

# **Towards a Combined Local and Global Explanation Framework for Deep Reinforcement Learning Agents with Visual Input: Novel Methods and Insights from Human Evaluation**

Dissertation  
zur Erlangung des Doktorgrades (Dr. rer. nat.) an der  
Fakultät für Angewandte Informatik  
der Universität Augsburg

vorgelegt von

Tobias Huber

2024

Erstgutachterin:	Prof. Dr. Elisabeth André
Zweitgutachter:	Prof. Dr. Bernhard Bauer
Drittgutachterin:	Prof. Dr. Ofra Amir

Tag der mündlichen Prüfung: 17. Juli 2024

# Abstract

In recent years, considerable advances have been made in the development of Deep Reinforcement Learning (DRL) algorithms. As a result of these advances, DRL agents are increasingly introduced into high-risk domains such as health-care or automated vehicles. To ensure proper use in these critical domains, users must understand the agents' strategies and know when to rely on them. At the same time, the increasing complexity and opacity of DRL algorithms present substantial hurdles to their explainability, especially when applied to large visual states.

This thesis is dedicated to improving the explainability of DRL agents with visual input. While there has been a resurgence of interest in developing explainable Artificial Intelligence (XAI), it has primarily focused on classification tasks. However, DRL presents its own set of challenges and requirements for explainability. First, DRL agents engage in sequential decision-making where actions are interconnected and contribute to a long-term strategy that is potentially influenced by delayed rewards. Second, DRL agents learn by interacting with an environment in which their goals are only indirectly defined by the rewards they receive for their actions. Consequently, the strategies developed by DRL agents might deviate from human expectations, even if they are optimal for the given reward function.

To address these specific challenges for explainable deep reinforcement learning, this dissertation pursues five objectives.

The first three objectives relate to the development of novel explanation methods that are tailored to the needs of DRL. First, this thesis introduces a novel saliency map algorithm that identifies relevant information for an agent's decision. Compared to other saliency map methods, this algorithm focuses on more selective areas within the input. As a result, it helps to quickly interpret multiple states and uncover their interrelationship within the agent's strategy. Second, this dissertation proposes a model-agnostic method for generating counterfactual explanations for visual DRL agents, illustrating how states can be changed to alter the agent's action. Third, to extend the insights from the local explanations to the global strategy of the agent, this dissertation introduces a novel combination of local explanations with global strategy summaries. Strategy summary methods identify representative states for the agent's strategy and thus allow users to gain a good understanding of the agent's strategy by examining a limited budget of states.

The last two objectives of this dissertation are concerned with the evaluation of XRL methods. Here, this thesis starts by evaluating the proposed methods and other local explanation methods with computational metrics that assess their fidelity to the agent’s internal reasoning. Finally, the complementary and individual contributions of the global and local explanations in the aforementioned combination are investigated in three user studies. These studies measure agent understanding, appropriate trust, and satisfaction with the explanations. The results of the experiments in this dissertation demonstrate the significant potential of combined explanations for DRL agents and identify challenges that inform the development of future explanation frameworks for DRL.

# Zusammenfassung

In den letzten Jahren wurden beträchtliche Fortschritte bei der Entwicklung von Algorithmen des tiefen bestärkenden Lernens (Deep Reinforcement Learning, DRL) erzielt. Diese Fortschritte haben dazu geführt, dass DRL-Agenten zunehmend in Hochrisikobereichen wie dem Gesundheitswesen oder automatisierten Fahrzeugen eingesetzt werden. Um den richtigen Einsatz in diesen kritischen Bereichen zu gewährleisten, müssen die Nutzer die Strategien der Agenten verstehen und wissen, wann sie sich auf sie verlassen können. Gleichzeitig stellen die zunehmende Komplexität und Undurchsichtigkeit der DRL-Algorithmen jedoch erhebliche Hürden für ihre Erklärbarkeit dar, insbesondere wenn sie auf große visuelle Zustände angewendet werden.

Diese Arbeit widmet sich der Verbesserung der Erklärbarkeit von DRL-Agenten mit visuellem Input. Während das Interesse an der Entwicklung von erklärbarer künstlicher Intelligenz (eXplainable Artificial Intelligence, XAI) wieder auflebt, konzentriert es sich hauptsächlich auf Klassifikationsaufgaben. DRL stellt jedoch eine eigene Reihe von Herausforderungen und Anforderungen an die Erklärbarkeit. Erstens treffen DRL-Agenten sequenzielle Entscheidungen, bei denen Aktionen miteinander verbunden sind und zu einer langfristigen Strategie beitragen, die möglicherweise durch verzögerte Belohnungen beeinflusst wird. Zweitens lernen DRL-Agenten durch Interaktion mit einer Umgebung, in der ihre Ziele nur indirekt durch die Belohnungen, die sie für ihre Handlungen erhalten, definiert sind. Folglich können die von DRL-Agenten entwickelten Strategien von den menschlichen Erwartungen abweichen, selbst wenn sie für die gegebene Belohnungsfunktion optimal sind.

Um diese spezifischen Herausforderungen für erklärbares tiefes bestärkendes Lernen anzugehen, verfolgt diese Dissertation fünf Ziele.

Die ersten drei Ziele betreffen die Entwicklung neuartiger Erklärungsmethoden, die auf die Bedürfnisse von DRL zugeschnitten sind. Zunächst wird in dieser Arbeit ein neuartiger Saliency-Map-Algorithmus vorgestellt, der relevante Informationen für die Entscheidung eines Agenten identifiziert. Im Vergleich zu anderen Saliency-Map-Methoden konzentriert sich dieser Algorithmus auf selektive Bereiche innerhalb des Inputs. Dadurch hilft er, mehrere Zustände schnell zu interpretieren und ihre Wechselbeziehungen innerhalb der Strategie des Agenten aufzudecken. Zweitens wird in dieser Dissertation eine modellunabhängige Methode zur Generierung kontrafaktischer Erklärungen für visuelle DRL-Agenten vorgeschlagen, die veranschaulicht, wie Zustände geändert

werden können, um die Aktion des Agenten zu verändern. Drittens: Um die Erkenntnisse aus den lokalen Erklärungen auf die globale Strategie des Agenten auszuweiten, wird in dieser Dissertation eine neuartige Kombination von lokalen Erklärungen mit globalen Strategiezusammenfassungen vorgestellt. Die Methoden der Strategiezusammenfassung identifizieren repräsentative Zustände für die Strategie des Agenten und ermöglichen es dem Benutzer, durch die Untersuchung einer begrenzten Anzahl von Zuständen ein gutes Verständnis für die Strategie des Agenten zu erlangen.

Die letzten beiden Ziele dieser Dissertation befassen sich mit der Evaluation von Erklärungsmethoden für DRL-Agenten. Dazu werden zunächst die in dieser Dissertation vorgeschlagenen Methoden und andere lokale Erklärungsmethoden mit rechnerischen Metriken bewertet, die ihre Übereinstimmung mit der internen Logik des Agenten messen. Schließlich werden die komplementären und individuellen Beiträge der globalen und lokalen Erklärungen in der oben genannten Kombination in drei Nutzerstudien untersucht. Diese Studien messen Agentenverständnis, angemessenes Vertrauen und die Zufriedenheit mit den Erklärungen. Die Ergebnisse der Experimente in dieser Dissertation zeigen das bedeutende Potenzial kombinierter Erklärungen für DRL-Agenten. Außerdem identifizieren sie Herausforderungen für die Entwicklung zukünftiger Erklärungssysteme für DRL-Agenten.

