Explanation Audience

When developing explanation mechanisms, knowing the audience/stakeholders, i.e., the users utilising the explanations, is critical to ensure an algorithm is effective. Belle and Papantonis (2021) identifies five types of people interested in explaining a machine learning model's decision-making, each with their own goals and desires for an explanation. This means that we expect explanations to differ depending on the audience. However, in our review of the literature (Section 3.2), it's common for work not explicitly to state for whom an explanation is designed. Figure 2.4 summarises stakeholders' motivations and primary considerations.

- **Data Scientists / Academic Researchers**: These stakeholders are interested in utilising, developing, and training different machine learning algorithms to achieve a task. Therefore, these users are more interested in understanding the model to debug it, in particular, how the model makes decisions when the model goes wrong. The second interest of these users is improving performance; the ability to compare and understand the differences between models is important to improving an algorithm, for which explanations might help.
- **Business Owner**: These stakeholders are primarily interested in a high-level understanding of the model (i.e. global explanations) and how the model will improve a business. This requires less technical and specialised explanations than a data scientist or academic researcher.
- **Model Risk**: An Actuary is an important job within businesses who predict the risk of certain events happening to ensure that stocks, insurance, etc., are correctly risk assessed. With the increasing growth of machine learning models, we predict that a similar role to actuaries will be required for machine learning models. Safe Artificial Intelligence is already an active field, and we expect that XAI will be an additional tool used to assess the risks of a model.
- **Regulator**: Being outside of businesses, regulators ensure that businesses follow a set of laws or guidelines. Already in GDPR (Council of European Union, 2016), there are guidelines on explanations for machine learning models, which we suspect will become a more significant policy issue for governments worldwide. We believe that the explainability or interpretability of machine learning models will be a critical element in these decisions. Therefore, regulators are interested in the truthfulness of explanation algorithms, and the capabilities explanations have for preventing specific undesirable behaviour.
- **Consumer**: As noted in Chapter 1, machine learning models are becoming increasingly important in our daily lives. Therefore, it is important that users who will soon be consumers and interact with these machine-learning models trust them. We believe that explanations are a critical way for consumers to learn to trust machine learning models by providing users with additional information on the model's decision-making process. Therefore, explanations must be simple and easy to understand; however, they must still accurately approximate the model's decision-making.

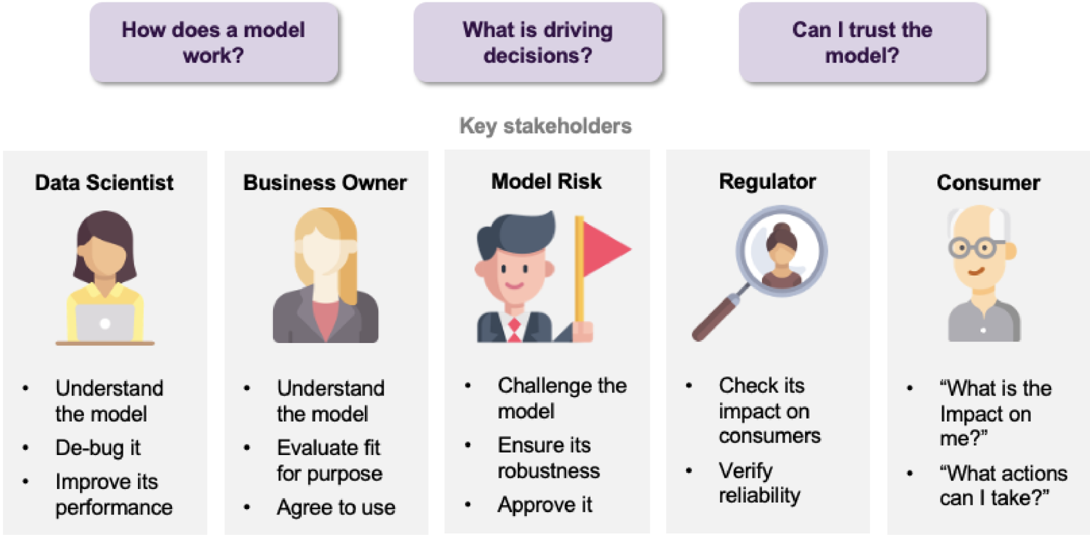


FIGURE 2.4: Outline of the motivations of different types of audiences in XAI. Source: Belle and Papantonis (2021)

2.3.2 Model Interpretability

In the way an explanation is generated, researchers have primarily explored two separate methods of providing explanations: intrinsically explainable algorithms and post-hoc explainable algorithms (Belle and Papantonis, 2021). These explanation techniques differ according to the interpretability of the underlying machine-learning algorithm.

- **Intrinsically explainable model** - For machine learning algorithms such as logistic and linear regression, decision trees, rule-based learners, and Bayesian models are referred to as intrinsically explainable as the models are often small enough for a human to directly understand how all parts of the model work (Lipton, 2018). Depending on the complexity of the algorithm, it is even possible to simulate the decision process that the model takes, the highest level of interpretability; however, this is only possible in a limited number of cases (Belle and Papantonis, 2021). Intrinsic algorithms can be more commonly decomposed so that the model can be divided into input, parameters, and computations with true and complete explanations generated for each part. However, even for these simple machine learning models, if they grow too large and contain too many parameters or computational steps, it may not be possible to provide effective or complete explanations for any of the inputs, parameters, or outputs to users. It should be noted that there is disagreement on the possible interpretability of even these small models, with Lipton (2018) arguing that such interpretability is a "myth" due to the need for features to be hard-crafted to achieve high performance. Although Rudin (2019) disagrees, stating that most problems can be solved with these models for effective explanations.

- **Post-hoc explainable model** - For machine learning models that use thousands, millions, and now billions of parameters for the decision-making process, no human could understand the model's decision-making. Unlike intrinsically explainable algorithms, these models, neural networks, ensemble models, and support vector machines are not directly explainable. Therefore, these models require post-hoc explanations where only an approximation of the decision-making can be explained (Lipton, 2018). The advantage of these models is due to their complexity; they can achieve higher performance on complex problems such as image recognition (He et al., 2016), text generation (Brown et al., 2020) and playing games (Mnih et al., 2015). However, this additional complexity has prompted concerns that these state-of-the-art algorithms can contain implicit biases, called algorithmic biases (Hajian et al., 2016), which are difficult to address once found due to the number of parameters involved in computing a model's output.

2.3.3 Explanation Type

Given the target audience and the type of model to explain, it is critical to select what information the explanation mechanism explains. These are split into Local and Global explanations that differ in explaining an agent's decision-making for a single example and the whole agent's decision-making for all examples, respectively.

- **Local explanations** explain a single input to the model. This can be done with both intrinsically explainable models and post-hoc explainable models, most commonly through explaining the decision-making process used by the intrinsically explainable model and for post-hoc explainable models through feature relevance. Local explanations are important for understanding individual model output, particularly incorrect model output, to see where the model went wrong in its decision-making process. We provide examples of local explanations in Section 3.2.1, where we examine saliency map explanations.

- **Global explanations** explain how the model works for all possible inputs. For intrinsically explainable models, such as rule-based learning or decision trees, this is possible by presenting all the rules the model uses to the user. This gives the user an explanation of the decisions made, given an input that can be investigated and evaluated. For post-hoc explainable models, as such complete model explanations are not possible, global explanations are often achieved through explanation through simplification. This will reduce a black-box model to an intrinsically explainable model that can be fully explained. However, this reduced model can have extremely different decision-making than the original black-box model.

2.3.4 Mechanism Categories

Explanation mechanisms can generally be grouped based on the type of explanation generates, we refer to these as the mechanism category. We note that this is not an exhaustive list of categories and that some algorithms utilise multiple categories.

- **Direct decision-making** mechanisms use intrinsically explainable models where complete decision-making is interpretable. These mechanisms use that information to present the way that the model reached its output for a global or local explanation.
- **Explanation by simplification** mechanisms are similar to the direct decision-making mechanism for post hoc explainable models where the true decision-making is too complex. These models attempt to simplify the model to an intrinsically explainable model that approximates the true black-box model. This approach may also be taken for models that are normally intrinsically explainable and too large for humans to comprehend.
- **Feature relevance explanation** mechanisms, also known as saliency maps, are the most common type of explanations that rank features for a model based on its importance to the decision-making. These mechanisms can use the input data or the model features to highlight the most relevant features when making a decision. Primarily, this is used with local explanations; however, global explanations are possible to highlight features that influence the model most across a dataset.
- **Counterfactual** mechanisms provide a local counterexample to explain why the decision was made. This differs from the other mechanisms that present a factual explanation of an input. These mechanisms can provide counterfactual inputs, decision-making, features, etc, as to what would happen in a similar situation.
- **Temporal explanations** provide an explanation for the future considerations of the agent's decision-making in terms of possible states, actions, and rewards. As explained in Chapter 1, as agents maximise their reward, they consider the future impact of an action.

2.3.5 Explanation Medium

As explained in Section 2.3, explainability depends on a user's understanding of the decision-making process of a model. Therefore, given the categories of the explanation mechanism, how this information is shown to the user is important. We group the possible options into distinct mediums:

- **Text explanations** explain concepts using natural language, similar to how humans talk or write explanations. However, this requires prior domain knowledge so that the explanation provides helpful and vital information for the user.
- **Visual explanations** are the most popular medium for explanations that use images or videos to convey highly complex information, often without the need for prior domain knowledge of a system, unlike text explanations.
- **Mathematical / Symbolic explanations** are an uncommon explanation medium; however, for particular audiences, e.g., data scientists and academic researchers, mathematical explanations can provide an essential technical explanation that textual and visual explanations cannot.

Having introduced reinforcement learning and summarised the key components and structures used in explanation mechanisms, the next chapter focuses on the research landscape of Explainable Reinforcement Learning.

Chapter 3

Literature Review

Agogino et al. (2019) identified that explainable artificial intelligence (XAI) provides "simple insights" for reinforcement learning agents, but it struggles to produce "good intuitive explanations".[1] Therefore, in this chapter, we first examine the challenges that XAI algorithms do not address for reinforcement learning agents, identifying five sources of difference (Section 3.1). Next, using the properties of the explanation mechanisms from Section 2.3, in Section 3.2, we survey the literature of Explainable Reinforcement Learning (XRL) algorithms, summarising a wide range in Table 6.2, listing their explanation properties, testing environment(s), and if a user survey was completed. Of these surveyed XRL algorithms, we provide a more detailed discussion of four mechanism types: saliency maps, natural language explanations, temporal explanations, and reward explanations in Sections 3.2.1 to 3.2.4, respectively. Next, in Section 3.3, for the literature that conducted user evaluations, we categorise the evaluation measurements used and compare them. Finally, we discuss recurring strengths and weaknesses from the literature review, particularly of temporal explanations (Section 3.4).

3.1 Challenges in Explainable Reinforcement Learning

Reinforcement Learning (RL) focuses on learning a sequence of actions that maximise an agent's total rewards over time. In contrast, supervised learning, an alternative field within machine learning, trains a model to learn labelled data where the correct input-output pair is known. RL, therefore, differs in two forms from supervised learning: time as an agent takes a sequence of actions and uses trial and error to learn rather than with a set of known right and wrong input/outputs. This distinction

[1]We reserve the term XAI to solely refer to explanation mechanisms designed primarily or initially for supervised learning models to help differentiate those algorithms from XRL which for explanation mechanisms are primarily or initially designed for reinforcement learning agents.

between RL and supervised learning propagates into a model's decision-making and, more importantly for this thesis, the type of explanations that could be generated. We outline several unique features of RL, not contained within supervised learning, that necessitate novel XRL algorithms:

- **Sequential Decision-Making**: For humans, explaining their decision-making in Chess compared to identifying if a cat is in a photo can produce vastly different explanations. One of these ways is rationalising the future impact of an action, a unique axis to their thinking compared to image classification (and all supervised learning), where no such future decision-making is necessary. As a result, applying XAI algorithms to RL is feasible but will implicitly miss out on the sequential/temporal nature of an agent's decision-making. Therefore, some XRL algorithms are designed to explain an agent's future/sequential decision-making, such as future rewards, actions, or goals, which we generally classify as temporal explanations.
- **Temporal Scale**: The reason for taking an action may have multiple valid explanations depending on the future scope examined. For example, in Chess, an explanation for a move could be: "This move captures an enemy rook", or "I remove the rook that was defending the enemy king to attack it in the future" or "This helps to set up a checkmating attack on the enemy king to win". All three explanations could be valid for the same Chess position; however, they are valid considering different time scales (i.e., how far in the future do we consider the effects of an action?). Thus, different XRL algorithms can range from focusing on an agent's decision-making in the near term, e.g., the impact of the next action, to the long-term impact, e.g., how an action helps achieve the agent's overall goal. Though linked to an agent's sequential decision-making, the challenge of how far to explain an action in the future is a unique challenge for algorithms to solve.
- **Multi-agent**: In situations with other agents, humans commonly explain how their actions might be prevented by competing agents or how to collaborate with another agent to achieve a common goal. For example, a chess player might explain that they "retreat my queen to prevent the opponent's rook from capturing it". Interacting with other agents is a unique feature of agent systems, requiring at least an implicit theory of mind for the other agents to understand their decision-making and how it might affect an agent's behaviour. Though not exclusive to RL anymore, with agentic systems being built with Large Language Models (Sypherd and Belle, 2024), multi-agent systems require novel explanation mechanisms from those traditionally built for XAI. Additionally, depending on whether agents are acting in cooperative, competitive or mixed cooperative-competitive environments would require the design of novel explanation algorithms.

- **Partially-observable environments**: For some environments, agents have incomplete or imperfect information about the environment; e.g. in Poker (Brown and Sandholm, 2019) agents are unaware of the opponent's cards, and in Starcraft 2 (Vinyals et al., 2019), the agent can only observe a small region of the environment at a time. These environments are formally referred to as Partially Observable Markov Decision Processes (POMDPs), which contain a large class of environments, including many real-world problems. Unlike supervised learning, where the model might similarly have to rationalise from incomplete data, in sequential systems, this incomplete information can change over time, appearing and disappearing, or can be partially inferred from other agent's behaviour. Thus, specialist XRL algorithms should be designed to explicitly account for agent behaviour where there is incomplete data from their observations.

- **Evaluation**: RL's temporal (sequential) nature poses difficulties in evaluating the effectiveness of explanation mechanisms. This is compared to supervised learning, where a given input will have a known (singular) true output that is expected. From this, using an explanation, an XAI algorithm's effectiveness can be measured through users' understanding or capability to reach the same output. In contrast, RL has no true outputs in the same sense but rather a range of decisions that lead to varying outcomes. This poses more options for what an explanation explains and is thus measured. See Gyevnar and Towers (2025) for a larger discussion on possible objective measurements for XRL, of which a few are discussed in Section 3.3.

It should be noted that it is feasible for all these unique challenges to be present in an environment, e.g., an automated driving agent where a car must drive to a location considering other drivers and pedestrians, in which an area of observation might be blocked by other cars. As a result of these unique challenges for XRL, a variety of algorithms have been designed, and we outline a selection in the next section.

3.2 Explainable Reinforcement Learning Survey

In this section, we present table 3.1, a non-exhaustive list of XRL algorithms containing a short description of the explanation mechanism, what environments the algorithm was tested on and whether a human evaluation was conducted. From this table, in Sections 3.2.1 to 3.2.4, we summarise several key explanation types related to novel work presented or that are utilised as a comparison in this thesis: saliency maps, natural language explanations, temporal explanations and reward explanations, respectively.

TABLE 3.1: Table of XRL techniques comparing their explanation classes, testing environment, and if they used a human study to evaluate their explanations.

Paper	Explanation Mechanism Properties	Testing Environment	User Survey
Bewley and Lawry (2021)	Direct Decision Making with Image / Logic explanations	Toy	✗
Coppens et al. (2019)	Visual explanations and explanation through simplification	Supermario Bro	✗
Costa et al. (2024)	Intrinsically interpretable decision tree	Cartpole, Mountain Car, Lunar Lander and Crop Management	✗
Cruz et al. (2019)	Text explanation using feature relevant data	Bounded and unbounded Gridworld	✗
Cruz et al. (2021)	Probabilistic-success of observation explanation	Deterministic and stochastic navigation task, and continuous visual sorting task	✗
Déletang et al. (2021)	Counterfactuals and domain knowledge	Gridworld	✗
Deproost et al. (2024)	Generic Programming for symbolic policy encoding	Toy	✗
Dodson et al. (2011)	Direct Decision Making and Text explanation	Recommendation System	✗
Gottesman et al. (2020)	Identify observations with the greatest influence on kernel and linear least square Q-value estimator	Intensive Care Unit dataset	✓
Greydanus et al. (2018)	Visual explanation (saliency map)	Breakout	✓
Grupen et al. (2022)	Intrinsically interpretable, concept-based policies	Meltingpot	✗
Guo et al. (2021)	Temporal explanation	Mujoco and Atari	✗
Gupta et al. (2020)	Saliency Map	Chess, Go	✓
Gyevnar et al. (2022)	Causal explanation with text	Carla Driving Simulator	✗
Gyevnar et al. (2024)	Counterfactual simulation with text	IPG2 (Top-Down Driving Simulator)	✓
Hayes and Shah (2017)	Text explanation	Gridworld, Cart-pole, Inspection task	✗

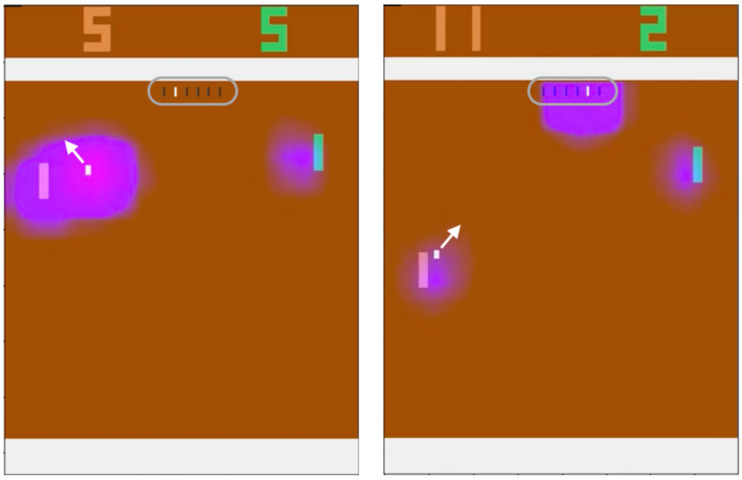
He et al. (2020)	Feature relevant explanations	UAV autonomous navigation	✗
Hein et al. (2017)	Explanation through simplification	Mountain Car, Cartpole balance and swing up	✗
Huber et al. (2021)	Strategy summarisation with saliency maps	Pacman	✓
Huber et al. (2023)	Counterfactual Observations	Pacman	✓
Iyer et al. (2018)	Object Identification Saliency Map	Pacman	✓
Juozapaitis et al. (2019)	Feature relevant explanation for the output	Cliff world and Lunar Lander	✗
Kenny et al. (2023)	Task and example-based explanation	Car Racing	✓
Liu et al. (2018)	Two feature relevant explanations and an explanation through simplification	Mountain Car, Cartpole and Flappy Bird	✗
Lu et al. (2023)	Reward Decomposition with LLM summarisation	Robotics hand grabber	✗
Lu et al. (2024)	Causal model of rewards	Gopher and Ms. Pacman	✗
Lyu et al. (2019)	Direct decision making with symbolic explanations	Toy and Montezuma's Revenue	✗
Madumal et al. (2020a)	Text explanations with counterfactuals	Starcraft 2	✓
Mahmud et al. (2024)	Sequential Feature Attribution	Autonomous Driving	✗
Pan et al. (2020)	Imitation learning based local and global feature-based explanations	Real-world Traffic data	✗
Ragodos et al. (2022)	Explain prototypical behaviour in situations	Supermario Bros	✗
Silva et al. (2019)	Intrinsically interpretable Decision tree	CartPole, Lunar Lander, Wildfire Tracking and Starcraft 2	✓
Sequeira et al. (2019)	Text explanation with feature relevant data	Frogger	✓
Septon et al. (2023)	Strategy summarisation with reward decomposition	Highway simulator and Pacman	✓

Terra et al. (2022)	Feature-based and reward decomposition with contrastive explanations	Telecom-antenna-tilt task, CartPole and Lunar Lander	✗
Topin and Veloso (2019)	Explanation through simplification	Gridworld	✗
Tsuchiya et al. (2023)	State transition Q-value Decomposition	Electricity Grid Simulator	✗
van der Waa et al. (2018)	Text explanations with counterfactuals	Gridworld	✓
Verma et al. (2018)	Text explanations	Car Racing Simulator	✗
Wang et al. (2020)	Text explanation through simplification	Ms. Pacman	✓
Yao et al. (2022)	Global Contrastive natural language explanations of policy changes	Toy	✗
Yau et al. (2020)	Future State Visitation Decomposition	Blackjack, CartPole & Taxi	✗
Yu et al. (2023)	Causal World Model	Lunar Lander and Starcraft 2	✗
Zahavy et al. (2016)	Visual explanation	Breakout, Seaquest, Ms. Pacman	✗

3.2.1 Saliency Map Explanations

There is a reason for the adage “a picture is worth a thousand words”, as visualisation is one of the most expressive forms of explanation. Saliency maps are visual explanations inspired by neuroscience, also called feature attribution explanations, that highlight areas of an observation that receive the most attention or are most important to the agent.

Initially intended for image classification models, Grad-CAM (Selvaraju et al., 2017) uses the convolutional layers of the model to find a feature’s importance for one of the outputs through differentiation. For a given convolutional layer, Grad-CAM computes the gradients from the layer’s features to one of the network’s outputs such that the gradient is proportional to the feature’s importance for the output. Computing a Grad-CAM explanation is very quick, requiring a single backpropagation through a neural network to the selected convolutional layer, and is highly versatile for any model that contains convolutional layers, including RL agents applied to image-based environments. Importantly, selecting the convolutional layer significantly impacts the explanations generated, as shallower layers generally identify higher-level general features, whereas deep layers find more detailed but sparse features. This produces interesting and visually appealing images; however, don't link these regions to what will happen in the future, the agent's rewards, or generally any sequential information. This makes producing actionable responses for users in practical RL applications difficult.



(a) Pong: control (b) Pong: overfit

FIGURE 3.1: A saliency map where the purple areas show the points that most influence the policy of the agent. Source: Greydanus et al. (2018)

Specially designed for RL, Greydanus et al. (2018) proposes adding noise to particular areas of an observation to measure its impact on a policy's output. The regions of the observation with the largest policy difference due to the noise are perceived to have the most significant importance for the policy when making decisions. We refer to this noise as perturbation and thus the algorithm as Perturbation-based Saliency Map (PBSM), and it is used in Chapter 4. More formally, Greydanus et al. (2018) modifies the observation with Gaussian noise applied to different regions using a circular mask of fixed radius (Eq. (3.1)). Using the perturbed images at each coordinate, (i, j), Eq. (3.2) computes the saliency map using the mean squared error between policy Q-values of the original image and the perturbed image. Figure 3.1 is an example saliency map produced.

$$\text{Perturbed Image}_{i,j} = \text{Image} \odot (1 - \text{Mask}_{i,j}) + \text{Blurred Image} \odot \text{Mask}_{i,j} \tag{3.1}$$

$$S_{i,j} = \frac{1}{2} \sum_{a \in A} ||Q(\text{Image}, a) - Q(\text{Perturbed Image}_{i,j}, a)||_2 \tag{3.2}$$

To validate PBSM, Greydanus et al. (2018) conducted a user survey with 31 students where participants were asked to identify which of two agents was acting; the first learnt to play from scratch using Atari observations, and the second where particular pixels encoded an expert agent's policy that the agent learnt to follow. Researchers found that the students trusted the expert-following agent more than the learnt-from-scratch agent without access to the saliency map. However, upon viewing the saliency map, the student's trust in the expert-following agent dropped from 48.4% to 25.8% as they could see the agent focused on several unimportant pixels which didn't match human expectations. Compared to Grad-CAM, PBSM is significantly slow to generate an explanation as the saliency map requires repeatedly computing the agent's policy, making the algorithm less scalable to large images.

Building on the perturbation-based method in Greydanus et al. (2018) with Eq. (3.1), Gupta et al. (2020) proposes SARFA, an improvement for better identifying regions of importance than computing the policy error. Gupta et al. (2020) reasoned that using the mean squared error across all actions can identify regions that change Q-values unrelated to the optimal action taken, possibly producing confusing saliency maps. Therefore, Gupta et al. (2020) proposes focusing on specific and relevant regions of the observation relevant only to the optimal action, $\hat{a}$. Eq. (3.3) computes the difference of softmax between the optimal action and other actions for the original observation, s, and the perturbed observation, s' for (i, j). To find only relevant changes to the policy, Eq. (3.4) computes the softmax difference without the optimal action, then the KL divergence between the original and perturbed observation, $P_{\text{rem}}(s', \hat{a}) || P_{\text{rem}}(s, \hat{a})$.[2] This equation discourages regions with significant changes between non-optimal actions for the original and perturbed observations. Using Eqs. (3.3) and (3.4), Gupta et al. (2020) computing the saliency map with $\frac{2K\delta p}{K+\delta p}$ where $K = \frac{1}{1+D_{\text{KL}}}$. To demonstrate SARFA's effectiveness, Gupta et al. (2020) presents illustrative examples for Atari, Chess, and Go against Greydanus et al. (2018) and Iyer et al. (2018). Further, they measured the time users took to solve a puzzle using a dataset of Chess puzzles with 40 proficient chess players (1600 to 2000 ELO). SARFA was fastest compared to Greydanus et al. (2018), Iyer et al. (2018), and no saliency map. Additional ablative experiments were conducted using different functions to combine Eqs. (3.3) and (3.4). SARFA produces significantly more interpretable saliency maps across a wide array of environments; however, it still struggles to explain sequential or RL-specific features of the agent's decision-making.

$$\Delta p_{a'} = \frac{\exp(Q(s, \hat{a}))}{\sum_{a \in A} \exp(Q(s, a))} - \frac{\exp(Q(s', \hat{a}))}{\sum_{a \in A} \exp(Q(s', a))} \tag{3.3}$$

$$P_{\text{rem}}(s, \hat{a}) = \frac{\exp(Q(s, \hat{a}))}{\sum_{a \in A / \hat{a}} \exp(Q(s, a))} \tag{3.4}$$

One of the saliency map's general limitations is that it only highlights areas of importance and does not explain the decision-making behind the policy. Local

[2] See Definitions and Abbreviations for the mathematical definition of the KL Divergence.

Interpretable Model-agnostic Explanations (Ribeiro et al. (2016a), LIME) is a model-agnostic mechanism that learns an interpretable model approximating the original black-box model, including deep RL agents. Rather than trying to approximate the model for the whole dataset, LIME learns an interpretable model for an individual input by generating a dataset of modified inputs and their outputs for the original model. Then, using a weighted importance function to compare how similar the modified input is to the original, an interpretable model (e.g., a decision tree) can learn a function for the input. One of the advantages of LIME is that it is model and data-agnostic, allowing it to work with any function and complex data types, e.g., text, tabular, and images. Figure 3.2 is an example of LIME using a superpixel (groups of pixels representing an object) to find the features in an interpretable model. This approach is popular within supervised learning; however, it hasn't had large-scale adoption to reinforcement learning, and it is unknown how effective the algorithm is.

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

FIGURE 3.2: An example of LIME for an image of a dog playing the guitar in the image and the top 3 classes for the Google Inception v3 model. Source: Ribeiro et al. (2016a)

3.2.2 Natural Language Explanations

In our daily lives, natural language-based explanations, including verbal explanations, are the most common type of explanation that humans produce. Therefore, text-based explanations interest researchers when designing explanations that humans use most. However, one of the limitations of natural language explanations is that there is a complexity to following the rules of a language, and it often requires additional domain knowledge. In contrast, visual explanations often do not require domain knowledge but are limited in the types of information provided to users.

To minimise the quantity of necessary domain knowledge, Wang et al. (2020) proposes first learning a simplified version of the agent's policy using a rule-based model created from a small quantity of domain knowledge. This rule-based model can have natural language descriptions of the policy programmatically generated, however, it doesn't match the performance of the original agent. Therefore, using the rule-based explanations as training data, Wang et al. (2020) proposed a feature encoder and verbal decoder to generate more comprehensive explanations to expand the list of rule-based explanations for more general situations. This learning structure is shown in Figure 3.3.

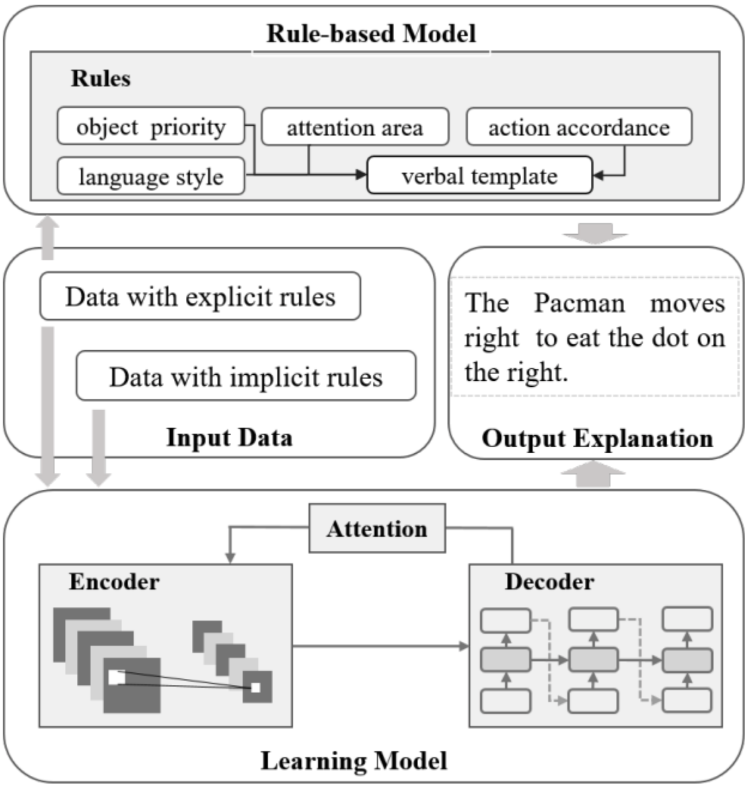


FIGURE 3.3: Shows how the rule-based and learner models use input data to generate explanations. Source: Wang et al. (2020)

They tested seven different encoder-decoder neural network models, finding that attention-based neural networks performed best. In a human evaluation trial, users were asked to rate their satisfaction and to predict an agent's next action. They found that their explanation improved the satisfaction rate of the system and that the extended neural network-based learner had higher satisfaction rates than the rule-based system.

Customers are a core audience for explanations, as noted in Section 2.3.1, due to the importance of building trust with users who may use a system regularly. Dodson et al. (2011) proposes an explanation mechanism for recommendation systems to explain the optimal action using argumentation-based techniques. The explanation mechanism incorporates domain-specific and independent information that Dodson et al. (2011) applies to a university's student module selection website. Using a tractable MDP and a policy of recommended actions, the mechanism extracts action-factored differential values that specify how much better the optimal action is regarding the actions in the next state. However, the explanations require computing the policy for every state in the MDP given an input. This, while possible for small environments, is infeasible for larger or multi-agent environments, which we develop explanations for in this thesis. Deploying the explanations for the module selection website, Dodson et al. (2011) found that students had high satisfaction and understood the short-term outcomes of a decision; however, long-term explanations were more difficult for users to accept.

When robots collaborate with humans to achieve a task, Hayes and Shah (2017) proposes a natural language query and response explanation mechanism. They implement this with a four-stage algorithm: understand the query, resolve the states necessary to explain, summarise the features of the states to explain, and compose the summarises into natural language to present to the user. This requires an environment where every state is tractable, such that the agent's policy for every state action can be queried and understood to answer questions. Further, Hayes and Shah (2017) utilises templating for language queries or responses, meaning that only particular questions are possible to answer, e.g., "When [do | will] you action take?". Testing this for a gridworld where a robot delivers packages, CartPole, and an inspection robot that locates and inspects components on a stock feeder. With detailed natural language descriptions of state components, Hayes and Shah (2017) can generate similar explanations to domain experts. However, no user survey was conducted to validate this. A core limitation is language templating, as this minimises the number of possible questions or responses that can be given to users and requires significant domain knowledge to implement.

3.2.3 Temporal Explanations

RL agents take a sequence of actions over time to achieve their goals. Therefore, their decision-making should incorporate future or temporal components into their reasoning. This matches the expectation of human explanations from psychological studies (Miller, 2019; Madumal et al., 2020a). Despite the strong reasons for these types of explanations, more thoroughly discussed in Chapter 1, there has been relatively little XRL research. In this section, we outline a majority of the literature on Temporal Explanations for XRL.

Psychology studies have found that humans are often more interested in why another action was not taken than why a particular action was taken based on their future impact (Miller, 2019). Therefore, van der Waa et al. (2018) explores contrastive explanations that use foil or counterfactual actions, an action or state that is not taken or viewed. To explain an agent, they propose contrasting the consequence of an agent's optimal action with a foil action that a user can query. For this foil action, a new policy was trained from the original policy that completes the foil action, from which a comparison of the original and counterfactual / foil policy can be generated as an explanation. In a human study, users found that explanations that explain the whole policy rather than a single action are more helpful, as the next action can be noticeable, but not the agent's plan. A limitation of van der Waa et al. (2018) is that this was only demonstrated for Gridworld environments and the need to retrain the policy for each explanation, making it difficult to scale to more complex domains.

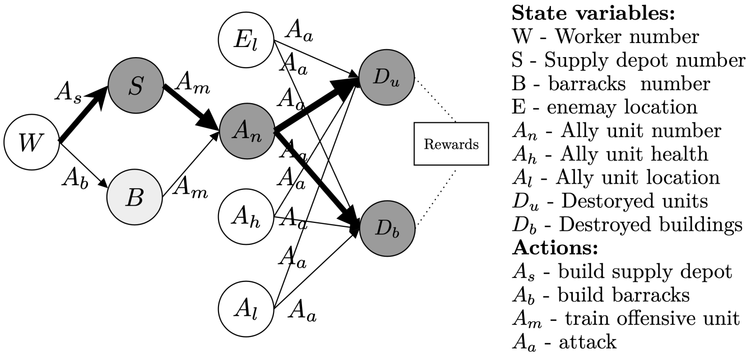
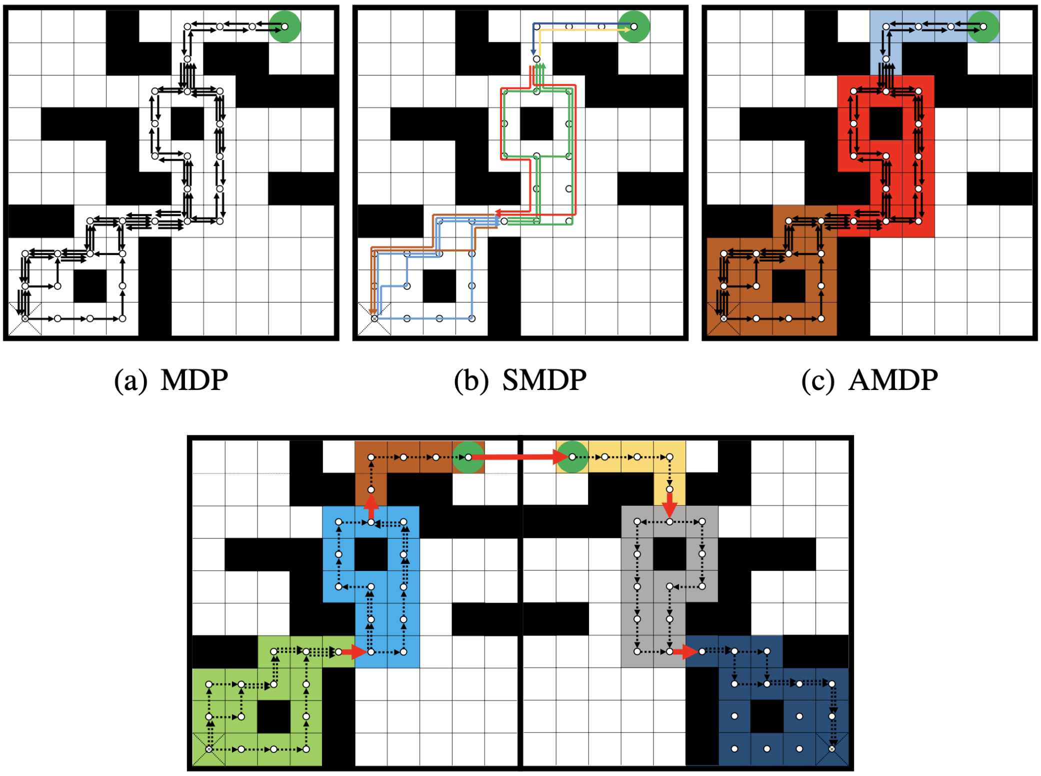


FIGURE 3.4: Directed Acyclic graph for the causal model is used to generate explanations. Source: Madumal et al. (2020a)

Using causality, humans commonly ask why or why not an action was taken in terms of its future outcome. Madumal et al. (2020a) proposes a text-based explanation generated from a preconstructed structured causal model where the relationship between the states and actions is learnt, allowing the causality between actions to be found. The advantage of using causal structures is the reduced search for counterfactual possibilities compared to van der Waa et al. (2018), where a search algorithm and tractable MDP must be used. Despite this, the structured causal model must be hand-generated for each environment (Figure 3.4), which may introduce errors or fail to account for the possible causes of action. Interestingly, in a human study, they found that while the explanation helped increase understanding, it did not increase user trust in agents. More research is required to understand if this is a unique problem for the setting or explanation, or a more general problem with prior work that did not ask this question. Madumal et al. (2020a) differs from work proposed in Chapter 4 as its model is hand-crafted and uses interpretable environment observations, whereas our work learns a similar temporal model without prior domain knowledge and can work with complex agent observation, e.g., images.

CEMA (Causal Explanations in Multi-Agent systems, Gyevnar et al. (2024)) is an alternative causal explanation of counterfactual actions that uses a probability model for forward simulation rather than the structured causal model of Madumal et al. (2020b). For questions like “Why did you not take Y instead of X?”, Gyevnar et al. (2024) first rolls back the environment simulation to before X was taken, then forward, taking action Y (achieved through Monte Carlo Tree Search). From the new state, a mechanistic or teleological (temporal is the terminology of this thesis) can be produced by analysing the system’s features that have changed with the largest causal effect. The mechanistic explanation focuses on the action of the other agents in the simulation, while the teleological explanation focuses on the effects on the agent’s future rewards. Implementing this for an automated driving simulator with predefined interesting features, Gyevnar et al. (2024) surveyed users for various situations, including roundabouts, T-junctions, and crossings. First, users were asked to write their own explanations of an agent's behaviour. Then, a second group of users were asked to compare the CEMA and the best user-generated explanations, ranking them and specifying their trust. Gyevnar et al. (2024) found that CEMA explanations were ranked similarly to human explanations, though they didn't affect the trust levels significantly. A limitation is that the explanation mechanism doesn't incorporate any information from the policy being explained, solely the effect on other agents if counterfactual actions were taken.



(a) MDP (b) SMDP (c) AMDP

(d) SAMDP

FIGURE 3.5: a. MDP diagram: the relationship between states and actions. b. SMDP diagram: Edge colours represent different skills. c. AMDP diagram: colours represent the clusters of the aggregated states. d. SAMDP diagram: colour clusters found after transforming the state space with dashed arrows for explaining skills and red arrows showing the overall policy. Source: Zahavy et al. (2016)

To anticipate changes in the future, Zahavy et al. (2016) proposes extending an environment's notation to group states that are similar and sequences of actions into skills. As most reinforcement learning environments use Markov Decision Processes (MDP), Zahavy et al. (2016) extends the state and action space to form a semi-aggregated MDP (SAMDP) that can group states and actions. Figure 3.5 shows how grouping states and actions can produce interpretable policies. Using an agent's observations dataset, they propose using internal neural network information (the final hidden layer for an Atari agent neural network) and t-SNE (Van der Maaten and Hinton, 2008) to embed the agent's policy in a 2D space. With the policy embedding, Zahavy et al. (2016) proposes a novel spatio-temporal distance metric for K-Means clustering that finds clusters of temporally close states meant to represent an agent's skills. From these groupings, researchers can provide in-depth explanations of the agent's policy from the skill clusters, a transition matrix of the skills, and domain knowledge of the environment. However, no human studies were ever completed to evaluate the effectiveness of the explanation, nor were any local explanation mechanisms proposed.