# Acknowledgments

I want to thank all the people who made this dissertation possible. I extend my deepest gratitude to my supervisor, Prof. Dr. Elisabeth André, whose guidance has been invaluable from the conception to the completion of this thesis. My sincere thanks also go to Prof. Dr. Bernhard Bauer and Prof. Dr. Ofra Amir for generously agreeing to serve as reviewers of this work. Thank you for spending your precious time on this thesis.

Furthermore, I would like to thank all my current and former colleagues at the Chair for Human-Centered Artificial Intelligence for the great cooperation and inspiring conversations. I am particularly grateful to Dominik Schiller for supporting me at the beginning of my academic journey, to Katharina Weitz for sharing her extensive knowledge of statistical analysis and study design, and to Silvan Mertes for the enriching discussions. I would also like to thank Ruben Schlagowski, one of the best teaching partners I could have hoped for.

I also want to thank my research collaborators. My deep gratitude goes to all my co-authors for the excellent cooperation. Moreover, I thank Dan Harborne for his valuable feedback on the paper on which chapter 9 is based. I thank Otto Grothe for his help with annotating the participants' textual responses for the study described in Chapter 10, and I thank Julian Stockmann and Simone Pompe for their help with implementing HRA for Pacman in the second study (Chapter 11).

A heartfelt thanks to my parents, who have always supported me throughout my academic journey. Finally, I am deeply grateful to my wife for her patience and understanding during the countless hours spent away from her in pursuit of this dissertation and to my son, my youngest “co-author”.

# Contents

<b>I. Introduction and Background</b>	<b>14</b>
<b>1. Introduction</b>	<b>15</b>
1.1. Motivation . . . . .	15
1.2. Research Objectives . . . . .	17
1.2.1. Developing Selective Explanation Methods that Focus on Specific Areas. . . . .	17
1.2.2. Creating Counterfactual Explanations for DRL Agents with Visual Input. . . . .	18
1.2.3. Combining Local and Global XRL Methods. . . . .	19
1.2.4. Computational Evaluation of XRL methods. . . . .	19
1.2.5. Holistic User Evaluations of Combined Global and Local Explanations for DRL. . . . .	20
1.3. Overview . . . . .	20
<b>2. Deep Reinforcement Learning</b>	<b>22</b>
2.1. Reinforcement Learning . . . . .	22
2.1.1. The Reinforcement Learning Setting and Notation . . . . .	22
2.1.2. Running Example: The Arcade Learning Environment . . . . .	24
2.1.2.1. Pacman . . . . .	27
2.1.3. Value-Based Reinforcement Learning and Q-Learning . . . . .	29
2.2. The deep Q-Network . . . . .	31
2.2.1. The deep Q-Network Architecture . . . . .	31
2.2.2. Neural Network Basics . . . . .	32
2.3. Training the DQN . . . . .	40
2.3.1. The DQN Loss Function . . . . .	40
2.3.2. Experience Replay . . . . .	41
2.3.3. Double DQN . . . . .	42
2.3.4. Alternative to the DQN: Actor-Critic Algorithms . . . . .	42
2.4. Conclusion . . . . .	43
<b>3. Explainable AI</b>	<b>45</b>
3.1. Terminology . . . . .	45



3.2.	Explanation Methods for Classifiers . . . . .	47
3.2.1.	Saliency Maps or Feature Attribution . . . . .	47
3.2.1.1.	Gradient-based Saliency Maps . . . . .	48
3.2.1.2.	Perturbation-based Saliency Map Approaches . . . . .	49
3.2.1.3.	Layer-Wise Relevance Propagation . . . . .	53
3.2.2.	Counterfactual Explanations . . . . .	57
3.3.	Evaluating XAI . . . . .	60
3.3.1.	Computational Metrics . . . . .	61
3.3.1.1.	Evaluating Saliency Map Fidelity . . . . .	61
3.3.2.	Human User Studies . . . . .	63
3.4.	Conclusion . . . . .	66
 <b>II. Related Work - Explainable Reinforcement Learning</b>		<b>68</b>
 <b>4. Explanation Methods for DRL</b>		<b>70</b>
4.1.	Local Explanations of Agent Behavior . . . . .	71
4.1.1.	Saliency Maps for DRL . . . . .	71
4.1.2.	Counterfactual Explanations for DRL . . . . .	74
4.1.3.	Intrinsic Explanation Methods . . . . .	75
4.1.3.1.	Reward Decomposition . . . . .	76
4.2.	Global explanations of agent behavior . . . . .	78
4.2.1.	Explainable Surrogate Models . . . . .	78
4.2.2.	Intrinsically Explainable Agent Architectures . . . . .	79
4.2.3.	Example-based Policy Explanations . . . . .	80
4.2.3.1.	Strategy Summarization and HIGHLIGHTS . . . . .	81
4.3.	Conclusion . . . . .	85
 <b>5. Evaluation of Explanation Methods for RL Agents</b>		<b>87</b>
5.1.	Human User Studies . . . . .	87
5.2.	Computational Metrics . . . . .	89
5.3.	Conclusion . . . . .	90
 <b>III. Approaches for Explaining DRL Agents</b>		<b>91</b>
 <b>6. LRP-Argmax: Selective Saliency Maps for DRL Agents</b>		<b>92</b>
6.1.	An argmax approach to LRP . . . . .	93
6.2.	LRP on Dueling Q-Networks . . . . .	95
6.3.	Illustration of the Selectivity of the argmax-rule . . . . .	97
6.4.	Sanity Checks . . . . .	100
6.5.	Conclusion . . . . .	102

<b>7. GANterfactul-RL: Counterfactual Explanations for RL Agents with Visual Input</b>	<b>103</b>
7.1. Approach . . . . .	104
7.1.1. The GANterfactul-RL Approach . . . . .	104
7.1.2. Dataset Generation . . . . .	106
7.1.3. Application to the Atari Domain . . . . .	107
7.2. Computational Evaluation . . . . .	108
7.2.1. Used Metrics . . . . .	108
7.2.2. Computational Results . . . . .	109
7.3. Discussion . . . . .	111
7.4. Conclusion . . . . .	112
<b>8. Combining Local and Global Explanations for Agent Behavior</b>	<b>113</b>
8.1. Combining Saliency Maps and HIGHLIGHTS as Video . . . . .	114
8.2. Combining HIGHLIGHTS and Local Explanations as Interactive Images . . . . .	118
8.2.1. Integrating Counterfactual Explanations and HIGHLIGHTS	119
8.2.2. Integrating Reward Decomposition and HIGHLIGHTS .	120
8.3. Discussion and Conclusion . . . . .	121
<b>IV. Computational Evaluation of XRL Approaches</b>	<b>122</b>
<b>9. Benchmarking Perturbation-Based Saliency Maps for DRL Agents</b>	<b>123</b>
9.1. Test-bed . . . . .	124
9.2. Evaluated Saliency Map Approaches . . . . .	125
9.3. Metrics . . . . .	125
9.3.1. Sanity Checks . . . . .	125
9.3.2. Insertion Metric . . . . .	126
9.3.2.1. How to Perturb the Input . . . . .	126
9.3.2.2. Which Output to Measure . . . . .	127
9.3.2.3. Final Setting . . . . .	128
9.4. Parameter Tuning . . . . .	129
9.4.1. Combining Insertion Metric Results . . . . .	129
9.4.2. Choosing a Test Set . . . . .	129
9.4.3. Used Saliency Map Parameters . . . . .	131
9.5. Results . . . . .	134
9.5.1. Sanity Checks . . . . .	134
9.5.2. Insertion Metric . . . . .	138
9.5.3. Run-time Analysis . . . . .	142
9.6. Discussion . . . . .	142
9.6.1. Sanity Checks . . . . .	142