How important is an observation or action (relative to others) for achieving the final environment's results? Guo et al. (2021) proposed a self-explainable model that augments a Gaussian Process (GP) with a custom kernel function and interpretable predictor, referred to as EDGE (strategy-level Explanations of Drl aGEnt). Assuming no direct access to the agent's Q-value or policy, Guo et al. (2021) learns to predict an episode's cumulative reward for embedding observations from $1 : T$ and the final observation of an episode. The embeddings are generated through an RNN for the observation sequence from $1 : T$ and a shallow MLP for the final observation. These embeddings are combined through a Gaussian Process to capture the timesteps' correlations. From this combined embedding, a linear regression model is used to predict an episode's cumulative reward, as the variable coefficients can be used to identify important timesteps across an episode. This enables strategy-level interpretation of an agent to be generated. EDGE was implemented for two robotics environments (You-Shall-Not-Pass and Kick-And-Defend) and Atari Pong. It was evaluated by replacing the top K important states (and the next N states) with random actions and then evaluating the difference in the resultant episode's cumulative rewards. Guo et al. (2021) conducts this fidelity evaluation with alternative self-explainable approaches such as Saliency, Attention, and Rational Nets, finding EDGE the most effective.

How does an agent's policy internalise their beliefs about the future? Yau et al. (2020) and Tsuchiya et al. (2023) proposed similar approaches to decompose an agent's beliefs about their future state transitions. For a tractable MDP where the set of possible states is known, Yau et al. (2020) proposes learning a novel future state-visitation function that computes the probability that a particular (future) state will be visited within an episode. Implementing this for a temporal difference (bootstrap-based) and Monte Carlo rolling training algorithm, Yau et al. (2020) demonstrates that this explanation could accurately predict what future states will be visited in Blackjack, CartPole, and Taxi environments. This requires knowledge of the whole environment state space and discretising continuous spaces, such as CartPole, limiting its applications. Inspired by Yau et al. (2020), rather than learning a separate future state visiting function, Tsuchiya et al. (2023) proposed directly modifying an agent's Q-value into the expected reward in several predetermined state types. This is similar to Juozapaitis et al. (2019), outlined in Section 3.2.4, where the Q-value is decomposed into reward components. For resource allocation of a power system environment, Tsuchiya et al. (2023) demonstrated that the decomposed Q-values could explain the quantity of rewards collected in each state type. However, unlike Yau et al. (2020), it required domain knowledge to categorise these states. Further, both algorithms were only applied to toy problems and contained inherent problems when scaling to more complex environments.

3.2.4 Reward Explanations

Reinforcement learning agents are optimised to maximise their expected rewards over time. Therefore, researchers have proposed explanation algorithms to explain how or what rewards the agent will receive.

The formulation of RL described in Section 2.2 utilises scalar rewards; however, for more complex environments, this scalar reward is often composed of sub-rewards from different sources/components. For environments that provide information on these reward components, Juozapaitis et al. (2019) identifies that the reward at timestep, R_t, is equivalent to the sum of reward components, $\sum_{c \in C} R_{t,c}$. As a result, Q-values, $Q(s, a)$, could be learned as the sum of expected rewards for each component, $\sum_{c \in C} Q_c(s, a)$. Therefore, Juozapaitis et al. (2019) proposes extending a deep reinforcement learning agent's output to include the reward components and a novel loss function to minimise the error for each reward component. Using these predicted future reward components, Juozapaitis et al. (2019) proposed a novel contrastive method that explains the minimal component differences between two actions. They demonstrate that this learning methodology can scale to complex domains with the lunar lander and Atari environments. However, Juozapaitis et al. (2019) requires that the environment provide the decomposed rewards, a feature uncommon in most RL environments but with growing support (Felten et al., 2023).

Further work has built on Juozapaitis et al. (2019). Integrating policy summary by Amir and Amir (2018) with reward decomposition, Septon et al. (2023) proposes combining the global and local capabilities of each to complement one another. This enables explaining important states (global) found during training and predicting the quantity of rewards for each source of each state (local). They demonstrate that this explanation is effective for Highway-Env and Atari MsPacman environments, where users predict an agent's strategy. For robotics environments, Lu et al. (2023) demonstrated applying reward decomposition for an abstracted action space by mapping an observation to task-specific rather than the base actions. Importantly, these task-specific actions have natural language summaries to describe them; therefore, Lu et al. (2023) proposes utilising Large Language Models to summarise the reward decompositions with natural language descriptions of the action taken. However, this requires significant quantities of domain knowledge (task-specific action specifications, LLM explanation templates, and the decomposed reward sources) to implement.

When interpreting a policy, an important question is why an agent believes it will receive a reward in the future. Lu et al. (2024) proposes learning a causal model of the state and actions to an agent's expected rewards. This differs from Madumal et al. (2020a), which learnt a causal relationship between states and future states. Driven by three objectives: sufficiency, sparsity, and orthogonality, Lu et al. (2024) proposes two separate training approaches to learning the causal model. The first learns to distil the reward components of an environment from a state, referred to as R-mask, and the second uses the reward components and state to distil the Q-value, referred to as Q-mask. From these two masks, Lu et al. (2024) use Grad-CAM (Selvaraju et al., 2017) to generate saliency maps of the mask's areas of importance. Utilising both approaches for two Atari games, Ms Pacman and Gopher, to identify observation regions with the most significant causal influence for determining the reward components and the Q-value. Despite the novel causal-based explanations, it is difficult to identify novel information that the causal inferences provide to the saliency maps that is not already available using algorithms provided in Section 3.2.1.

3.3 Evaluating XRL Explanations

Given the XRL algorithms surveyed in Section 3.2, we summarise several human-based evaluation procedures. As noted in Section 3.1, evaluation is a novel challenge within XRL, and more generally, XAI, with minimally agreed upon or common evaluation procedures. Gyevnar and Towers (2025) have identified two general applications for explanations: debugging, where researchers and developers validate an agent's behaviour before deployment, or for fault investigation and human-agent teaming, where explanations could be used as communication for coordination, accountability, etc.

RL policies map states to actions with some explanation mechanisms focusing on illuminating this function to users. To test this, researchers have evaluated whether users shown an explanation for a policy can determine the agent's next action. We refer to this approach as Next Action Prediction, with several papers implementing this evaluation style (Silva et al., 2019; Madumal et al., 2020b). Modelling an agent's policy as a decision tree that is specialised for RL, Silva et al. (2019) ran an interpretability study (alongside computational evaluations) comparing their discrete decision tree, a decision list, and a binarized neural network trained on the CartPole toy problem. Fifteen participants were asked to predict the policy output for five different states and rank their feelings of interpretability for each model. Silva et al. (2019) found that the decision tree and decision list had the lowest time at 2.3 and 2.1 seconds, compared to the binarized MLP at 7.8 seconds. These results matched the self-reported feelings, with the decision tree having the highest rating, with the decision list a couple of points lower (40 and 36, respectively), whereas the MLP scored 25 points. Madumal et al.

(2020b) ran a more complex survey using the Starcraft 2 environment where 120 participants were asked to answer "what will the agent do next?" along with two subjective questions of Explanation Satisfaction Scale and Trust Scale (proposed by Hoffman et al. (2018)) using a 5 Point Likert scale (Likert, 1932). For 8 questions, Madumal et al. (2020b) found that users scored the highest with their causal explanation at 11 out of 16, however, for confidence, predictability, reliability and safety, no explanation had a significant difference in score. Overall, a limitation of Next Action prediction is that for environments with hundreds or thousands of actions, understanding an agent's behaviour might require reviewing vast numbers of explanations. Compared to other explanatory objectives below, the Next Action prediction explanation gives a microscopic understanding of an agent's policy, focused solely on a particular state action and not providing a more extensive, general understanding of the policy for users.

In RL, a policy is optimised to maximise the cumulative rewards for an environment's reward function. Researchers can modify that reward function to produce multiple policies for the same environment dynamics to maximise different rewards and achieve different goals. Several papers conduct variations on these evaluations with different reward functions to produce different policies where, using explanations, users must identify the underlying goal (Amir and Amir, 2018; Huber et al., 2023; Septon et al., 2023). Huber et al. (2023) implemented a variant of Goal prediction where users select the items they believe were most important to the agent (not the goal itself). Three agents were trained in the Atari MsPacman environment with different reward functions (standard, power pill, fear-ghosts). Users were further tested by asking which agent they would select to achieve different goals, and the user's subjective satisfaction with the explanation. Huber et al. (2023) found that the explanations tested achieved scores of 0.8, 1.18, and 1.65 out of a maximum score of 3. However, user satisfaction didn't significantly vary between users. For two environments (Highway-Env and Atari MsPacman), each with two goals, Septon et al. (2023) proposed asking users to rank which goal was prioritised by an agent or if neither was using an explanation. Users were further asked to rank their opinions of the explanations using a 7-point Likert Scale for five questions. For the four explanation mechanisms, the highest accuracy found for either environment was around 60%, where random is 33%. Overall, a limitation of the Goal prediction evaluation outlined above is that the different goals implemented appear to have some significant overlap, meaning that if two agents perform a common task, it might not be possible to differentiate between the agents' behaviour/goal. This can be solved by using more clearly differentiated goals for agents.

For human-agent teaming applications, Gupta et al. (2020) evaluates their explanations as support for players attempting chess puzzles, a common training technique to improve players' intuition for challenging positions. Gupta et al. (2020) proposed SARFA, a saliency map algorithm to highlight pieces of importance to help users when solving. Surveying 40 chess players ranked between 1600 and 2000 ELO, each was shown 15 puzzles taken from `chess.com` with either no saliency map or a saliency map from Greydanus et al. (2018), Iyer et al. (2018), or Gupta et al. (2020). The explanation's performance was measured based on the time taken to solve and its accuracy for the puzzle. Gupta et al. (2020) found their explanation was 10 seconds faster than no saliency map (67.02 seconds to 77.53 seconds, respectively) and over 15% better at solving the puzzle (72.41% to 56.67%). It is important to note that one of the explanations (Iyer et al., 2018) significantly worsens player performance, taking 102.26 seconds on average with 24.60% accuracy. This reinforces the importance and value of human evaluations, as an explanation can appear effective while being more ineffective than no explanation for users solving a task.

3.4 Discussion

Using the outlined unique challenges in XRL (Section 3.1), a survey of the literature (Section 3.2) and evaluation methodologies (Section 3.3), in this Section, we discuss the general strengths and weaknesses of XRL research we identify. A primary strength of the literature is the wide range of explanation techniques explored, as seen in Table 3.1, many of which offer interesting and novel approaches. However, despite the number of XRL papers we surveyed, we found three recurring weaknesses:

1. The definition of explainability specified in Section 2.3 combines an understanding of a policy's content and human comprehension of an explanation. The problem is that of the 43 surveyed papers, only 14 (32.6%) completed a human study to evaluate the effectiveness of their proposed explanation. This is important as researchers reviewing the literature cannot know how effective over half of the proposed algorithms are. This has several knock-on effects. One is that researchers cannot know if future work should build upon prior work or go in new directions. This could explain why so little literature has built upon a common base, unlike RL, where a handful of algorithms are regularly improved, e.g., DQN (Mnih et al., 2015), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018).

2. Along with whether a human study was completed, for the literature surveyed in Table 3.1, we record the setting(s) used to test the effectiveness of the proposed algorithms. The number of papers that use Toy environments, e.g., gridworld or CartPole, represents 37.2% (16 of 43) of the papers surveyed. This is critical as toy problems normally have small state and/or action spaces, whereas the XRL algorithm must be scalable to real-world problems that are common complex image-based observations with large action spaces.

3. Although not recorded in Table 3.1, while surveying the research, we found that very few papers provided open-source code that allows for a better understanding of the algorithms proposed and an easy implementation for future work to use either as a building block or comparison. We believe this has partially resulted in only a handful of research, including prior work as a comparison/baseline, further limiting the capability to compare research.

Collectively, the lack of human surveys, standardised testing environments, and limited open source code mean that it is *completely unknown* the current state-of-the-art algorithm for XRL. In addition, several important questions cannot be answered due to these problems: Are certain algorithms effective depending on who evaluates them (general public vs academics)? Are visual explanations better than text explanations? How important are counterfactual explanations? These challenges pose a significant problem for the long-term health of XRL that we are working on addressing, see Chapter 7 for more detail.

Section 3.2.3 summarises almost all approaches taken for Temporal Explanation in the literature. There is a strong variety of approaches to explain different properties of agents, the most popular being counterfactual futures (van der Waa et al., 2018; Madumal et al., 2020b; Gyevnar et al., 2024). One of the weaknesses across the approaches is the need for domain knowledge in most approaches, with Guo et al. (2021) being the only algorithm that doesn't require it and scales to complex domains. A problem we investigate in Chapter 4. A second weakness in the approaches is that only Yau et al. (2020) and Tsuchiya et al. (2023) explain an agent's future rationale based on the internal properties of the neural network and only of the future states that could be taken. Whereas all others build externally on the policy or are based on the observations of how the agent acts. Therefore, we explore internal explanations of agent beliefs in Chapters 5 and 6.

Chapter 4

Video-Based Explanations of Deep Reinforcement Learning Agents

Reinforcement learning agents work to maximise their rewards over time, meaning that to completely explain an agent's decision-making is to explain an action's future impact on the agent. We refer to these as "temporal explanations" which explain an agent by highlighting the future outcomes of a decision. Investigating prior work on temporal explanation, Chapter 3 found that most heavily relied on domain / expert knowledge for the explanation, e.g., natural language descriptions of the state components (Madumal et al., 2020b). Although useful, domain knowledge can be costly to gather in terms of time, money, and availability, and may contain errors or biases. Therefore, this chapter investigates temporal explanation mechanisms for a state's outcome that require no or limited domain knowledge.

Our approach is inspired by two common human strategies for understanding agent behaviour: contextualising an observation based on previously observed behaviour to predict future actions, and grouping prior behaviour by its goals to identify and explain new observations' current goal. For example, having previously observed an agent take a sequence of actions from a state, humans might infer that in a similar setting, the agent will take the same or similar sequences of actions. Alternatively, in understanding the objective/goal that an agent is completing, humans might infer the sequence of actions that would be required to complete it. As this conception of a goal is human imposed upon an agent, not learn as an objective or necessary component of the agent. We utilise skill identifications from agent behaviour to group similar behaviour, referred to as skills that, taken collectively, can be viewed as an agent's high-level plan of an environment using the skills. Therefore, we propose three video-based temporal explanations designed to enable the user to contextualise and visualise these possible future decision paths of agents for an observation (Section 4.1). Two of the mechanisms require no domain knowledge, and the third requires minimal knowledge only of what each of the agent's skills achieves.

To assess their effectiveness, we conducted a user survey comparing them against two feature-based explanation mechanisms surveyed in Section 3.2.1. We find that all our temporal explanations had a higher average rating than both prior mechanisms across four different statements. Further, when users were presented with two explanations, they preferred our temporal explanations 83.3% of the time over the prior explanations. Finally, as two of our explanations rely on agent skills (i.e. a repeated sequence of actions with a common goal), we propose two novel metrics to measure skill coherence. In summary, this chapter makes the following contributions to explainable reinforcement learning:

- We propose a Dataset Similarity Explanation mechanism (Section 4.1.1) that doesn't require domain knowledge. Using a dataset of pre-collected observation-action pairs, essentially a memory of how an agent has behaved in the past, to explain an observation, we retrieve the most similar observation from the memory of prior agent behaviour and present a video to the user of the next n observation-actions from memory following the retrieved observation. This provides users with a video of possible future behaviour, assuming an agent behaves similarly in similar prior situations, which, for fixed agents, should be true.
- We propose a Skill Explanation mechanism in Section 4.1.2 that doesn't require domain knowledge. To explain a new observation, we identify the skill (repeated sequences of actions with a common goal) an agent is currently performing and samples an example skill from a set of prior recorded examples, presenting it as a video to the user. This contextualises an agent's behaviour based on the agent's current goal.
- In Section 4.1.3, building upon the Skill Explanation, we propose a Plan Explanation mechanism. This uses a small amount of domain knowledge by annotating each skill and the transitions between them. To apply this domain knowledge, we augment the Skill Explanations to incorporate a natural language description of an agent's skill within the video.
- As our Skill and Plan Explanations require annotating agent behaviour with discrete skills, in Section 4.2, we propose two novel metrics to measure the skill's quality, Skill Alignment and Skill Length Distribution. We demonstrate their effectiveness for human-labelled skills as a baseline against algorithmically labelled skills proposed by Zahavy et al. (2016) in Section 4.3.2.

4.1 Explanation Mechanisms

We propose three novel video-based temporal explanation mechanisms that present possible future decision-making of an agent using a dataset of prior observation-actions: Dataset Similarity Explanation, Skill Explanation, and Plan Explanation. Dataset Similarity Explanations and Skill Explanations do not require domain knowledge, whereas Plan Explanations require small amounts of domain knowledge. Examples of each explanation mechanism and source code are provided on Github [1].

4.1.1 Dataset Similarity Explanations

One way to explain an agent's behaviour is to identify how it has acted in similar situations in the past and extrapolate the expected behaviour for the current state. Using a large dataset of known observation-action pairs and an embedding function, the Dataset Similarity Explanation explains the agent by presenting a video of possible future observation-action pairs based on similar observations from memory.

We maintain a dataset of observation-action pairs of length L, $D = ((o_1, a_1), (o_2, a_2), \ldots, (o_L, a_L))$, an observation encoder $E(o) \rightarrow e$ to map an observation to a vector embedding, and observation to explain, $\hat{o}$, assumed to be not contained within the dataset. To find the most similar observation in the dataset, we find the smallest Euclidean embedding distance between the observation to explain and those in memory: $\arg\min_{i \in \{0,1,2,\ldots,L\}} \|E(o_i) - E(\hat{o})\|_2$. Using the most similar observation in memory, we generate a video of the next n future observation-actions for the user, $(D[i], D[i+1], \ldots, D[i+n])$. Importantly, due to randomness in the environment and agent policy (epsilon greedy), there can be numerous future decision paths that the agent could take. We extend the dataset similarity explanation to generate multiple non-overlapping explanations and present some of these future paths to the user. We achieve this through iteratively generating explanations with a mask to ignore all observations within n timesteps of previously selected observations when computing the most similar observation in memory.

In this work, we use the embedding method from Zahavy et al. (2016) outlined in Section 3.2.3 that computes the t-SNE projections of PCA reduction for a neural network's hidden layer activations of the dataset. There are other and possibly better embedding functions (e.g. autoencoders that are designed to learn embeddings), but we focus on using such embeddings to generate explanations rather than the embedding algorithms themselves. For our user survey (Section 4.3.1), we present 2 explanation videos each with $n = 50$ resulting in videos of 5 second at 10 observations (frames) per second. We selected these values as we believed there was a trade-off in explanation

[1] https://github.com/pseudo-rnd-thoughts/temporal-explanations-4-drl

Algorithm 1 Pseudocode implementation of the Dataset Similarity Explanation

Require: $D = ((o_1, a_1), (o_2, a_2), \ldots, (o_L, a_L))$ ▷ Observation-action Dataset
Require: $\hat{o}$ ▷ Observation to Explain
Require: $E : O \rightarrow \mathbb{R}^m$ ▷ Embedding function
Require: n ▷ Explanation length
Require: K ▷ Number of Explanations
  $D_d = (||E(\hat{o}) - E(o_i)||_2 \quad \forall i \in \{0, 1, \ldots, L\})$ ▷ Compute the dataset distance
  $e = \{\}$ ▷ Explanations
  for $k \in \{1, \ldots, K\}$ **do** ▷ Generate explanations
    $i = \arg\min D_d$
    $e = e \cup (D[i], D[i+1], \ldots, D[i+n])$
    $D_d[(i-n, i-n+1, \ldots, i+n-1, i+n)] = \infty$
  end for

complexity (e.g., presenting 10 different 1 second videos) compared to simplicity (e.g., presenting 1 video of 10 seconds). We believed that 2 videos of 5 seconds at 10 frames per second would provide adequate time for users to understand both explanations while providing multiple future decision paths for the user to view.

4.1.2 Skill Explanation

When humans describe agent behaviour, we commonly describe the goal that an agent is working towards by grouping sequences of actions into discrete skills. For example, to describe someone making a cup of tea, we could describe the sequence of actions taken: collect a mug, place a teabag in the mug, boil water, and add water to the mug. This type of description provides temporal information about the agent's behaviour that leads to a goal. A difference to the Dataset Similarity Explanations is the dynamic length of skills enables the Skill Explanation to adapt to the observation context rather than a predetermined fixed length (n). For example, baking a cake and making a cup of tea requires significantly different time lengths/scales to properly explain the agent's behaviour. Secondly, in explaining a skill, this can include observations from "before" the observation to explain would occur in the skill's sequence, providing additional context to the user on what could have led to this state and what could happen after. Skill Explanations, therefore, aim to contextualise an agent's decision-making based on an estimate of the skill is currently completing through visualising the decision path that the agent will take.

Skill Explanations rely on a skill identification function, $I(o) \rightarrow k$, mapping an observation to a skill (e.g. a hierarchical RL policy or an offline skill discovery algorithm), a dataset of known agent skills D_K and an observation to explain $\hat{o}$. We find all skill instances from the dataset of the same type as the observation: $K = \{k \in D_S | I(\hat{o}) = k\}$. With the set of skill instances, K, we select one, $k \in K$, to present to the user as a video, $(k[0], k[1], \ldots)$.

For this work, we use the skill discovery mechanism proposed by Zahavy et al. (2016). This mechanism can produce a large number of example skills in K, so we filter the skills to find instances that contain similar observations using the same embedding distance procedure as the Dataset Similarity Explanation (Section 4.1.1). Additionally, due to the large variance of skill lengths (see Figure 4.9), we find that some skills were only a couple of actions long (less than a second in a video) and others that were hundreds of actions, resulting in several minute long videos. As a result, we limit the videos in length to avoid. We do not believe this is a limitation of the explanation mechanism itself, but rather the skill identification algorithm that it is paired with, Zahavy et al. (2016) in this case. For this reason, in Section 4.2, we propose two skill similarity metrics for evaluating the quality of the skill collected and utilised in the explanations.

4.1.3 Plan Explanation

Over an episode, an agent will aim to maximise the rewards it receives by using a series of skills to accomplish various goals and sub-goals. These skills can be represented as nodes in a directed network, where the edges (referred to as skill transitions) represent the probabilities of the agent changing from one skill to another. For instance, after getting out of bed, the agent may make coffee. This skill-based plan offers a simplified global representation of the agent's policy and can be enriched with minimal domain knowledge for each skill and skill transition. Figure 4.1 is an example plan for an Atari breakout agent where, after starting, the agent will only take Skill 2, from which, a majority of the time, it will take Skill 0. We developed the Plan Explanations (PE) to utilise text-based domain knowledge to complement our Skill Explanation mechanism (Section 4.1.2) and provide additional domain-specific insights into the agent's decision-making process.

To construct a plan, we create a complete directed network, Q, with $N + 2$ nodes where N represents the number of unique skills along with two skills to indicate the beginning and end of an episode. Using a list of skills from the dataset, D_K, we assign each edge weight to an adjacency matrix, Q_{ij}, the probability that given skill i is taken, the agent's next skill is j. To incorporate domain knowledge into the plan, we reviewed examples of the corresponding skill (Q_k) or skill transition. Given an annotated plan to explain an agent's decision-making for observation, $\hat{o}$, the domain knowledge associated with the observation skill, $Q_{I(\hat{o})}$, is included with the Skill Explanation video as above. In this work, as the skill discovery mechanism only predicts the skill taken, we are limited to presenting the current skill's domain knowledge.

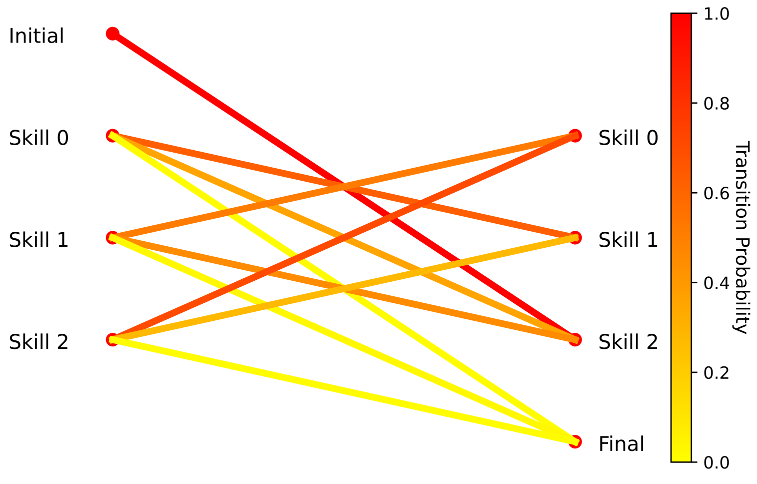


FIGURE 4.1: Example skill network for Atari Breakout using hand-labelled skills.

4.2 Skill Similarity Metrics

Our Skill and Plan Explanation mechanisms (Section 4.1.2 and 4.1.3) utilise past skill examples to contextualise an agent's decision-making. It's important, therefore, that instances of the same skill resemble each other as closely as possible. We propose two novel metrics to quantify this similarity: the first evaluates the similarity in actions for two skill instances, while the second metric assesses the similarity of skill instances in terms of their lengths.

4.2.1 Skill Alignment Metric

Given two instances of a skill, we expect both to share a common sequence of actions to complete a common goal. To quantify this alignment of actions, we propose the Skill Alignment metric that scores the similarity between two skill instances by identifying the longest subsequence of common actions between them. We employ the Smith-Waterman algorithm (Smith and Waterman, 1981) to implement our skill alignment metric, which is commonly utilised in bioinformatics for identifying comparable sequences in protein and DNA databases (Pearson, 1991), as well as in text analysis (Su et al., 2008). Importantly, the algorithm can skip sequence elements to create a more optimal alignment, e.g. to ignore a mutation in a DNA sequence. This ability to skip elements is critical in evaluating skills as policies are frequently stochastic, and minor variations between instances of a skill are expected.

The Smith-Waterman algorithm (Smith and Waterman, 1981) uses dynamic programming to build a scoring matrix for two sequences along with a similarity (or scoring) function and penalty (or gap) score through iteratively computing the highest score of possible subsequences by choosing to include the sequence element or skip with a penalty. As a result, the optimal subsequence is the maximum score in the matrix. We adapt the Smith-Waterman algorithm for computing the alignment of two skill instances, K^1 and K^2. Using the scoring matrix H, for each element of the matrix, we solve Eq. (4.1) where $s(a,b) \rightarrow \mathbb{R}$ is a similarity scoring function to compare two actions and W is gap penalty for allowing particular actions to be skipped/ignored at a cost. The equation evaluates the score at the previous subsequence plus the score of moving to ij, either including the actions K^1_i and K^2_j in the subsequence or skipping either K^1_i or K^2_j in the subsequence.

$$H_{i,j} = \max(H_{i-1,j-1} + s(S^1_i, S^2_j), H_{i-1,j} + W, H_{i,j-1} + W, 0) \tag{4.1}$$

Importantly, to provide a comparable score between skill instances of different lengths, we normalise the metric to $[0,1]$. Selecting the best sequence's score, $\max(H)$, we divide it by the highest possible score of any two skill instances with the same length as K^1 and K^2. Therefore, for two skill instances that are equivalent, their normalised score will be 1, while two skills with zero common action sequences will have a score of 0, no matter the skill instance's length.

Algorithm 2 Adapted Waterman-Smith Algorithm for Normalised Skill Alignment Metric using Discrete Actions

Require: $K^1 = (a^1_0, a^1_1, \ldots, a^1_{L_1})$ and $K^2 = (a^2_0, a^2_1, \ldots, a^2_{L_2})$ ▷ Skill Instances to compare
Require: $w : \mathbb{R}$ ▷ Penalty
Require: $s(a^1, a^2) \rightarrow \mathbb{R}$ ▷ Score function
  $H_{0,j} = 0 \quad \forall j \in \{0, 1, \ldots, L_1\}$ ▷ Initialise Alignment Matrix
  $H_{i,0} = 0 \quad \forall i \in \{0, 1, \ldots, L_2\}$
  for $i \in (1, \ldots, L_1)$ **do**
    for $j \in (1, \ldots, L_2)$ **do**
      $H_{i,j} = \max(H_{i-1,j-1} + s(K^1_i, K^2_j), H_{i-1,j} + W, H_{i,j-1} + W, 0)$ ▷ Eq. 4.1
    end for
  end for
  Score $= \max(H)$ ▷ Score
  Normalised score $= \max(H) \div \min(L_1, L_2)$ ▷ Normalised score for discrete actions

We use the Arcade Learning Environment (Bellemare et al., 2013), commonly known as Atari, in which agents take discrete actions. We therefore define our scoring function as $s(a_1, a_2) = a_1 \equiv a_2$, such that the score is 1 if the actions are equivalent and 0 otherwise. We use a linear gap penalty of -1 for W. To normalise the metric, given our scoring function, the maximum score occurs when one of the skills is a subsequence of the other. Consequently, the maximum possible score is the length of the shorter skill instance. Algorithm 2 provides a pseudo-code implementation for our normalised skill alignment metric.

4.2.2 Skill Distribution Metric

If instances of a skill contain a similar sequence of actions, we would expect them to contain a similar number of actions; for example, each time an agent makes a cup of coffee, we expect it to take roughly the same amount of time. For a set of skill instances, the Skill Distribution metric quantifies the probability that their lengths could have been sampled from the same probability distribution.

We use the Kolmogorov-Smirnov test to assess the likelihood that a set of skill lengths could have been sampled from a given distribution. It measures the probability that the distance between the actual skill length distribution and the expected reference distribution matches. Given skill lengths are all positive discrete values, we use the Poisson distribution, commonly used for modelling counting events, e.g. the number of mutations in DNA or the number of times a web server is accessed. Following (Paszek, 2007, Section 3.6), the reference Poisson distribution has a mean, λ, equal to the value of the average skill length.

4.3 Quality of Explanations and Skills Extracted

In this section, we present an analysis of our novel temporal explanations and skill similarity metrics (Sections 4.3.1 and 4.3.2 respectively). We use the Arcade Learning Environment (Bellemare et al., 2013), commonly known as Atari, for its image observations, long episodes, and complex objectives in this analysis. We selected the Breakout, Space Invaders, and Seaquest games from the 60 available Atari games for their variety of skills and the expected prior knowledge of users, which was confirmed by our users, see Figure 4.5. To support the reproducibility of this work, we used open-source pre-trained DQN (Mnih et al., 2015) and Rainbow (Hessel et al., 2018) agents released by Castro et al. (2018). Furthermore, our code is open source to promote future comparative user surveys with our explanations [1]. For our explanations, we used a dataset of $60,000$ observation-action pairs and, following Zahavy et al. (2016), conducted a grid search across different window sizes and the number of skills for the skill clustering algorithm to find the optimal number of skills. See Section 3.2.1 on Zahavy et al. (2016) for more details.

4.3.1 Human Evaluation of Explanations

The ultimate goal is to enable people to understand the behaviour of autonomous agents so that the quality of explanations can only be evaluated by people. We, therefore, conducted a user survey to compare our three novel temporal explanation mechanisms with two state-of-the-art feature-based explanation mechanisms:

Grad-Cam (Selvaraju et al., 2017) and Perturbation-based Saliency Maps (Greydanus et al., 2018) referred to as PBSM (both are explained in Section 3.2.1). We adapt Grad-CAM for Atari agents using the agent's action as the discrete neural network output to differentiate by and the second convolutional layer of the DQN neural network architecture as the target layer. Appendix B includes example Grad-CAM saliency maps for each possible convolutional layer within the neural network. We decided not to use Madumal et al. (2020b) or other temporal explanation surveyed in Section 3.2.3 for two reasons. First, that Madumal et al. (2020b) requires the use of an interpretable environment feature for the structured causal model, which Atari does not provide and that it requires significantly more domain knowledge compared to our Plan Explanations.

For the survey, users evaluated the explanations using two methods: individually rating observation/explanation for four statements and comparing two explanations for the same observation, selecting the preferred explanation for four statements (referred to as individual and contrastive evaluations). On average, the user survey took 20 minutes to complete for 26 volunteer university students.[2]

For the individual evaluation, users are shown three observation explanations for each explanation mechanism (one for each testing environment). Users are asked to rate their agreement on four statements, "From the explanation, I understand how the agent works.", "This explanation of how the agent works has sufficient detail" and "This explanation of how the agent works is satisfying" (all inspired by Hoffman et al. (2018)) and "Using the explanation, I can predict the agent's next action" (inspired by Madumal et al. (2020b)), from Strongly disagree to Strongly agree, based on the Likert scale (Likert, 1932). See Figure 4.2 for an example figure shown to the user. This evaluation generates 60 ratings per user (4 statements $\times$ 3 explanation-observations $\times$ 5 explanation mechanisms). To minimise bias due to the order in which explanations are presented to the participants, some saw the temporal explanations first, while others saw the feature-based explanations first.

For the contrastive evaluation, users were presented with an observation and two explanations and asked to select their preferred explanation ("This is more true of explanation 1 than explanation 2", "This is more true of explanation 2 than explanation 1", and "This is equally true for explanation 1 and 2") across the same four statements as the individual evaluation. We presented users with nine observations/explanations, one for each permutation of the explanation mechanism (excluding the Skill and Plan explanation pair, as the only difference is the addition of domain knowledge). We infer the effect of adding domain knowledge from the performance difference for the Skill and Plan Explanations to another explanation mechanism. This contrastive evaluation enables us to elicit explicit preferences among explanation mechanisms in addition to user ratings from the individual evaluation to confirm user opinions on explanations.

[2]We obtained ethical approval for the survey from our institutional review board (ERGO FEPS/78297).

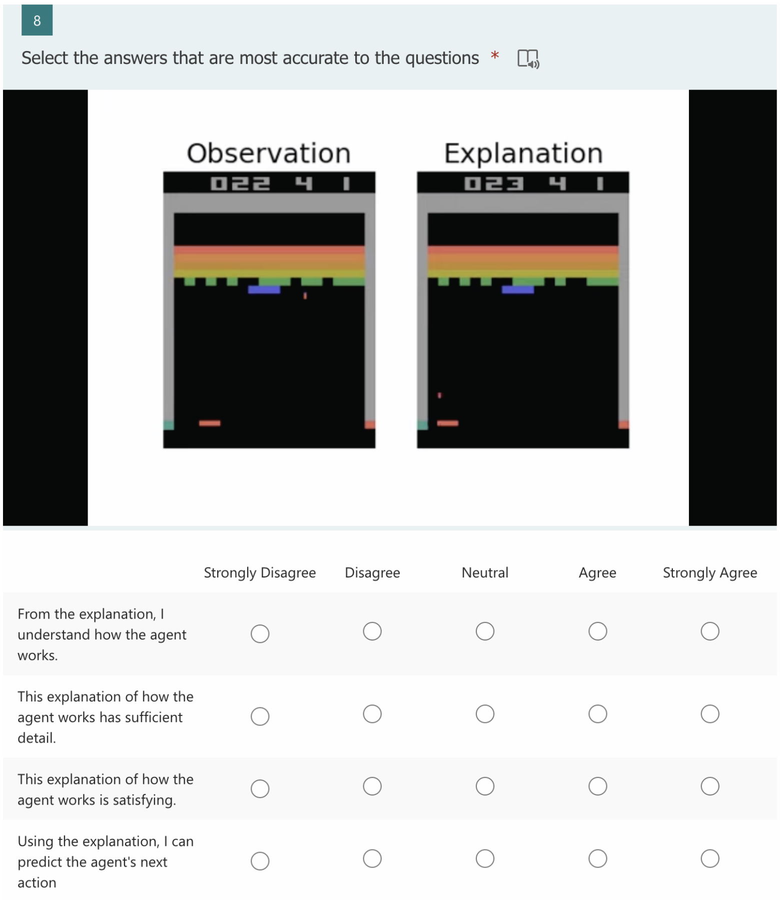


FIGURE 4.2: Example individual question shown to the user.

To mitigate potential sources of selection bias introduced by researchers when choosing observations for the explanation mechanisms, we adopted an unsupervised environment-agnostic approach to generate observations to be explained. We assume that unimportant observations should exhibit low variance in their Q-values across their actions, as if there are minimal differences; none of the agent's actions has a greater impact on the agent's future rewards than any other action for the observation. Therefore, we focus on explaining observations with the highest Q-value variances for a dataset of example observations from the agent. Furthermore, we implemented a masking mechanism to ensure a diverse selection of observations across time. Specifically, we mask out observations that occur within 20 timesteps of any previously selected observation within the same episode. This approach is similar to the HIGHLIGHT-DIV algorithm (Amir and Amir, 2018) though using a previously collected dataset of observations rather than online sampling.

For the individual user survey, Figure 4.4 presents the user ratings for each explanation mechanism and statement where strongly disagree has a score of 1 and strongly agree has a score of 5. Our Plan Explanation (PE) had the highest average rating of 4.0 across the statements, while Grad-CAM and PBSM had the lowest average rating of 2.5. In comparison, our Dataset Similarity Explanation (DSE) and Skill Explanation (SE),

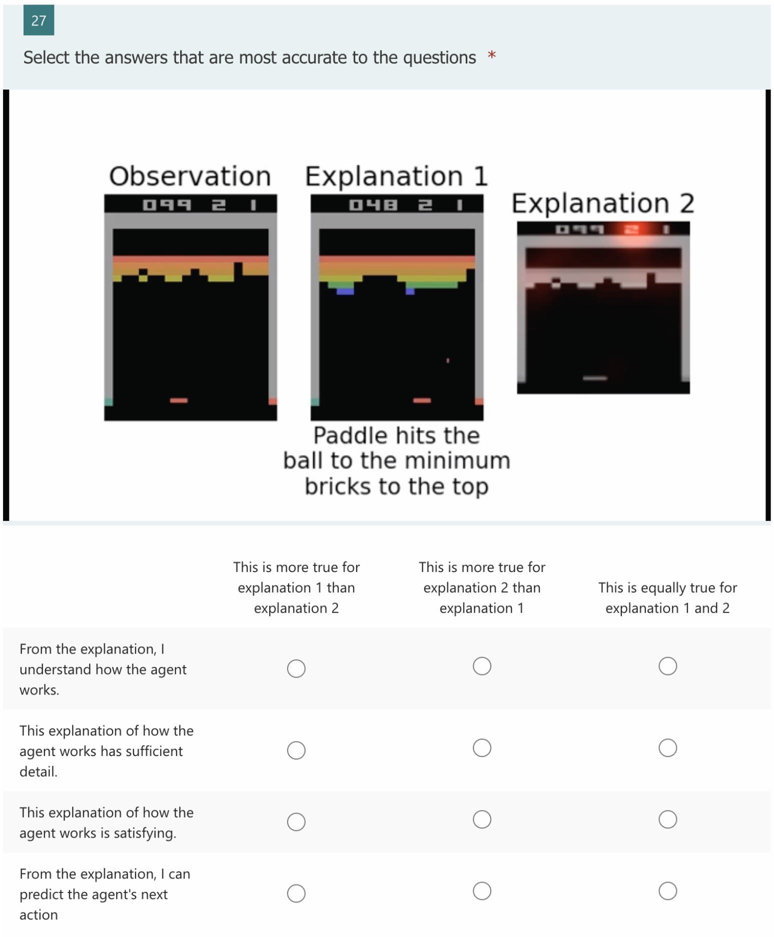


FIGURE 4.3: Example contrastive question shown to the user.

which, like the feature-based mechanisms, do not require domain knowledge, had a higher average rating of 3.5 and 3.3, respectively. In particular, users found our explanations had significantly higher ratings than Grad-CAM and PBSM regarding explanation detail and the predictability of the next action. This is unsurprising due to the additional information shown in our video explanations compared to the image-based feature explanations of the prior work.

Interestingly, we found differences in the explanation mechanism ratings based on the environment tested. Before completing the survey, we asked users to self-rate their prior experience in the three environments, using "Never played and have no idea how the game works", "Have played once or twice and have minimal knowledge of the game", and "Have played and understand the game well" with ratings 1, 2 and 3 respectively. We found that, on average, users had a score of 2.5 and 2.3 for Breakout and Space Invaders, respectively, while Seaquest had a score of 1.3, indicating that participants had generally strong prior knowledge of Breakout and Space Invaders while few had prior knowledge of Seaquest. As shown in Figure 4.5, despite the majority of users having minimal prior knowledge of the Seaquest environment, we found that user ratings for our temporal explanation had the smallest average difference between

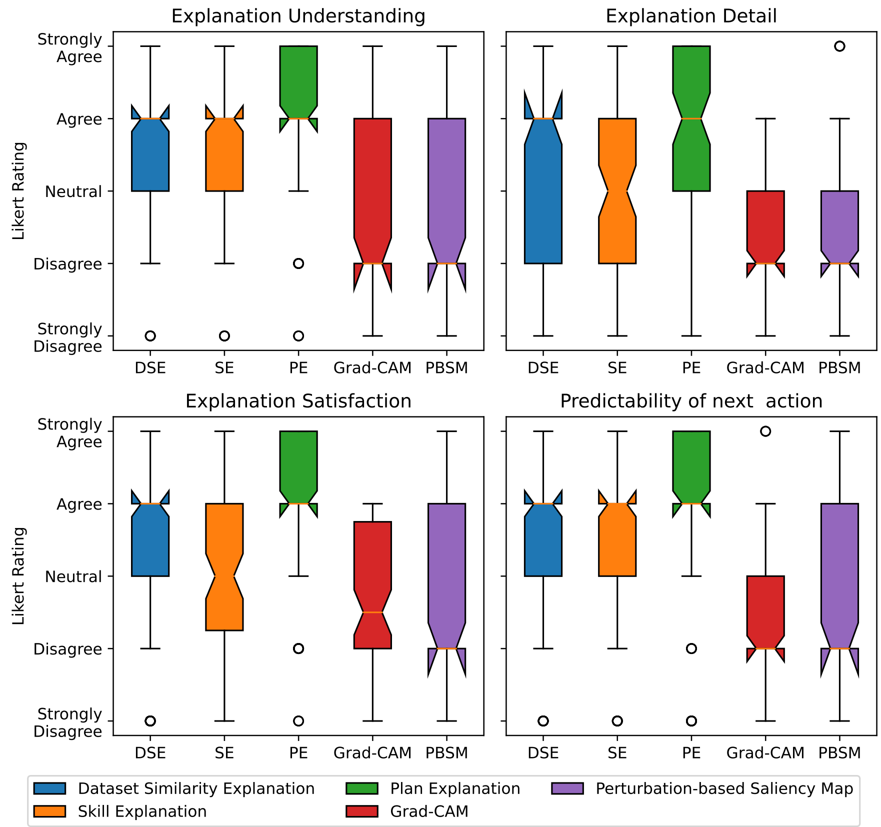


FIGURE 4.4: Box plot of the user ratings of each explanation mechanism and question as part of the individual evaluation section in the user survey.

Breakout or Space Invaders and Seaquest (-0.15, $+0.25$ and -0.1 for DSE, SE and PE respectively). In comparison, Grad-CAM and PBSM had the largest average decrease from Breakout or Space Invaders to Seaquest of -0.2 and -0.25, respectively. We hypothesise this performance difference is due to the saliency maps requiring users to understand why an agent focuses on specific observation features, requiring prior knowledge of the environment, without which the explanation is more difficult to understand. These findings indicate that our temporal explanations can be equally effective regardless of the users' prior knowledge of the environment. Furthermore, it suggests that while Grad-CAM and PBSM do not need domain knowledge to generate explanations, users still need familiarity with the environment to comprehend the explanations. Further work is required to understand if this phenomenon is reproducible for more explanation mechanisms and environments.

In Figure 4.6, we present the results from contrastive questions, with each cell giving the percentage of users who explicitly prefer the row explanation over the column explanation. For example, with respect to the two benchmarks, Grad-CAM and PBSM, 67% of users prefer Grad-CAM over PBSM, and 17% of users expressed the opposite preference, with 16% of users expressing no preference. Similarly, 50% of users were

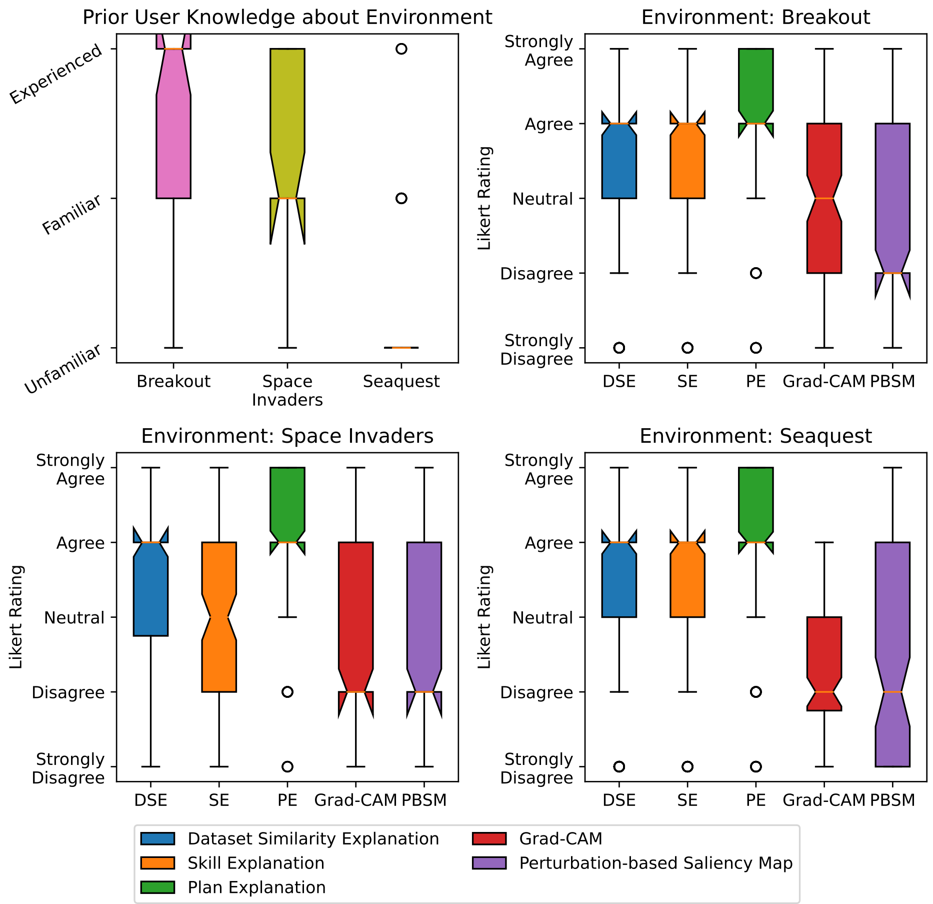


FIGURE 4.5: A boxplot of prior environment knowledge and per environment explanation ratings.

more satisfied with Grad-CAM explanations than PBSM explanations, with 17% of users more satisfied with PBSM and a third of users expressing no preference.

With respect to our temporal explanations, we observe that most participants had an equal preference for DSE and SE, with only 17% of users *explicitly* preferring one explanation over the other (except for Explanation understanding with 33% of users preferring DSE to SE). In comparison, 67% of users explicitly preferred PE over DSE, a 50% improvement over SE where the sole difference is due to the inclusion of domain knowledge. Comparing our temporal explanations to the prior feature-based explanations, we find that they are explicitly preferred 63%, 87% and 100% (DSE, SE and PE, respectively) for an average of 83.3% over Grad-CAM and PBSM. While there is minimal preference for DSE over SE, we do find that user preference for DSE or SE for the feature-based explanation does increase, indicating that the skill-based approach appears to have inherent advantages. Additionally, the inclusion of domain knowledge from SE to PE has a similar increase in performance, recognising that users prefer it when domain knowledge is included. Lastly, comparing the feature-based explanations, we find that Grad-CAM is explicitly preferred on average 63% to DSE while vice versa, DSE is preferred 17% on average. This is despite PBSM being specifically designed for

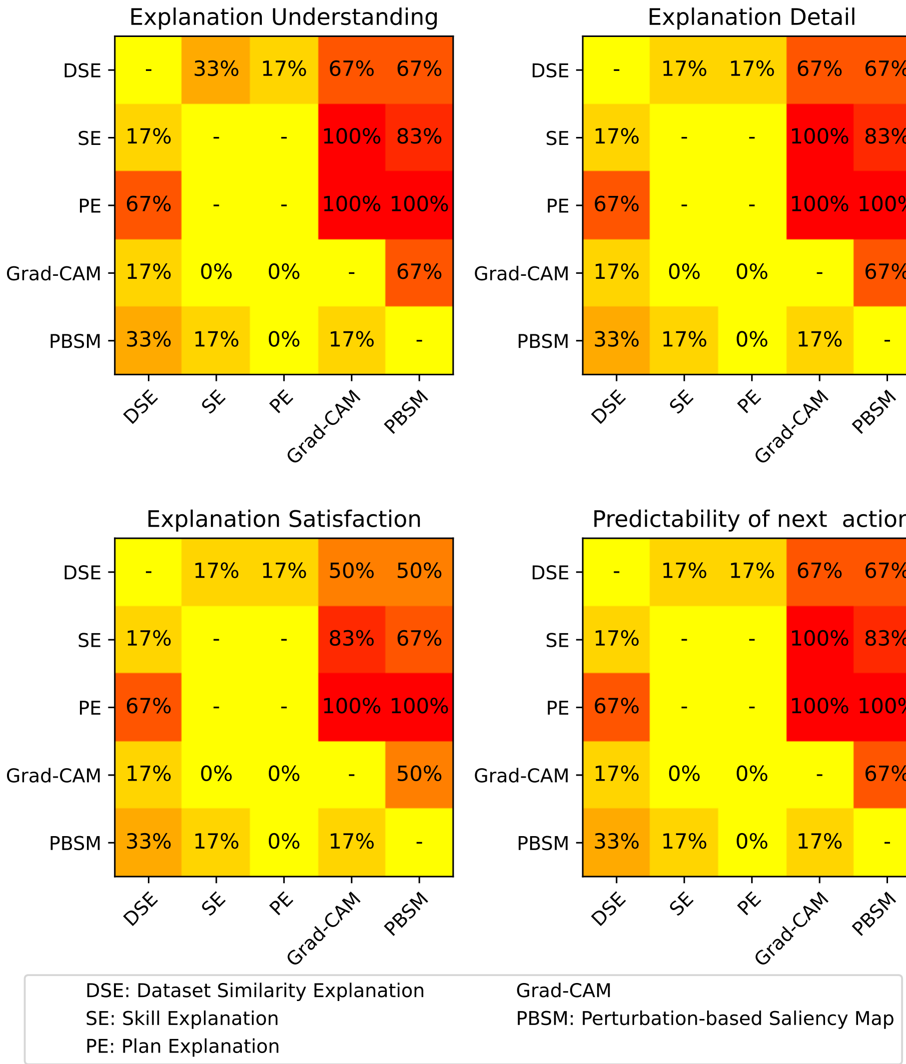


FIGURE 4.6: Heatmap for each statement of the user preference for each explanation mechanism. Grid elements are the percentage of users who explicitly prefer the row explanation mechanism over the column explanation mechanism.

RL agents, while Grad-CAM was originally designed to explain image recognition neural networks. In summary, these contrastive evaluations reinforce our finding from the individual evaluation that our novel temporal explanations are significantly preferred over the tested prior feature-based explanation mechanisms.

Finally, we analysed the variances of user ratings for each explanation mechanism to see if users had significantly different answers based on the statements or explanations presented. In Figure 4.7, we present a stacked histogram of the user rating variance for each explanation mechanism irrespective of the question asked. For each explanation mechanism, we found, on average, 79% of users' ratings stayed within 1 rating for each explanation mechanism, showing that user opinion was consistent given an observation/explanation for each mechanism. Of the explanation mechanisms, we found that our novel Dataset Similarity, Skill and Plan Explanations had the largest

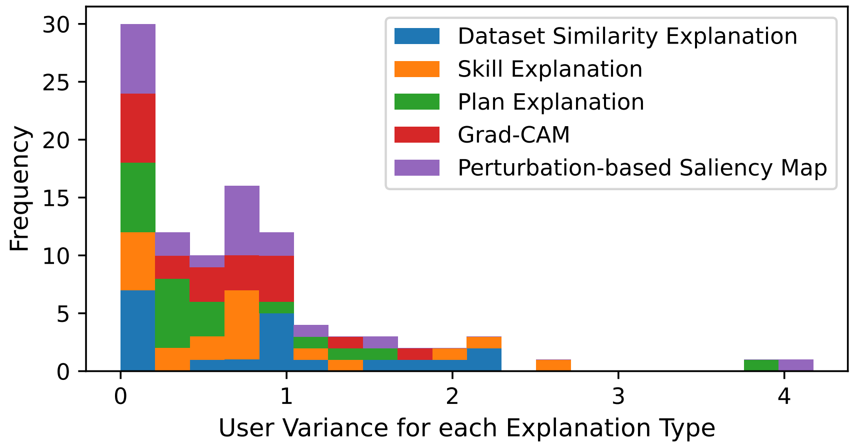


FIGURE 4.7: A stacked histogram of per-user variance in the Explanation type ratings.

variances, with the variance at the 80th percentile being 1.65, 1.17 and 1.01 respectively. The Grad-CAM and Perturbation-based Saliency Map had a variance of 0.89 and 0.88 at the 80th percentile. This shows that the users were consistent in their lower rating of the feature-based explanations regardless of observation, whereas our temporal explanations were more variable in rating depending on observation. Further work is required to understand the reason for these variances.

4.3.2 Skill Extraction Analysis

As our Skill and Plan Explanation requires a dataset of skills, we investigated the extracted skills from Zahavy et al. (2016), referred to as algorithmically labelled, using our two novel skill similarity metrics, Skill Alignment and Skill Distribution defined in Section 4.2. Using an Atari Breakout Rainbow agent, we extracted four episodes worth of skills and additionally hand-labelled the episodes as a baseline comparison using the following skills: "Skill 0 - Agent isn't moving, waiting for the ball to get lower"; "Skill 1 - Agent moves left to hit the ball"; or "Skill 2 - Agent moves right to hit the ball". These skill labels were selected from previously viewing the episodes and summarising the agent's simple behaviour of sticking to the left or right of the screen until the ball moved low enough to hit.

For well-differentiated skills, we expect our Normalised Skill Alignment Metric (Section 4.2.1) to find higher alignment given two instances of the same skill type compared to different types. In Figure 4.8, we plot the mean Normalised Skill Alignment for 200 randomly selected instances of the column and row skill type for both the hand and algorithmically labelled skills. We found that hand-labelled skills exhibit this expected feature where the diagonal (same skills) has higher alignment than the rest of the matrix (the alignment between different skills). In comparison, such a feature does not exist for the algorithmically labelled skill, with only Skill 1 presenting an above-average skill

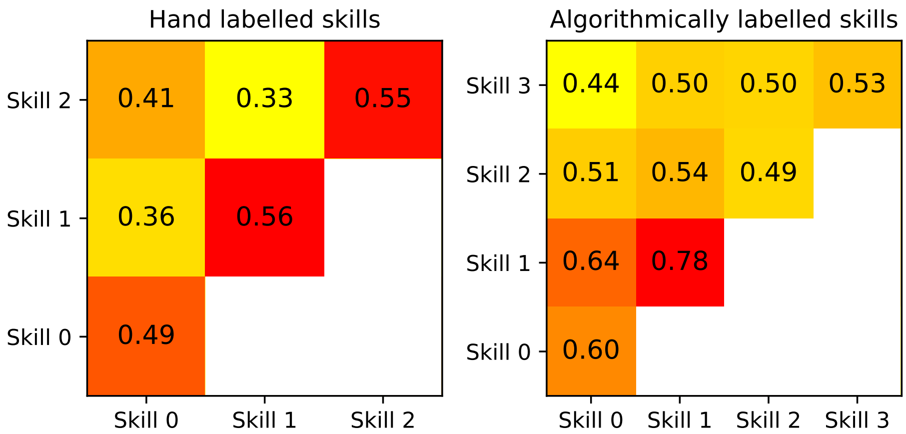


FIGURE 4.8: The skill alignment of two sets of skills for the hand-labelled and algorithmically-labelled trajectories.

alignment of 0.78 compared to the average of 0.52. This implies that algorithmically labelled skills 0, 2 and 3 all lacked a distinctive subsequence of common actions compared to the other skills and, possibly, a common purpose. This is partially unsurprising as Zahavy et al. (2016)'s clustering algorithm does not consider the actions taken by the agent.

In common with the Skill Alignment metric, for the Skill Length Distribution (Section 4.2.2), using well-defined skills, we expected each skill to have a similar length. Figure 4.9 presents a stacked histogram of skill lengths for the hand and algorithmically labelled skills along with the optimal Poisson distribution for the Skill Length Distribution metric. We again found that the algorithmically labelled skills did not follow the expected Poisson distribution largely due to the range of skill lengths, with some being just a single action while others being several hundred. In comparison, we found that the distribution of hand-labelled skill lengths more closely matched the expected Poisson distribution.

Using our two novel skill metrics, we found that the algorithmically labelled skills often lacked a common sequence of actions or pattern in their length, particularly when compared to human-labelled skills. Therefore, further work should investigate the effectiveness of alternative skill discovery algorithms using our metrics.

4.4 Discussion

In this chapter, we developed novel explanation mechanisms for understanding and visualising agent behaviour and future decision paths based on a dataset of prior collected observation-action. Two mechanisms do not utilise domain knowledge, while

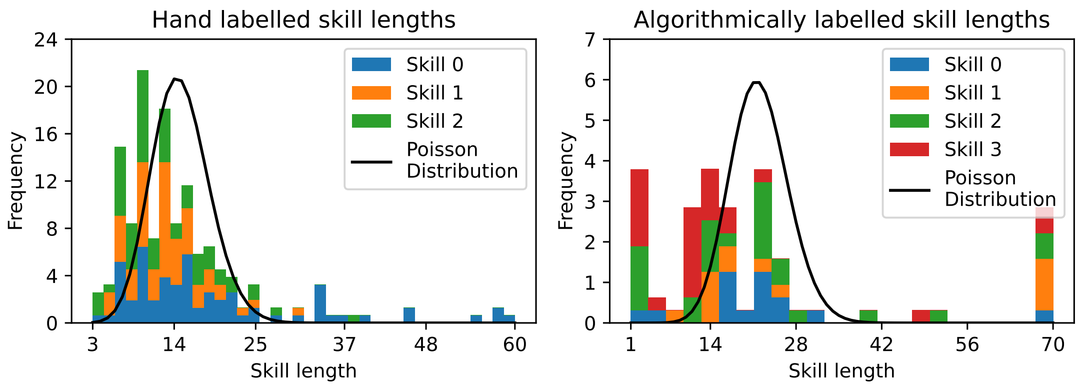


FIGURE 4.9: A stacked histogram of skill lengths of hand-labelled and algorithmically discovered skills. The skill lengths were clipped to 70.

the third requires knowledge only for each agent's skills (Section 4.1). An additional impact is that these are one of the first RL explanations designed around videos as the medium to communicate an agent's decision-making. These explanations are demonstrated in the Atari 2600 environments, and we anticipate that the approach will be scalable to robotics or more complex domains such as chess with minimal changes. One of these changes is modifying the embedding encoder to an Autocoder (Bank et al., 2023) that learns the vital data from observations or neural network activations rather than filtering using PCA and embedding with t-SNE, as in this chapter. We utilise this approach in Chapter 6. Furthermore, we propose two novel skill metrics for measuring the coherence of agent skills, as we assume that explaining an agent's skill requires them to be consistent and intelligible (Section 4.2). Importantly, we found that using these metrics for algorithmically discovered skills, using Zahavy et al. (2016), lacked the coherence of skills labelled by a domain expert (Section 4.3.2).

Finally, in this chapter, from our user evaluation, we find that users rated all our proposed explanations higher across four areas (understanding, detail, satisfaction, and prediction) than two feature-based explanations. Further, when asked to explicitly select the preferred explanations for an observation, users preferred our explanation on average 83.3% over the feature-based explanations. This was raised to 100% for the plan explanations against both feature-based explanations tested.

Despite these remarkable user preferences in Figures 4.4 and 4.6, it is unclear how to unify the results from Sections 4.3.1 and 4.3.2. In Section 4.3.1, we find that users rate our explanations, in particular Skill and Plan Explanations, above the feature-based explanations. In Section 4.3.2, however, we find that the skills discovered lacked coherence compared to human-labelled skills. We hypothesise two reasons for the difference in results. First, with the significant difference in what the explanation methods describe, feature-based explanations highlight an agent's focus while the temporal explanation showcases the possible near future decisions of the agent, users might significantly prefer this future knowledge of the agent's observational importance. This means that even if the temporal explanations had flaws, users still preferred them over the feature-based explanations. For example, when validating the feature-based explanation implementations, we found that the saliency-map image could be unintelligible, with seemingly unimportant areas of observation for humans being highlighted or viewed as highly important. In contrast, our temporal explanation videos are significantly more straightforward to understand, demonstrating what the agent has previously done. Secondly, the user survey conducted only used self-reported/subjective questions, i.e., user preference or understanding, without any questions that validate if users actually understood or could use the explanations to do something. This lack of an objective question means we can only infer that users significantly preferred our temporal explanations, not that the explanations are actually more effective. This limitation of the user survey is rectified in Chapter 6, where users are asked to use various explanations to predict known objective data about several agents, for which we find the Dataset Similarity Explanation still highly effective.

Chapter 5

Explaining an Agent's Future Beliefs through Temporally Decomposing Future Reward Estimators

While found to be effective in a user survey, the video-based explanations of Chapter 4 required extracting skills from pre-trained agents, which we found lacked coherence and consistency compared to human-extracted skills (Section 4.3.2). This highlights one of the weaknesses of post-hoc explanations that neural networks, when not explicitly trained on a task, i.e., learning discrete skills, extracting such information is highly complex and difficult to confirm truthfulness to the original model's decision making. In this Chapter, we investigated how agent training and architecture can be modified to explicitly output temporal information about the agent's decision-making that can be explained, avoiding the problems associated with post-hoc explanations.

Central to RL agents is a future reward estimator (Q-value, Eq. (2.2) or state-value function, Eq. (2.3)) predicting the sum of future rewards for a given state. These functions are used either explicitly in the policy itself (e.g., DQN (Mnih et al., 2015)) or for learning with a critic (e.g., PPO (Schulman et al., 2017) and TD3 (Fujimoto et al., 2018)). However, few XRL algorithms have devised methods to explain these functions directly. One problem is that their scalar outputs provide no information on its composition (i.e., when and what future rewards the agent believes it will receive), just its expected cumulative sum.

An example of this problem is illustrated in Figure 5.1, where a drone has two paths: up or down. Depending on the path taken, the drone can receive coins for 1 point each or the treasure chest for 4 points. Using a discount factor of 0.95, the drone's Q-value for moving up is 3.26 while moving down is 3.52. Despite this small difference in Q-values, the quantity and temporal immediacy of these expected rewards are radically different; moving up, the drone receives a single large reward, while moving down receives many

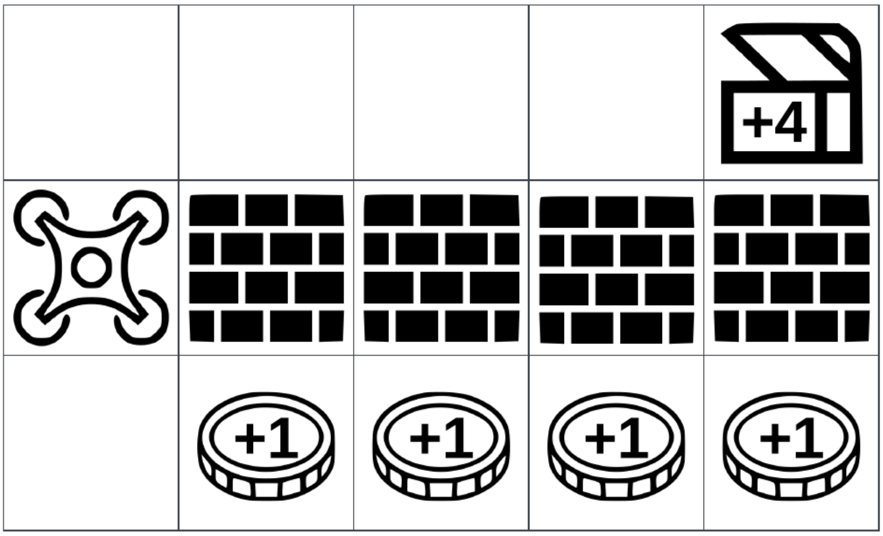


FIGURE 5.1: Example Gridworld with an agent and two paths (up and down) that contain different rewards.

smaller rewards. A fact unknown from observing the Q-values alone, although critical to agent behaviour in selecting whether to move up or down and importantly is impossible to diagnose in more complex environments, e.g., Atari or Robotics.

To solve this, we propose a novel future reward estimator that predicts the agent's next N expected rewards for a given state, referred to as Temporal Reward Decomposition (TRD) (Section 5.1). We prove that TRD is equivalent to the scalar Q-value and state-value functions. In this way, TRD can report the temporal immediacy and quantity of future rewards for different action choices, enabling decisions to be explained and contrasted. For example, using Figure 5.1, the agent's TRD Q-value for moving down is $[0, 0.95, 0.90, 0.86, 0.81]$ and moving up is $[0, 0, 0, 0, 3.26]$, enabling us to produce explanations such as "while the sum of actual rewards is equal, taking the route down has more immediate rewards, which are preferred by the drone due to its discount factor".

Implementing TRD requires only two changes to a deep RL agent's future reward estimator: increase the network output by $N{+}1$ for predicting the future rewards and a novel element-wise loss function of future rewards (Section 5.1). Importantly, TRD can achieve similar performance as DQN (Mnih et al., 2015) agents for Atari environments (Bellemare et al., 2013) across a wide range of N (Section 5.2).

Building on this direct access to an agent's predictions for individual future rewards, we explore three novel applications for understanding an agent's decision-making (Section 5.3). The first is to generate explanations for an action choice based on when and what rewards it will receive and, for particular environments, the agent's confidence in collecting a reward (Section 5.3.1). Importantly, as TRD exposes the agent's predictions of the future, we refer to these as future beliefs or confidence of collecting particular individual rewards. The second explains how the importance of an observation feature changes depending on how far into the future a reward is expected, e.g., identifying features that are more relevant for earlier rewards (Section 5.3.2). Thirdly, we produce contrastive explanations using the difference in future expected rewards for two actions, which can reveal changes in expected rewards the agent will receive between them (Section 5.3.3). Together, these three explanation mechanisms demonstrate the value of Temporal Reward Decomposition for XRL.

In summary, this chapter contributes the following to the state-of-the-art in XRL:

Temporally Decomposed Future Reward Estimator (Section 5.1) - We propose a novel future reward estimator that outputs the expected rewards for the next N timesteps (Eq. (5.6)). Using the Linearity of Expectation, we prove our decomposed future reward estimator is equivalent to the standard scalar estimator. We propose a novel loss function to learn the estimator using an element-wise mean-squared error.

Scaling TRD to Atari environments (Section 5.2) - We show that pretrained DQN agents for Atari environments can be efficiently retrained to incorporate TRD with minimal impact on agent performance. Testing across a range of hyperparameters for the number of decomposed timesteps and timestep widths demonstrates neither has a significant impact on final agent performance. Additionally, we find that the retrained agent learns Q-values similar to those of the original agent.

Explaining Decomposed Future Rewards (Section 5.3) - We propose three explanation approaches that utilise TRD's decomposed rewards. First, we show how TRD explicitly provides the agent's understanding of when and what quantity rewards it will receive and their confidence in them. Second, by adapting prior saliency map algorithms, we show that an observation feature has temporal importance, such that its influence on an agent's decision-making varies depending on the temporal immediacy of the predicted reward. Third, we present a contrastive explanation for two actions to highlight the difference in when and what rewards the agent expects to receive.

5.1 Temporally Decomposed Future Reward Estimators

As illustrated in Figure 5.1, due to the scalar output of future reward estimators (i.e., Q-value and state-value functions), their reward composition cannot be known, preventing understanding when and what future rewards the agent expects to receive. We, therefore, propose a novel future reward estimator (Eqs. (5.6)), referred to as Temporal Reward Decomposition (TRD), that predicts an agent's next N expected rewards. Furthermore, we prove its equivalence to scalar future reward estimators and provide a bootstrap-based loss function to learn the estimator (Eq. (5.14)). For consistency, all equations in this section are for the Q-value with TRD-based state-value equations in Appendix A.

Before defining our TRD-based future reward estimators, to prove their equivalence to scalar future reward estimators (Eq. (5.7)), we first prove that the expected sum of future rewards is equivalent to the sum of expected future rewards enabling the decomposition of rewards in Eq. (5.6): Theorem 5.1.[1] This theorem swaps the position of the summation and expectation operators ($\sum$ and $\mathbb{E}$), proving that rather than learning the expected sum of rewards for all time (Eq. (2.2)), this equation can be decomposed. We rely on this theorem for Eqs. (5.7) and (5.9) to prove that our novel functions are equivalent to standard scalar Q-values, preserving any properties of the original agent to be explained.

Theorem 5.1. *Given a state s and action a, the expected sum of rewards is equal to the sum of expected rewards, more precisely* $\mathbb{E}_\pi\left[\sum_{i=0}^{\infty}\gamma^i R_{t+i}|S_t = s, A_t = a\right] \equiv \sum_{i=0}^{\infty}\mathbb{E}_\pi[\gamma^i R_{t+i}|S_t = s, A_t = a]$ *for all $s \in S$ and $a \in A$.*

Proof.

$$\mathbb{E}_\pi\left[\sum_{i=0}^{\infty}\gamma^i R_{t+i}\middle|S_t = s, A_t = a\right] \tag{5.1}$$

$$=\mathbb{E}_\pi\left[R_t + \sum_{i=1}^{\infty}\gamma^i R_{t+i}\middle|S_t = s, A_t = a\right] \tag{5.2}$$

$$=\mathbb{E}_\pi[R_t|S_t = s, A_t = a] + \mathbb{E}_\pi\left[\sum_{i=1}^{\infty}\gamma^i R_{t+i}\middle|S_t = s, A_t = a\right] \quad \text{(given LoE}^1\text{)} \tag{5.3}$$

$$\begin{aligned}=&\mathbb{E}_\pi[R_t|S_t = s, A_t = a] + \mathbb{E}_\pi[\gamma R_{t+1}|S_t = s, A_t = a]\\ &+ \mathbb{E}_\pi\left[\sum_{i=2}^{\infty}\gamma^i R_{t+i}\middle|S_t = s, A_t = a\right]\end{aligned} \tag{5.4}$$

$$=\sum_{i=0}^{\infty}\mathbb{E}_\pi[\gamma^i R_{t+i}|S_t = s, A_t = a] \tag{5.5}$$

□

Using the notation in Section 2.1, we propose Eq. (5.6) that outputs a vector of the next N expected rewards with the last element being equal to the cumulative sum of expected rewards from N to ∞. Each element i refers to the expected reward in $t+i$ timesteps, with the final element being the sum of rewards beyond N timesteps. Using Theorem 5.1, Eq. (5.6) is provably equivalent to the scalar Q-value by summing over the

[1] Linearity of Expectation (LoE) is a property that any expectation can be split into its linear components, even for *dependent* random variables (Stirzaker, 2003, Page 166).

array elements (Eq. (5.7)) through expanding Eq. (5.4) with $N+1$ expectations. Critically, this equivalence is not reversible such that given a scalar Q-value, Eq. (5.6) cannot be known.

Implementing TRD within a deep RL agent's future reward estimator requires two primary changes. The first is increasing the neural network output by $N+1$; i.e., the size of Eq. (5.6) for predicting the next N future rewards. The second is the loss function (Eq. (5.14)) for the network to learn Eq. (5.6). Additionally, as the network now outputs a vector of future rewards rather than a scalar, for action selection and other applications, q_π can be recovered by summing across vector elements before being applied as normal. We provide a single-file implementation of a DQN-modified TRD training algorithm using Gymnasium (Towers et al., 2024c) on Github[2].

$$q_\pi^{\text{TRD}}(s,a) = \begin{pmatrix} \mathbb{E}_\pi[R_t | S_t = s, A_t = a] \\ \mathbb{E}_\pi[\gamma R_{t+1} | S_t = s, A_t = a] \\ \vdots \\ \mathbb{E}_\pi[\gamma^{N-1} R_{t+N-1} | S_t = s, A_t = a] \\ \mathbb{E}_\pi\left[\sum_{i=N}^{\infty} \gamma^i R_{t+i} | S_t = s, A_t = a\right] \end{pmatrix} \tag{5.6}$$

$$\sum q_\pi^{\text{TRD}}(s,a) \equiv q_\pi(s,a) \qquad \forall s \in S, \forall a \in A \tag{5.7}$$

For long-horizon environments where an agent may take hundreds or thousands of actions, TRD is limited in scale as the number of predicted rewards scales linearly with the number of output neurons. We, therefore, propose an alternative approach to preserve the temporal distance that can be explained using a fixed number of output neurons. Rather than each vector element predicting an individual reward, Eq. (5.8) groups rewards in each vector element; e.g., for pair grouping $[R_t + R_{t+1}, R_{t+2} + R_{t+3}, \dots]$. This approach, denoted w for the reward grouping size, scales linearly with the number of future rewards by w for a fixed N such that the total number of predicted rewards $N \cdot w$. Importantly, like Eq. (5.6), Eq. (5.8) is equivalent to the Q-value by summing across elements (Eq. (5.9)) using $N \cdot w + 1$ expansions of Eq. (5.4). Additionally, for $w = 1$, Eq. (5.8) is equivalent to Eq. (5.6) and implementation only requires utilising an N-step (Sutton and Barto, 2018) experience replay buffer to compute the sum of the first w rewards and the next observation in w timesteps.

[2]https://github.com/pseudo-rnd-thoughts/temporal-reward-decomposition

$$q_\pi^{\text{TRD}}(s,a) = \begin{pmatrix} \mathbb{E}_\pi[R_t|S_t = s, A_t = a] + \cdots + \mathbb{E}_\pi[\gamma^{w-1}R_{t+w-1}|S_t = s, A_t = a] \\ \mathbb{E}_\pi[\gamma^{w}R_{t+w}|S_t = s, A_t = a] + \cdots + \mathbb{E}_\pi[\gamma^{2w-1}R_{t+2w-1}|S_t = s, A_t = a] \\ \vdots \\ \sum_{i=(N-1)w}^{Nw} \mathbb{E}_\pi[\gamma^{i}R_{t+i}|S_t = s, A_t = a] \\ \mathbb{E}_\pi[\sum_{i=Nw}^{\infty} \gamma^{i}R_{t+i}|S_t = s, A_t = a] \end{pmatrix} \tag{5.8}$$

$$\sum q_\pi^{\text{TRD}}(s,a) \equiv q_\pi(s,a) \qquad \forall s \in S, \forall a \in A \tag{5.9}$$

As a result, N and w present a trade-off between the reward vector size (N) and precise knowledge of each timestep's expected reward (w). For example using Figure 5.1, if $w = 2$ and $N = 2$ then the q_π^{TRD} for moving up is $[0, 0, 3.26]$ as $[0+0, 0+0, 3.26]$ and moving down is $[0.95, 1.76, 0.81]$ as $[0+0.95, 0.90+0.86, 0.81]$. Furthermore, to predict, for example, the next 30 rewards, $N = 30, w = 1$ and $N = 6, w = 5$ are both valid parameters. We explore the impact of these parameters on training in Section 5.2.

Through experiment, we found that converting q_π^{TRD} to q_π by summing over elements (Eq. (5.7)), then using the scalar loss function (Eq. (2.7)) does not converge to q_π^{TRD}. Therefore, based on the Q-learning loss function (Eq. (2.7)), we define a novel element-wise mean squared error of reward vectors (Eq. (5.14)) where a' denotes the optimal next action ($\arg\max_{a\in A} \sum q_\pi^{\text{TRD}}(s_{t+w}, a)$) and we use the following notation to index an element of the reward vector:

$$q_\pi^{\text{TRD}_0}(s,a) = \mathbb{E}_\pi[R_t|S_t = s, A_t = a] \tag{5.10}$$
$$q_\pi^{\text{TRD}_1}(s,a) = \mathbb{E}_\pi[\gamma R_{t+1}|S_t = s, A_t = a] \tag{5.11}$$
$$q_\pi^{\text{TRD}_N}(s,a) = \mathbb{E}_\pi[\gamma^{N-1} R_{t+N-1}|S_t = s, A_t = a] \tag{5.12}$$
$$q_\pi^{\text{TRD}_{N+1}}(s,a) = \mathbb{E}_\pi\left[\sum_{i=N}^{\infty} \gamma^{i} R_{t+i}|S_t = s, A_t = a\right] \tag{5.13}$$

For Eq. (5.14), we construct a predicted and bootstrap-based target value (cf. Q-learning), computing the element-wise mean squared error of the predicted and target reward vectors. The prediction is the reward vector for the action taken in state t, $q_\pi^{\text{TRD}}(s_t, a_t)$. For the target, the first element is the actual reward collected (R_t to R_{t+w}) followed by the reward vector for the optimal action in s_{t+w}, $q_\pi^{\text{TRD}}(s_{t+w}, a')$, shifted along one position with the last two elements combined. We do this because element i of the reward vector, $q_\pi^{\text{TRD}_i}(s_t, a_t)$, refers to the predicted reward in $t+i$ timesteps, for the next observation, $t+w$, the equivalent reward vector element is $i-1$ in the target vector, $\gamma q_\pi^{\text{TRD}_{i-1}}(s_{t+w}, a')$.

$$L_{\text{TRD}} = \mathbb{E}_{(s_t, a_t, R_{t+i}, s_{t+w}) \sim D} \begin{bmatrix} \left(q_\pi^{\text{TRD}_0}(s_t, a_t) - \sum_{i=0}^{w} R_{t+i}\right)^2 \\ \left(q_\pi^{\text{TRD}_1}(s_t, a_t) - \gamma^w q_\pi^{\text{TRD}_0}(s_{t+w}, a')\right)^2 \\ \left(q_\pi^{\text{TRD}_2}(s_t, a_t) - \gamma^w q_\pi^{\text{TRD}_1}(s_{t+w}, a')\right)^2 \\ \vdots \\ \left(q_\pi^{\text{TRD}_N}(s_t, a_t) - \gamma^w q_\pi^{\text{TRD}_{N-1}}(s_{t+w}, a')\right)^2 \\ \left(q_\pi^{\text{TRD}_{N+1}}(s_t, a_t) - \gamma^w (q_\pi^{\text{TRD}_N}(s_{t+w}, a') + q_\pi^{\text{TRD}_{N+1}}(s_{t+w}, a'))\right)^2 \end{bmatrix} \tag{5.14}$$

Importantly, this loss function can be efficiently implemented into code with our example pseudocode in Appendix D.

5.2 Retraining Pretrained Agents for TRD

The goal of Temporal Reward Decomposition (TRD) is to provide information about an agent's future expected rewards over time so that we can use this information to better understand its behaviour. For this to be practically effective, TRD agents should be capable of achieving performance similar to the associated RL agents. In this section, we evaluate the performance of DQN agents (Mnih et al., 2015) that incorporate TRD for a range of Atari environments (Bellemare et al., 2013) and assess the impact of TRD's two hyperparameters on training: reward vector size, N and reward grouping, w.

We conduct hyperparameter sweeps across each independently, varying N, w, and $N \cdot w$, across three Atari environments (Breakout, Space Invaders and Ms. Pacman), each containing different reward functions. To account for variability in training, we repeat our hyperparameter sweeps three times. The training hyperparameters and hardware used in training, along with the agent's final scores, are presented in Appendix B. Rather than training agents from scratch for these environments, we use open-sourced pretrained Atari agents (Huang et al., 2023) and the QDagger training workflow (Agarwal et al., 2022) described in Section 2.2. Training scripts and final neural network weights for all runs are provided in the associated GitHub repository.

Using periodic evaluation on the same ten seeds, Figure 5.2 plots the teacher normalised interquartile mean (Agarwal et al., 2021) of the episodic reward. We find that all three hyperparameter sweeps enable the agent to approach the pretrained (teacher) agent's score, with neither parameter having a significant detrimental impact. Only the offline training for a constant temporal distance ($N \cdot w = 24$) does agents with smaller values of w show a greater initial performance, however, this difference is resolved during the online training stage.