9.6.2. Insertion Metric . . . . .	143
9.6.3. Limitations . . . . .	144
9.7. Conclusion . . . . .	145

## **V. User Studies Evaluating the Effect of Local and Global XRL Approaches 147**

### **10. First Study: Strategy Summaries and Saliency Maps 149**

10.1. Study Design . . . . .	150
10.1.1. Research Question . . . . .	150
10.1.2. Experimental Conditions . . . . .	150
10.1.3. Dependent Variables and Main Tasks . . . . .	152
10.1.4. Hypotheses . . . . .	157
10.1.5. Procedure and Compensation . . . . .	159
10.1.6. Participants . . . . .	159
10.2. Results . . . . .	160
10.3. Discussion & Future Work . . . . .	166
10.4. Conclusion . . . . .	171

### **11. Second Study: Strategy Summaries and Reward Decomposition 172**

11.1. Study Design . . . . .	173
11.1.1. Research Question and Hypothesis . . . . .	173
11.1.2. Dependent Variables and Main Task . . . . .	173
11.1.3. Experimental Conditions . . . . .	175
11.1.4. Procedure . . . . .	176
11.1.5. Participants . . . . .	176
11.2. Results . . . . .	176
11.3. Discussion . . . . .	178
11.4. Conclusion . . . . .	178

### **12. Third Study: Counterfactual Explanations 180**

12.1. Study Design . . . . .	181
12.1.1. Research Question and Hypothesis . . . . .	181
12.1.2. Dependent Variables and Main Tasks . . . . .	182
12.1.3. Conditions and Explanation Presentation . . . . .	183
12.1.4. Procedure and Compensation . . . . .	183
12.1.5. Participants . . . . .	184
12.2. Results . . . . .	185
12.3. Discussion . . . . .	186
12.4. Conclusion . . . . .	188

<b>13. Conclusions From All Three Studies</b>	<b>189</b>
13.1. Potential of Combining Local and Global Explanations . . . . .	189
13.2. Global Explanations Contributed to Appropriate Trust and Agent Understanding . . . . .	189
13.3. Contribution of Local Explanation . . . . .	190
13.3.1. Intrinsic Explanations Outperformed Post-Hoc Explanations . . . . .	190
13.3.2. Assessing Different Post-Hoc Explanations . . . . .	191
13.4. Interactive Explanation Presentation More Effective than Videos	192
13.5. Subjective Explanation Satisfaction Still Lacking . . . . .	192
13.6. Generalization of Results . . . . .	192
<b>VI. Conclusion</b>	<b>194</b>
<b>14. Contributions</b>	<b>195</b>
14.1. Conceptual Contributions . . . . .	195
14.1.1. Novel Approaches to Explaining DRL Agents . . . . .	195
14.1.2. Novel Evaluation Methods for XRL . . . . .	197
14.1.2.1. Advancing Computational Evaluation Methods for XRL . . . . .	197
14.1.2.2. Innovative User Study Design for XRL . . . . .	198
14.2. Technical Contributions . . . . .	199
14.3. Empirical Contributions . . . . .	200
14.3.1. Computational Evaluation of the Fidelity of Post-Hoc Explanation Methods . . . . .	200
14.3.2. User Studies Comparing the Complementary Benefits of Local and Global Explanation Methods for DRL Agents	201
<b>15. Future Work</b>	<b>203</b>
15.1. Advancing Computational Metrics for XRL . . . . .	203
15.2. Towards a Combined Local and Global Explanation Framework for Deep Reinforcement Learning . . . . .	204
15.2.1. Conversational Explanation Interfaces . . . . .	204
15.2.2. Addressing Low Subjective Explanation Satisfaction . . . . .	205
15.2.3. Developing Local Explanations that Foster Appropriate Trust . . . . .	206
15.3. Explanations for Multi-Agent Reinforcement Learning . . . . .	207
<b>Bibliography</b>	<b>208</b>

<b>VII.Appendix</b>	<b>223</b>
<b>A. My Publications</b>	<b>224</b>
A.1. Main Publications . . . . .	224
A.2. Other Publications . . . . .	225
<b>B. Appendix to the First User Study</b>	<b>228</b>
B.1. Participants Demographics . . . . .	228
B.2. Supplementary Results . . . . .	232
B.3. Evaluation of the Retrospection Task . . . . .	236
B.4. Questionnaire . . . . .	239
<b>C. Appendix to the Second User Study</b>	<b>250</b>
C.1. Survey Information . . . . .	250
<b>D. Appendix to the Third User Study</b>	<b>258</b>
D.1. Implementation Details . . . . .	258
D.1.1. Training Data . . . . .	258
D.1.2. Training GANterfactual-RL . . . . .	259
D.1.3. Training the Counterfactual State Explanations Model . . . . .	259
D.2. User Study Demographics . . . . .	260
D.3. Agent Performance . . . . .	263
D.4. Example Counterfactuals . . . . .	264
D.5. Full User Study . . . . .	269

## **Part I.**

# **Introduction and Background**

# 1. Introduction

## 1.1. Motivation

I want to begin with a personal story that illustrates the motivation behind this thesis. Some time ago, my father got a new car that was able to recognize traffic signs automatically and combined this with adaptive cruise control to brake and accelerate accordingly. While he was demonstrating this feature, which was still novel and interesting to us, we encountered a situation where the car slowed down far more than necessary. This incident was concerning to us. Was the unexpected behavior due to the system misinterpreting a traffic sign, was it due to a malfunction in the adaptive cruise control, or was there no malfunction and we just overlooked a traffic sign that the system recognized accurately? We spent the rest of the drive trying to figure out what happened and whether it would happen again. We carefully observed how the car reacted to each new traffic sign, and my father was ready to hit the brakes at any moment. Any initial trust in the system was gone after experiencing this unexpected behavior. We were concerned about the potential dangers of a similar error causing inadequate braking in a more dangerous situation. An explanation mechanism that could clarify why the car was excessively slowing down would have allowed us to judge whether we could continue to trust the system.

This anecdotal story illustrates the motivation of my dissertation. As artificial intelligence (AI) continues to evolve, Reinforcement Learning (RL) agents are introduced into increasingly high-risk domains such as healthcare, automated vehicles, and robotic navigation [Yu et al., 2021; Kiran et al., 2022; Fan et al., 2020]. Since these systems are used by humans in such high-stakes domains, users must be able to understand and anticipate their behavior to facilitate human-agent cooperation [Silva et al., 2022]. For instance, in the example above, we wanted to anticipate in which situations we could rely on the automated vehicle. Similarly, a clinician must understand the treatment plan proposed by an agent to assess whether it aligns with the patient’s preferences.

The growing recognition of the importance of human understanding of agent behavior, coupled with the increasing complexity of modern AI systems, has sparked a rising interest in developing Explainable AI (XAI) methods [Doshi-Velez and Kim, 2017; Gunning and Aha, 2019]. The idea of making AI systems explainable is not new. It has been a topic of discussion since the early days of

expert systems [Swartout, 1983; Chandrasekaran et al., 1989]. However, modern AI algorithms use more complex representations and algorithms (such as deep neural networks), making them more difficult to interpret. For instance, in classical planning approaches such as the Belief-Desire-Intention (BDI) framework [Rao and Georgeff, 1995] the agent’s goals are explicitly defined. In contrast, current RL agents often employ policies that have been trained using complex reward functions and high-dimensional feature representations that are difficult for humans to understand [Heuillet et al., 2021].

This thesis addresses the challenge of explaining the behavior of Deep Reinforcement Learning (DRL) agents with visual input. Several factors make explaining the decisions of such DRL agents particularly challenging.

For one, RL agents are employed in sequential decision-making tasks – their actions are not isolated. These actions are part of a long-term strategy that might be influenced by delayed rewards. For example, consider a warehouse robot trained to place packages on shelves. An observer watching the robot navigate through empty space may not immediately understand the purpose of the robot’s movement. Understanding the robot’s *current* actions requires knowledge of its *future* goals (e.g., reaching an empty shelf).

Second, RL agents are not trained on a given ground truth strategy. RL agents learn by interacting with an environment and observing what reward they receive for each action. Here, the reward only indirectly specifies the agent’s goals [Langosco et al., 2022]. Thus, the emerging strategies might differ from what humans expect, even if the strategy is optimal for the reward function. A prominent example of this phenomenon is the chess-playing DRL agent AlphaZero [Silver et al., 2018]. This DRL agent learned to play chess at a superhuman level only by self-play without being exposed to human strategies. As a result, it adopted strategies that were rare in human play before AlphaZero. Today, these strategies have begun influencing human play after players thoroughly analyzed AlphaZero’s games [Sadler et al., 2019].

Finally, DRL agents are trained by employing deep neural networks. Such networks typically train millions of interconnected parameters when they are used on vast visual input spaces. This complexity adds another layer of difficulty to interpreting and explaining the behavior of DRL agents.

When I started working on this thesis, the majority of previous work in XAI concentrated on classifiers trained by supervised learning. The unique challenges associated with explaining DRL agents, as outlined above, received less attention. Research on Explainable Reinforcement Learning (XRL) leaned towards *global* explanations that describe the agent’s overall policy [Alharin et al., 2020]. Such methods include demonstrating informative interactions or distilling the agent’s policy into more comprehensible models, such as decision trees [Amitai and Amir, 2023]. *Local* explanations, which analyze individual decisions of an RL agent, have mainly applied XAI methods for image classifiers directly to



DRL [Alharin et al., 2020], especially in the context of DRL with visual input.

This dissertation addresses this research gap by proposing a combined global and local explanation framework for DRL. To this end, it builds on existing work from XAI for image classifiers to develop local explanation methods that are specifically tailored to DRL with visual input. The effectiveness of these local explanation methods is initially assessed through computational metrics. Such computational metrics provide a good way to choose promising methods before moving on to more extensive user studies. Furthermore, they are crucial for selecting explanation methods that accurately reflect the agent’s internal reasoning rather than producing explanations that only appear convincing to users [Mohseni et al., 2021b]. Finally, this dissertation describes a method to integrate local explanations for DRL agents with global strategy summaries that demonstrate the agents’ behavior. The individual and complementary benefits of the local and global explanations in this integration are evaluated in three user studies using the Atari game Pacman. Such user studies are equally important as computational metrics since they determine whether the explanations actually help users understand the agents [Mohseni et al., 2021b]. An explanation that perfectly reflects the agent’s reasoning but is incomprehensible to users is not helpful [Miller et al., 2017].

The following section elaborates on the individual research objectives of this dissertation.

## 1.2. Research Objectives

The main research question in this thesis is *how we can make the behavior of deep reinforcement learning agents explainable*. To address this overarching question, this thesis delineates five specific research objectives. The first three deal with the development of novel explanation methods that address the specific challenges of DRL agents, while the remaining ones address the evaluation of such approaches.

### 1.2.1. Developing Selective Explanation Methods that Focus on Specific Areas.

The most common explanation method for neural networks with visual input is the creation of saliency maps, which highlight the most important pixels for the network’s decision [Arrieta et al., 2020]. Previous methods for such saliency maps have focused primarily on classification tasks, attempting to uncover multiple detailed reasons behind a decision. However, as we have seen above, DRL involves long-term decision-making where it is often necessary to analyze multiple states to understand individual actions. This nature of DRL necessitates

a departure from previous saliency map approaches, as the amount of information becomes unwieldy when attempting to comprehend detailed explanations for multiple states. Recognizing this, one research objective of this thesis is the development of more selective saliency maps that focus on specific areas within states. This selective approach not only helps users to parse information more effectively but is also supported by existing literature, which indicates that DRL agents tend to concentrate on specific objects within their visual input [Iyer et al., 2018; Goel et al., 2018].

In particular, this thesis:

- introduces a selective saliency map method for explaining DRL agents, which does not overwhelm the user with unnecessary information.
- demonstrates how this approach can be applied to state-of-the-art DRL agents.

### **1.2.2. Creating Counterfactual Explanations for DRL Agents with Visual Input.**

Another prominent and successful family of explanation approaches for classification systems are counterfactual explanations [Arrieta et al., 2020]. In the context of reinforcement learning, counterfactual explanations answer "Why not?" or "What if?" questions by demonstrating the smallest modification required in a state to prompt the agent to select an alternative action. Creating counterfactual explanations for DRL agents with visual input poses unique challenges compared to image classification. This is due to the fact that the counterfactual explanations have to account for the agent's overarching policy, which requires long-term decision-making. Furthermore, many counterfactual generation methods for classification models rely on the models' training data. Because of the absence of direct training datasets for DRL agents, these methods cannot be directly applied to DRL [Wells and Bednarz, 2021]. Consequently, at the start of this thesis, there was only a single method for generating counterfactual explanations for deep RL agents with visual inputs - the counterfactual state explanation method by Olson et al. [2021]. However, this approach is quite dependent on the model architecture, making it difficult to apply to different agents, and it only indirectly incorporates the agent's actions.

This thesis

- proposes a novel model-agnostic approach to create counterfactual explanations for DRL agents with visual input.
- shows that our approach outperforms the previous method in computational metrics and a user study.

### 1.2.3. Combining Local and Global XRL Methods.

Explainable reinforcement learning methods can broadly be divided into two classes based on their scope: local and global explanations [Alharin et al., 2020]. Local explanations analyze specific actions of the agent, whereas global explanations attempt to describe the overarching policy of an agent. These methods offer complementary perspectives, with local explanations providing insight into the agent’s reasoning in specific instances and global explanations offering a broader view of the agent’s strategy without detailing the specific reasons for each decision. To leverage these complementary perspectives, this thesis investigates the integration of different local and global explanation methods for DRL agents.

In particular, this thesis

- presents the first combination of local explanations and global strategy summaries for DRL agents.
- evaluates their joint and separate contributions through human user studies.

### 1.2.4. Computational Evaluation of XRL methods.

Computational metrics provide a systematic approach to assess explanations by quantifiable measures that can be calculated automatically without human intervention. This approach facilitates the cost-effective evaluation of a wide range of explanation methods prior to resource-intensive user studies. Moreover, it enables the evaluation of the fidelity of explanations to the internal reasoning of the agent [Mohseni et al., 2021b].

DRL agents pose unique challenges to computational evaluation due to the inherent complexity of their sequential decision-making, which is integrated into overarching policies involving long-term considerations [Heuillet et al., 2021]. For instance, in value-based DRL algorithms, the output encodes both the value of the current state and the expected future reward after performing each possible action in that state. This ambiguity should be taken into account when evaluating explanations for DRL agents.

Computational evaluation is particularly important for post-hoc explanation methods that are not intrinsically built into the agent’s model but are applied after the agent has been trained. Further, it is even more important for model-agnostic methods that do not interact with the agent’s internal model at all.