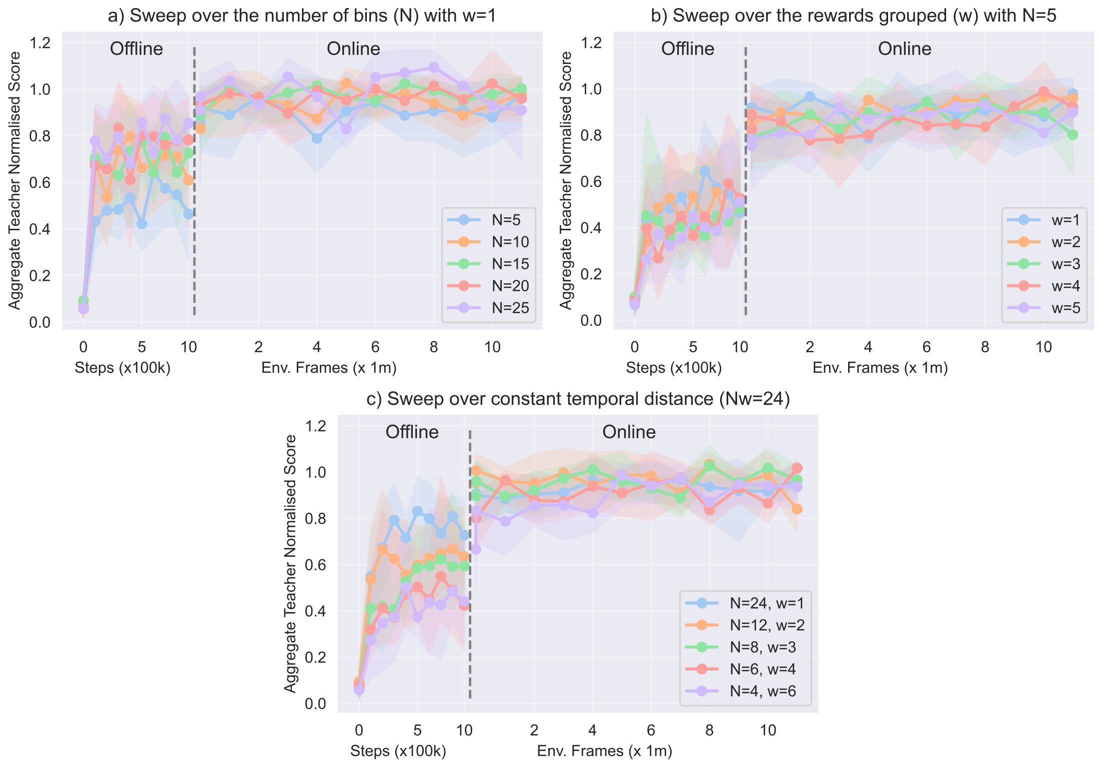


FIGURE 5.2: Interquantile mean training curves for Atari TRD-DQN agents for three environments (Breakout, Space Invaders, and Ms Pacman) with three repeats, normalised by the teacher's score. Offline and Online indicate where training used the offline replay buffer and the online environment steps.

Interestingly, for the sweep of N, we did not find any degradation in performance, which was unexpected as we believed that larger values of N would require more training to reach the same performance. As a result in Section 5.3, we trained agents with $N{=}40, w{=}1$. Further work is required to understand if these performance curves hold for larger values of N and more complex environments or agents.

To verify that our TRD loss function (Eq. (5.14)) converges to a policy similar to the pretrained agent's scalar Q-value. Figure 5.3 plots the mean squared error between the pretrained DQN agent's and TRD agent's Q-values during training. We find all parameters get close to the pretrained agent's Q-value with $w{=}1$ being an important factor.

Regarding the computation impact of incorporating TRD, we found that our QDagger+TRD DQN agents took $\approx$ 10% fewer steps per second than our base DQN agents, 248 to 274 steps per second, respectively. This performance will be jointly caused by QDagger requiring an additional forward pass from the teacher agent and TRD using a larger network output and a more complex loss function.

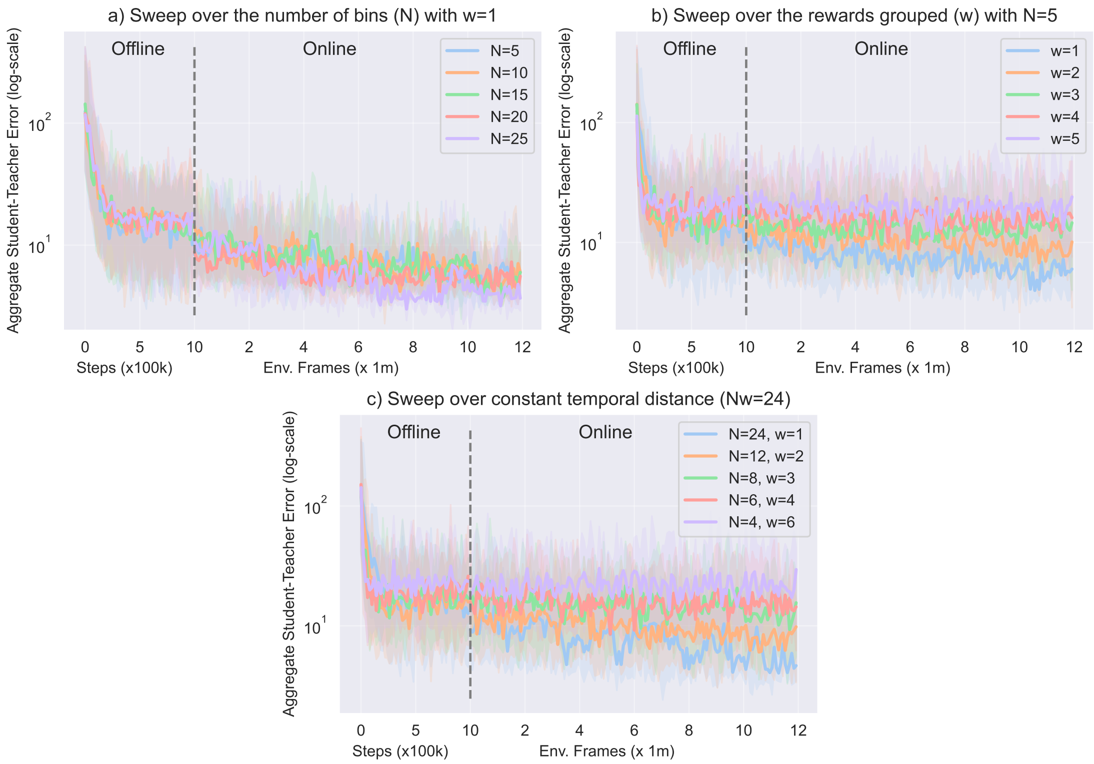


FIGURE 5.3: The Mean Squared Error between the student (TRD agent) and pretrained teacher agent averaged over three Atari environments with three repeats. Offline and Online indicate where training used the offline replay buffer and the online environment steps.

5.3 Explaining an Agent's Future Beliefs and Decision-Making

Utilising TRD's access to the agent's future rewards, in this Section, we explore how this information can be used to inform our understanding of the agent's beliefs about the future for a given observation, i.e., their future beliefs. We present three novel explanation mechanisms using TRD's future expected rewards: understanding what rewards the agent expects to receive and when, and their confidence in this prediction; visualising an observation feature's importance for predicting rewards at near and far timesteps; and a contrastive explanation using the difference in future rewards to understand the impact of different action choices (Sections 5.3.1, 5.3.2, and 5.3.3 respectively). We showcase these applications using three different Atari environments with more examples in Appendices C, D, and E. All agents were retrained DQN agents incorporating TRD using N=40 and w=1.

5.3.1 What Rewards to Expect and When?

For environments with complex transition dynamics or reward functions such as Atari, understanding how an agent maximises its future rewards or predicting what rewards it will receive and when is impossible, unlike with the toy example illustrated in Figure 5.1. We show here how a TRD agent's predicted future rewards supply this information, presenting an important explanatory perspective for understanding agent decisions. Furthermore, for environments with binary reward functions (i.e., where the rewards are either zero or a constant value), the agent's expected reward can be further decomposed into the probability of the reward function components. Atari uses integer rewards, and DQN agents clip rewards to -1 to 1, so for these examples, the agent's probability of collecting a reward is equivalent to the reward's expectation.

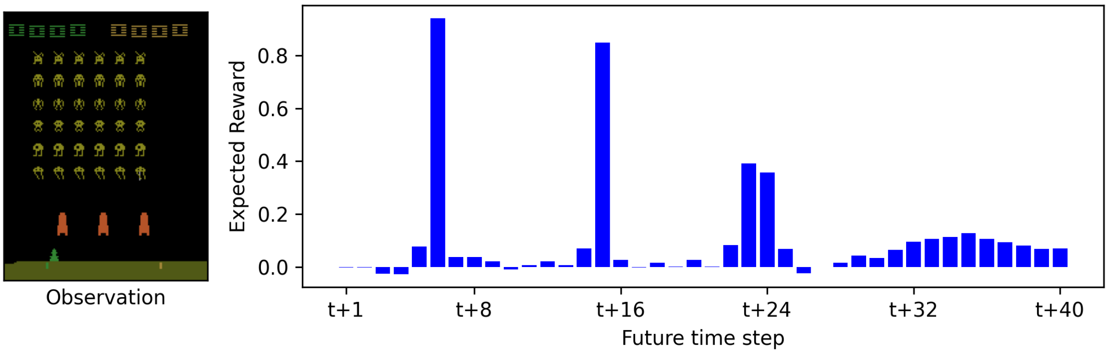


FIGURE 5.4: A Space Invaders observation (left) with the respective predicted next 40 future expected rewards (right)

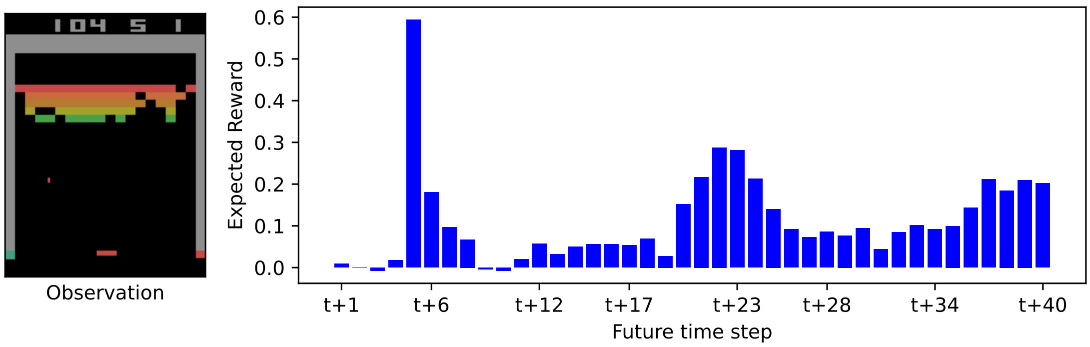


FIGURE 5.5: A Breakout observation (left) with the respective predicted next 40 future expected rewards (right)

Figures 5.4 and 5.5 plot the agent's expected rewards over the next 40 timesteps for the observation on the left. As w=1, the discount factor is constant for each predicted timestep, and so we factor it out, leaving just the expected reward. Without domain knowledge of each environment and its reward functions, we can observe from the expected rewards plots that the agent expects periodic non-zero rewards every 8 to 9 timesteps for Space Invaders and every 15 timesteps for Breakout. Additionally, considering that the expected rewards (for these environments) are equivalent to the agent's confidence (probability) in receiving a reward for a particular timestep, users can infer that the agent's confidence reduces over time for the specific timestep that the agent will receive a reward. As such, for space invaders, the agent has high confidence for the close timesteps (t+6 and t+15), with the expected rewards for the third and fourth rewards being distributed across several timesteps (t+23 to t+24 and t+30 to t+40).

Further, utilising domain knowledge of each environment, Figures 5.4 and 5.5 correlate with our understanding as agents can only shoot aliens or break bricks periodically. Additionally, as the policy is stochastic due to epsilon-greedy action selection and with randomness in the environment, the uncertainty of timesteps far in the future is unsurprising and matches with human expectations.

Building on the two figures, we can generate videos of the agent's expected rewards across entire episodes, plotting the expected reward for each observation. For the Atari environments tested, we find that future rewards move like waves towards the current timestep, demonstrating the agent's clear understanding of future rewards. Example videos are provided in the associated GitHub repository and contain significant additional context for users to visualise how the agent's predicted future rewards change over time as the environment's state evolves.

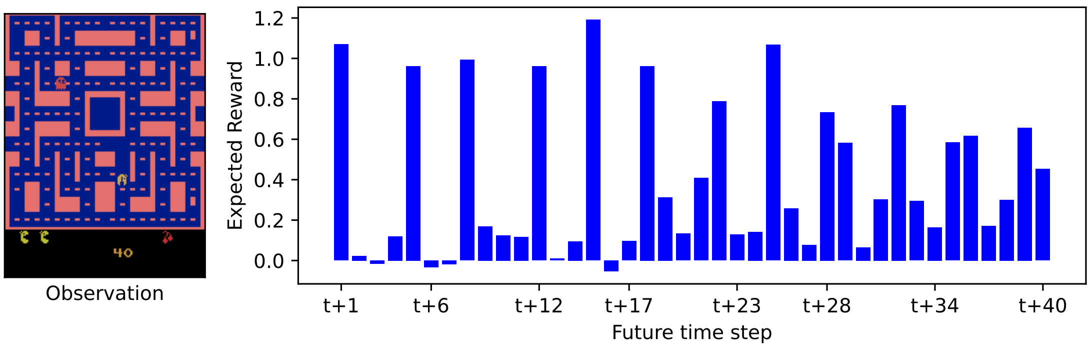


FIGURE 5.6: A Ms. Pac-man observation (left) with the respective predicted next 40 future expected rewards (right)

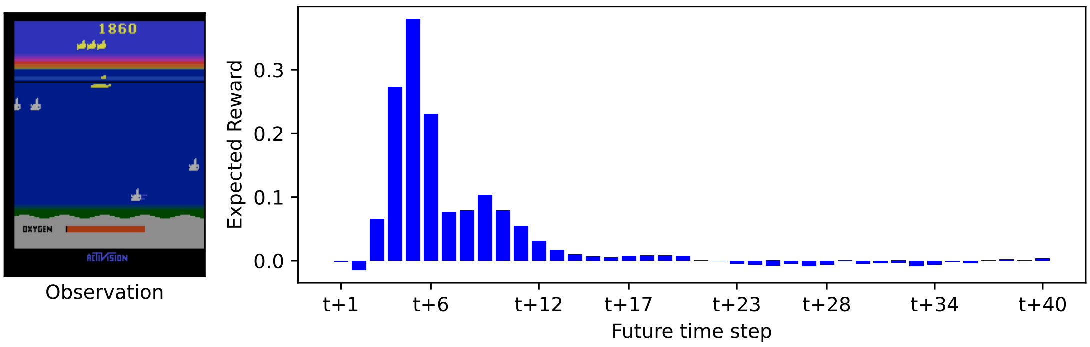


FIGURE 5.7: A Seaquest observation (left) with the respective predicted next 40 future expected rewards (right)

As a result, we anticipate that TRD has the potential to aid researchers and developers in debugging RL agents; Figure 5.4 and the related videos provide novel information about an agent's future beliefs and its understanding of an environment's reward function.

5.3.2 What Observation Features are Important?

Understanding the areas of an input that have the greatest impact on a neural network is a popular technique for generating explanations, called saliency maps. These allow users to visualise what features of an observation most influence an agent's decision. With access to an agent's beliefs about its future expected rewards, TRD provides novel saliency map opportunities to understand how the agent's focus with respect to an observation varies.

Utilising GradCAM (Selvaraju et al., 2017) (a popular saliency map algorithm described in Section 3.2.1), we can select individual expected rewards as the output to discover its feature importance. Figure 5.8 plots an Atari Breakout observation and the normalised feature importance for the expected reward of the next timestep ($t+1$) and the most distance expected reward ($t+40$) along with their normalised difference. The feature importance plots highlight areas of focus (red), influencing its decision and ignored areas (blue). We find that the agent's focus on the ball and bricks varies depending on how far in the future a reward is predicted. For the $t+1$ feature importance, the agent is highly focused (shown in red) on the ball in the centre. In comparison, for $t+40$, the agent focuses more on the bricks than the ball. Using domain knowledge of the environment validates human expectations as the number of bricks left and their position will have greater long-term importance to the agent than the ball. This difference is highlighted when subtracting the feature importance of $t+1$ from $t+40$ such that the ball's importance is significantly lower (shown in blue) and the bricks have relatively greater importance (shown in red).

To help visualise this change in an observation feature's importance across each predicted future reward, we provide a video of Figure 5.8 within the associated GitHub repository. Additionally, we provide a video of an episode plotting the first and last predicted reward's feature importance for each timestep. Like visualising an agent's expected reward in Section 5.3.1, Figure 5.8 and videos can help researchers and developers understand in what context a feature has importance for an agent. Previously, it was only possible to understand a feature's importance to predict the agent's total reward, whereas TRD provides us with the ability to investigate the importance of features in a more granular way.

5.3.3 What is the Impact of an Action Choice?

Within an environment, there are often multiple (possibly similar) paths to complete a goal, with humans interested in understanding the differences between them (e.g., Figure 5.1). Contrastive explanations are a popular approach to understanding the reasons for taking one decision over another. In our case, this is the choice between two alternative actions in some state (Miller, 2021). With the future expected rewards, TRD

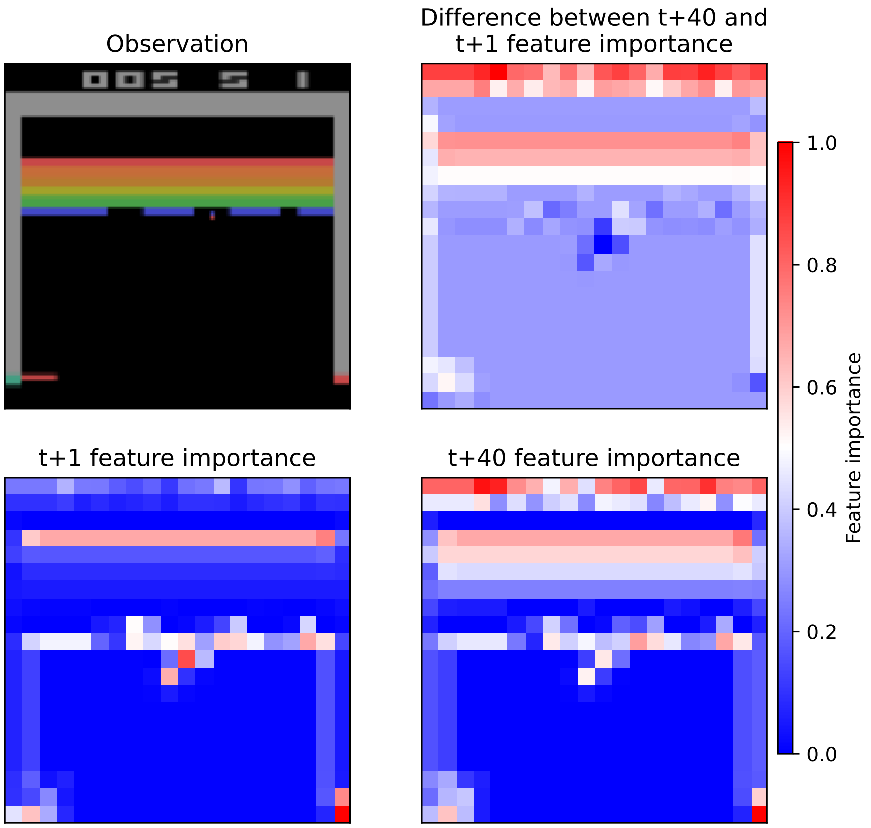


FIGURE 5.8: GradCAM saliency maps for the $t+1$ and $t+40$ expected reward along with their difference for a Breakout observation.

provides additional information to compare and contrast states and actions using what rewards the agent expects to receive and when along different paths. In this section, we show how simple explanations only using the timestep-wise difference in expected rewards can help understand an action's impact on an agent's future rewards.

Figure 5.9 shows the expected reward for taking no action (noop) and firing and the differences between the expected reward for noop and firing in the Atari Seaquest environment. The right-hand side figure shows that the difference in future rewards produces a positive and negative spike, after which the expected rewards converge. We can infer from these spikes that if the agent fires rather than noop then there is a more immediate reward, whereas if the agent waits, taking no action, the reward is delayed resulting in a later spike. Crucially, this difference in reward outcomes is resolved afterwards, causing no long-term difference in the agent's expected rewards. Using domain knowledge, we can assume that this means if the agent doesn't fire in this timestep, it will most likely fire in the following timestep or soon after, thus receiving a slightly delayed reward.

Collectively, with the explanations from Sections 5.3.1 and 5.3.2, contrastive explanations provide a greater understanding of an action's impact on an agent's future rewards.

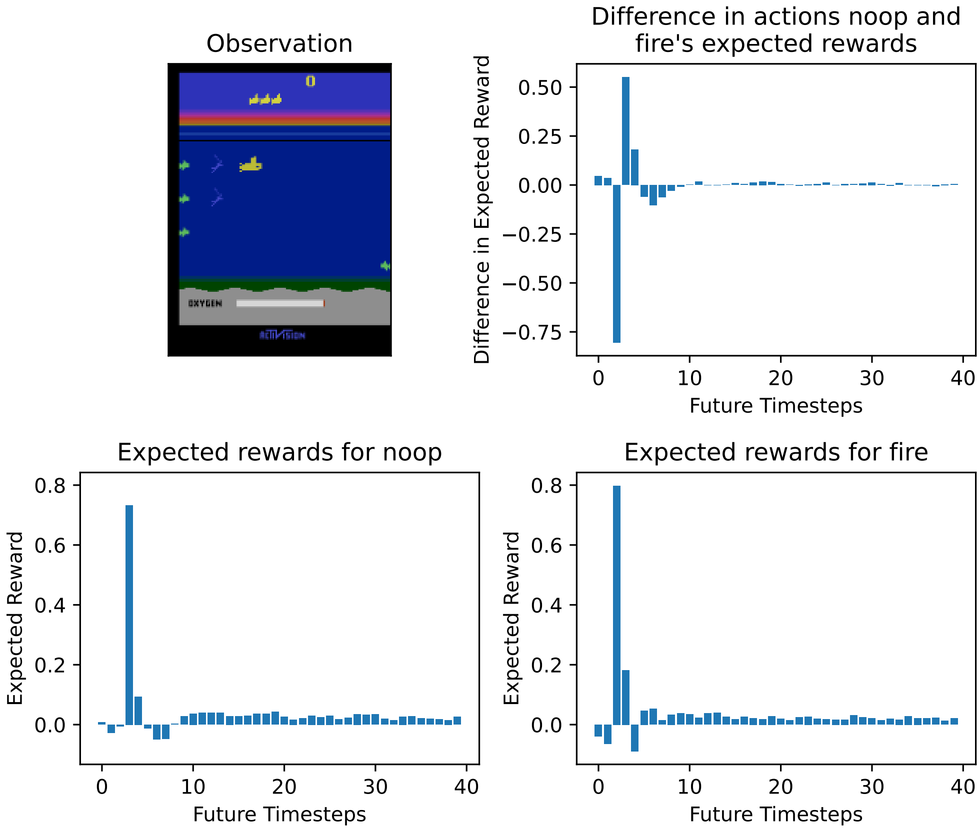


FIGURE 5.9: The difference of each future expected reward for taking Fire and Noop actions of the observation for the Atari Seaquest environment.

5.4 Discussion

In this chapter, we identified a novel problem for XRL when describing future reward estimators (i.e., Q-value or state-value) that their scalar nature makes explaining the differences between two actions impossible in some cases. To address this novel problem, we propose TRD using a novel future reward estimator to output the next N expected rewards, which can be easily implemented through modifying an agent's neural network output and their loss function (Eqs. (5.6) and (5.14)). Critically, this estimator cannot be reverse-engineered from a state's reward and must be learnt as a separate function. Due to the TRD estimator's provable equivalence to scalar future reward estimators (Theorem (5.1) and Eq. (5.7)), TRD can be incorporated into all deep reinforcement learning algorithms, adding the capability to output and explain an agent's future expected rewards.

Extending DQN, we show in Section 5.2 that TRD agents can achieve performance approximating standard scalar DQN agents within 4 million frames (with the help of QDagger (Agarwal et al., 2022)), 10x less than the base agent. Importantly, we found that this performance held across a range of TRD hyperparameters, the number of reward bins, bin widths, and the reward distance. We were surprised by this result as

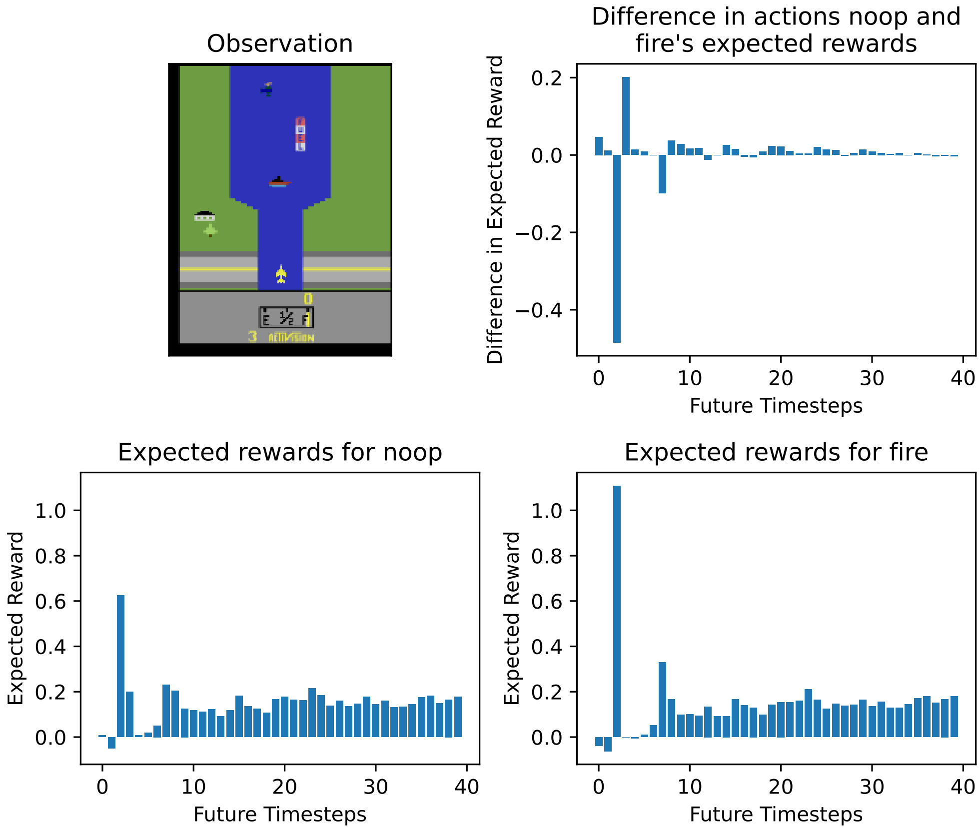


FIGURE 5.10: The difference of each future expected reward for taking Left and Right actions of the observation for the Atari Riverraid environment.

we expected the number of bins to negatively impact training efficiency and possible performance. In particular, we feared the problem of bootstrapping (Sutton and Barto, 2018, Page 264), where the agent uses itself to improve, would be exacerbated compared to scalar agents due to numerous distinct though interlinked outputs with each bin. We believe that QDagger's offline training stage, originally utilised to minimise training time, had the unexpected impact of strongly preventing individual reward outputs from diverging during training, which could cause unexpected policy collapse.

As TRD agents learn similar Q-values (and performance) to the base agent, closely mimicking the base scalar DQN agent, the TRD agent can be used either as an explainer of the base DQN agent or as the actual policy replacing the DQN agent with incorporated explainable capability. However, this research does have limitations. Primarily is that TRD can only explain the next N rewards that can cause problems explaining environments with highly sparse rewards for which we propose a solution in Chapter 7. An additional weakness of this research is the lack of user evaluation and that the explanations provided in Section 5.3 require a prior understanding of RL to interpret the future expected rewards. For both of these weaknesses, we address them in Chapter 6.

Chapter 6

A Comparative User Evaluation of Goal Identification using Explanations

In Chapters 4 and 5, we researched and developed novel temporal explanations for deep reinforcement learning, explaining the impacts of actions on future states and rewards for pre-trained agents and architecturally modified agents, respectively. In this Chapter, we primarily focus on the design of a novel comparative user evaluation methodology for XRL, addressing limitations identified in prior research (Section 3.3) and Chapter 4's user survey (Section 4.3.1).

Before proposing our method for comparative user evaluation of XRL methods, however, we explore one of the limitations of our explanation mechanisms for TRD (Section 5.3): the need for users to have sufficient technical knowledge to interpret the presentation of an agent's future expected rewards (e.g., Figure 5.6). This limitation is expected to be more significant when presenting explanations to a general audience rather than technical experts. Therefore, in Section 6.1, we improve TRD's interpretability by adding natural language summarisation of an agent's future expected rewards through a large language model (LLM), referred to as TRD Summarisation. This was opted for over natural language templating as Brown et al. (2020) has demonstrated that LLMS have the capacity, with a few example questions and answers, to generalise for unseen tasks not in their training dataset. Thus, the approach for generating explanations is not limited to the structure or context designed in the language template, but can adapt to a range of domains without further fine-tuning due to the general knowledge and understanding of LLMs. We showcase this novel summarisation for when and what rewards an agent expects to receive with GPT4o (Hurst et al., 2024), a state-of-the-art multi-modal LLM in Table 6.1 across several atari environments and with prompts that contain no domain knowledge and prompts that describe the environment's reward function and the current observation.

Next, building upon the surveys discussed in Section 3.3 and conducted in Chapter 4, Section 6.2 sets out a novel comparative user evaluation centred on whether explanations can be used to debug agent behaviour. This evaluation aims to simulate the real-world task of: (i) understanding what an agent is doing; (ii) what (sub)goals it is working towards; and (iii) why. In particular, for the same environment, we train unique agents to optimise different reward functions, producing policies that work to accomplish different goals with different goals. Using these different policies, selecting one at random, we test whether an explanation allows users to identify the agent's goal. We pair each "Which task?" question with questions to elicit the user's confidence in their selection, the clarity of the explanation, and how easy the user found identifying the task. This combination of questions allows us to investigate correlations among users' judgements with their confidence and assessments of how good/helpful an explanation is.

In Section 6.3, we analyse the survey results for 100 participants, evaluating how effective each explanation mechanism was, the relationship between user confidence and goal accuracy, and overall user opinions of the explanations. Of the four tested mechanisms, our temporal explanations (Dataset Similarity Explanations and TRD Summarisation) had the highest accuracy of 53.0% and 34.9%, respectively, compared to two baseline algorithms (Optimal Action Description and Specific and Relevant Feature Attribution) of 28.7% and 22.5% where 25% is the expected accuracy for random guessing. We find that explanation mechanism accuracy can vary depending on the goal being explained, the time taken can correlate to the user's accuracy, and no user characteristic was correlated to user performance.

In summary, this Chapter contributes the following to the state-of-the-art in XRL:

- We demonstrate the application of LLMs to summarise an agent's future expected rewards, testing several prompts containing varying levels of domain knowledge across several environments. For carefully designed prompts, we find that high-quality summaries of an agent's future expected rewards can be generated even without environmental domain knowledge. Furthermore, including domain knowledge of the reward function further improved the summaries, allowing simpler and contextually aware explanations.
- We propose a novel evaluation methodology for testing if survey participants can debug/identify an agent's goal. This methodology involves asking users to identify the task an agent is completing from among a fixed set of alternatives to assess mechanism effectiveness. We pair these objective questions with subjective questions, testing whether algorithmic effectiveness correlates with user preference.

- Across 100 participants, we demonstrate that users' self-reported confidence, understanding, and ease of identification are weakly correlated with their goal prediction accuracy and vary between explanation mechanisms. Further, we find that a user's average confidence is very weakly correlated with their accuracy.

6.1 Natural Language Summarisation of Future Rewards

To explain a TRD agent's future expected rewards, we plot a bar chart of each reward for each timestep and describe the figures to the reader (e.g., Figures 5.4, 5.5, and 5.6). Interpreting these figures requires some technical knowledge of reinforcement learning: what an expected reward means and, optionally, domain knowledge of the particular environment to understand what might lead to a reward. This is a limitation of the explanation mechanisms that, without some technical knowledge, users may be confused or misunderstand an agent's future expected rewards. For this reason, we investigate the use of natural language to explain the figure's meaning to end-users who may lack the technical knowledge to initially understand it. To generate this natural language, we utilise LLMs rather than natural language templating due to their ability to more easily adapt to new environments, contextual information, and implementation speed. We investigate whether these natural language explanations of an agent's future expected rewards can reduce the quantity of technical or domain knowledge necessary to understand the explanations. We refer to this novel explanation mechanism as TRD Summarisation.

Large Language Models (LLMs) based on the Transformer architecture (Vaswani et al., 2017) are designed to process sequential data and iteratively predict the next element of the sequence. This approach has been successfully applied to text prediction, language translation, image recognition, and more (Islam et al., 2023). Crucial for this work, Brown et al. (2020) and Kaplan et al. (2020) have found that LLMs can exhibit novel capabilities as the number of parameters increases. This is an important property for us, as explaining an agent's future reward will not be contained in any model's training dataset; however, explaining sequential data will be. Using domain knowledge to condition the model on the task being asked and providing several examples for the agent, we would expect LLMs of adequate size to generate reasonable summarisations of the agent's future expected rewards to users.

Prompt engineering is important when utilising LLMs, determining what information the model acts upon. We consider three use cases of LLMs; the first is where the model is provided with a technical description of what the data means and a list of rewards but doesn't contain a description of the environment or reward source; we refer to this as a "zero-environment knowledge prompt". This is important for applications with minimal knowledge of the underlying environment or where we wish not to bias the user by including domain knowledge. The second includes environmental information, describing what the game is, how it works, and when or why rewards are given to the agent, referred to as an "environment knowledge prompt". This prompt, using the environment description, enables the model to infer possible sources of rewards for a list of future expected rewards. Finally, the previous two methods did not incorporate the agent's observation into the model's context, requiring the LLM to induce the cause of the rewards for an unseen state. Utilising multi-modal LLMs allows the images, text, sound or videos to be simultaneously processed to condition the model's output. For multi-modal LLMs, we can prompt the agent with the environmental details and observation context, referred to as an "environment+observation knowledge prompt". Including the observation for the model could enable it to link the observational features to the future expected futures to improve the contextual information of the future reward summary.

System You are an assistant providing summaries of an agent's predicted rewards in future timesteps playing a sequential game. Help describe in a one-sentence summary the pattern of rewards, highlighting how far in the future this happens, if the pattern is periodic or consistent, positive or negative, etc. Ignore rewards close to zero compared to the rest of the pattern, as this is related to data noise.

User [0, 0, 0, 0, 4, 0, 0]

Assistant The agent expects a reward of 4 in 5 timesteps

User [1, 1, 1, 1, 1, 1]

Assistant The agent expects consistent positive rewards

Prompt 1: Zero-enviornment knowledge prompt

In designing these prompts, we found several important components crucial to their effectiveness. For the zero-environment knowledge prompt (Prompt 1), first, "help describe in a one-sentence summary the pattern of rewards" limits the model's output as we found GPT4o would produce longwinded explanations and additionally forces the LLM to summarise the important aspects of the data, e.g., when the rewards happen, if the reward pattern is periodic or constant, and if the rewards were positive or negative. Second, as the TRD models were not extensively trained, the future expected rewards can be relatively noisy around zero. As Atari environments have sparse rewards, it was common for most timesteps to be around zero, causing the model to describe this data, not the non-zero rewards. Therefore, the sentence "ignore rewards close to zero compared to the rest of the pattern" helped prevent the LLM from becoming distracted. Finally, we include two example prompts for the LLM, referred to as few-shot examples, to help prime the model on the expected outputs it should produce. To prevent biasing the model for the task, we selected the expected future rewards from Figure 5.1 with a very short summary of both moving up and down, as they describe cases of dense and sparse future rewards to the model and the type of answers we are interested in. Without these examples, we found that the model reverted to answers closer to its training data, unrelated to what we wanted. Finally, TRD agents predict an agent's future rewards to five or six digits of precision, which is unnecessary for the model or any human when understanding an agent's beliefs. Therefore, we round each future reward to 2 decimal places, helping reduce the complexity of the data for the model and making any quoted expected rewards shorter.

System You are an assistant providing summaries of an agent's predicted rewards in future timesteps playing the Atari game Breakout. In the game Breakout, the agent receives a reward of 1 each time the ball breaks a brick. Help describe in a one-sentence summary the pattern of rewards, highlighting how far in the future this happens, the agent's confidence (as the reward is either 1 or 0, then expected rewards are equivalent to the probability of the agent breaking a brick), etc. Ignore rewards close to zero compared to the rest of the pattern, as this is related to data noise.

User [0, 0, 0, 0, 1, 0, 0]

Assistant The agent will break a brick in 5 timesteps

User [1, 1, 1, 1, 1, 1]

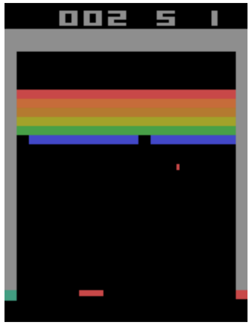
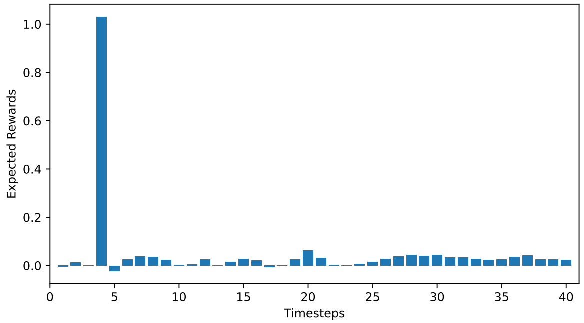
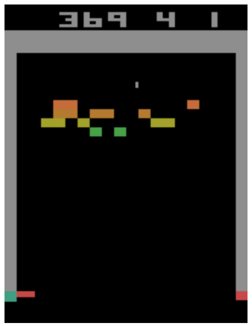
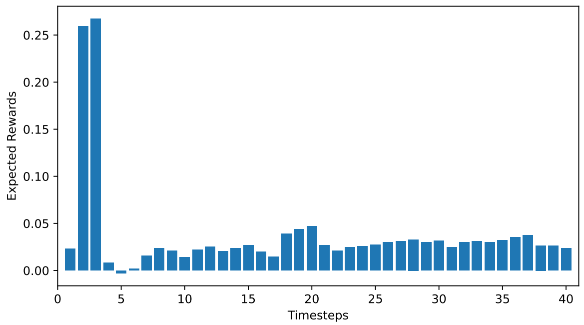
Assistant The agent expects to break a brick every timestep

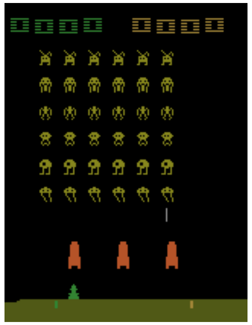
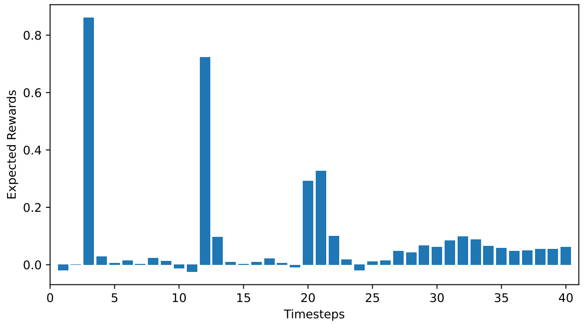
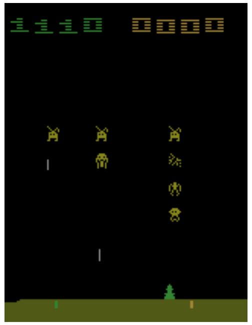
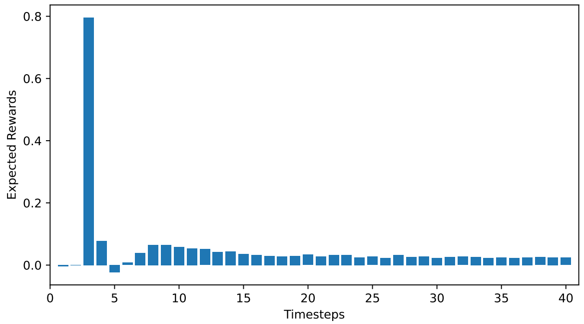
Prompt 2: Environment knowledge prompt for Atari Breakout

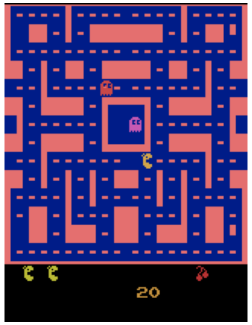
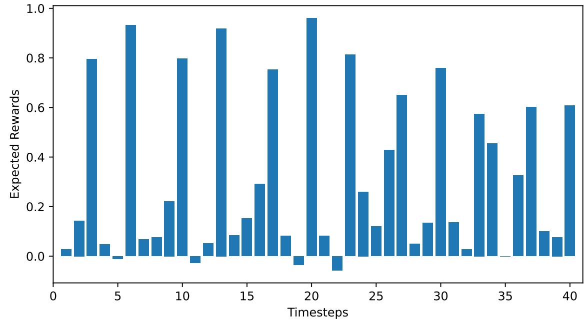
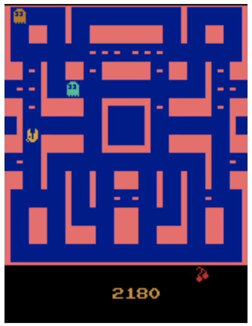
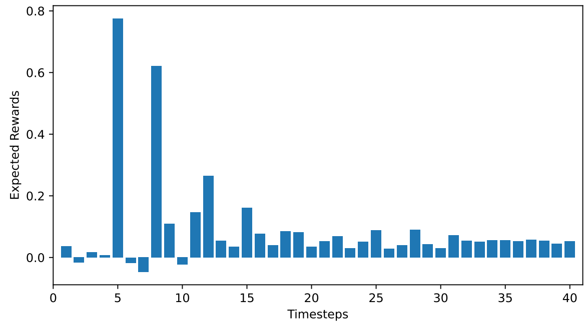
For the environment knowledge prompts (Prompt 2), we change two details from the zero-knowledge prompt. First, we add a description of the environment's reward function, explaining what the rewards are related to, e.g., for Breakout, "the agent receives a reward of 1 each time the ball breaks a brick". Given this, we updated the few-shot examples for each environment to match the reward function description. Second, as the Atari rewards are clipped between -1 and 1 for environments with only positive or negative rewards, a timestep's expected reward is equivalent to the probability of the non-zero reward happening. We include the "the agent's confidence ..." statement in our prompt, explaining that the expected rewards should be interpreted as the probability of the reward, enabling the model to rationalise the expected rewards in terms of overall confidence in the reward happening. This prevented the model from believing that, for example reward of 0.84, the agent would receive a reward of 0.84, but rather, there is an 84% chance of getting a reward of 1. Despite not using this confidence analysis in our few-shot prompts, we found the models highly capable of incorporating this knowledge into their summaries. See Table 6.1 for examples.