This dissertation targets the aforementioned challenges and considerations in two ways:

- It proposes a computational evaluation methodology for saliency maps that is tailored to the specific needs of DRL.

- It computationally evaluates two model-agnostic post-hoc explanation methods for DRL agents: perturbation-based saliency maps and counterfactual explanations.

### 1.2.5. Holistic User Evaluations of Combined Global and Local Explanations for DRL.

In addition to computational metrics, user studies are necessary to ensure explanations are not only technically accurate but also comprehensible and practical for human users [Miller et al., 2017; Mohseni et al., 2021b]. Previous user studies on XRL have mostly focused on individual explanation techniques in isolation, overlooking the potential synergistic effects of combining different types of explanations. Before this dissertation, no study evaluated the combined and individual benefits of global and local explanations for DRL agents despite their potential complementary benefits. Another problem is that many XRL studies focused on individual dimensions of the explanation process, such as the participants’ mental model of the agent or their subjective satisfaction with the explanations.

This dissertation addresses these shortcomings by conducting comprehensive user studies that evaluate combinations of global and local explanations in the context of DRL agents. Moreover, it presents a holistic user study design that evaluates three distinct dimensions: agent understanding, appropriate trust, and explanation satisfaction. By exploring these dimensions in tandem, the study aims to uncover the unique and combined contributions of different local and global explanation approaches. The insights gained from the studies in this dissertation will inform the future development of XRL frameworks.

## 1.3. Overview

This thesis is structured as follows. The remainder of Part I provides the theoretical background that is necessary to understand the thesis. Chapter 2 gives a brief introduction to deep reinforcement learning. In particular, it introduces the Deep-Q Network (DQN) and its application to the Arcade Learning Environment (ALE). This will be the main test-bed throughout this work. Chapter 3 serves as an introduction to Explainable AI (XAI). It provides an overview of key XAI concepts and XAI methods for image classifiers that have served as the basis for explainable RL methods.

Part II gives an overview of the related work on Explainable Reinforcement Learning (XRL). It focuses on XAI works that specifically target RL agents. Chapter 4 deals with related methods on how to explain RL agents in contrast

to classification models. Chapter 5 shows related work on how to evaluate explanations for (deep) RL agents. The chapter starts by introducing computational metrics and then describes different user studies.

Part III presents novel concepts and techniques for explaining RL agents. Chapter 6 presents a novel algorithm that selectively highlights the information that was important for an agent’s decision. Chapter 7 introduces a novel algorithm for generating counterfactual explanations for RL agents and evaluates these explanations computationally. Chapter 8 presents the first combination of local explanations with global strategy summaries for RL agents.

Part IV deals with the computational evaluation of explanations for RL agents. Chapter 9 computationally benchmarks model-agnostic XRL methods with respect to their fidelity to the agents’ internal reasoning and their dependence on the agents’ learned parameters.

Part V describes user studies that evaluate different combinations of global and local explanation methods for RL agents, including the methods proposed in the previous parts. The studies were conducted over the course of three different experiments that used a very similar setup, but each evaluated different explanation approaches. All three studies use the Atari game Pacman and evaluate the users’ agent understanding and subjective explanation satisfaction. Chapters 10 and 12 also measure appropriate trust. Chapter 10 examines a combination of strategy summaries and saliency maps. Chapters 11 and 12 investigate combinations of strategy summaries with reward decomposition and counterfactual explanations, respectively. Finally, Chapter 13 discusses the individual study results in a common context.

Finally, Part VI concludes the thesis by summarizing the contributions in Chapter 14 and outlining future work in Chapter 15.

## 2. Deep Reinforcement Learning

The background of this thesis is divided into two parts.

First, this chapter will provide the theoretical background of the Reinforcement algorithms that stand at the center of this thesis since they are the agents that we want to explain. The chapter starts by providing a basic background on reinforcement learning. Then, it will introduce the DQN as a specific example of a DRL algorithm.

Second, Chapter 3 will provide an overview over general XAI concepts and XAI methods that focus on classification models.

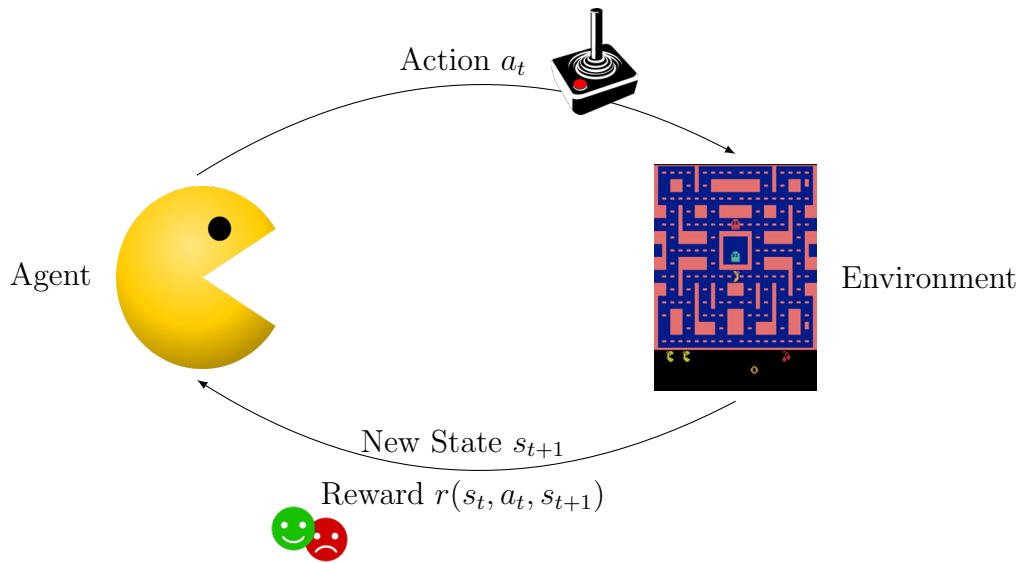
### 2.1. Reinforcement Learning

This section lays out the foundations of Reinforcement Learning (RL) that are necessary for this thesis.

#### 2.1.1. The Reinforcement Learning Setting and Notation

In the context of reinforcement learning, the system that is supposed to solve the problem is called **agent**, and everything that this agent can interact with is called **environment**. For example, in an Atari game like Pacman (see Section 2.1.2.1), the game itself is the environment, and the player controlling Pacman is the agent. To make the abstract environment tangible, one defines **states**  $s$ , which represent different states of the environment, and subsumes them in the so-called **state space**  $\mathcal{S}$ . In the Atari example, the states can, for example, be defined based on the visual frames of the game (more specifics in Section 2.1.2.1).

The agent also has a set  $\mathcal{A}$  of **actions**  $a$  that it can use to influence the environment. This thesis is limited to agents with discrete actions  $\{a_1, a_2, \dots\}$  (e.g., Pacman can move up, down, left, etc.) instead of a continuous action space. After each action, the agent receives a **reward**  $r \in \mathcal{R}$  that describes how good the chosen action was in this situation (e.g., Pacman receives points for eating pellets). These rewards implicitly specify the agent’s goal. To ease the notation in this thesis, we assume that the reward can be described by a deterministic **reward function**  $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ . The concepts within this



**Figure 2.1.:** Interaction of an RL agent with its environment.

thesis can be extended to stochastic reward functions as long as  $\mathcal{R}$  is bounded. The basic interaction loop of an RL agent is shown in Figure 2.1.

A sequence of successive states and actions  $s_1, a_1, \dots, a_{t-1}, s_t$  is called a **trajectory**. For reinforcement learning, one usually assumes that the environment has one or more **final states**, after which the interaction ends. In a computer game like Pacman, this occurs when the agent loses or wins the game. A sequence of successive states and actions  $s_1, a_1, \dots, a_{T-1}, s_T$  that ends with a final state  $s_T$  is called an **episode**.

The behavior of an agent is determined by its **strategy or policy**  $\pi$ . This thesis mostly focuses on **deterministic policies**  $\pi : \mathcal{S} \rightarrow \mathcal{A}$ , where  $\pi(s)$  directly specifies the action to be performed in the state  $s \in \mathcal{S}$ . There are also **stochastic policies** where  $\pi(s)$  describes the agent's action probability distribution for each action  $a \in \mathcal{A}$  in state  $s \in \mathcal{S}$ . Every deterministic policy can be written as a stochastic policy by setting the likelihood of the deterministic action, which the agent should execute, to 1 and the likelihood of all other actions to 0. Unless stated otherwise, this thesis assumes deterministic policies.

The agent's goal is to learn an optimal strategy  $\pi^*$ . Here, optimal means that the agent receives the maximum amount of cumulative reward over the course of an episode or trajectory if it follows this strategy. To assess how successful a time-step  $t$  was within a given episode  $s_1, a_1, \dots, a_{T-1}, s_T$ , we define an episode-dependent **return**  $G_t$ . The idea here is that the time-step  $t$  also participated to some extent in the rewards of the next steps. For example, in Pacman, every action on the way to a pellet contributes to the points received by the pellet.

Thereby, the share of the time-step  $t$  in the subsequent rewards decreases more and more since these were less and less influenced by the step – the action in which Pacman eats a pellet contributes more than the actions on the way to the pellet. To formulate this, we choose a so-called **discount value**  $\gamma \in [0, 1]$  and define the **discounted return**:

$$G_t(s_1, a_1, \dots, a_{T-1}, s_T) := \sum_{i=t}^{T-1} \gamma^{i-t} r(s_i, a_i, s_{i+1}).$$

This definition also works for environments without final states and potentially infinite trajectories since  $\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i, s_{i+1})$  converges for  $\gamma < 1$  and bounded rewards.

In addition to this general problem definition, many RL algorithms assume that they operate in a Markov Decision Process, which we will define now.

Let  $P(s_{t+1} = s' | s_1, a_1, \dots, a_t, s_t)$  denote the conditional **transition probability** that the agent ends up in state  $s'$ , given the condition that it has so far passed through states  $s_1, \dots, s_t$  and performed actions  $a_1, \dots, a_t$ . In computer games like Pacman, the transition probability  $P$  is given by the game rules. In this thesis, we assume a finite number of states and actions since it allows us to use simple probabilities to ease notation. The concepts work analogously for infinite state and action spaces, but we would have to use probability densities.

A tuple  $(\mathcal{S}, \mathcal{A}, P, r)$  of a finite state-space  $\mathcal{S}$ , a finite action-space  $\mathcal{A}$ , transition probabilities  $P$  and a reward function  $r$  is called a **finite Markov Decision Process (MDP)** if  $\mathcal{S}$ ,  $\mathcal{A}$  and  $P$  satisfy the Markov property.

$\mathcal{S}, \mathcal{A}$  and  $P$  satisfy the **Markov property** if, for any trajectory  $s_1, a_1, \dots, s_t, a_t$  of successive states  $s_i \in \mathcal{S}$  and actions  $a_i \in \mathcal{A}$ , the next state depends only on the current state  $s_t$  and the currently chosen action  $a_t$ . That is, for all  $s' \in \mathcal{S}$  holds

$$P(s_{t+1} = s' | s_t, a_t) = P(s_{t+1} = s' | s_1, a_1, \dots, s_t, a_t).$$

Given a MDP, we can shorten the notation for the probability that state  $s' \in \mathcal{S}$  follows state  $s \in \mathcal{S}$  and action  $a \in \mathcal{A}$  to:

$$P(s' | s, a) := P(s_{t+1} = s' | s_t = s, a_t = a)$$

### 2.1.2. Running Example: The Arcade Learning Environment

As an example of an MDP, this section introduces the Arcade Learning Environment (ALE) [Bellemare et al., 2013], which will be used as the running example and use case throughout this thesis.

The ALE contains over 55 Atari 2600 game environments, serving as a standardized test-bed for reinforcement learning algorithms. Since the ALE provides



a common interface for all of its games, it allows for RL algorithms to be tested across various games without the need for substantial modifications. Each game has a unique set of rules, objectives, and dynamics. This diversity ensures that the algorithms are not just tailored to a specific kind of task but are capable of handling a wide range of scenarios.

Another feature of the ALE is the provision of existing human baselines through the scores within the Atari 2600 games. These baselines enable a direct comparison of the performance of RL algorithms with human-level performance.

In the remainder of this section, we will look at how the ALE defines the transition probabilities  $P$ , the action space  $\mathcal{A}$ , the rewards  $r$ , and the state space  $\mathcal{S}$  necessary for an MDP  $A(\mathcal{S}, \mathcal{A}, P, r)$ . This will include some adjustments to the ALE by Mnih et al. [2015].

**Transition Probabilities  $P$ .** The ALE uses the Stella<sup>1</sup> emulator to simulate Atari 2600 games. Thus, this emulator defines the transition probabilities  $P(s'|s, a)$  based on the different game rules.



**Figure 2.2.:** The Atari 2600 Controller.

**Action Space  $\mathcal{A}$ .** The actions that the Stella emulator can use correspond to all 18 possible actions of the Atari 2600 joystick (see Figure 2.2). Depending on the game, only the actions that are meaningful within the game are used. For example, in Pacman, the agent has nine different actions to choose from (*do nothing, up, down, left, right, up-left, up-right, down-left, down-right*). An addition by Mnih et al. [2015] is that the agent chooses an action only every four frames, and this action is repeated for the next four frames. This is often called

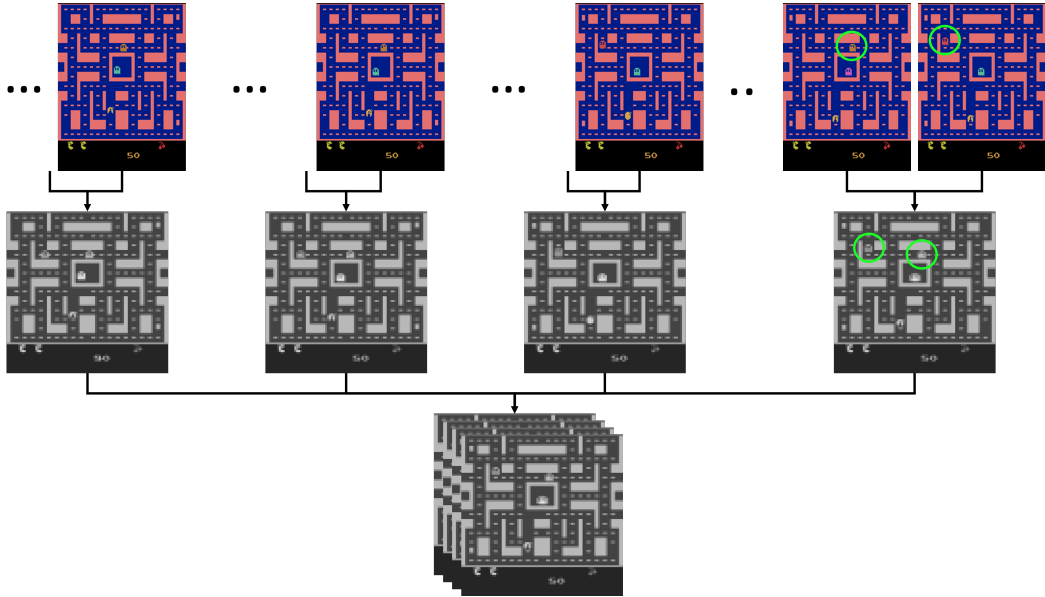
**frame skipping.** In the first row of Figure 2.3, these skipped frames are represented by "...".