To compare these prompts, we test across three Atari environments (Breakout, Ms Pacman, and Space Invaders) with GPT4o (Hurst et al., 2024), a state-of-the-art multi-modal LLM. We decided against using time series LLMs (Zhang et al., 2024), designed for very long series of high-dimensional data, as the data used in this work is scalar and relatively short (40 elements long), meaning that time series specialisation was unnecessary. Table 6.1 showcases the three prompts across six observations. Prompt 1 is the zero-environment knowledge prompt, and Prompt 2 is the environment-knowledge prompt specialised for Atari Breakout. Appendix E contains the specialised prompts for Space Invaders and Ms Pacman.

TABLE 6.1: Table of example observations and future expected rewards with the outputs for three different prompts: zero-knowledge, environment knowledge, and environment+observation knowledge

Observation	Agent's Future Expected Rewards	Zero-Environment Knowledge Prompt	Environment Knowledge Prompt	Environment + Observation Knowledge Prompt
		The agent predicts a significant positive reward of 1.03 around 4 timesteps, followed by small fluctuations.	The agent is expected to break a brick in 3 timesteps, with diminishing confidence several timesteps after.	The agent is likely to break a brick in 3 timesteps with high confidence due to the sharp peak, and maintains a low probability of breaking bricks in subsequent steps.
		The agent experiences a notable positive reward spike to 0.27 in the third timestep, while other rewards remain consistently small and positive.	The agent is likely to break a brick in 3 timesteps, with a high level of confidence.	The agent is highly confident in breaking a brick around 2 or 3 timesteps from the present.

		The agent expects notable positive rewards of approximately 0.86 in 2 timesteps and 0.72 in 11 timesteps, with smaller consistent positive rewards starting from 19 timesteps onward.	The agent is highly likely to shot an alien in 3 timesteps and then again in 12 timesteps, with a moderate likelihood at 19 and 20 timesteps.	The agent is highly likely to shoot an alien in 2 timesteps and again in 11 timesteps.
		The agent expects a peak reward of 0.8 after 2 timesteps, followed by a gradual decline and stabilization around 0.02-0.03 for future timesteps.	The agent is highly confident of shooting an alien in 3 timesteps.	The agent is most likely to shoot an alien in the next timestep with a high confidence of 0.8.

		The agent experiences periodic positive rewards peaking around 0.8 to 0.96 approximately every 6 to 7 timesteps.	The agent is likely to eat dots at 2, 5, 10, 12, 16, 19, 22, 29, and 34 timesteps with high confidence.	The agent is likely to eat a dot in 2 and 5 timesteps, with high certainty.
		The agent anticipates a notably high reward of 0.78 in 5 timesteps and another significant reward of 0.62 in 8 timesteps, followed by generally low and consistent rewards.	The agent is highly confident about eating a dot in 5 and 8 timesteps.	The agent will confidently eat a dot in 5 timesteps, with another significant dot consumption expected in 8 timesteps.

6.2 Comparative User Evaluation Design

Outlining XAI in Section 2.3, we adopted a definition of explainability that requires an explanation to accurately describe a model's working and present this information in a way that users can understand. Therefore, we view user surveys as a critical feedback component of XAI research, testing whether a user understands an explanation and whether it is effective in describing a model's decision-making. This section outlines a novel comparative user evaluation methodology centred on the practical problem of identifying the goal that an agent is completing and testing if explanations can be utilised for this task, referred to as goal identification.

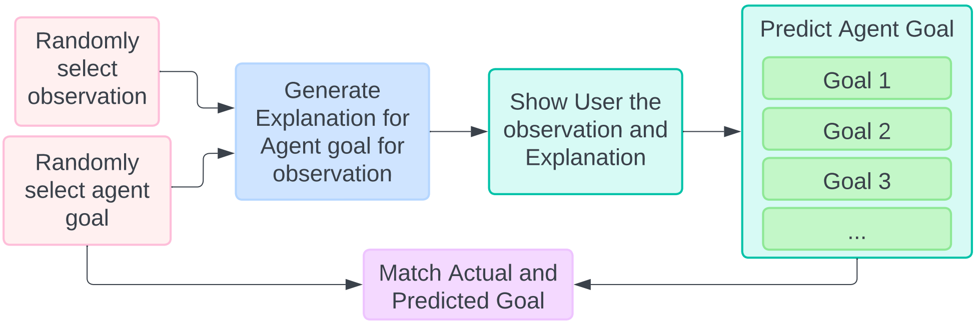


FIGURE 6.1: Flowchart of the Goal Identification Task

Figure 6.1 illustrates how we implement goal identification questions. For a common environment, we implement several non-overlapping goals by modifying the reward function.[1] By training a unique agent for each reward function (i.e., goal), each agent's decision-making should produce unique reasoning to complete each goal. Given a selected environment state and an agent trained for a goal, we can generate an explanation of why the agent took the decision it did. From this environment state and explanation of the agent's decision-making, we ask users to select the goal description that best describes the agent's decision-making. This requires users to rationalise from the explanation content to what goal could cause this explanation; i.e., users must debug a model's decision-making through explanations to determine the agent's goal.

We implement this goal identification evaluation methodology for the Atari (Bellemare et al., 2013) Ms Pacman environment, where we devise four goals (reward functions) by intercepting and filtering the environment's standard rewards. As two of the agent's goals are to live for as long as possible and die as quickly as possible, we terminate an episode when an agent loses a life, rather than the standard four lives given in Ms Pacman. Videos of agents pursuing these goals are available on GitHub[2]. Figure 6.2 presents an example observation from Ms Pacman with reward sources labelled.

[1] Importantly, with a common environment, all possible observations are viable for each agent to act with distinct decision-making and goals to complete.
[2] https://github.com/pseudo-rnd-thoughts/eval-xrl-goal-identification

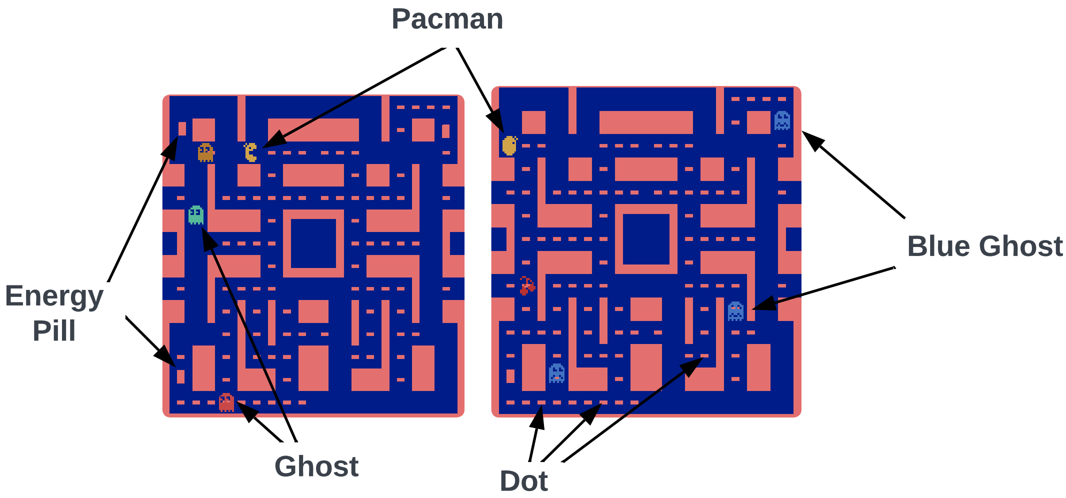


FIGURE 6.2: Ms Pacman with the different reward sources labelled

Eat Dots - Each time the agent eats a dot, the agent is given a reward of 1. This gives the agent a regular positive reward of 1 every two to three timesteps. To maximise this reward, the agent's policy should be to try to eat every dot on the screen and avoid ghosts.

Eat Energy Pills and Ghosts - The Ms Pacman environment has four energy pills that, upon eating, the agent will receive a reward of 1 and turn the ghosts blue for 500 timesteps. Being blue, the ghosts can now be eaten by the Pacman (rather than dying when non-blue) and will give the agent a reward of 1. This gives the agent a relatively sparse positive reward of 1 every 20 to 30 timesteps from the energy pills or eating ghosts. To maximise the reward, the agent's policy aims to eat an energy pill and then all the blue ghosts, repeating for all four energy pills. As the agent can learn to eat all four energy pills, after which the agent would have no goal left, we truncated the episode when all energy pills are eaten and the ghosts have returned to their normal colour, i.e., 500 timesteps after the fourth energy pill is eaten.

Survive - Atari reports agents' lives on each timestep. We can use this data to determine if a ghost has eaten the agent, losing a life. Therefore, for each timestep, the agent is rewarded +0.5 if it survives and is not eaten, providing a consistent positive reward for every timestep. To maximise the rewards, the agent policy will move to maximise their distance from the ghosts and use the energy pills when in danger (turning the ghosts blue and preventing Pacman from being eaten).

Lose a Life - The opposite of the survival goal, the agent receives a reward of -0.5 every timestep it is alive. This negative reward encourages the agent to lose a life quickly. Due to Ms Pacman's deterministic initial states, to help train the agent on a variety of initial states, we used a pretrained Ms Pacman agent for the standard reward function to take the first 100 to 400 timesteps before the Lose a Life agent starts acting. To maximise the rewards, the agent will move towards ghosts to be eaten unless the ghosts are blue (as contact will eat the ghost, not the Pacman) when it will move away until the ghosts return to their normal colour.

The first three goals represent rational (sub)goals that we could easily expect a normal agent to pursue, while the Lose a Life goal is irrational with respect to the norms of the game. These goals were selected because we believe it is critical for explanations to recognise rational and irrational behaviour from a human perspective due to their use to confirm expected behaviour and understand unexpected behaviour. For each goal, we train a standard DQN agent (Mnih et al., 2015) for 10 million steps with an Impala vision encoder (Espeholt et al., 2018), proven to improve performance significantly (Clark et al., 2024).

With the evaluation procedure outlined above, we incorporate it into our user survey as shown in Figure 6.3 with screenshots of the implemented survey provided in Appendix F. The first stage of the survey informs the user of the survey's purpose and content, and then we collect user information: age, gender, level of education, prior knowledge of AI, and Ms Pacman. We collect this data to investigate questions of correlation between participants' characteristics and goal accuracy. Next, we outline the goal identification problem with comprehension testing to ensure users understand the questions. This includes two questions on the survey's objective of identifying agent goals and selecting the corresponding goal option for a description. Users who do not get one of the questions correct are removed from the survey and are not allowed to continue. For users who continue, we evaluate each explanation mechanism in a random order containing an explanation summary, comprehension question, 4 goal identification questions, and an overall explanation rating section. For the goal-identification questions, from 20 observations (5 from each goal), we generate an explanation for each goal, producing 80 unique observation-goal questions. From these questions, each was randomly selected to be shown an equal number of times across all participants. Included in each goal-identification question, alongside selecting the agent's goal, we ask users to select their confidence in the prediction, from a 5-point Likert Scale (Likert, 1932) of "Very Unconfident" to "Very Confident". Additionally, we include a hidden timer to measure the time users take to answer each question. Once the user has answered the four goal identification questions, the survey moves to elicit the user's opinions on the explanation mechanism. Again, we use a 5-point Likert Scale to measures users' overall confidence from "Very Unconfident" to "Very Confident", their ease of identifying the agent goals from "Very Difficult" to "Very Easy", and the user's general understanding of the explanation from "Did Not Understand At All" to "Completely Understood".

This survey structure has two advantages over the survey presented in Chapter 4. The "Which goal?" questions provide concrete data on how good an explanation is for goal

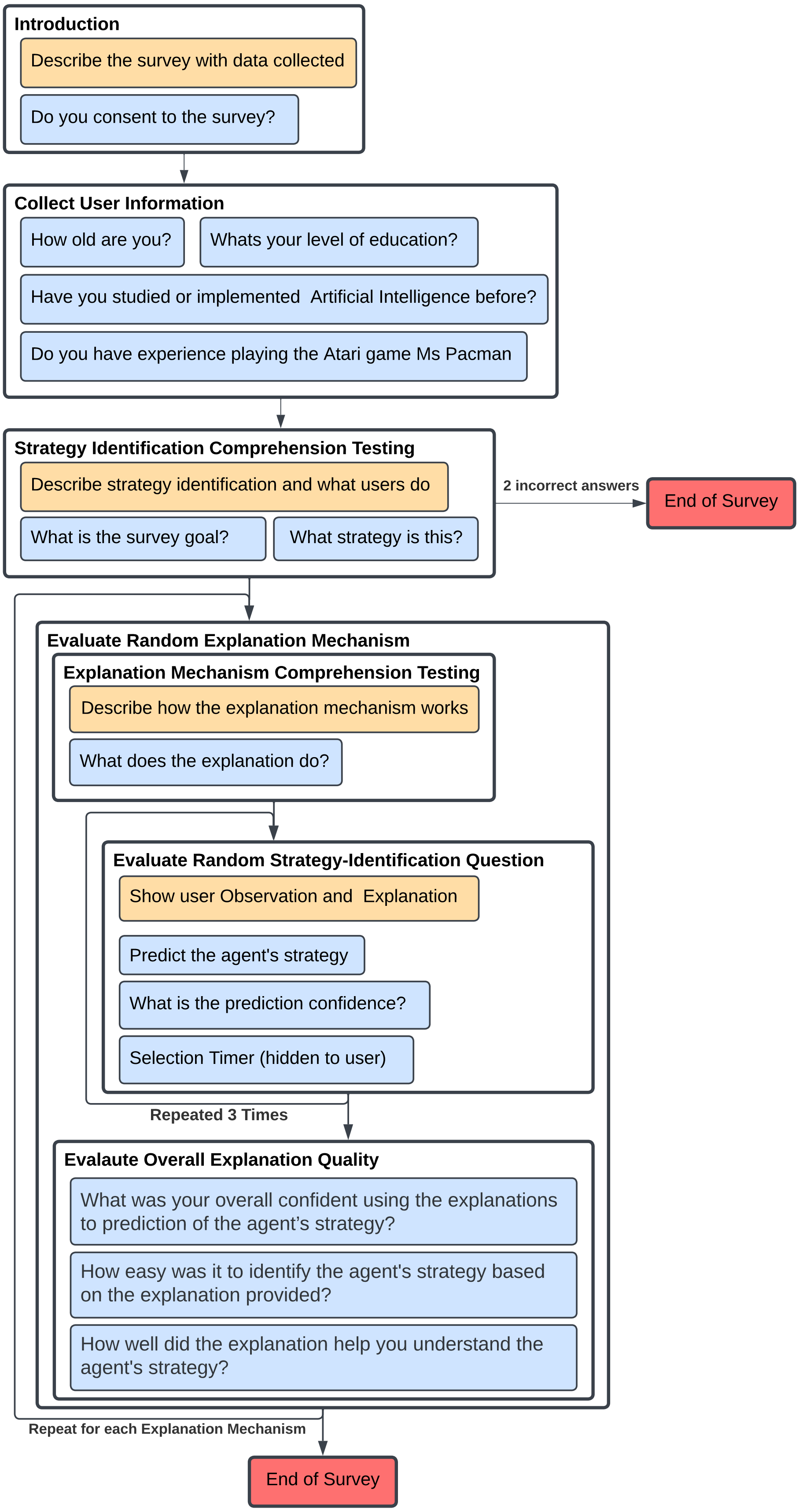


FIGURE 6.3: Flowchart of the comparative user evaluation. Blue boxes represent user questions, and the orange boxes represent information given to the user.

identification, allowing us to measure correlations between the user's self-reported views and the explanations' accuracy. Additionally, we added comprehension testing and question randomisation to help ensure the results collected are more robust to confirm user understanding and question variance. Compared to the literature in Section 3.3, our survey design differs through the use of self-reported questions on both individual questions (confidence) and the overall explanation (confidence, ease of identification, and understanding). Second, for goal prediction surveys like Huber et al. (2023) and Septon et al. (2023), we provide the user four goal options and ask them to pick the agent's, compared to selecting characteristics that the agent is focusing on (Huber et al., 2023) or with binary options (Septon et al., 2023).

We use four explanation mechanisms: TRD Summarisation (Section 6.1), Dataset Similarity explanations[3] (Chapter 4), Gupta et al. (2020) called Specific And Relevant Feature Attributed (SARFA) described in Section 3.4 (and improved upon Perturbation-based Saliency Maps (Greydanus et al., 2018) used in Chapter 4). Finally, to understand if users can predict an agent's goal with just a description of the next action as an explanation, referred to as Optimal Action Description (OAD). Table 6.2 summarises each mechanism's medium and content.

TABLE 6.2: Table of Explanation Mechanisms used in the Survey with their explanatory medium and content.

Explanation Mechanism	Medium	Content
TRD Summarisation (Section 6.1)	Text & Image	Natural language summary of future expected rewards
Dataset Similarity Explanation (Section 4.1.1, DSE)	Video	Video of similarity observations in a previously collected dataset
Specific and Relevant Feature Attribution (Gupta et al. (2020), SARFA)	Image	Saliency map for the optimal action.
Optimal Action Description (OAD)	Text	A description of the agent's optimal action

It is essential to control for bias when selecting observations for the survey. Prior work largely uses importance-based metrics rather than hand-selecting observations (Huber et al., 2023; Amir and Amir, 2018; Septon et al., 2023). Most use importance-based metrics (the difference between a state's maximum and minimum Q-values) from an agent's rollout to select states with the highest importance. This can be highly effective in picking states that could be considered interesting to explain, as a right or wrong action will result in significantly worse or better outcomes for the agent. However, this

[3]As suggested in Section 4.4, we use an Autoencoder to learn the observation embedding, which was trained from 40,000 observations gathered from across all four agent goals. For each agent goal, using the pretrained agent, we extract the hidden layer activations for the 40,000 observations, with a two-layer encoder and decoder neural network with a hidden layer of 128 and ReLU activation with the bottleneck layer containing 16 features.

selection can bias observations to a limited subset of the state space with high Q-value importance but minimal observational differences.

To promote the selection of diverse and interesting observations across all agents, we propose a novel greedy selection algorithm that uses a state's importance and embedding distances between it and previously selected states to explain. To compute the embedding, we train an autoencoder on a dataset of an agent's hidden layer activations generated from rolling out each agent for $10,000$ steps and removing duplicates. We test this approach against alternative approaches with the average importance and embedding distance to other selection algorithms and present the results in Table 6.3. The combined importance and embedding approach maximises the product of the importance and diversity (represented by the embedding distance).

TABLE 6.3: Table of State Selection Algorithms with their average importance and embedding distance of the selected observations.

Selection algorithm	**Average Importance**	**Average Embedding Distance**
Importance	0.476	0.162
Importance with Masking	0.452	0.213
Embedding Distance	0.234	0.372
Embedding Cosine Similarity	0.258	0.172
Importance & Embedding Distance	0.408	0.296

6.3 Analysing User Comprehension

We conducted the comparative user evaluation of the four explanation mechanisms considered (Table 6.2) with 100 participants, using Prolific, a crowdsourcing website that provides access to participants across the globe, and Qualtrics for hosting and implementing our online surveys.[4] The pool of participants was filtered to those in North America and the UK whose first language is English and with at least High School GEDs / A-levels. Each user was paid £5 to complete the survey, with the median competition time being 14 minutes and 8 seconds.

For the survey, we have four agent goals, four explanation mechanisms, and 20 observations (five from each goal), generating 320 observation-goal-explanation questions to evaluate. With 100 participants in the survey, each was asked four randomly selected questions per explanation mechanism, giving 1600 answers. This resulted in all questions being independently evaluated five times. The raw anonymised survey results are provided in the associated GitHub project [2]. From these results, we investigate various research questions regarding how well users identify an

[4]We obtained ethical approval for the survey from our institutional review board (ERGO FEPS/99644).

agent's goal given an explanation (Section 6.3.1) and the subjective self-reported answers (Section 6.3.2).

6.3.1 Can Users Accurately Predict Agent Goals?

The comparative evaluation methodology proposed in Section 6.2 is centred on goal identification, where users are given an observation and an explanation, and they must select the goal description that most closely matches the explanation's content. This provides an objective measure to compare various components of users' accuracy for each explanation mechanism.

TABLE 6.4: Each explanation mechanism's accuracy across all agent goals for the 100 survey participants and an expert user.

Explanation mechanism	Survey Participant Accuracy
Dataset Similarity Explanation (DSE, Section 4.1.1)	53.0%
TRD Summarisation (TRD Sum, Section 6.1)	34.9%
Optimal Action Description (OAD)	28.7%
Specific and Relevant Feature Attribution (SARFA, Gupta et al. (2020))	22.5%

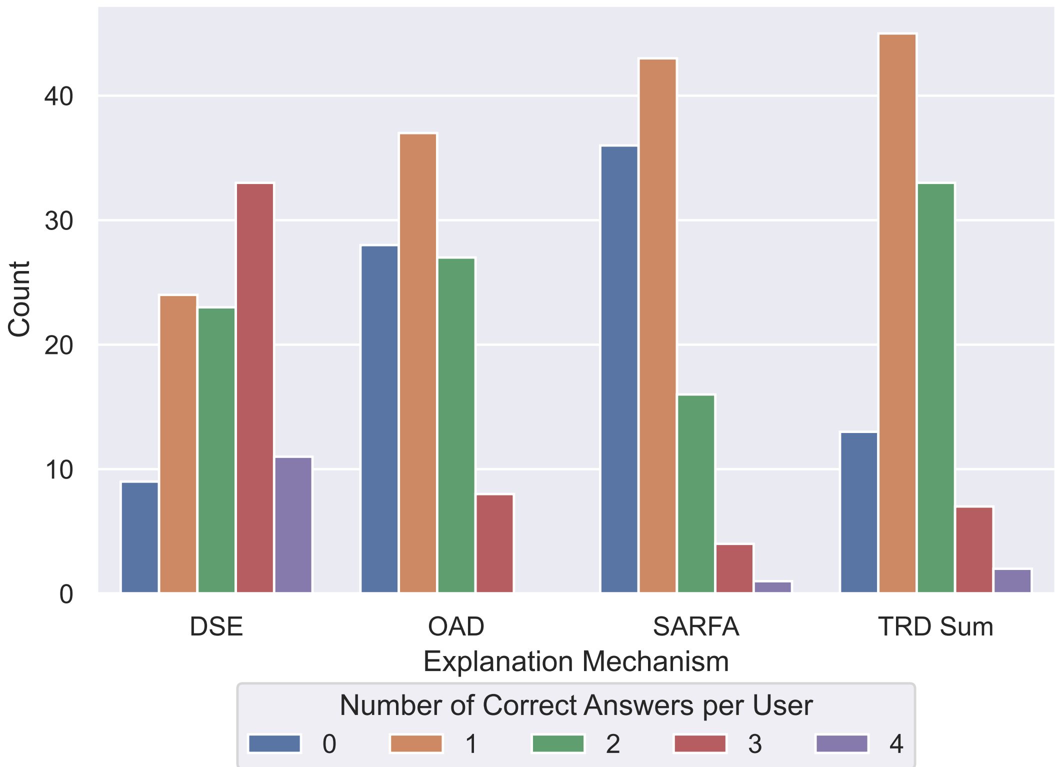


FIGURE 6.4: The number of users who got each possible number of correct answers for each explanation mechanism.

Table 6.4 lists the accuracy of each explanation mechanism across all agent goals. Our temporal explanations (DSE and TRD Sum) have the highest accuracy of 53.0% and 34.9%, respectively, while the two baseline algorithms (OAD and SARFA) had the lowest accuracies of 28.7% and 22.5%, respectively. If users were to guess randomly, their expected accuracy would be 25.0%. Investigating each user's performance for each explanation mechanism, Figure 6.5 plots how many users got 0, 1, 2, 3, or 4 answers correct for each explanation mechanism. Our temporal explanations had the most users who got at least one answer correct, at 91 and 87 for DSE and TRD Sum, respectively, compared to 72 and 67 for OAD and SARFA. However, DSE is the only algorithm where a consistent number of users could correctly identify 3 or 4 agent goals with 33 and 11, respectively, compared to 8, 5, and 9 for OAD, SARFA and TRD Sum that got either 3 or 4 correct answers.

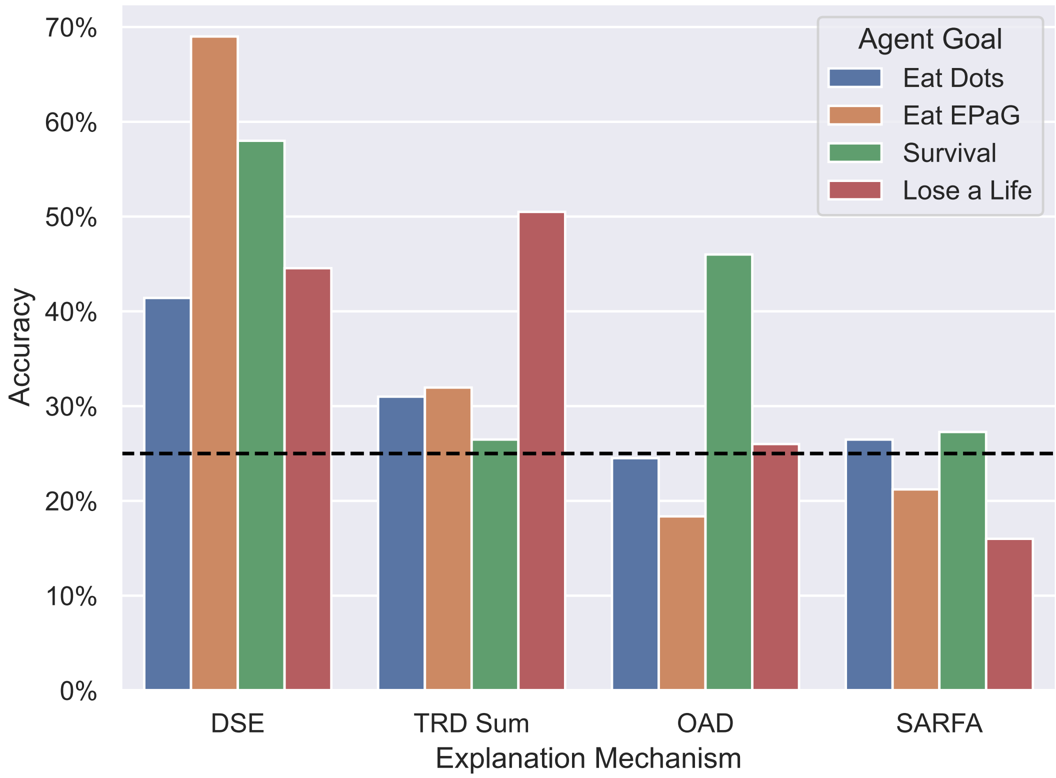


FIGURE 6.5: A bar chart for each explanation mechanism with the accuracy of the explanation mechanism across each agent goal. Each bar's accuracy is averaged across 100 answers from users. The dotted line is 25%, the expected accuracy for random guesses.

Is an explanation mechanisms' accuracy influenced by the agent's goal being explained? Figure 6.5 plots each explanation mechanism's accuracy across each agent's goal. Ideally, each explanation mechanism should achieve the same performance for all goals. For DSE, there is nearly a 30% difference in accuracy with 69.3% for Eat Energy Pills and Ghosts (Eat EPaG) and 58.0% for Survival, while its performance for Eat Dots and Lose a Life drops to 40.5% and 44.1%, respectively. Why the difference in performance?

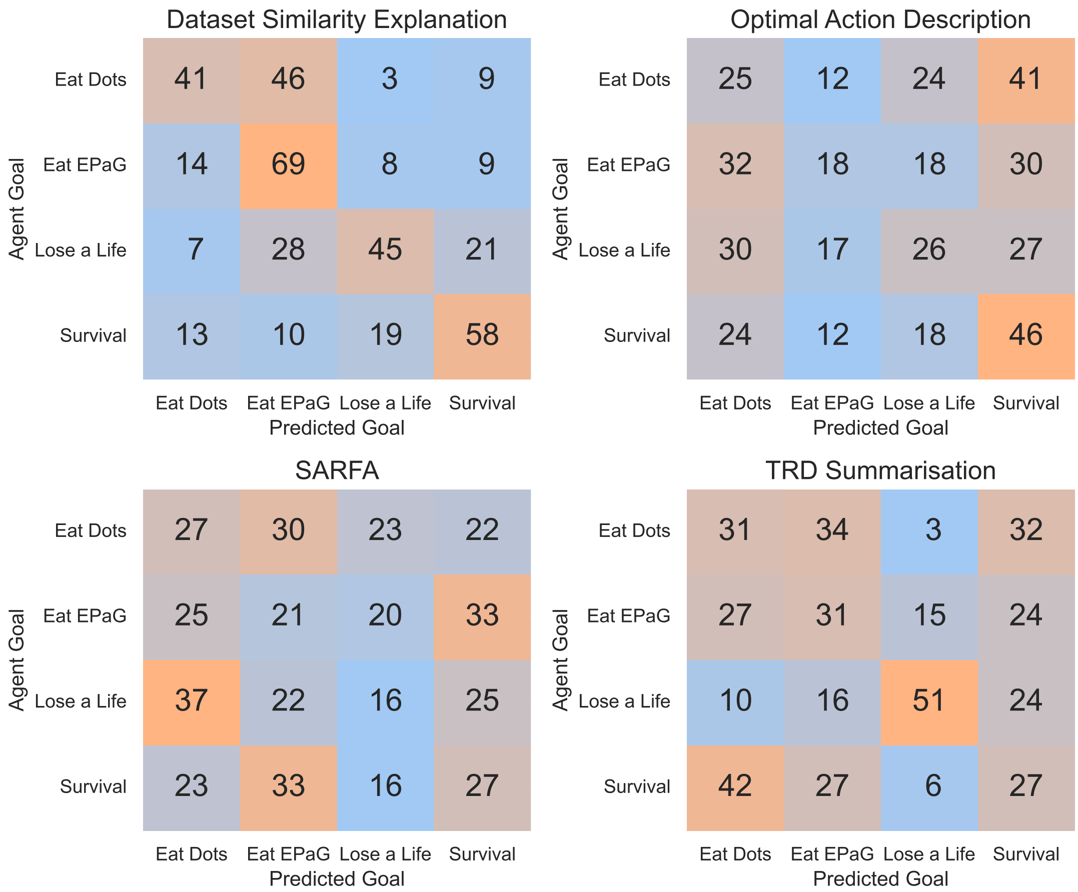


FIGURE 6.6: Confusion matrix for the number of times the predicted and true agent goal for each explanation mechanism occurs, where the rows are the actual agent goal and the column is the predicted. Correct answers are found on the top left to bottom right diagonal. Each row will sum to 100 as the 20 observations are each evaluated five times.

Using Figure 6.6 that plots the number of instances of each combination of true and predicted agent goals selected. For DSE, we can identify that for the Eat Dots goal, users are highly likely to misidentify it as Eat EPaG (41 to 46 selections), while for the Lose a Life goal, users mistook the explanation for the Eat EPaG or Survival goal, with 28 and 21 selections compared to 45 for the correct answer. Overall, for DSE, we can infer that users can generally understand the video of agent behaviour and link it to the related goal. However, for goals that have some overlap, users can struggle to differentiate.

For TRD Summarisation and OAD, we find a significant difference between each agent goal's overall and individual accuracy. For three of the agent goals, their average accuracy is between 29.8% and 23.0%, very close to that expected of random guessing; then, for Lose a Life and Survive, respectively, their accuracy increases to 50.5% and 46.5%. For TRD Summarisation, the reason for this difference in performance can be explained by the reward functions of the four goals. For Eat Dots, Eat EPaG, and Survival, the reward functions are all positive with different quantities and frequencies of rewards, while the Lose a Life goal is the only reward function with negative rewards. This means that users could accurately predict the Lose a Life goal solely by observing the signs of future rewards, while the other three goals required additional attention and knowledge of the goal's reward functions to comprehend the future rewards. Utilising the confusion matrix of Figure 6.6, we can decipher that for Eat Dots and Eat EPaG goals, users selected the other three positive reward goals (Eat Dots, Eat EPaG, and Survival) roughly evenly with 31, 34 and 32 for Eat Dots and 27, 31, and 24 for Eat EPaG. For Survival, we observe that users often confuse the goal for the Eat Dots goal, unlike the other two positive reward goals, with Eat Dots being selected 42 times compared to 27 and 28 for Eat EPaG and Survival, respectively. In summary, the TRD Summarisation mechanism for end-users is effective when the reward function differs in a significant and obvious quality; however, most users cannot identify the subtleties of reward frequencies or quantities. The reason for this is likely a lack of technical knowledge about how RL agents act to maximise their reward over time; thus, users struggle to identify the reward frequency and quantities and correlate this to goals' reward functions. Therefore, TRD Summarisation could be viewed as user-specific (which requires necessary technical knowledge to understand) for its effective deployment. We note in Section 6.4 how future surveys and TRD Summarisation could be improved to alleviate this problem.

Optimal Action Description was included within this survey as a baseline algorithm to understand how users predict agent goals only with the next action, requiring that users "simulate" the goal options to select the correct answer. Like TRD Summarisation, users could predict one of the goals, Survival, more accurately than the other three. Further, from Figure 6.6, we can observe that users unevenly predict the Survival and Eat Dots goals with 144 and 111 selections compared to 59 and 86 for the Eat EPaG and Lose a Life goals. One of the reasons for this skewed selection distribution is that OAD doesn't consider if other agents share the optimal action, such that if a significant number of Eat Dots and Survival agents shared actions, then this could explain the sampling bias. Reviewing the optimal actions for the different agent goals for the 20 observations, we found a common optimal action for nine, of which five were for Eat Dots and Survival. While this may partially explain the oversampling, we don't believe it can fully account for the significant difference in the observed sampling quantities. An alternative contributing factor might be human bias for particular goals we prefer, e.g., Eat Dots and Survival, compared to Lose a Life, when favouring outcomes or behaviours.

We observe the lowest average accuracy for SARFA (Gupta et al., 2020) of 22.5%, below that expected of random guessing. A unique feature of user predictions for SARFA is that users consistently misinterpret the agent's decision-making for a different goal. From Figure 6.6, we can observe that users consistently predict different goals for each goal option: the Eat Dots goal is predicted as the Eat EPaG goal, Eat EPaG as Survival, Lose a Life as Eat Dots, and Survival as Eat EPaG. We believe the cause of this confusion is that for saliency maps to be effective, the features humans view as important for an agent's goal must align with the neural network's feature importance. Otherwise, users will consistently misinterpret an agent's feature importance, mistaking an agent's goal for a different one. We observe this most strongly for the Lose a Life goal, where the most selected goal is actually the Eat Dots goal, with 37 compared to the next highest of 25.

TABLE 6.5: Table of p-values for each explanation mechanism of the time taken for correct and incorrect answers.

Explanation Mechanism	Number of correct / incorrect samples	Test Statistic	p-value
DSE	213 / 187	0.055	0.900
OAD	115 / 285	0.158	0.029
SARFA	91 / 309	0.086	0.632
TRD Summarisation	140 / 260	0.064	0.819

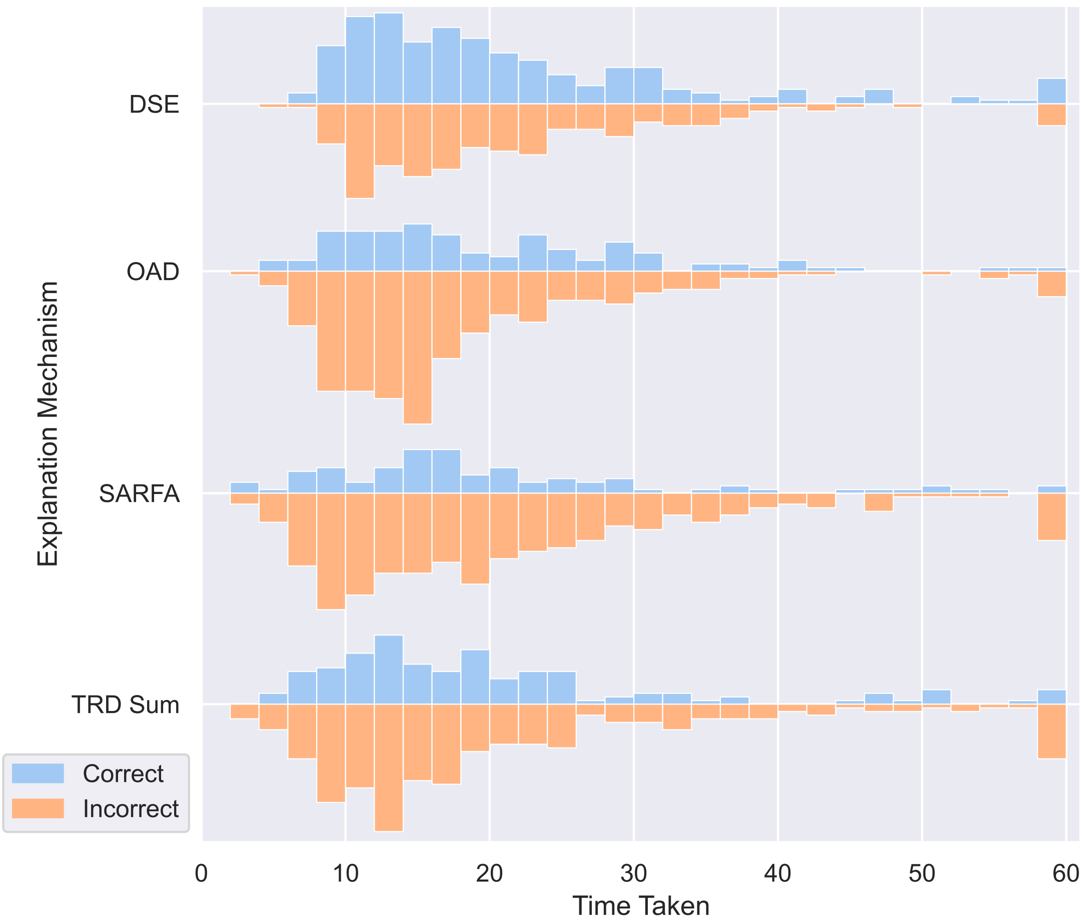


FIGURE 6.7: A histogram of the time taken for each explanation mechanism. The time taken is clipped at 60 seconds.

From Figures 6.4, 6.5, and 6.6, we have identified each explanation mechanism's accuracy, but is this accuracy affected by the time users take to select a goal? Figure 6.7 plots a histogram of the time taken for each question split by whether the user answered the question correctly or not. To formally test this, we compute the p-value using the KS test (Massey Jr, 1951) where the null hypothesis is that the time taken for correct and incorrect answers is from the same distribution. We set the significance level at 0.05 (5%), meaning we must reject the null hypothesis if the p-value is less than the significance level. Table 6.5 lists each explanation mechanism with the number of samples from both distributions, the Test Statistic measuring the degrees of differences between samples, and the p-value representing the probability of observing the results from the null hypothesis. For DSE, SARFA, and TRD Summarisation, their p-values are significantly greater than the significance level at 0.900, 0.63,2 and 0.819, respectively. This means that we can't reject the null hypothesis, and most likely, the distributions for the time taken for incorrect and correct answers are the same. However, for Optimal Action Description, the p-value is 0.029, meaning there is evidence to reject the null hypothesis, and we cannot conclude that the time taken for incorrect and correct answers is sampled from the same distribution. Figure G.2 plots the cumulative density functions of each explanation mechanism, from which we can observe that for OAD, users who took more time were more likely to get the correct answer. This is because, with OAD, users need to simulate or imagine each goal using the action description to see if they match before selecting the goal. This additional thinking time was most likely used to simulate the goal, helping with the user performance. This makes OAD the only goal where the longer users thought about the explanation, the higher their performance was, indicating a higher complexity to fully understand and predict the agent's goal.