**Reward  $r$ .** The ALE's base reward is the increase in the in-game score between the beginning of the four skipped frames and the frame after them. This thesis uses different variations of this reward, which will be explained when they are employed.

**State Space  $\mathcal{S}$ .** As observations, the ALE uses  $210 \times 160 \times 3$  RGB images with a 128-color palette consisting of the raw pixel values of each frame of the

---

<sup>1</sup><http://stella.sourceforge.net/>



**Figure 2.3.:** The state preprocessing steps used by the Atari environment in this thesis. The first row shows the in-game frames of the Atari emulator. The dots represent skipped frames, where the agent repeats the action chosen in the last state. Only the penultimate frame shows a skipped frame to illustrate the *max*-operation during preprocessing. For each observed frame, we take the maximum of the color values of the frame and the immediately preceding skipped frame. The reason for this is that the Atari 2600 did not display every object in every frame to conserve computing power. The green circles in the last two frames highlight this phenomenon. In the last frame of the first row, the red ghost is visible (green circle), but the yellow ghost is not. In contrast, the red ghost is not visible in the penultimate image, but the yellow ghost is (green circle). By taking the maximum value of each color channel before converting to grayscale, both ghosts become visible in the last frame of the second row, as highlighted by the green circles. Finally, the last four preprocessed frames (second row) are stacked to encode temporal information in the final states (third row).

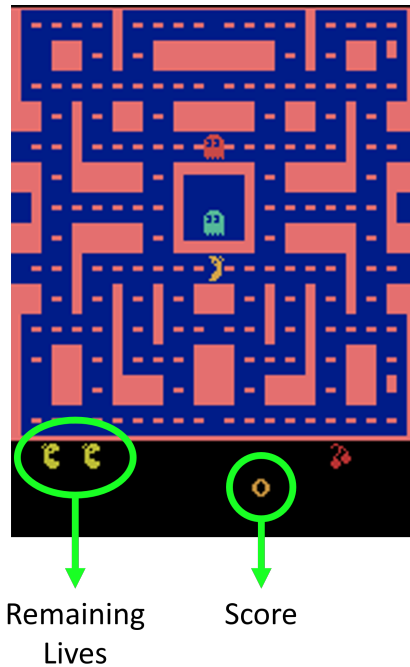
in-game screen. Mnih et al. [2015] further preprocess these frames. First, to encode a single frame, they take the maximum value for each pixel color value over the frame being encoded and the immediately preceding frame (a skipped frame where the agent did not even choose an action). This is done since the Atari 2600 did not show every object in every frame to save computing power (illustrated by the green circles in Figure 2.3). Second, the resulting frame is converted to grayscale and resized down to  $84 \times 84$  pixels. In Figure 2.3, the transition from the first to the second row represents these two preprocessing steps.

Looking only at single frames does not result in an MDP since, for example, the direction of movement of game objects like Pacman is not observable. Therefore, the states used by the agents in this thesis and by Mnih et al. [2015] consist of the last four preprocessed frames for which the agent chooses an action. These frames are stacked, resulting in final states of size  $84 \times 84 \times 4$ . So-called **stacked frames**. In Figure 2.3, this stacking process is represented by the transition from the second to the third row. The skipped frames are only implicitly included in the state through the *max*-operation described above. If we count the skipped frames, each state contains information about the last 16 frames.

**Episodes.** Originally, an ALE episode starts with the first frame of the game and ends when the game ends. However, Mnih et al. [2015] adjust this for games that contain lives, such that a new episode starts whenever a life is lost. This thesis uses these episodes proposed by Mnih et al. [2015]. Furthermore, to introduce randomness into deterministic games, Mnih et al. [2015] force the agent to repeat the “do nothing” action at the beginning of each game for a random amount of steps between 0 and 30 until it is allowed to choose actions based on its policy.

### 2.1.2.1. Pacman

One Atari game that will be used as an example throughout all parts of this thesis is Pacman. Specifically, we will use the Atari 2600 version called *MsPacman*, which we will refer to as Pacman for simplicity and because it is commonly known by that name. Since this game will be used throughout the thesis, this section explains the rules of the game. In the game, the player controls Pacman by moving the joystick of the Atari controller (Figure 2.2) in a specific direction (*up*, *down*, *left*, *right*, *up-left*, *up-right*, *down-left*, *down-right*) or by doing nothing. The ambiguous actions, like *down-right*, always prioritize changing Pacman’s current direction of movement, e.g., at an intersection or when turning around.










**Figure 2.4.:** A typical screen of the game Pacman. The bottom of the screen displays the remaining lives and the score (green circles). The other objects are explained in Table 2.1.

Pacman earns points by eating food pellets while navigating through a maze (see Figure 2.4) and avoiding ghosts. There are two types of pellets: regular pills, for which Pacman receives 10 points, and power pills, which are worth 50 points. Power Pills also turn the ghosts blue, which makes them edible by Pacman. Pacman receives 200, 400, 800, 1600 points for each ghost it eats successively. After a blue ghost is eaten, it turns into eyes and moves back to the center of the maze. At random intervals, cherries spawn and move through the labyrinth. Eating a cherry is worth 100 points. The total score is displayed at the bottom of the screen. Table 2.1 shows a list of all relevant objects in Pacman.

Pacman has three lives, shown at the bottom left of the screen. When Pacman is touched by ghosts, it loses a life and respawns in the center of the maze. When Pacman loses its last life, the game ends.

After eating every normal pill in the maze, Pacman is transferred to a new maze with a modified layout. This sometimes even changes the color scheme of the game and is one of the reasons why RL agents struggle to perform well in Pacman [Mnih et al., 2015].

**Table 2.1.:** Important objects within the game Pacman.

Visual Representation	Object Name	Function within the Game
	Pacman	Controlled by the player.
	Ghost	When Pacman touches a ghost, it loses a life.
	Normal Pill	Gives 10 points when eaten.
	Power Pill	Gives 50 points when eaten, turns ghost blue for a short duration
	Blue Ghost	Pacman receives 200, 400, 800, 1600 points for each blue ghost it eats successively.
	Ghost Eyes	When Pacman eats a blue ghost, the ghost turns into eyes and returns to the center of the maze.
	Cherry	Gives 100 points when eaten.

### 2.1.3. Value-Based Reinforcement Learning and Q-Learning

So-called **value-based reinforcement learning** approaches, like the DQN that will be used as the running example in this thesis, do not learn the policy  $\pi$  directly. Instead, they approximate the expected future return of states and actions and choose the actions that promise the biggest return.

To this end, these algorithms define the **state value function**  $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$  that assigns to each state  $s \in \mathcal{S}$  the estimated discounted return that the agent will achieve if it starts in state  $s$  and then acts according to the policy  $\pi$ . For example, a Pacman state where Pacman is close to an edible pellet should be more valuable than a state where Pacman is far away from any pellet. To define  $V^\pi$ , we set

$$V^\pi(s) = \mathbb{E}_\pi\{G_t(s_1, a_1, \dots, a_{T-1}, s_T) | s_t = s\},$$

where  $\mathbb{E}_\pi\{G_t(s_1, a_1, \dots, a_{T-1}, s_T) | s_t = s\}$  computes the expected value of all episodes played with strategy  $\pi$  and where  $s$  appears at the  $t$ -th position of the episode. We need the expected value because the policy and the problem do not have to be deterministic. That is, different actions  $a_t$  and successor states  $s_{t+1}$  can follow the state  $s_t$ .