Does a user's total time taken for all questions correlate with their overall accuracy? Figure 6.8 plots a scatter graph of each user's total time to answer all 16 goal identification questions and their accuracy with weak correlation ($R = 0.325$). Therefore, a link cannot be drawn between the two user properties. Of the time taken, most users took between 3 and 6 minutes, which could indicate that some might be randomly selecting an answer and moving on. Contrastingly, almost all users who took more than 8 minutes achieved over 35% average accuracy. To address this, in Section 6.4, we discuss the application of attention questions to discourage users from not engaging with explanations or fully reading descriptions.

Because the survey is sequential, and users iteratively answer similar questions, we investigate whether users exhibit a learning effect over time. As we provide no feedback to users after each question, we do not expect any significant change. In Figure 6.9, we plot the average user accuracy and time taken for the ordinal position of the question (irrespective of the explanation mechanism, goal or observation). The figure shows no clear change in accuracy over time, neither consistently increasing nor decreasing, with the average user accuracy for the first and last questions being 31.7% and 30.7%. This matches expectations that users do not explicitly learn more information over time, as they are not told if they are correct or incorrect for each question. Interestingly, this also implies that observing the other explanation

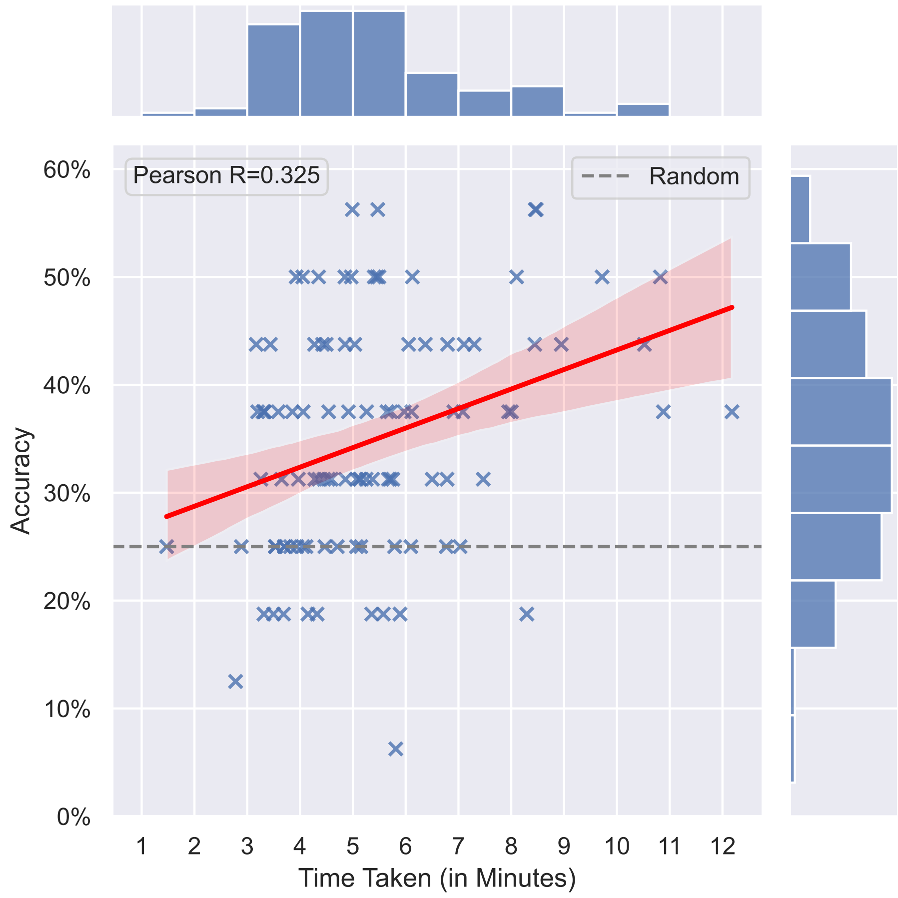


FIGURE 6.8: A scatter plot of each user's cumulative time taken against their accuracy across all questions, with the red line being a plot of the linear regression. Individual question time taken is clipped at 60 seconds. The dashed line is at 25%, representing the expected accuracy if users are selected randomly.

mechanisms doesn't influence the users for future explanations and that each explanation is independent, not helping users if they had seen particular explanations previously. In Figure 6.9, we also plot the average time for users to answer each question. In contrast to the user accuracy, the time taken progressively reduces as the survey progresses from an average of over 30 seconds on the first question to around 15 seconds for the last question. An additional quantity visible in Figure 6.9 is that the time taken substantially increases every fourth question, i.e., 1, 5, 9, and 13. This is as in every fourth question, users are shown a new explanation mechanism. This demonstrates that while user accuracy doesn't substantially change over an explanation mechanism's four questions, users generally take more time the first time they see each explanation mechanism and then tend to speed up over subsequent questions using the same explanation mechanism. This indicates that additional questions could be shown for an explanation mechanism and wouldn't substantially increase the overall time taken for the survey, a change we suggest for future user surveys (Section 6.4).

Finally, we investigate whether user characteristics collected from the survey, i.e., prior

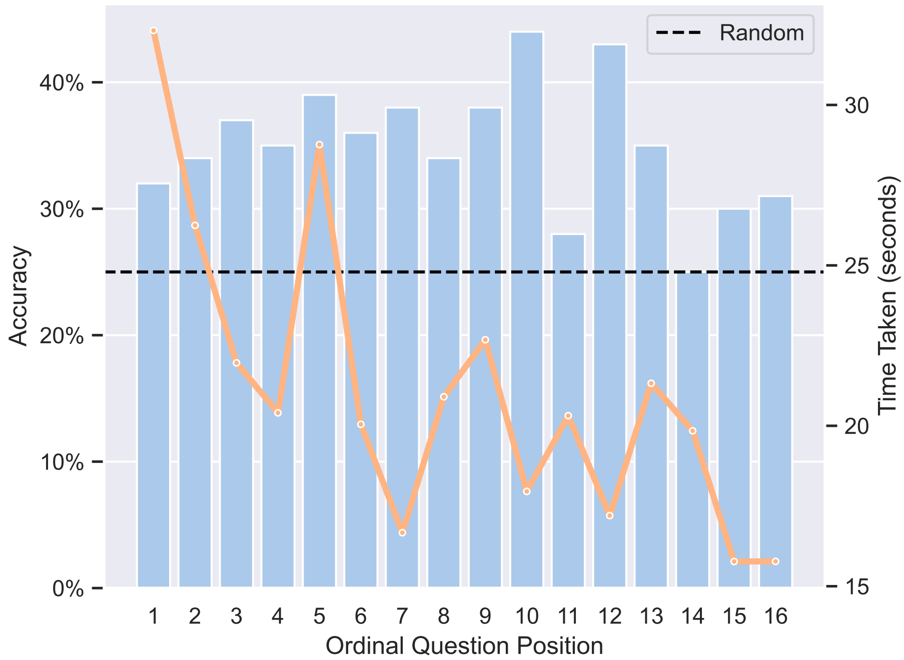


FIGURE 6.9: Barchart of the average accuracy (in blue) and a line plot of the time taken (in orange) for an ordinal question.

knowledge of AI or MsPacman, age, gender, and education level, affect user accuracy. Table 6.6 lists the number of users, average accuracy, and time taken in seconds for each characteristic. This data shows no significant impact on accuracy or time taken for any characteristic. To confirm this, we trained a linear regression model (using ordinary least squares) to predict a user's accuracy using their reported characteristics. We use this linear regression model to indicate if user characteristics can be correlated to their accuracy using the overall error (R^2) and the coefficients indicating a variable's importance to the predictions. As characteristics are categorical data, we one-hot encode each characteristic and include a constant column for a total of 19 columns. We find the R^2 value is 0.221, where 1.0 equals the perfect prediction, meaning the linear regression model can't reliably predict the user's accuracy from their characteristics. Further, the characteristic with the greatest coefficient was the constant, and the second being "Gender - Non-binary / prefer not to say" (Table G.1). This means no user characteristic (prior knowledge of AI / MsPacman, level of education, age, or gender) had a significant independent correlation to their accuracy. We further investigated the Variance Inflation Factor (James, 2013, Page 108) that measures multi-column correlations, also finding no column had a multicollinearity value greater than five, a standard cutoff value.

TABLE 6.6: A Table of User Characteristics with the number of users that selected the characteristics, the average accuracy, and the time taken (in seconds) for each question.

Characteristic	Number of Users	Accuracy (%)	Time Taken (s)
Prior experience with Atari MsPacman			
Never played and have no idea how the game works	3	31.3%	23.6
Have played once or twice and have minimal knowledge of the game	46	35.6%	22.2
Have played and understand the game well	51	34.6%	20.1
Do you have prior experience with AI?			
No	61	36.2%	20.0
Yes	39	33.0%	22.9
What level of education do you have?			
High School	31	35.3%	20.0
Bachelor's Degree	49	35.3%	21.0
Master's Degree	17	33.8%	22.8
Doctorate or Professional Degree (e.g., PhD, MD or JD)	3	31.3%	26.0
What is your age?			
18–24	16	39.1%	21.2
25–34	30	33.1%	20.3
35–44	27	39.1%	22.5
45–54	15	30.8%	21.5
55–64	8	27.3%	20.2
65 and older	4	34.4%	18.2
What is your gender?			
Female	57	33.1%	19.8
Male	42	37.2%	22.9
Non-binary / prefer not to say	1	43.8%	22.7

6.3.2 What do Users believe about the Explanation?

Within the survey, alongside the goal identification questions, we ask users to self-report their confidence for each answer and, for each explanation mechanism, their overall confidence, ease of identification, and understanding. In this section, we first analyse this subjective data for each "Which goal?" question to understand the relationship between the user's confidence and accuracy. We then investigate how users' overall ratings of each explanation mechanism correlate with their accuracy. Finally, we look at the optional comments made by users about each explanation mechanism and the survey as a whole.

TABLE 6.7: Table of Explanation Mechanism and user confidence with the user accuracy and their count in brackets.

Explanation Mechanism	Very Unconfident	Unconfident	Neutral	Confident	Very Confident
DSE	80.0% (5)	31.2% (32)	48.9% (92)	55.9% (213)	60.3% (58)
OAD	55.6% (9)	28.6% (42)	28.7% (129)	29.7% (172)	20.8% (48)
SARFA	21.4% (14)	16.9% (83)	25.2% (115)	23.8% (143)	24.4% (45)
TRD Sum	56.2% (16)	30.6% (72)	32.6% (132)	36.9% (149)	35.5% (31)

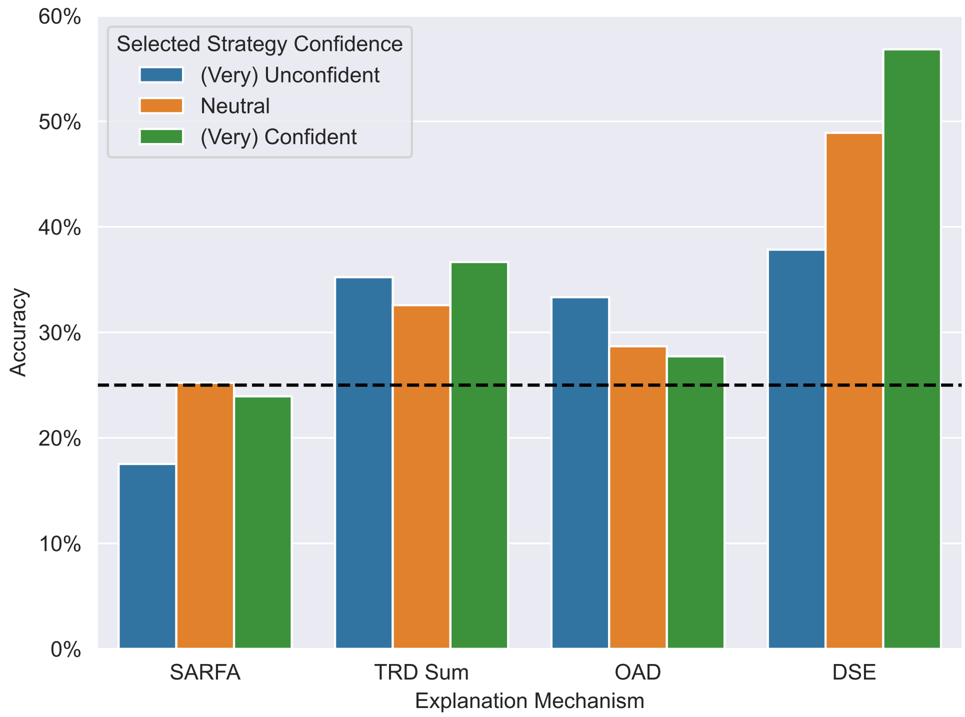


FIGURE 6.10: Bar chart of user accuracy with their selected confidence. Very unconfident and unconfident are grouped as (very) unconfident, and very confident and confident are grouped as (very) confident. The dashed line represents the expected accuracy with random guesses.

For each goal identification question, users select the goal that matches the explanation content and their confidence using a 5-point Likert Rating between Very Unconfident and Very Confident. Table 6.7 lists each explanation mechanism's accuracy for each confidence level. For Figure 6.10, to visualise Table 6.7, we group Very Confident and Confident into (Very) Confident due to the limited number of answers for Very Confident. We repeat this for Very Unconfident due to the same respective result. For DSE, there is a clear correlation between the user's confidence and their accuracy, with each confidence rank increasing the expected accuracy. In contrast, OAD has the opposite confidence-accuracy correlation, decreasing for each confidence rank. For TRD Summarisation, user accuracy isn't correlated with accuracy, with the lowest number of users who selected Very Confident at 31 (out of 400) questions and the most with Neutral confidence at 136. For SARFA, it had the most users who were (Very)

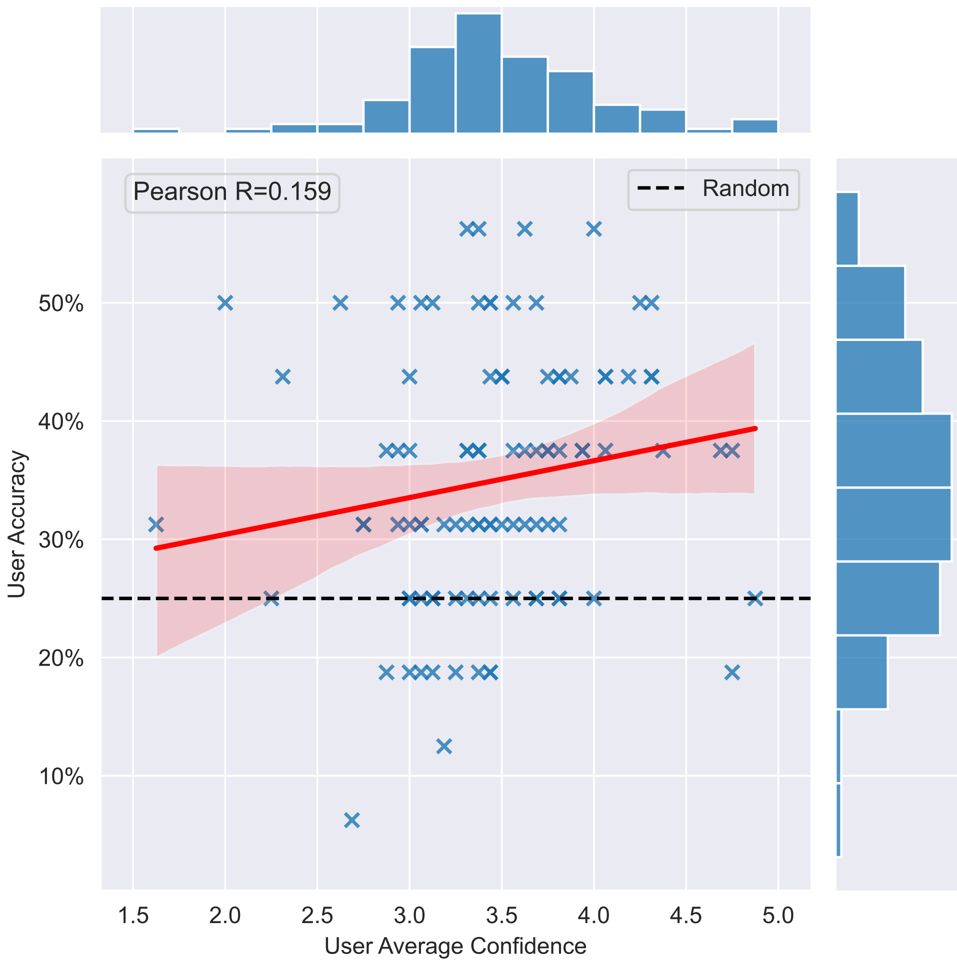


FIGURE 6.11: Scatter plot of each user's average accuracy and confidence for all explanation mechanisms. For the user average confidence, we discretise to Very Unconfident=1, Unconfident=2, Neutral=3, Confident=4, and Very Confident=5.

Unconfident at 97 (out of 400), for which they have significantly lower accuracy at 17.5% than the other confidence ranks, Neutral and (Very) Confident, at 25.2% and 23.9% accuracy. However, it should be noted that the users selecting Very Confident still only achieved an accuracy very close to that expected of random selection. This highlights a general problem of only using self-reported metrics for understanding an explanation's effectiveness with user confidence (and other measures), which are sometimes positively correlated, but in other cases, there is no correlation, and even a negative correlation.

From our testing on the individual question correlation between user confidence and accuracy, is this a feature of general user behaviour in their confidence to accuracy? In Figure 6.11, we present a scatter plot of each user's average accuracy and confidence (converted to discrete values of 1.0 for Very Unconfident to 5.0 for Very Confident) for all 16 questions. We observe that average user confidence is weakly positively correlated to their accuracy ($R = 0.159$). However, as seen in the histogram above, most

TABLE 6.8: Average user rating for each explanation mechanism on the three overall rating questions with the Likert Ratings discretised (1.0 for the lowest rating and 5.0 for the highest rating).

Explanation Mechanism	**Selection Confidence**	**Ease of Identification**	**Explanation Understanding**
DSE	3.48	3.23	3.56
OAD	3.25	3.06	3.50
SARFA	2.98	2.72	3.23
TRD Summarisation	2.94	2.61	3.01

of the average user confidence is concentrated between Neutral (3.0) and Confident (4.0). It is also interesting to note that there are more outliers with lower average confidence and high accuracy (top left) than users with high confidence and low accuracy (bottom right). This has important consequences for research that finds users have low confidence in an explanation, as it doesn't necessarily mean the explanation is ineffective.

After answering each explanation mechanism's goal identification questions, users select their overall confidence, ease of identification, or understanding of the explanations. Therefore, we investigate how users view their overall rating related to their performance in the explanation mechanism. Figure 6.12 plots the number of users who select a Likert Rating and the number of questions correctly answered.[5] From this data, we can observe that DSE is the only algorithm with more users who rate their selection confidence as Confident rather than Neutral, and that user performance positively correlates with accuracy. Meanwhile, SARFA has the highest level of users who select Unconfident in their predictions, and TRD Summarisation has a significantly larger number of users who selected that they are Very Unconfident in their prediction, compared to the other mechanisms. Despite this, for TRD Summarisation, OAD and SARFA, there isn't a correlation between the average user accuracy and confidence rating selected. Additionally, these overall confidence ratings don't significantly differ from the individual question user confidence result, either in accuracy or count. Viewing the average overall ratings in Table 6.8, each question follows the same explanation preference order (DSE, OAD, SARFA, and TRD Summarisation), with DSE having the highest average rating and TRD Summarisation the lowest.

For the overall ratings, we asked two more questions of users compared to the goal identification questions: "How easy was it to identify the agent's goal based on the explanation provided?" and "How well did the explanation help you understand the agent's goal?". Like overall user confidence, TRD Summarisation had significantly higher numbers of users who selected that they found the explanation "Very Difficult"

[5]Due to the limited numbers of users who rated any explanation mechanism as either the highest or lowest Likert Rating, e.g., Very Confident or Very Unconfident, Figure 6.12 combines these selections into the second highest and lowest ratings. Table G.2 includes the accuracy and user counts for all ratings, and Figure G.5 for the per-user data.

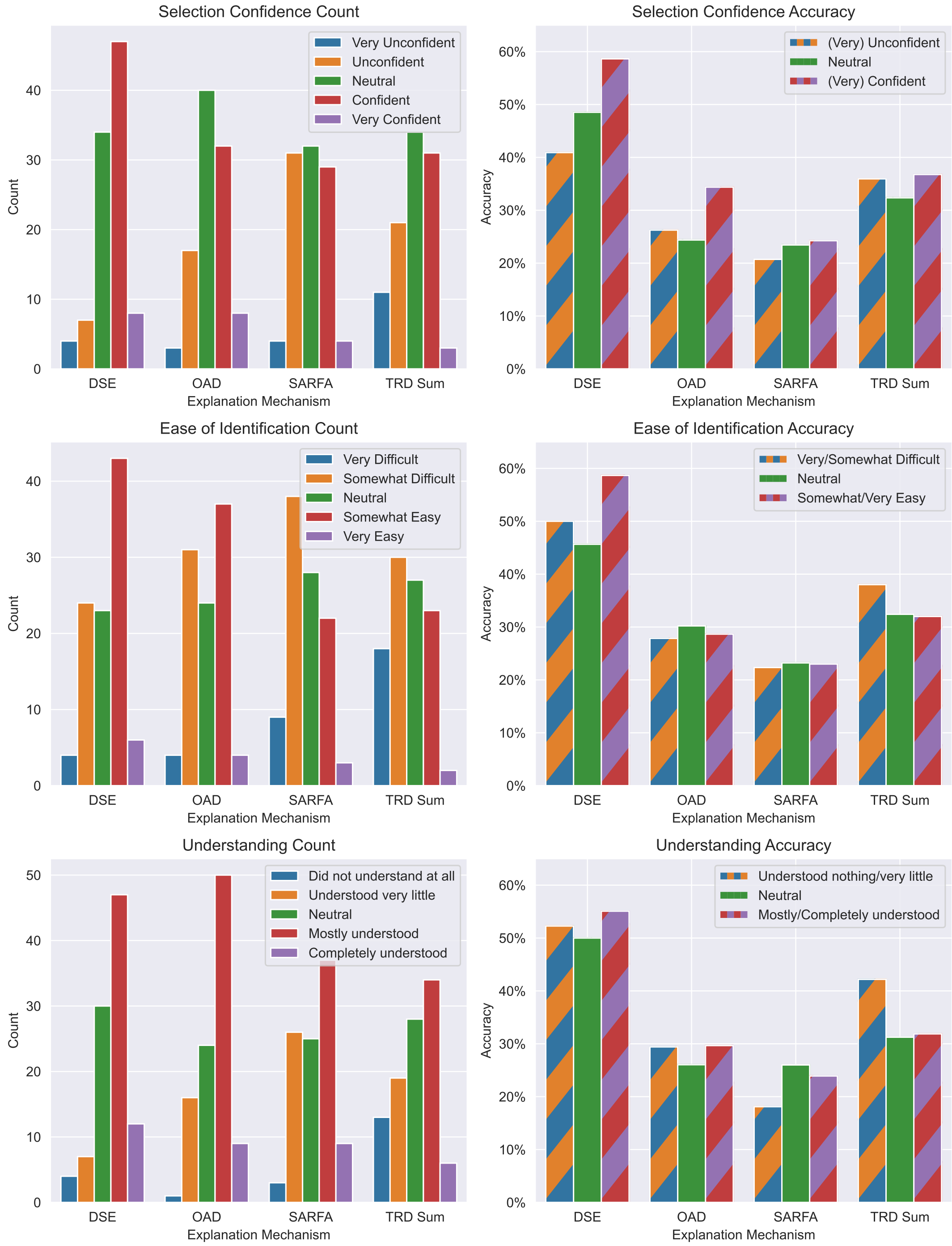


FIGURE 6.12: Histogram of the number of times on the left and the accuracy for each explanation mechanism, confidence, ease of identification, or understanding is selected by users.

in ease of identification for the goal and “Did not understand anything at all” from the explanation. This presents a probable failure to explain the explanation mechanism to users before the goal identification questions and/or that users lacked the skills to interpret the explanations. This is ironic as the natural language summarisation (Section 6.1) attempted to alleviate this anticipated problem.

Finally, we summarise the optional additional thoughts that users provided for each explanation mechanism and the whole survey (with all user comments provided in Tables G.3 and G.4). As the questions were optional, there is a sampling bias in the responses. For TRD Summarisation, most comments were on the difficulty of interpreting the explanations: “this was more difficult to interpret”, “... I didn’t understand the graph part in relationship to the Pacman part”, and “this one was more difficult than the others”. These comments reflect the overall user ratings for TRD Summarisation, which showed that users found the explanation more difficult to understand than the other (Figure 6.12). One user commented, “... without being more involved in that sort of mathematics I would not feel confident identifying them from a graphical analysis”. This implies that the explanation summary stage for TRD failed to adequately explain what the explanation means to users. Though one user did comment on the explanation’s effectiveness, “I think this seemed like a better way to see into the Agent’s ‘mind’ and make a guess as to what it was planning.” reflecting the goal of the explanation mechanism, showcasing the agent’s expected future rewards. We discuss in Section 6.4 how this problem can be rectified. For SARFA, users commented that “whilst the heat map gives a clue, it was much harder to discern without movement or the wider context” and “the heat maps don’t show the chronology of movement, just an average of all movement”. This highlights a generally noted problem with saliency maps in RL, as they are static and struggle to present the temporal component of agent decision-making. For OAD, several users commented that the optional action was not possible: “action described wasn’t possible”, “this was tricky when it suggested directions that would block” and “there were frames were ***[sic]*** it said ‘move right’ or ‘move down’ when this was not possible”. This can be fixed in future surveys by including a comment during the explanation summary that some actions might not be possible to complete. One user commented that they valued knowing the agent’s intentions “I found it a bit easier as I knew the intentions”, something that is not specified for the other three mechanisms. For DSE, the comments don’t share a common theme: “I think it’s a small screen and difficult to see”, “Some videos I thought I understood, others I didn’t understand the intention”, and “the videos weren’t long enough to judge the goal accurately”. These comments highlight survey parameters that can be addressed (video size and length) rather than systemic problems with the mechanism. For general comments on the survey, we had both users who expressed difficulty at the survey (“Its very tricky for me”, “I feel dumb after that” and “it was difficult to be confident without feedback”) and those who enjoyed the survey (“A very enjoy survey.”, “It was a really interesting survey” and “interesting.

love the game. thanks"). A general point noted in explanation mechanisms is that some images were very small on some users' screens. This can be solved by enabling users to increase image sizes or specifying image widths as a percentage of the screen width.

6.4 Discussion

Building upon the prior research in Chapters 4 and 5, we have explored improving the interpretability of visualising TRD's future expected rewards through natural language summarisation using an LLM (Section 6.1, TRD Summarisation) and proposed and conducted a novel evaluation methodology for XRL containing objective and subjective questions (Sections 6.2 and 6.3).

When implementing the TRD Summarisation, we found that prompt engineering, the fine-tuning of the prompt, was critical. Once refined, in Table 6.1, we show that the LLM could produce effective summarisations both with and without domain knowledge for sparse and dense rewards. Disappointingly, including observations in the prompts did not significantly change the summaries generated. We believe this is because LLMs, GPT4o in our case, are most likely not trained to understand Atari frames, limiting their capability to link the expected reward and the observation features. Despite this, overall, we believe these summarisations should significantly increase the accessibility of TRD to users without technical knowledge of reinforcement learning.

Taking inspiration from the application of explainability to debug/understand a trained RL agent, either pre-deployment to assess for flaws or post-deployment to identify why a decision was taken. We devised a novel evaluation methodology, proposed in Section 6.2, conducted with 100 participants, with the user answers analysed in Section 6.3. Overall, our Dataset Similarity Explanation in Section 4.1.1 had the highest accuracy (53.0%) and user ratings. Meanwhile, our TRD Summarisation explanation (Section 6.1) had the second highest accuracy of 34.9% but the lowest overall user ratings for every category (Table 6.8). The two baseline explanations achieved close to expected random performance at 28.7% and 22.5% for Optimal Action Description and SARFA (Gupta et al., 2020). Overall, we find the following features of the survey:

- **Low accuracy** - The average user accuracy for the survey was 34.9%, with the lowest being 6.3% and the highest being 56.3%. Against an expected random accuracy of 25.0%, only one of the explanation mechanisms (DSE) achieves accuracy success significantly above random for all agent goals. While we have not tested a wide variety of explanation mechanisms, we believe this survey demonstrates that there are still substantial performance gaps within the literature. We discuss in Chapter 7 how this can begin to be solved.

- **Self-reported questions cannot determine effectiveness** - In Section 3.3, our literature survey finds that a significant quantity of research on XRL with user surveys solely uses self-reported questions (including our research in Chapter 4). We identified that self-reported measures can only specify an explanation's preference, not its effectiveness for an application. The evaluation methodology proposed in this Chapter centres on an objective question to assess an explanation's effectiveness, "What goal?". Comparing users' answers for the self-reported and objective questions, we find a weak correlation between user confidence and their accuracy, and mostly no correlation between the overall explanation ratings and accuracy. These results highlight a graver issue: not just that self-reported measures can only specify an explanation preference, but that these answers provide very weak or even misleading information on the explanation's effectiveness. This poses further problems in analysing user survey results from the literature, as only a few (though growing) number of papers conduct user surveys with objective user questions.
- **Overconfidence** - Like self-reported questions, user confidence is a critical feature to correlate correctly with user accuracy, as in real-world situations where true answers don't exist, then we don't want users to be misled by an explanation. Ideally, in XRL, we hope that users only select that they have high confidence in an answer when they are correct. However, in Section 6.3.2, we find that users' confidence is not correlated with their accuracy, implying that users were tricked into believing they knew the answer when they didn't. This possibly poses a critical problem for applying explanations in real-world situations where misunderstanding an explanation can result in significant consequences.

If conducting the survey again, we would consider changing the survey in the following ways:

- **Improve survey/explanation descriptions** - In the optional thoughts from users, several noted that they were either confused by what they needed to do in the survey or by the explanations themselves. If conducted again, further time would have been spent optimising the descriptions of the explanation mechanisms to ensure that the users could interpret the explanation most effectively. This could be achieved through a pilot study.
- **Increase the number of questions asked** - The median time for users was 14 minutes to complete the survey, including 16 goal identification questions across four explanation mechanisms. With only four questions per explanation mechanism, this provides relatively little data that is more sensitive to user mistakes. Therefore, if conducted again, we would explore increasing the number of goal identification questions for each explanation mechanism to five or six;

alternatively, reducing the number of explanation mechanisms to three could increase the number of questions to ten or twelve. This should make the results more resilient to user mistakes.

- **Add attention checks** - For this survey, we included comprehension questions at the beginning to check that users understood the survey. However, we did not include attention checks throughout the survey to validate that users consistently checked and read the prompts/explanations. Adding such checks could help prevent super-speedy users from completing the survey without the necessary attention.

- **Policy Optimality** - For this survey, we trained deep reinforcement learning agents for 10 million steps with a relatively strong algorithm, DQN (Mnih et al., 2015) with an IMPALA neural network (Espeholt et al., 2018). However, we know from observing policy rollouts (videos of the agents) that they occasionally take sub-optimal actions from a human's perspective with knowledge of the action's goal. It is difficult to assess, but the sub-optimality of agents could cause issues explaining behaviour that is irrational (sub-optimal) from a human's perspective. The use of more powerful training algorithms, e.g., Rainbow DQN (Hessel et al., 2018) or Beyond the Rainbow (Clark et al., 2024) that are trained for longer, e.g., 200 million steps, might minimise this issue if it exists.

Chapter 7

Conclusion and Future Work

In this thesis, we have investigated the importance of temporality for Explainable Reinforcement Learning (XRL), a relatively undeveloped research topic within the field. From this desire to understand and explain an agent's decision-making about the future, we developed several explanation mechanisms for implicit and explicit policy features: skills and future rewards with Chapters 4, 5, and 6, respectively. In particular, Chapter 4 investigated post-hoc agents' groups of actions (skills) and which might be taken in the future, and how skills might be used together to complete the environment's goal of maximising its rewards, referred to as the agent's plan. In Chapter 5, we took a different direction to investigate an agent's predictions about their future rewards, providing greater understanding to researchers on the agent's beliefs/confidence about collecting individual future rewards. To validate these algorithms, we conducted two user surveys using baseline algorithms, finding that our temporal explanations are preferred by users and, in the second survey, were better at identifying agent strategies. We've shown the potential of temporal explanations for XRL and believe that further research is warranted. Next, we identified patterns from this thesis that we think other researchers and the field should learn, as well as the limitations of the research conducted, and then extensions and improvements that could be made.

Across our research, we identify two common features that impact, and we think, should be learned by the field more widely. First, researchers should not trust user self-reported questions as a source of objective explanation performance, only as a measure of user preferences. In Section 6.3.2, we found that self-reported user preferences rarely correlated with the user's performance with the explanation for our task. Therefore, researchers should not infer a causal link between user preferences and an explanation's true effectiveness. Adopting this would influence the design and conduct of future work evaluating explanations, such that researchers utilise evaluation methodologies that centre on objective user questions with known answers, like the one proposed in Section 6.2, where user preference can be collected as a secondary quality. Secondly, neural networks and reinforcement learning agents do not necessarily learn policies or features similar to those of humans. We found this when evaluating the discovered skills in Section 4.3.2 and using saliency maps to identify agent strategies in Section 6.3. Researchers should be aware of how to confirm that the feature accurately represents how the agent thinks and to evaluate whether a feature matches human expectations.

For the broader field, it is important to note that the user survey results collected are limited to the video games, i.e., Atari. The application of video games, while a common setting for testing reinforcement learning agents, is unknown how explanation's effectiveness will translate to more real-world, safety-critical settings, such as robotics, healthcare, or automated driving. One reason to think there may be a difference is how users treat video games as more playful, with minimal consequences for their decisions, compared to safety-critical agents' decisions, which may be more closely examined due to their potential impacts. This problem is widespread, with the majority of XRL research being conducted within video-game-like environments (Section 3.2). A challenge that future researchers could address by developing more explicitly safety-critical environments. To adapt our explanation mechanisms to these safety-critical settings, minimal changes would be necessary, as our algorithms were developed around RL principles rather than video game components. For example, Chapter 4 is based around visualising an agent's actions and using a dataset of prior actions, and Chapters 5 and 6 on agents receiving rewards over time. These features are expected in all RL settings, though they are particularly prevalent within video games with fast actions and easy-to-understand scores to maximise. Adapting for safety-critical settings, we believe, will primarily centre on how to present the explanations or how to contextualise them for users, rather than needing to modify the central working of the explanation mechanisms proposed in this thesis.

The research methodology used in this thesis is that we first developed algorithms to explain a particular RL agent's behaviour, and then explored how this could be presented to users. The approach that appears most commonly in the XRL literature. An alternative approach, most common in Human-Computer Interface (HCI) research (Interaction Design Foundation - IxDF, 2016) and present in a few XRL papers (Lucas et al., 2024; Madumal et al., 2019), is first to gather user requirements or designs and then develop explanations to replicate these specifications. A result of the difference in methodology means that there may be numerous ways our explanations could be improved, solely through how the information is presented or contextualised for users, beyond any technical changes to improve our explanations. Further, in this thesis and XRL literature evaluations have centred on user preferences and the actionability explanation for users to answer questions. Future research could be inspired by HCI for evaluations of explanations, testing user experience of explanations beyond preferences, e.g., eye tracking to understand user focus, mental load using physiological signals (skin response, heart rate, EEG), or contextual inquires where structured interviews are conducted to understand how users utilise explanations and difficulties with them (Song et al., 2023).

After reviewing the research we completed, we believe that several components of the proposed explanation mechanisms can be further improved or combined with prior work for greater potential impact. In Chapter 4, we propose video-based explanations, one of the most under-utilised explanation mediums (Table 3.1) compared to images, natural language, or symbolic/mathematical approaches. For RL, where agents take sequences of actions over time, videos should be a more prevalent medium, which we demonstrate their effectiveness in Section 6.3 against saliency maps (images) and text summarisation (natural language) explanations. For Temporal Reward Decomposition (TRD) proposed in Chapter 5 and extended in Section 6.1, we believe this provides a significant capability in any other XRL algorithm, explaining interesting and important qualities of deep RL algorithms that are scalable to complex environments. We can envision several extensions that combine TRD's capabilities with prior XRL and RL algorithms to improve its explainability.

1. The advantage of the Dataset Similarity Explanation (DSE, Section 4.1.1) is that the videos produced are easy to understand for users (Section 6.3). While the advantage of TRD is that it provides access to the "true" thinking of the agent, we found that users struggled to interpret the graphical explanations (e.g., Figure 5.5). Therefore, we believe it is feasible to combine the two approaches to get both of their advantages, providing a video of an agent's behaviour and a description of the anticipated rewards of an agent. These explanations could be similar to [1], though from a memory of an agent's prior behaviour, rather than online.
2. Rewards in RL are traditionally scalar, which recent work (Felten et al., 2024) has challenged, recognising that complex environments often have several competing reward sources for agents to optimise, referred to as Multi-Objective Reinforcement Learning (MORL). TRD assumes the use of scalar rewards while prior work (Juozapaitis et al., 2019) has proposed decomposing an agent's future reward estimator into the environment multi-objective reward components, e.g., coins and chests in Figure 5.1 using this MORL framework. Reward Component Decomposition (Juozapaitis et al., 2019) can be combined with our Temporal Reward Decomposition to explain the when and quantity of each reward component for the next N timesteps. This should improve TRD for environments like Figure 5.1 or robotics environments where researchers wish to understand how agents trade off different reward sources in time. This can be implemented by expanding TRD's network output for the number of reward components and the loss function to minimise the error for each reward component and timestep.

[1] https://github.com/pseudo-rnd-thoughts/temporal-reward-decomposition/blob/main/figs/Breakout-expected-reward.mp4

3. In Section 5.3.1, we show how the agent's confidence in a future reward can be computed, conditional on the environment reward having a binary structure. This limits TRD for RL environments that don't contain a binary reward function, which is common outside Atari. Therefore, we propose utilising research from distributional RL (Bellemare et al., 2017) where agents learn the probability distribution of the Q-value rather than the scalar sum of future rewards. We believe this probability distribution learning approach could be applied at the individual reward level, computing the distribution of each predicted reward. This would enable understanding an agent's confidence in the next N future reward, regardless of an environment's reward function structure. As a result, extending the explanation mechanisms proposed in Section 5.3.1.

4. TRD is primarily effective in environments containing dense rewards where several elements of the reward vector are non-zero. However, this is not true in sparse or episodic reward scenarios where almost all rewards are zero. One approach to solving this could be to scale N, the number of time steps explained, beyond that tested in this work, e.g., 100. However, we believe there will be a practical limit to the scale, and more importantly, noise will cause interference when interpreting the expected future rewards. Addressing this weakness would require learning the Q-value distribution over time. This might be possible by modelling the Q-value as a cumulative density function over time, where Dabney et al. (2018) showed infinitely wide probability distributions can be modelled with neural networks. We propose constructing a separate function that can be learned and explain a Q-value to infinity (and beyond) through a novel recursive optimisation function and loss function.

In Section 3.4, we lament the inability to identify the state-of-art for XRL, limiting any discussion about the effectiveness of different algorithms from the lack of comparative user surveys on common benchmarks. Using the evaluation methodology proposed in Chapter 6 as the foundation, we plan to submit a competition proposal to NeurIPS 2025. The competition would allow researchers to submit explanation mechanisms that are evaluated to explain agents for a common problem alongside other submitted algorithms. If accepted and completed, this will provide a leaderboard of explanation mechanisms, all tested on the same problem case using a high-quality user survey containing objective and self-reported subjective questions. A longer-term hope would be that this competition could continue as a yearly competition, with the complexity of the problem increasing each year, allowing iterative development of new algorithms over time, with dreams that this will increase the health and growth of the field.

For inspirations from this thesis beyond XRL, as noted in Chapter 1, supervisor learning differs from reinforcement learning as a known answer exists for each output. This means that models take a single "step" before computing their error to the true label.

This is still true for text-based systems, but there is some sense that they are also trying to optimise for a sequence of characters, not just the next character in the sequence. More formally, in the case of Large Language Models, they output a probability distribution over possible tokens and aim to maximise the cumulative probability for a batch of tokens, $\sum_i^S P_{t_i}$ where t_i is the correct token at index i of the sequence. This function resembles the Q-value (Eq. (2.2)) where t_i would be the action a_i taken at timestep i. We posit that LLMs could expand their output probability distribution like TRD does for time to enable an explicit understanding of when a token could appear in the future sequence. If realised, this could allow researchers to understand when the model expected different tokens in the future sequence based on the prior tokens shown. This idea is in confluence with Qi et al. (2024) critiquing prior work for only investigating safety alignment "a few tokens deep" and wishing to understand LLMs for large sequences of tokens and the future impacts on token selection.

References

12 C.F.R. § 1002.1(b). Equal credit opportunity act of 2017, 2017. URL https://www.ecfr.gov/cgi-bin/text-idx?SID=1bb14fdfd1afc7d3f2d4756d223aeadb&mc=true&node=se12.8.1002_19.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *Advances in Neural Information Processing Systems*, 35:28955–28971, 2022.

Adrian Agogino, Ritchie Lee, and Dimitra Giannakopoulou. Challenges of explaining control. *OpenReview.net*, 2019.

Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Edward Lee, Sergey Levine, Yao Lu, Isabel Leal, Sharath Maddineni, Kanishka Rao, Dorsa Sadigh, Pannag Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Steve Xu, and Zhuo Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

Dan Amir and Ofra Amir. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems*, pages 1168–1176, 2018.

Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023.

Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *Frontiers in Big Data*, 4:39, 2021. ISSN 2624-909X. doi: 10.3389/fdata.2021.688969. URL https://www.frontiersin.org/article/10.3389/fdata.2021.688969.

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.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

Tom Bewley and Jonathan Lawry. Tripletree: A versatile interpretable representation of black box agents and their environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11415–11422, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019. ISSN 0036-8075. doi: 10.1126/science.aay2400. URL https://science.sciencemag.org/content/365/6456/885.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

Tyler Clark, Mark Towers, Christine Evers, and Jonathon Hare. Beyond the rainbow: High performance deep reinforcement learning on a desktop pc, 2024. URL https://arxiv.org/abs/2411.03820.

Youri Coppens, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*, pages 1–6, 2019.

Vinícius G Costa, Jorge Pérez-Aracil, Sancho Salcedo-Sanz, and Carlos E Pedreira. Evolving interpretable decision trees for reinforcement learning. *Artificial Intelligence*, 327:104057, 2024.

Council of European Union. Council regulation (EU) 2016/679, 2016. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679.

Francisco Cruz, Richard Dazeley, and Peter Vamplew. Memory-based explainable reinforcement learning. In *Australasian Joint Conference on Artificial Intelligence*, pages 66–77. Springer, 2019.

Francisco Cruz, Richard Dazeley, Peter Vamplew, and Ithan Moreira. Explainable robotic systems: Understanding goal-driven actions in a reinforcement learning scenario. *Neural Computing and Applications*, pages 1–18, 2021.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1:

Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Grégoire Déletang, Jordi Grau-Moya, Miljan Martic, Tim Genewein, Tom McGrath, Vladimir Mikulik, Markus Kunesch, Shane Legg, and Pedro A Ortega. Causal analysis of agent behavior for ai safety. *arXiv preprint arXiv:2103.03938*, 2021.

Senne Deproost, Denis Steckelmacher, and Ann Nowé. Human-readable programs as actors of reinforcement learning agents using critic-moderated evolution. *arXiv preprint arXiv:2410.21940*, 2024.

Thomas Dodson, Nicholas Mattei, and Judy Goldsmith. A natural language argumentation interface for explanation generation in markov decision processes. In *International Conference on Algorithmic Decision Theory*, pages 42–55. Springer, 2011.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno C. da Silva. A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

Florian Felten, El-Ghazali Talbi, and Grégoire Danoy. Multi-objective reinforcement learning based on decomposition: A taxonomy and framework, 2024. URL https://arxiv.org/abs/2311.12495.

FICO. Fico announces winners of inaugural xml challenge, Jan 2019. URL https://www.fico.com/en/newsroom/fico-announces-winners-inaugural-xml-challenge.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Omer Gottesman, Joseph Futoma, Yao Liu, Sonali Parbhoo, Leo Celi, Emma Brunskill, and Finale Doshi-Velez. Interpretable off-policy evaluation in reinforcement learning by highlighting influential transitions. In *International Conference on Machine Learning*, pages 3658–3667. PMLR, 2020.

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801. PMLR, 2018.

Niko Grupen, Natasha Jaques, Been Kim, and Shayegan Omidshafiei. Concept-based understanding of emergent multi-agent behavior. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Wenbo Guo, Xian Wu, Usmann Khan, and Xinyu Xing. Edge: Explaining deep reinforcement learning policies. *Advances in Neural Information Processing Systems*, 34: 12222–12236, 2021.

Piyush Gupta, Nikaash Puri, Sukriti Verma, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations (ICLR)*, 2020.

Balint Gyevnar and Mark Towers. Objective metrics for human-subjects evaluation in explainable reinforcement learning, 2025. URL https://arxiv.org/abs/2501.19256.

Balint Gyevnar, Massimiliano Tamborski, Cheng Wang, Christopher G Lucas, Shay B Cohen, and Stefano V Albrecht. A human-centric method for generating causal explanations in natural language for autonomous vehicle motion planning. *arXiv preprint arXiv:2206.08783*, 2022.

Balint Gyevnar, Cheng Wang, Christopher G. Lucas, Shay B. Cohen, and Stefano V. Albrecht. Causal explanations for sequential decision-making in multi-agent systems. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, page 771–779, Richland, SC, 2024. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400704864.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Sara Hajian, Francesco Bonchi, and Carlos Castillo. Algorithmic bias: From discrimination discovery to fairness-aware data mining. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2125–2126, 2016.

Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 303–312. IEEE, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Lei He, Aouf Nabil, and Bifeng Song. Explainable deep reinforcement learning for uav autonomous navigation. *arXiv preprint arXiv:2009.14551*, 2020.

Daniel Hein, Alexander Hentschel, Thomas Runkler, and Steffen Udluft. Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence*, 65:87–98, 2017.

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 *Thirty-second AAAI conference on artificial intelligence*, 2018.

Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

Shengyi Huang, Quentin Gallouédec, Florian Felten, Antonin Raffin, Rousslan Fernand Julien Dossa, Yanxiao Zhao, Ryan Sullivan, Viktor Makoviychuk, Denys Makoviichuk, Cyril Roumégous, Jiayi Weng, Chufan Chen, Masudur Rahman, João G. M. Araújo, Guorui Quan, Daniel Tan, Timo Klein, Rujikorn Charakorn, Mark Towers, Yann Berthelot, Kinal Mehta, Dipam Chakraborty, Arjun KG, Valentin Charraut, Chang Ye, Zichen Liu, Lucas N. Alegre, Jongwook Choi, and Brent Yi. Open rl benchmark: Comprehensive tracked experiments for reinforcement learning, 2023. URL https://github.com/openrlbenchmark/openrlbenchmark.

Tobias Huber, Katharina Weitz, Elisabeth André, and Ofra Amir. Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps. *Artificial Intelligence*, 301:103571, 2021.

Tobias Huber, Maximilian Demmler, Silvan Mertes, Matthew L Olson, and Elisabeth André. Ganterfactual-rl: Understanding reinforcement learning agents' strategies through visual counterfactual explanations. *arXiv preprint arXiv:2302.12689*, 2023.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Interaction Design Foundation - IxDF. What is user centered design (ucd)? https://www.interaction-design.org/literature/topics/user-centered-design, 2016. URL https://www.interaction-design.org/literature/topics/user-centered-design. Retrieved April 21, 2025.

Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Najat Drawel, Gaith Rjoub, and Witold Pedrycz. A comprehensive survey on applications of transformers for deep learning tasks, 2023. URL https://arxiv.org/abs/2306.07303.

Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. Transparency and explanation in deep reinforcement learning neural networks, 2018. URL https://arxiv.org/abs/1809.06061.

Gareth James. An introduction to statistical learning, 2013.

Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Eoin M Kenny, Mycal Tucker, and Julie Shah. Towards interpretable deep reinforcement learning with human-friendly prototypes. In *The Eleventh International Conference on Learning Representations*, 2023.

Greg Kohs. Alphago, 2017. URL https://www.youtube.com/watch?v=WXuK6gekU1Y.

Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. Toward interpretable deep reinforcement learning with linear model u-trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 414–429. Springer, 2018.

Wenhao Lu, Xufeng Zhao, Sven Magg, Martin Gromniak, Mengdi Li, and Stefan Wermterl. A closer look at reward decomposition for high-level robotic explanations. In *2023 IEEE International Conference on Development and Learning (ICDL)*, pages 429–436. IEEE, 2023.

Wenhao Lu, Xufeng Zhao, Thilo Fryen, Jae Hee Lee, Mengdi Li, Sven Magg, and Stefan Wermter. Causal state distillation for explainable reinforcement learning. In *Causal Learning and Reasoning*, pages 106–142. PMLR, 2024.

Caroline Lucas, Balint Gyevnar, Cheng Wang, Sanja Cohen, and Stefano Albrecht. Headd: Human explanations for autonomous driving decisions, 1 2024. URL https://doi.org/10.7488/ds/7867.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2970–2977, 2019.

J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. A grounded interaction protocol for explainable artificial intelligence, 2019. URL https://arxiv.org/abs/1903.02409.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Distal explanations for model-free explainable reinforcement learning. *arXiv preprint arXiv:2001.10284*, 2020a.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2493–2500, 2020b.

Saaduddin Mahmud, Marcell Vazquez-Chanlatte, Stefan Witwicki, and Shlomo Zilberstein. Explaining the behavior of pomdp-based agents through the impact of counterfactual information. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1346–1354, 2024.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

Tim Miller. Contrastive explanation: A structural-model approach. *The Knowledge Engineering Review*, 36:e14, 2021.

Tim Miller, Piers Howe, and Liz Sonenberg. Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Christoph Molnar. Interpretable machine learning, Aug 2021. URL https://christophm.github.io/interpretable-ml-book/shapley.html#advantages-13.

Menghai Pan, Weixiao Huang, Yanhua Li, Xun Zhou, and Jun Luo. xgail: Explainable generative adversarial imitation learning for explainable human decision analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1334–1343, 2020.

Ewa Paszek. Introduction to statistics. 2007.

William R Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms. *Genomics*, 11(3):635–650, 1991.

Erika Puiutta and Eric MSP Veith. Explainable reinforcement learning: A survey. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 77–95. Springer, 2020.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep, 2024. URL `https://arxiv.org/abs/2406.05946`.

Yunpeng Qing, Shunyu Liu, Jie Song, and Mingli Song. A survey on explainable reinforcement learning: Concepts, algorithms, challenges. *arXiv preprint arXiv:2211.06665*, 2022.

Ronilo Ragodos, Tong Wang, Qihang Lin, and Xun Zhou. Protox: Explaining a reinforcement learning agent via prototyping. *Advances in Neural Information Processing Systems*, 35:27239–27252, 2022.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016a.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016b.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Yael Septon, Tobias Huber, Elisabeth André, and Ofra Amir. Integrating policy summaries with reward decomposition for explaining reinforcement learning agents. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 320–332. Springer, 2023.

Pedro Sequeira, Eric Yeh, and Melinda T Gervasio. Interestingness elements for explainable reinforcement learning through introspection. In *IUI Workshops*, volume 1, 2019.

Andrew Silva, Taylor Killian, Ivan Dario Jimenez Rodriguez, Sung-Hyun Son, and Matthew Gombolay. Optimization methods for interpretable differentiable decision trees in reinforcement learning. *arXiv preprint arXiv:1903.09338*, 2019.

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):484–489, 2016.

Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

Xintai Song, Minxia Liu, Lin Gong, Yu Gu, and Mohammad Shidujaman. A review of human-computer interface evaluation research based on evaluation process elements. In *Human-Computer Interaction: Thematic Area, HCI 2023, Held as Part of the 25th HCI International Conference, HCII 2023, Copenhagen, Denmark, July 23–28, 2023, Proceedings, Part I*, page 262–289, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-35595-0. doi: 10.1007/978-3-031-35596-7_17. URL https://doi.org/10.1007/978-3-031-35596-7_17.

David Stirzaker. *Elementary Probability*. Cambridge University Press, 2003.

Zhan Su, Byung-Ryul Ahn, Ki-Yol Eom, Min-Koo Kang, Jin-Pyung Kim, and Moon-Kyun Kim. Plagiarism detection using the levenshtein distance and smith-waterman algorithm. In *2008 3rd International Conference on Innovative Computing Information and Control*, pages 569–569. IEEE, 2008.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Chris Sypherd and Vaishak Belle. Practical considerations for agentic llm systems, 2024. URL https://arxiv.org/abs/2412.04093.

Ahmad Terra, Rafia Inam, and Elena Fersman. Beerl: Both ends explanations for reinforcement learning. *Applied Sciences*, 12(21):10947, 2022.

Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2514–2521, 2019.

Mark Towers, Yali Du, Christopher Freeman, and Timothy J. Norman. Explaining an agent's future beliefs through temporally decomposing future reward estimators. In *ECAI 2024*, pages 2790–2797. IOS Press, 2024a.

Mark Towers, Yali Du, Christopher Freeman, and Timothy J. Norman. Temporal explanations of deep reinforcement learning agents. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, pages 99–115. Springer, 2024b. doi: 10.3233/FAIA240814.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024c.

Yuta Tsuchiya, Yasuhide Mori, and Masashi Egi. Explainable reinforcement learning based on q-value decomposition by expected state transitions. *CEUR Workshop Proceedings*, 2023.

UK Parliament. Data protection act 2018, 2018. URL https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, and Mark Neerincx. Contrastive explanations for reinforcement learning in terms of expected consequences. *arXiv preprint arXiv:1807.08706*, 2018.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad, 2018. URL https://arxiv.org/abs/1812.02648.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko

Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Xinzhi Wang, Huao Li, Hui Zhang, Michael Lewis, and Katia Sycara. Explanation of reinforcement learning model in dynamic multi-agent system. *arXiv preprint arXiv:2008.01508*, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

Stephen Wu. Tracing ai decisions: Ai explainability and the gdpr, Dec 2020. URL https://www.airoboticslaw.com/blog/ai-explainability-gdpr.

Jiayu Yao, Sonali Parbhoo, Weiwei Pan, and Finale Doshi-Velez. Policy optimization with sparse global contrastive explanations. *arXiv preprint arXiv:2207.06269*, 2022.

Herman Yau, Chris Russell, and Simon Hadfield. What did you think would happen? explaining agent behaviour through intended outcomes. *Advances in Neural Information Processing Systems*, 33:18375–18386, 2020.

Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. Explainable reinforcement learning via a causal world model. *arXiv preprint arXiv:2305.02749*, 2023.

Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*, pages 1899–1908. PMLR, 2016.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*, 2024.

Appendix A

Impact Assessment

As described in Chapter 1, the motivation for this work within Explainable Artificial Intelligence (XAI) and Explainable Reinforcement Learning (XRL) is the perceived social need for explanatory tools to understand machine learning algorithms, which are becoming an increasingly prevalent part of daily life. In this appendix, we utilise the Orbit Responsible Research and Innovation framework, AREA 4Ps, adopted by EPSRC (UK Engineering and Physical Science Research Council) to ensure that this work produces impactful research in academia and for the general public. Following the 4Ps of the framework (process, product, purpose and people), in Section A.1, we first consider the purpose of the research. Then, in Section A.2, we consider the importance of human studies to evaluate the effectiveness of explanations as a critical way of informing impactful research. This ensures that the product and the explanatory mechanisms match the users' (people) needs. Finally, in Section A.3, we consider how this research is conducted to ensure that the other 3Ps (product, purpose, and people) are being achieved.

A.1 The Motivation and Purpose of this Research

In Chapter 1, we presented a case study of AlphaGo that shows the importance of explanations for machine learning as a mechanism to increase trust and confidence that the algorithms work as intended. As a result, explanations have been viewed as an increasingly important feature of machine learning by academia, governments, and regulatory agencies, each with their motivations and applications (see Figure 2.4 for example stakeholders).

In academia, as Figure 2.3 shows, the number of papers with the keywords “explainable artificial intelligence” or “explainable reinforcement learning” has increased exponentially over the last few years. Currently, one of the primary applications of XAI

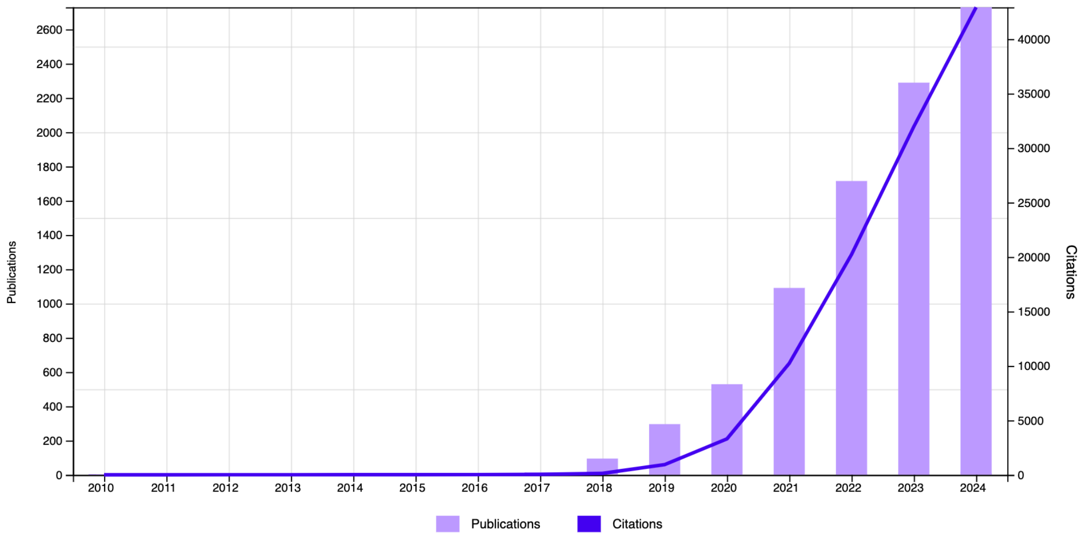


FIGURE A.1: The number of papers published each year from 2010 to 2024 that include "Explainable Artificial Intelligence" or "Explainable Reinforcement Learning".

is for researchers who wish to understand what their machine learning models are learning to debug and improve them. An example algorithm for this is LIME (Local Interpretable Model-agnostic Explanations, Ribeiro et al. (2016b)), which uses simpler machine learning models to try to understand the sections of an input with respect to an output. LIME has been used for text and image explanations, highlighting the most important pixels or words.

Arguably, a second important motivation has been the recent development of laws across the globe on the "right to explanation" (e.g., GDPR in the EU and UK, and credit scores in the US), the idea that people should have the right to understand how or why an algorithm outputted a particular decision for them. In the United States, it is federally mandated that credit risk scores being given to customers that determine if a bank will offer you a credit card, loan, mortgage, etc. (12 C.F.R. § 1002.1(b), 2017), must be "explainable" to ensure that customers are not denied these important services due to algorithmic bias or human prejudice. This is currently an open challenge to provide the most accuracy while explainable credit risk score algorithms with FICO, one of the leading credit scoring companies, who organised a competition in 2019 to explore this (FICO, 2019).

In the European Union, similar laws have been enacted with GDPR (General Data Protection Regulation, Council of European Union (2016)), which came into effect in 2018, requiring companies in certain situations to provide "an explanation for the decision reached".[1] This is important because while the law provides example settings where the explanation is required (e.g., online credit applications or e-recruiting

[1]GDPR has been enacted into British law, post-Brexit, as the Data Protection Act 2018 (UK Parliament, 2018).

practices), it is not a complete list, making it unknown how far this law could be applied without legal precedence.

> The data subject should have the right not to be subject to a decision, which may include a measure, evaluating personal aspects relating to him or her which is based solely on automated processing and which produces legal effects concerning him or her or similarly significantly affects him or her, such as automatic refusal of an online credit application or e-recruiting practices without any human intervention.
>
> ...
>
> In any case, such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such an assessment and to challenge the decision.

As a result of such regulations, some research has explored explainable algorithms that provide mathematical guarantees to the explanations provided. The most popular approach for this is SHAP (SHapley Additional exPlanations, Lundberg and Lee (2017)), which uses game theory to provide a precise mathematical approach to understanding which input parts are most important for an algorithm. As a result, some researchers have suggested that SHAP is the only legally compliant explainable method due to its mathematical properties (Molnar, 2021).

We believe that these two laws (12 C.F.R. § 1002.1(b), 2017; Council of European Union, 2016) are likely to be expanded in the future (Wu, 2020) to include cases involving reinforcement learning, and the current academic interest gives strong empirical evidence for the social desirability of this research.

A.2 The Use of Human Studies for Impactful Research

As explained in Section A.1, there are strong motivations for the research we conduct in this thesis. In this section, we discuss how human studies ensure that the research conducted is not just of academic interest but is impactful and important for a range of real-world applications and people.

By placing human studies front and centre in this research, we believe that this forces researchers to consider the system's potential applications and ethics and actively engage with public stakeholders. These actual people would use the system as the primary metric of success. To assess the possible applications of XAI and XRL, we will consider the places where humans most commonly and importantly require explanations. These are primarily environments where machine learning interacts or collaborates with humans such that the resulting decision by a model can have a noticeable impact on a person, e.g., medical diagnosis or jail sentencing. It should be noted that this list of examples will continue to grow as machine learning is applied to an increasing number of areas in people's lives.

When considering the application of research, it is important to consider possible misuse of the technology. The only misuse we believe with this work is the problem of *uncritical trusting*, a problem that also exists with humans in that people overly trust another person (or a machine learning model in this case). When this happens, people do *not* critically assess whether the explanations provided are true and can have unintended consequences. This can be very difficult when the machine learning models are better than humans in the application, making evaluating model output difficult even with expert knowledge. An example is chess, where computer engines are popularly used to evaluate a board position. However, commentators have often noted that this is unreliable for assessing a board state for the two players, as it assumes that both players make the optimal move each turn, which is normally impossible for humans. In addition, it is more difficult because, depending on the strength of the chess engine and the time allowed to evaluate the board position, computer engines can arrive at completely different evaluations and move selections. We believe that it is likely that an analogue problem could exist in XAI where, depending on the complexity of the explanatory mechanism, different explanations could be provided that are difficult to compare and, importantly, difficult to assess which explanations are true.

For the stakeholders involved, we believe that human studies help survey them to ensure that the research is impactful and effective for them. In addition, this allows researchers to assess what training is required to understand the project and the intended and unintended consequences for people. As shown in Figure 2.4, there are a range of stakeholders, each with motivations, applications, and expected explanations. For this work, our primary stakeholders are non-expert users and developers, who we believe are a critical demographic for explanations in a wide range of applications in the future.

A.3 Conducting Research with the Right Purpose and People

In the previous two sections (Sections A.1 and A.2), we have discussed the role of designing research with the right purpose and people in mind so that the resulting research is impactful. In this section, we discuss the actual research completed, how it links to the previous two sections and if it is impactful.

In Chapter 4, we presented three video-based temporal explanation mechanisms for image-based and complex environments. This research helps produce important incremental improvements to post-hoc temporal explanations that do not require domain knowledge. This is confirmed by the user survey in Section 4.3.1 showing that users significantly preferred our novel temporal explanation over prior state-of-the-art feature-based explanations (Selvaraju et al., 2017; Greydanus et al., 2018). In Chapter 5, we demonstrate how to extract normally implicit beliefs for RL agents by modifying an agent's output function. Our explanations can extract previously hidden information on an agent's beliefs about the future (Section 5.3). Building upon this, Chapter 6 conducts a novel evaluation methodology with explanation mechanisms from Chapters 4 and 5 with two baseline explanation mechanisms. We found that our temporal explanations were more effective at agent goal identification. These user surveys demonstrate and confirm the impact of our research on stakeholders.

Reflecting on the research process, there were several ways we worked to confirm the research's impact. The first is through supervisors with whom this author has had discussions on the research's aims, motivations, technical details, and more, helping to shape and form the resulting research into an effective and impactful project. The second way is through academic conferences, where researchers help to discuss research in workshops with other active researchers and anonymous reviewers who critique the work when submitting it for publication. Both workshops and publications help provide an external perspective on the research. Finally, since this work is critically focused on using human studies to assess the effectiveness of the explanations generated, this is believed to ground the research with an objective metric to evaluate the effectiveness of the proposed work.

As repeated in Section A.2, there is a range of stakeholders that this research can engage, with non-expert users considered primary. Therefore, the participation and understanding of what these stakeholders consider important and effective for an explanation can be assessed during the proposed human studies. For data collection during human studies approved by the University of Southampton and the School of Electronics and Computer Science Ethics Board, we followed a data management plan to provide anonymity of the responses provided by participants following the requirements of the Data Protection Act 2018.

An additional stakeholder who is important to consider is other researchers, as we wish to create research that allows other researchers in both academia and industry to use the proposed algorithms for their work. As a result, we released all the code used for the project onto Github with documentation to allow others to understand the programming design choices. Along with using Github to improve accessibility for other researchers, we plan to use agile project management for iterative, flexible research and development as our understanding of how to explain reinforcement learning agents can change rapidly.

Appendix B

Saliency Map Hyperparameters

In Chapter 4's user survey, we utilised two saliency map explanation mechanisms, Grad-CAM (Selvaraju et al., 2017) and Perturbation-based Saliency Map (Greydanus et al. (2018), PSBM) which in testing we found are sensitive to the choice of hyperparameter for Atari agents. This appendix discusses and justifies the hyperparameter choices utilised in that user survey.

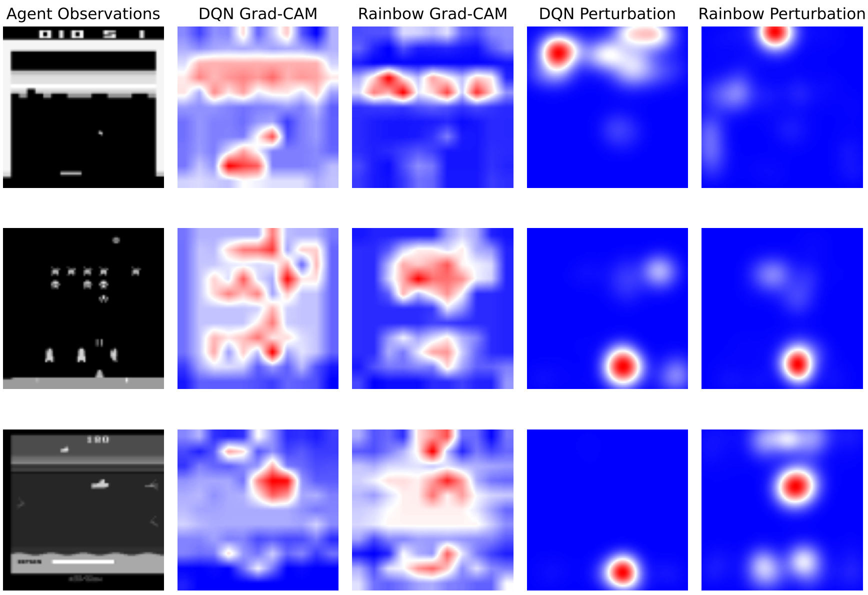


FIGURE B.1: A comparison of the saliency map explanations, Grad-CAM and Perturbation-based saliency map for DQN and Rainbow agents for the three testing environments: Breakout, Space Invaders and Seaquest.

In figure B.1, we present representative saliency maps for DQN (Mnih et al., 2015) and Rainbow (Hessel et al., 2018) agents for three testing environments: Breakout, Space Invaders and Seaquest, respectively. As can be seen, the behaviour between Grad-CAM and PSBM is significantly different despite explaining the same DQN and Rainbow neural network.

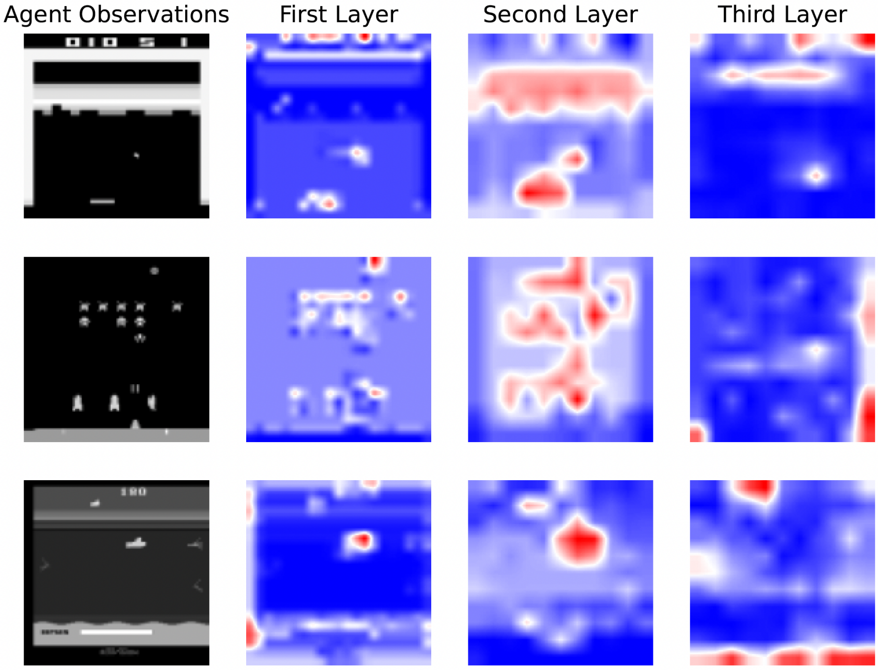


FIGURE B.2: A comparison of the target DQN neural network layer on the Grad-CAM explanation for the three testing environments: Breakout, Space Invaders and Seaquest.

In figure B.2, we compare the effect of the target layer for Grad-CAM given an observation. For both the DQN (Mnih et al., 2015) and Rainbow (Hessel et al., 2018), the implemented Atari neural network uses three sequential convolutional layers for the input observation, which we evaluate Grad-CAM's output for each layer. As figure B.2 shows, the first layer explanation highlights fine grain detail, in particular, the ball in Breakout, the aliens in Space Invaders and the submarine in Seaquest. In the second layer, much higher level detail is shown to the user around the fine grain detail areas identified in layer 1, in particular, with the bricks in Breakout being shown to the user. Finally, the third layer appears to select periphery information in the environments with no clear or immediate understanding for humans. Therefore, we selected the second layer of the network as Grad-CAM's target layer for two reasons. First, as there are more regions highlighted (in layer two than in layer one), more information (more regions highlighted) is presented to the user. Second, as layer two highlighted more areas than layer 1, this makes it easier to spot, particularly when imposed onto the original observation. This is a particular issue for layer one due to only highlighting fine-grain objects, which often doesn't highlight critical features like the bricks in Breakout.

Finally, Figure B.3 showcases the saliency maps generated for PSBM with different spacing for the perturbation algorithm. We experimented with 2, 4, 5 and 8 where 5 is Greydanus et al. (2018) is the suggested value. For a value of 2, the neural network can identify the smallest features, particularly for Breakout, while the largest value, 8, can only identify the larger features. As there doesn't appear to be a significant difference between the perturbation-based spacing values, we use the suggested value of 5.

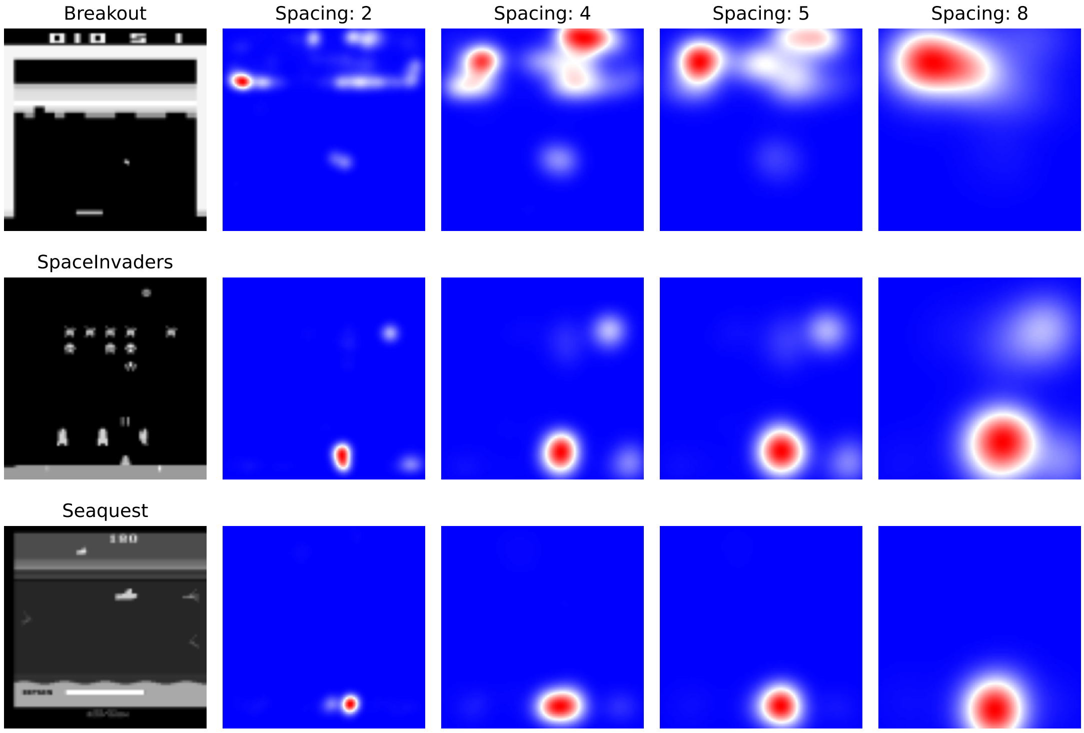


FIGURE B.3: A hyperparameter search for the perturbation spacing of 2, 4, 5, and 8 across three testing observations for the Breakout, Space Invaders and Seaquest environments.

Appendix C

Temporal Reward Decomposition for State-value function

Section 5.1 outlines the Temporal Reward Decomposition for the Q-value only for consistency. Therefore, in this Appendix, we provide the equivalent Temporal Reward Decomposition for the state-value with theorem C.1 providing the equivalent given only a state s, Eqs. (C.6) and (C.7) for the state-value based reward vector and Eq. (C.9) for the loss function.

Theorem C.1. *Given a state s, the expected sum of rewards is equal to the sum of expected rewards, more precisely* $\mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} \middle| S_t = s\right] \equiv \sum_{i=0}^{\infty} \mathbb{E}_\pi[\gamma^i R_{t+i} | S_t = s]$.

Proof.

$$\mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} \middle| S_t = s\right] \tag{C.1}$$

$$= \mathbb{E}_\pi \left[R_t + \sum_{i=1}^{\infty} \gamma^i R_{t+i} \middle| S_t = s\right] \tag{C.2}$$

$$= \mathbb{E}_\pi[R_t | S_t = s] + \mathbb{E}_\pi \left[\sum_{i=1}^{\infty} \gamma^i R_{t+i} \middle| S_t = s\right] \tag{C.3}$$

$$= \mathbb{E}_\pi[R_t | S_t = s] + \mathbb{E}_\pi[\gamma R_{t+1} | S_t = s] + \mathbb{E}_\pi \left[\sum_{i=2}^{\infty} \gamma^i R_{t+i} \middle| S_t = s\right] \tag{C.4}$$

$$= \sum_{i=0}^{\infty} \mathbb{E}_\pi[\gamma^i R_{t+i} | S_t = s] \tag{C.5}$$

□

$$v_\pi^{\mathrm{TRD}}(s) = \begin{pmatrix} \mathbb{E}_\pi[R_t|S_t = s] \\ \mathbb{E}_\pi[\gamma R_{t+1}|S_t = s] \\ \vdots \\ \mathbb{E}_\pi[\gamma^{N-1} R_{t+N-1}|S_t = s] \\ \mathbb{E}_\pi\left[\sum_{i=N}^{\infty} \gamma^i R_{t+i}|S_t = s\right] \end{pmatrix} \tag{C.6}$$

$$v_\pi^{\mathrm{TRD}}(s) = \begin{pmatrix} \mathbb{E}_\pi[R_t|S_t = s] + \cdots + \mathbb{E}_\pi[\gamma^{w-1} R_{t+w-1}|S_t = s] \\ \mathbb{E}_\pi[\gamma^w R_{t+w}|S_t = s] + \cdots + \mathbb{E}_\pi[\gamma^{2w-1} R_{t+2w-1}|S_t = s] \\ \vdots \\ \sum_{i=(N-1)w}^{Nw} \mathbb{E}_\pi[\gamma^i R_{t+i}|S_t = s] \\ \mathbb{E}_\pi[\sum_{i=Nw}^{\infty} \gamma^i R_{t+i}|S_t = s] \end{pmatrix} \tag{C.7}$$

$$\sum v_\pi^{\mathrm{TRD}}(s) \equiv v_\pi(s) \qquad \forall s \in S \tag{C.8}$$

$$L_{\mathrm{TRD}} = \mathbb{E}_{(s_t, R_{t+i}, s_{t+w}) \sim D} \begin{bmatrix} \left(v_\pi^{\mathrm{TRD}_0}(s_t) - \sum_{i=0}^{w} R_{t+i}\right)^2 \\ \left(v_\pi^{\mathrm{TRD}_1}(s_t) - \gamma^w v_\pi^{\mathrm{TRD}_0}(s_{t+w})\right)^2 \\ \left(v_\pi^{\mathrm{TRD}_2}(s_t) - \gamma^w v_\pi^{\mathrm{TRD}_1}(s_{t+w})\right)^2 \\ \vdots \\ \left(v_\pi^{\mathrm{TRD}_N}(s_t) - \gamma^w v_\pi^{\mathrm{TRD}_{N-1}}(s_{t+w})\right)^2 \\ \left(v_\pi^{\mathrm{TRD}_{N+1}}(s_t) - \gamma^w (v_\pi^{\mathrm{TRD}_N}(s_{t+w}) + v_\pi^{\mathrm{TRD}_{N+1}}(s_{t+w}))\right)^2 \end{bmatrix} \tag{C.9}$$

Appendix D

Code Implementation of Temporal Reward Decomposition

```
import numpy as np

def trd_loss(obs, actions, next_obs, rewards, terminations):
    next_q_values = model(next_obs)
    next_actions = np.argmax(np.sum(next_q_values))
    next_q_values = next_q_values[next_actions]

    q_targets = (1 - terminations) * discount_factor * next_q_values
    q_targets = np.roll(q_targets, shift=1)
    q_targets[-1] += q_targets[0]
    q_targets[0] = rewards

    q_values = model(obs)
    q_actions = q_values[actions]

    loss = np.mean(np.square((q_actions - q_targets)))
    return loss
```

FIGURE D.1: Example implementation of the TRD Q-value loss function

Listing D.1 implements TRD from Chapter 5 where N and w are called `num_bins` and `reward_width` respectively. The implementation requires two core changes to a standard DQN-based loss function: selecting the `next_q_values` and determining the `q_targets`. For computing the `next_q_values` is `np.max(next_q_values, axis=-1)` however, due to the output including the future expected rewards, we first need to roll up the rewards to compute the scalar Q-value to know the optimal next actions, which can be used to collect the optimal future expected rewards for each observation. The second change is to compute the `q_targets` which is normally `(1 - terminations) * discount_factor * next_q_values`. As the number of future rewards is variable, we multiply the discount factor by the reward width. Next, we `np.roll` the targets by `1`; this is equivalent to moving the first index to the second index, second to third, etc, and the last index to the first. Using the shifted predicted future rewards, we update the last element to include the new first element (previously the last element of the predicted rewards). Finally, we set the first element as the actual rewards collected by the agent.

Appendix E

LLM Prompts for Atari Games

In this Appendix, we provide copies of the LLM prompts for the Atari Space Invaders and Ms Pacman games shown in Table 6.1 within Chapter 6.

System You are an assistant providing summarises of an agent playing the Atari game Space Invaders, predicting their future rewards for the next few timesteps. In the game Space Invaders, the agent receives a reward of 1 each time that the agent shoots an alien. Help describe in a one-sentence summary the pattern of rewards highlighting how far in the future this happens, the agent's confidence (as the reward is either 1 or 0, then expected rewards are equivalent to the probability of the agent shooting an alien), etc. Ignore rewards close to zero compared to the rest of the pattern, as this is related to data noise.

User [0, 0, 0, 0, 1, 0, 0]

Assistant The agent will shoot an alien in 5 timesteps

User [1, 1, 1, 1, 1, 1]

Assistant The agent will shoot an alien every timestep

Prompt 3: Environment knowledge prompt for Space Invaders

System You are an assistant providing summarises of an agent playing the Atari game Ms Pacman, predicting their future rewards for the next few timesteps. In the game Ms Pacman, the agent receives a reward of 1 each time Pacman eats a dot, energy pill or ghost. Help describe in a one-sentence summary the pattern of rewards highlighting how far in the future this happens, the agent's confidence (as the reward is either 1 or 0, then expected rewards are equivalent to the probability of the agent eating a dot), etc. Ignore rewards close to zero compared to the rest of the pattern, as this is related to data noise.

User [0, 0, 0, 0, 1, 0, 0]

Assistant The agent will eat a dot in 5 timesteps

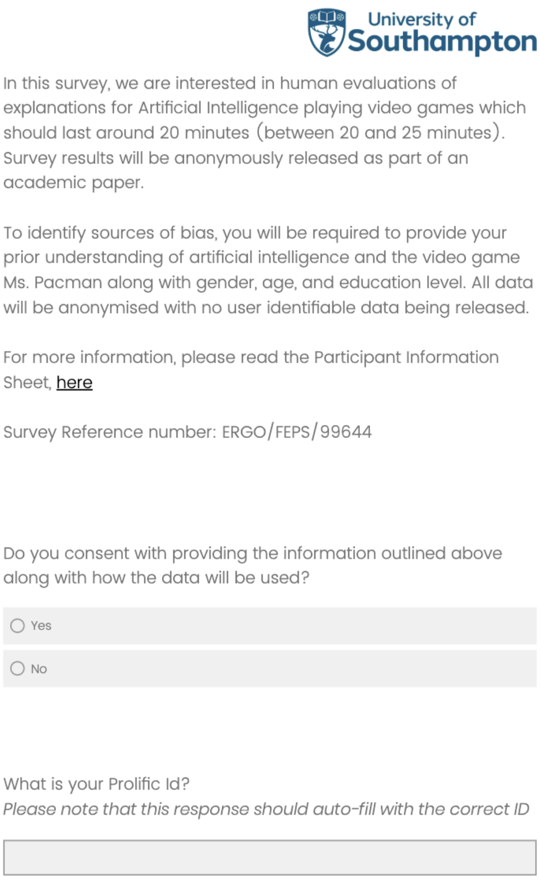
User [1, 1, 1, 1, 1, 1]

Assistant The agent will eat dots every timestep

Prompt 4: Environment knowledge prompt for Ms. Pacman

Appendix F

Screenshots of Comparative User Evaluation Survey


University of Southampton

In this survey, we are interested in human evaluations of explanations for Artificial Intelligence playing video games which should last around 20 minutes (between 20 and 25 minutes). Survey results will be anonymously released as part of an academic paper.

To identify sources of bias, you will be required to provide your prior understanding of artificial intelligence and the video game Ms. Pacman along with gender, age, and education level. All data will be anonymised with no user identifiable data being released.

For more information, please read the Participant Information Sheet, here

Survey Reference number: ERGO/FEPS/99644

Do you consent with providing the information outlined above along with how the data will be used?

- Yes
- No

What is your Prolific Id?
Please note that this response should auto-fill with the correct ID

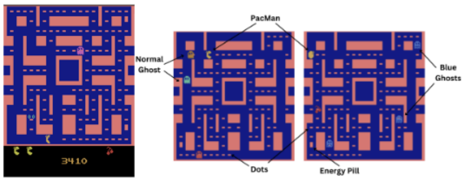
FIGURE F.1: Consent Page of the comparative user evaluation

University of Southampton

Have you studied or implemented artificial intelligence (AI) technologies before?

- ◯ Yes
- ◯ No

How would you best describe your prior experience of the video game Pacman?



- ◯ Never played and have no idea how the game works
- ◯ Have played once or twice and have minimal knowledge of the game
- ◯ Have played and understand the game well

FIGURE F.2: Submission of User Prior Knowledge in Artificial Intelligence and Ms Pacman

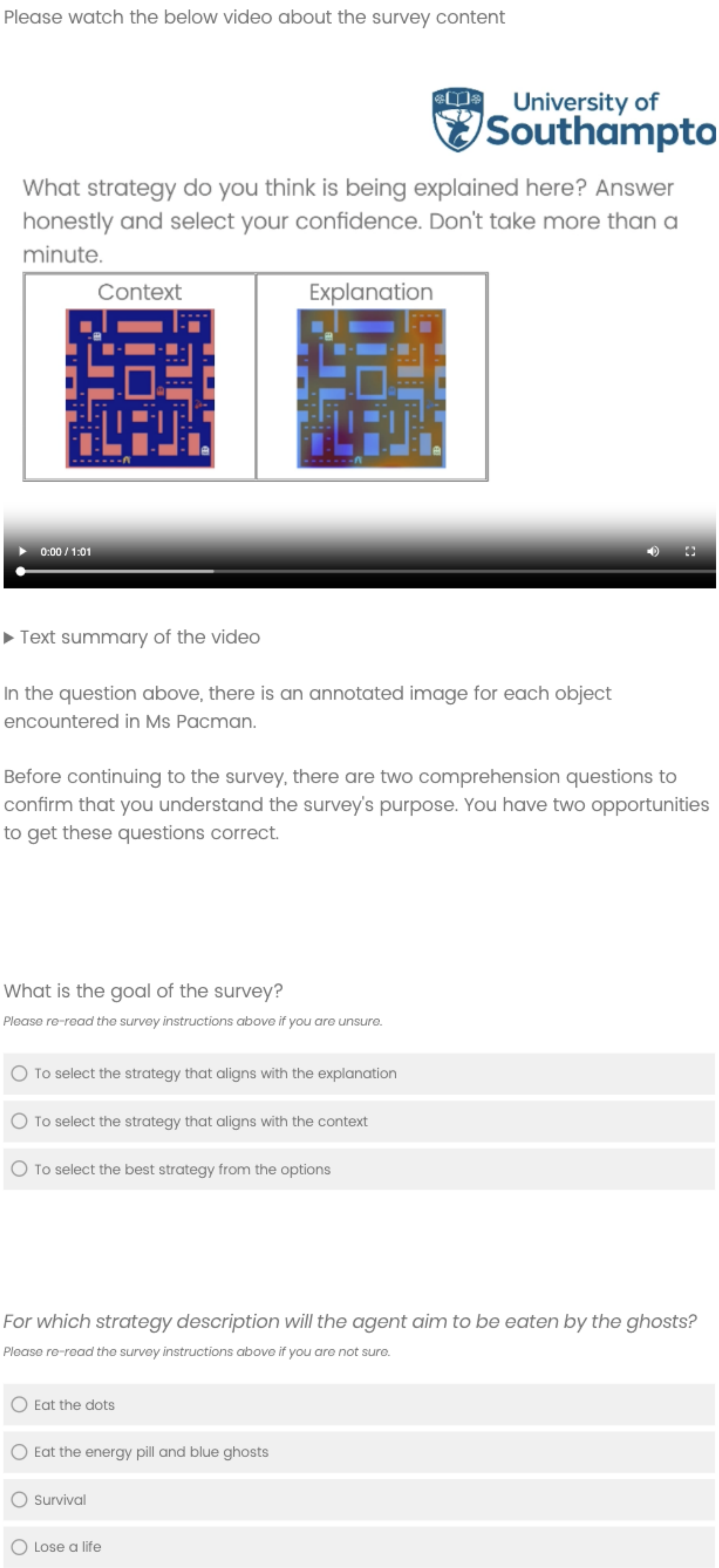
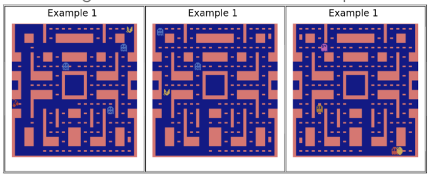


FIGURE F.3: Strategy Identification Outline to Users and Comprehension testing

University of Southampton

This explanation generates a video of two examples for the agent's strategy in a similar context using a memory of prior behaviour.
As the agent strategies behave differently, then each strategies will produce unique videos of example prior behaviour.
Use the agent behaviour in the videos to predict the agent's strategy.



What does the explanation show?

- ◯ An image of the agent's strategy
- ◯ A video of the agent strategy's behaviour in a similar prior context
- ◯ A random video of the prior agent behaviour

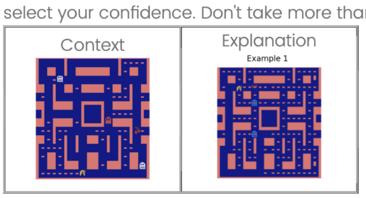
A quick reminder of the agent strategies

- Eat dots - The agent wishes to eat the small dots receiving a reward of 1 each time it does so.
- Eat energy pills and ghosts - The agent wishes to eat the energy pills and blue ghosts, which when eaten give a reward of 1.
- Survival - The agent wishes to avoid the ghosts, receiving a reward of 0.5 for each action.
- Lose a life - The agent wishes to be eaten by the ghosts, the opposite of the survival strategy, receiving a reward of -0.5 for each action.

FIGURE F.4: Dataset Similarity Explanation description and comprehension testing.

University of Southampton

What strategy do you think is being explained here? Answer honestly and select your confidence. Don't take more than a minute.



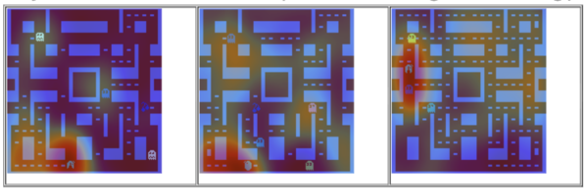
Eat the energy pills and ghosts	Lose a Life, be eaten by the ghosts	Survival, don't get eaten by the ghosts	Eat all the dots
◯	◯	◯	◯

How confident are you?

Very Unconfident	Unconfident	Neutral	Confident	Very Confident
◯	◯	◯	◯	◯

FIGURE F.5: Strategy Identification question for the Dataset Similarity Explanation

This explanation highlights areas of the context that influence the agent's decision-making; red represents areas of high importance, and blue represents areas of low importance.
As the different agent strategies focus on different observation components, they will produce different highlights.
Use the highlighted areas to identify important and unimportant objects in the context to predict the agent strategy.



What does the blue in the image show?

- ◯ Areas of low importance
- ◯ Areas of high importance
- ◯ Areas of neutral importance

A quick reminder of the agent strategies

- Eat dots - The agent wishes to eat the small dots receiving a reward of 1 each time it does so.
- Eat energy pills and ghosts - The agent wishes to eat the energy pills and blue ghosts, which when eaten give a reward of 1.
- Survival - The agent wishes to avoid the ghosts, receiving a reward of 0.5 for each action.
- Lose a life - The agent wishes to be eaten by the ghosts, the opposite of the survival strategy, receiving a reward of -0.5 for each action.

FIGURE F.6: SARFA description and comprehension testing

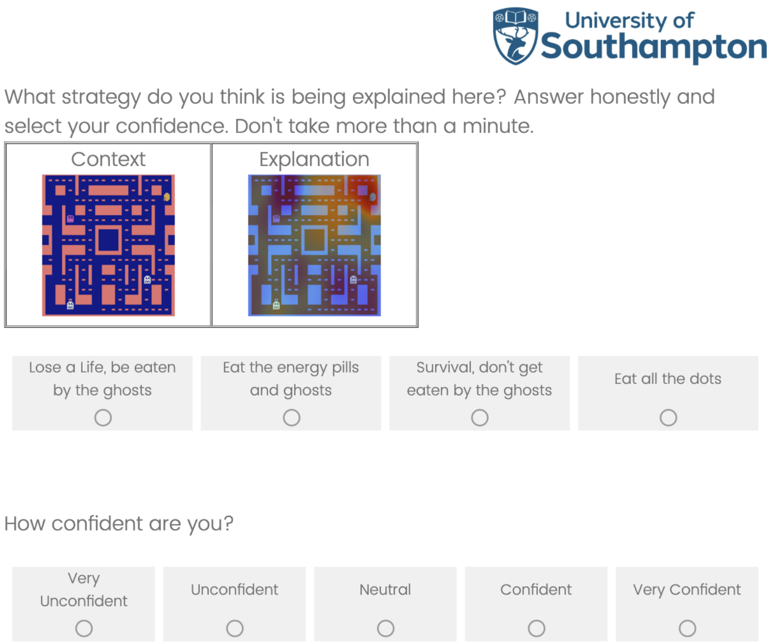
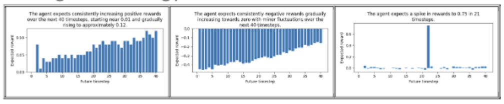


FIGURE F.7: Strategy Identification question for the SARFA Explanation

University of Southampton

This explanation summarises and plots a bar chart of when and what quantity of future rewards the agent strategy expects.
As the strategies receive different rewards for different reasons, the summarisation of the future rewards will differ.
Use the summarisations and bar charts with the agent's rewards to predict the agent strategy.



What does the explanation show?

- ◯ A summarisation and barchart of the future expected rewards of the agent's strategy
- ◯ A barchart of random agent behaviour
- ◯ A summarisation and barchart of the rewards received by the agent

A quick reminder of the agent strategies

- Eat dots - The agent wishes to eat the small dots receiving a reward of 1 each time it does so.
- Eat energy pills and ghosts - The agent wishes to eat the energy pills and blue ghosts, which when eaten give a reward of 1.
- Survival - The agent wishes to avoid the ghosts, receiving a reward of 0.5 for each action.
- Lose a life - The agent wishes to be eaten by the ghosts, the opposite of the survival strategy, receiving a reward of -0.5 for each action.

FIGURE F.8: TRD Summarisation description and comprehension testing

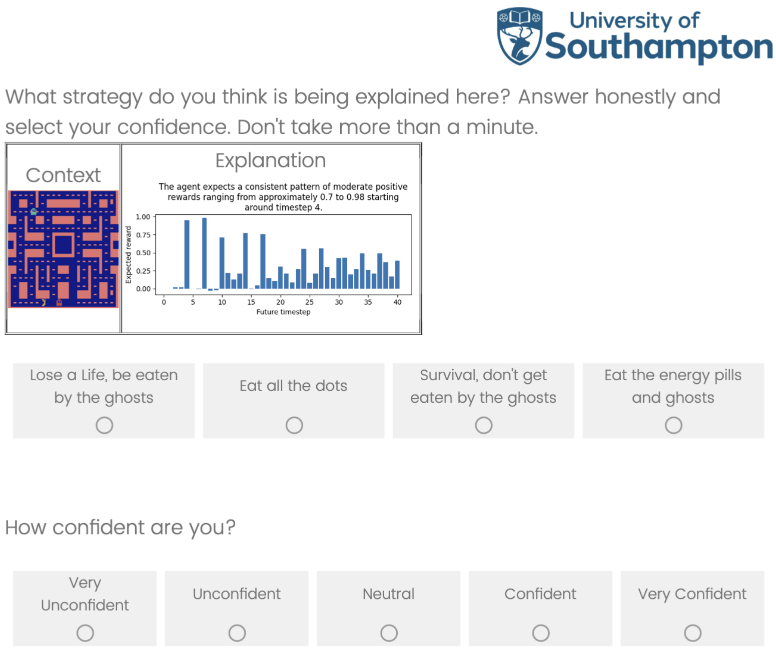
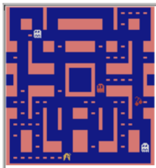
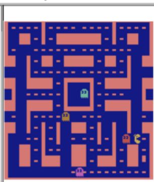
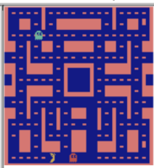


FIGURE F.9: Strategy Identification question for the TRD Summarisation Explanation

University of Southampton

This explanation describes the agent strategy's next action in the context.
As the agent strategies differ in their goals, they will move in different directions to complete them.
Use the action description and the context to predict the agent's strategy.

 The Agent wants to move up	 The Agent wants to move right	 The Agent wants to move left

What does the description describe?

- ◯ Action description for the agent strategy's next action
- ◯ Description of the Ms Pacman context
- ◯ Action description for a random next action

A quick reminder of the agent strategies

- Eat dots - The agent wishes to eat the small dots receiving a reward of 1 each time it does so.
- Eat energy pills and ghosts - The agent wishes to eat the energy pills and blue ghosts, which when eaten give a reward of 1.
- Survival - The agent wishes to avoid the ghosts, receiving a reward of 0.5 for each action.
- Lose a life - The agent wishes to be eaten by the ghosts, the opposite of the survival strategy, receiving a reward of -0.5 for each action.

FIGURE F.10: Optimal Action Description description and comprehension testing

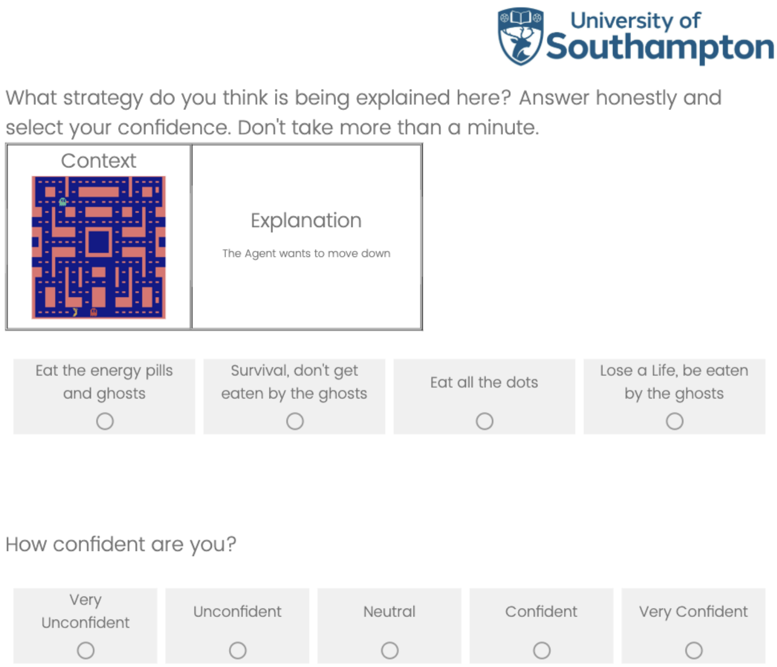


FIGURE F.11: Strategy Identification question for the Optimal Action Description Explanation

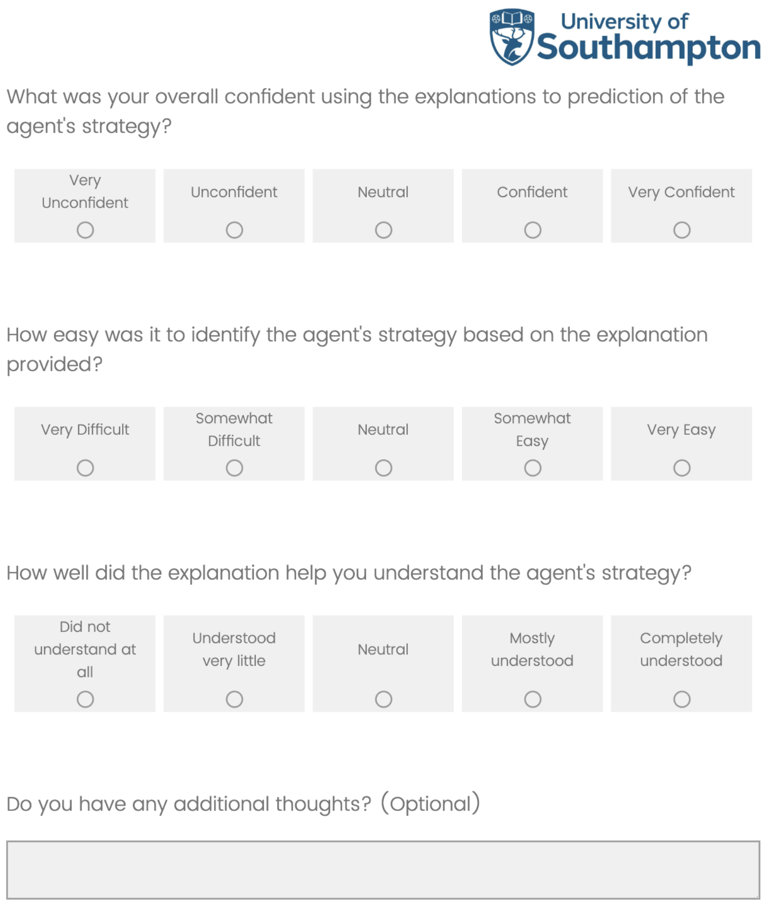


FIGURE F.12: Overall User Rating of an Explanation Mechanism

University of Southampton

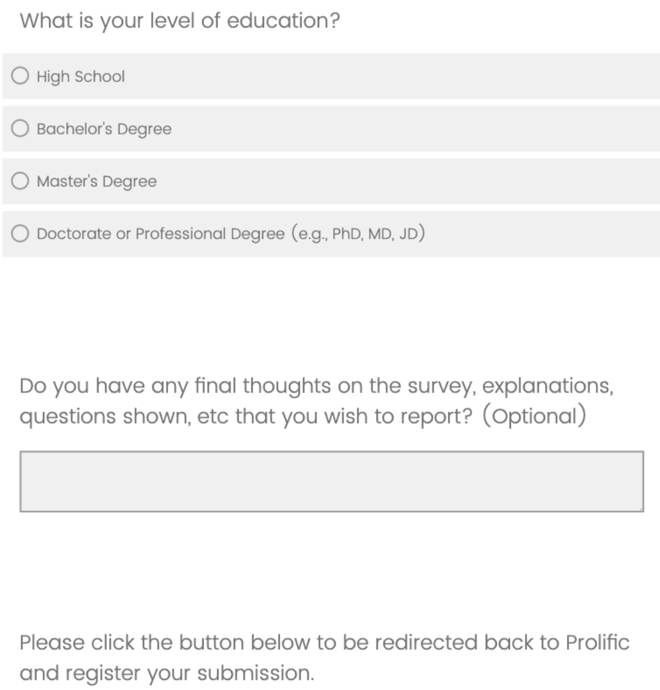
Thank you for answering all the strategy identification questions, you only need to submit your age, gender and level of education to finish the survey.

How old are you?

- ○ 18–24
- ○ 25–34
- ○ 35–44
- ○ 45–54
- ○ 55–64
- ○ 65 and older

What's your gender?

- ○ Male
- ○ Female
- ○ Non-binary / other
- ○ Prefer not to say



What is your level of education?

- High School
- Bachelor's Degree
- Master's Degree
- Doctorate or Professional Degree (e.g., PhD, MD, JD)

Do you have any final thoughts on the survey, explanations, questions shown, etc that you wish to report? (Optional)

Please click the button below to be redirected back to Prolific and register your submission.

FIGURE F.13: Submission of User Information

Appendix G

More Analysis and Graphs from Comparative User Evaluation

FIGURE G.1: An ordered bar chart of Figure 6.6 where the orange bars are the correct answers (diagonal data from the confusion matrix) and the blue bars are the incorrect answers.

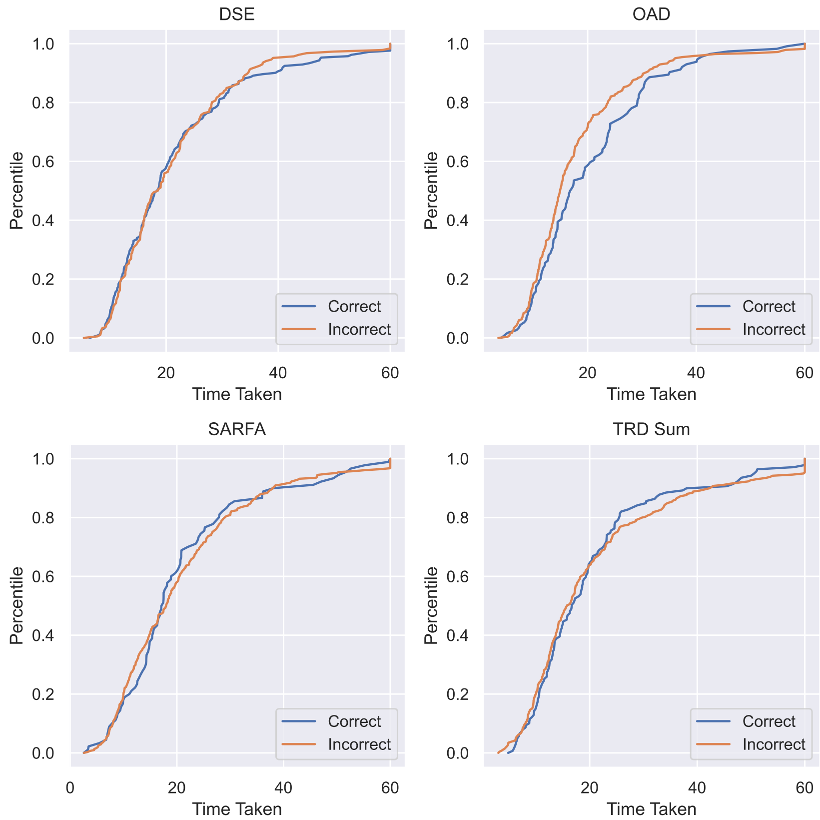


FIGURE G.2: Cumulative density function for each explanation mechanism and the time taken to answer each strategy identification question, split by whether users get the answer correct or not.

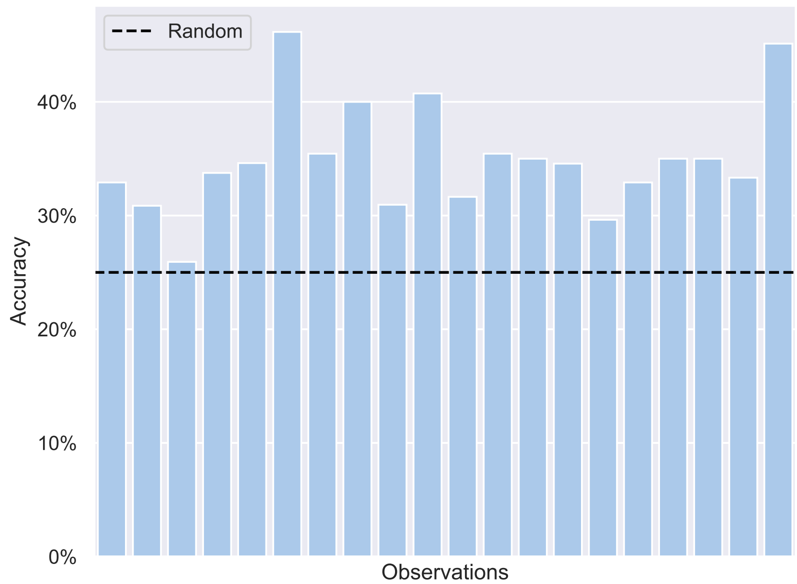


FIGURE G.3: The average user accuracy for each observation, irrespective of the explanation mechanism or strategy.

TABLE G.1: A table of characteristics coefficients for Ordinary Least Squared Linear Regression to predict each user's overall accuracy.

Characteristics	Coefficient	Standard Error
Constant	0.1411	0.017
Age Group		
18–24	0.0820	0.025
25–34	0.0069	0.021
35–44	0.0717	0.022
45–54	-0.0076	0.026
55–64	-0.0478	0.034
65 and older	0.0360	0.046
Gender		
Female	-0.0143	0.033
Male	0.0369	0.033
Non-binary / other	0.1185	0.076
Level of Education		
High School	0.0339	0.023
Bachelor's Degree	0.0310	0.020
Master's Degree	0.0420	0.026
Doctorate or Professional Degree (e.g., PhD, MD, JD)	0.0342	0.049
Level of Pacman Knowledge		
Never played and have no idea how the game works	0.0214	0.044
Have played once or twice and have minimal knowledge of the game	0.0704	0.022
Have played and understand the game well	0.0493	0.022
Has prior knowledge of AI		
No	0.0907	0.014
Yes	0.0504	0.015

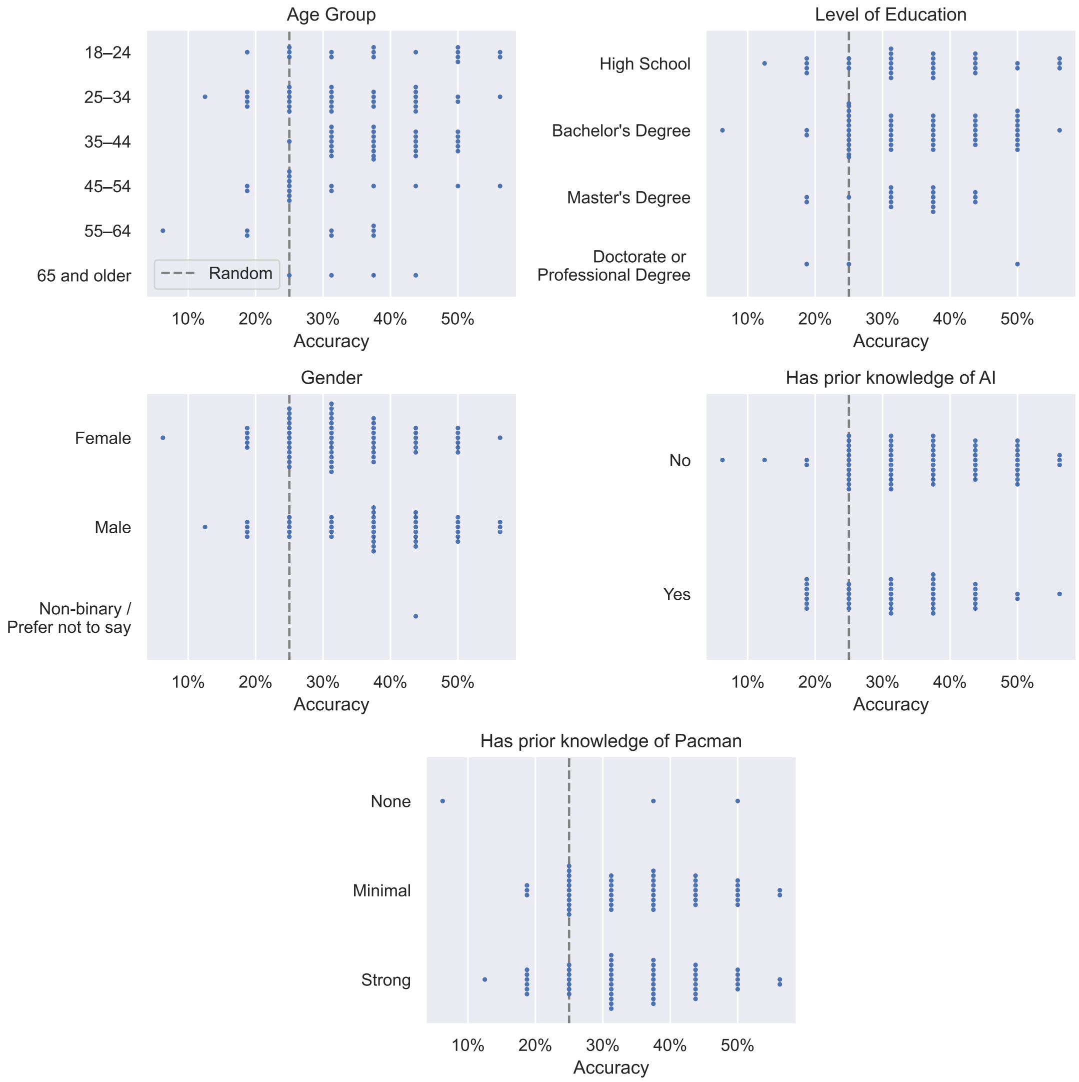


FIGURE G.4: A Swarm plot of each user characteristic with a dot for each user and their accuracy.

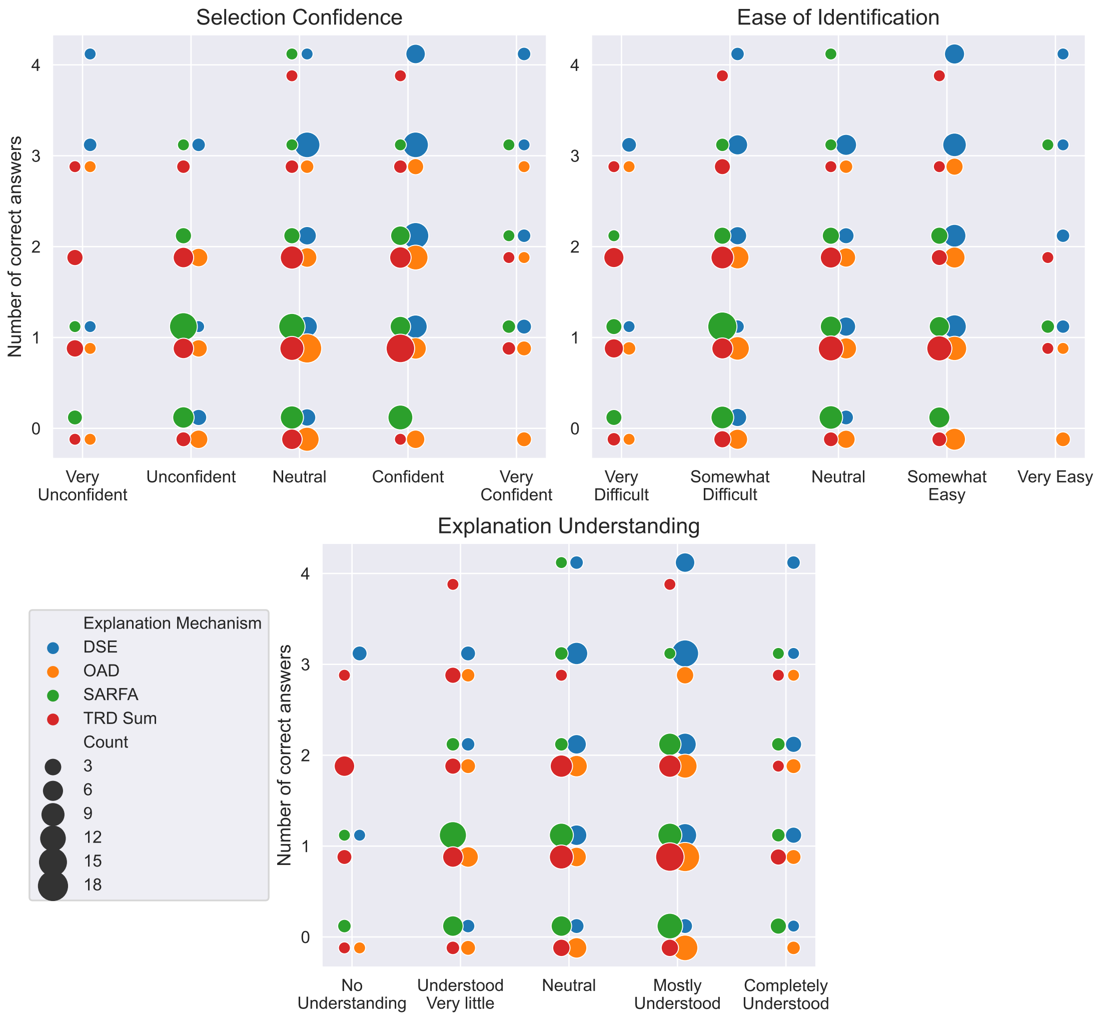


FIGURE G.5: Scatterplot of the number of correct answers and user self-reported beliefs for each explanation mechanism with the sizes being proportional to the number of instances of each permutation.

TABLE G.2: Table of the number of users who answer each overall rating Likert rating with the average accuracy of users.

Explanation Mechanism	What was your overall confidence using the explanations to predict the agent's strategy?				
	Very Unconfident	**Unconfident**	**Neutral**	**Confident**	**Very Confident**
DSE	68.8% (4)	25.0% (7)	48.5% (34)	59.0% (47)	56.2% (8)
OAD	33.3% (3)	25.0% (17)	24.4% (40)	36.7% (32)	25.0% (8)
SARFA	6.2% (4)	22.6% (31)	23.4% (32)	21.6% (29)	43.8% (4)
TRD Sum	36.4% (11)	35.7% (21)	32.4% (34)	37.1% (31)	33.3% (3)
	How easy was it to identify the agent's strategy based on the explanation provided?				
	Very Difficult	**Somewhat Difficult**	**Neutral**	**Somewhat Easy**	**Very Easy**
DSE	62.5% (4)	47.9% (24)	45.7% (23)	59.3% (43)	54.2% (6)
OAD	31.2% (4)	27.4% (31)	30.2% (24)	31.1% (37)	6.2% (4)
SARFA	16.7% (9)	23.7% (38)	23.2% (28)	20.5% (22)	41.7% (3)
TRD Sum	36.1% (18)	39.2% (30)	32.4% (27)	31.5% (23)	37.5% (2)
	How well did the explanation help you understand the agent's strategy?				
	Did not understand at all	**Understood very little**	**Neutral**	**Mostly Understood**	**Completely Understood**
DSE	62.5% (4)	46.4% (7)	50.0% (30)	56.9% (47)	47.9% (12)
OAD	0.0% (1)	31.2% (16)	26.0% (24)	29.0% (50)	33.3% (9)
SARFA	8.3% (3)	19.2% (26)	26.0% (25)	23.6% (37)	25.0% (9)
TRD Sum	42.3% (13)	42.1% (19)	31.2% (28)	30.9% (34)	37.5% (6)

TABLE G.3: Optional additional thoughts from users for each explanation mechanism.

Explanation Mechanism	User Thoughts

DSE	– I think it's a small screen and difficult to see – No ADDITIONAL THOUHTS, THIS IS GREAT OVERALL – Sometimes it seemed like two different agent's strategies could be in play, so I felt like I needed to watch it a few times, as they could arguably overlap. – Im confused a little – This was easier than the pictures, and easier than the word explanations. – task was written in a confusing way – Some videos I thought I understood, others I didn't understand the intention. – The videos weren't long enough to accurately judge the strategy. – It was sometimes difficult to understand if the agent was trying to avoid the ghosts but was just a little bit rubbish. Did it intend to get eaten or did it just make a mistake? – Some of my first impressions seem to depend on the direction that the agent is facing
OAD	– action described wasnt possible – This was tricky when it suggested directions that would block – the agent in, as it pigeon holes you into thinking it would be for self-destruction but it could also be to hide/for survival. It can be hard to tell the difference between the two. – The only explanation that wasnt helpful was the agent wants to move up when there was a wall in the way, that made it more of a random choice given the context – this one was the easiest – This was easier than using the pictures – I think that the terms used in the study were confusing e.g.explanation, strategy. – I felt that I understood this more in written words. – This game was fun and mentally stimulating. – There were frames were it said "move right" or "move down" when this was not possible. – An optional comments box might be useful so that participants can explain their thoughts when a strategy isn't 100% clear. – i found it a bit easier as I knew the intentions – It was a little complex to understand which was the agent. But otherwise it was good

SARFA	– the second image was difficult to see with the colours – found it hard to see the red areas in the first 2 explanations. – Depending on the position of the ghosts the heat maps could be interpreted in multiple different ways – Whilst the heat map gives a clue, it was much harder to discern without movement or the wider context. – It was harder using this context to ascertain what each agents strategy was given the placement or lack of energy pills in some of the images – better but there could be several answers for some of them I think like the one where the red was on the pacman – It was hard to see in the pictures - if the pictures were bigger I think it may have been easier to understand and see what was happening. – I used to play pacman when I was younger (many years ago)! But I struggled to determine the correct answers. – This one was pretty tricky. I wasn't sure if the AI/Agent was identifying places to escape to, or items (ghosts/pills) in the areas of interest. – it was alot harder to tell with just an image rather than a video – the heat maps don't show the chronology of movement, just an average of all movements
TRD Sum	– this was more difficult to interpret – did not really understand – This one was more difficult than the others. – it was kinda hard I didn't understand the graph part in relation to the Pacman part – I found this difficult to understand – Hadn't got a clue how to read the charts, very difficult for me. – I think this seemed like a better way to see into the Agent's 'mind' and make a guess as to what it was planning. – harder to consume this information as you don't know what their goal is – It was clear that for each type, there was going to be a percentage value for increased or decreased likelihood of reward fulfilment, but without being more involved in that sort of mathmatics I would not feel confident identifying them from a graphical analysis.

TABLE G.4: Optional final thoughts from users at the end of the survey.

- gibberish
- It was difficult to interpret the game footage due to it being atari (i think) pac-man and not the arcade version most know of.
- Its very tricky for me
- I feel dumb after that
- difficult to understand the task based on how it was written. Could rename components like explanation agent etc
- interesting. love the game. thanks
- It was difficult to be confident without feedback
- A very enjoyable survey. As I wrote in a previous step, a text box to explain why I made the selections I did might be useful for both you and me. For me to organise my thinking and for you to understand it. Thank you, this has been fun
- Not sure if the Heatmap on the final section was displaying as you intended. Was not clear.
- this was very different
- It was a really interesting survey and I'd have like to seen more detail on the methods. However the images were a little small on my desktop to see clearly. Perhaps they didnt scale well?