Analogously we can define a **state-action-value or Q-value function**  $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  that assigns to each state-action pair  $(s, a) \in \mathcal{S} \times \mathcal{A}$  the estimated reward that the agent will achieve if it performs action  $a$  in state  $s$  and then follows strategy  $\pi$ . Going back to the Pacman example from above, a state-action pair that moves closer to a pellet should be more valuable than a state-action pair that moves away from a pellet. To define  $Q^\pi$ , we set

$$Q^\pi(s, a) = \mathbb{E}_\pi\{G_t(s_1, a_1, \dots, a_{T-1}, s_T) | s_t = s, a_t = a\},$$

where  $\mathbb{E}_\pi\{G_t(s_1, a_1, \dots, a_{T-1}, s_T) | s_t = s, a_t = a\}$  computes the expected value of all episodes played with strategy  $\pi$  and where  $s$  followed by  $a$  appears at the  $t$ -th position of the episode. Again, we need the expected value because the problem does not have to be deterministic. That is, different successor states  $s_{t+1}$  can follow the state-action pair  $(s_t, a_t)$ .

To ease notation, the policy  $\pi$  is often omitted from  $V^\pi(s)$  or  $Q^\pi(s, a)$  if it is clear which policy is meant.

The prominent reinforcement learning algorithm family of **Q-Learning** approaches, aims to learn the optimal Q-Value function  $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ . If  $Q^*(s, a)$  was known, then we could define an optimal strategy  $\pi^*$  by choosing in each state  $s \in \mathcal{S}$  the action  $a \in \mathcal{A}$  for which  $Q^*(s, a)$  is largest. We call such a strategy  $\pi(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a)$  for a Q-value function  $Q$  a **greedy strategy**.

To approximate  $Q^*$ , in reinforcement learning, one assumes that the problem is a finite MDP. In this case, there exists a unique  $Q^*$  and it satisfies the **Bellman optimality equation**:

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

See [Sutton and Barto, 2018] equation (3.20).

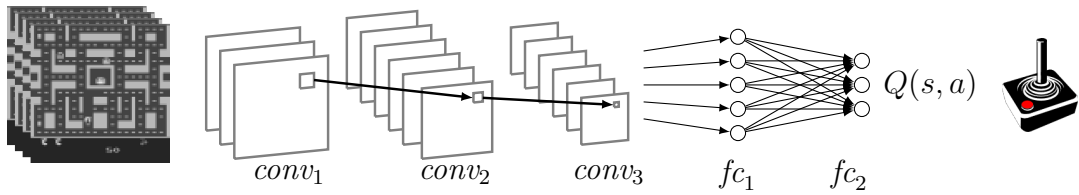
Q-learning methods use the Bellman optimality equation to iteratively adjust the Q values. That is, in each step  $i$  one chooses

$$Q^{i+1}(s, a) = \sum_{s' \in \mathcal{S}} P(s'|s, a) (r(s, a, s') + \gamma \max_{a' \in \mathcal{A}} Q^i(s', a')). \quad (2.1)$$

where  $Q^0$  often sets all values to 0.

This converges to  $Q^*$  due to the Bellman equation for  $i \rightarrow \infty$  (see [Sutton and Barto, 2018] chapter 4.4).

In practice, it is usually not possible to calculate this limit. Therefore, Q-value-based reinforcement learning uses different approximation functions  $Q(s, a; \theta)$ , where we adjust the parameters  $\theta$  to approximate  $Q^*$ .



**Figure 2.5.:** A simplified illustration of the architecture of the DQN. It consists of three convolutional and two fully connected layers.

## 2.2. The deep Q-Network

The running example for Deep Reinforcement Learning agents used in this thesis is the deep Q-Network (DQN) [Mnih et al., 2015]. It uses a neural network  $Q(s, a; \theta)$  with trainable parameters  $\theta$  to approximate the optimal strategy  $Q^*$ . In this section, we will take a detailed look at the architecture of this network as it will be the use case for all XAI algorithms in this thesis. Section 2.2.1 will describe the specific architecture of the DQN. Section 2.2.2 will define all the neural network basics needed to understand the DQN architecture. If you are already familiar with this background, you can skip Section 2.2.2.

### 2.2.1. The deep Q-Network Architecture

The architectural graph of the DQN is depicted in Figure 2.5. We use the DQN version of Dhariwal et al. [2017], which differs slightly from Mnih et al. [2015] in the number of neurons in the hidden, fully connected layer. It consists of three 2D convolution layers followed by two fully connected layers. The first convolutional layer  $conv_1$  applies 32 filters with a width and height of 8 and a stride of (4, 4) to convolve the  $84 \times 84 \times 4$  ALE input states. The second convolutional layer,  $conv_2$ , consists of 64 filters of size  $4 \times 4$  with a stride of (2, 2) and the last convolutional layer,  $conv_3$ , consists of 64 filters of size  $3 \times 3$  with a stride of (1, 1). All convolutional layers use same-padding. The output of  $conv_3$  is flattened and fed into the first fully connected layer,  $fc_1$ , which has 256 neurons. Finally,  $fc_1$  is succeeded by the output layer  $fc_2$ , which is a fully connected layer with one neuron for each possible action of the current game. All layers except the output layer  $fc_2$  use a ReLU activation function. The output layer does not use an activation function to allow for Q-values in the full range of  $\mathbb{R}$ .

This architecture covers the most common building blocks of DRL architectures while avoiding domain-specific techniques. Therefore, the results of this thesis can be applied to a broad range of DRL variations.

### 2.2.2. Neural Network Basics

To understand the architecture of the DQN, we have to understand how the underlying neural network works. Therefore, this section briefly introduces the necessary theory about artificial neural networks.

In the most general description, artificial neural networks are functions of the form  $f : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}^m$ . Here, we call  $\theta \in \Theta \subset \mathbb{R}^k$  the **parameters** of the network. The goal of such a network is to approximate given functions. Let  $f^* : \mathbb{R}^n \rightarrow \mathbb{R}^m$  be a given function, which is called **ground truth**. In the examples presented in this thesis, the ground truth function is the optimal policy,  $\pi^*$ , for solving an Atari game such as Pacman. Then we try to find parameters  $\theta \in \Theta$  such that  $f(\cdot, \theta)$  approximates this function as well as possible. It can be shown that any continuous function can be approximated by simple neural networks, albeit with an arbitrary amount of parameters (see for example [Pinkus, 1999]). The process of determining optimal parameters is commonly referred to as **training**. The act of applying  $f(\cdot, \theta)$  to an input  $x$  is called **forward pass**. Employing a neural network after training is known as **inference**.

In the remainder of this section, we will take an in-depth look at how the network used by the DQN works during the forward pass, since this thesis focuses on explaining trained DRL agents during inference.

**Neurons.** Inspired by biological neural networks, we want to trace artificial neural networks back to neurons. So first, we need to define neurons, the basic building blocks of neural networks. The following definition can be found, for example, in [Haykin, 2009] page 11 and in [Goldberg, 2017] page 41.

A **neuron** is a function  $f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$  of the form

$$f(x; \omega, b) = \sigma(\omega \cdot x + b)$$

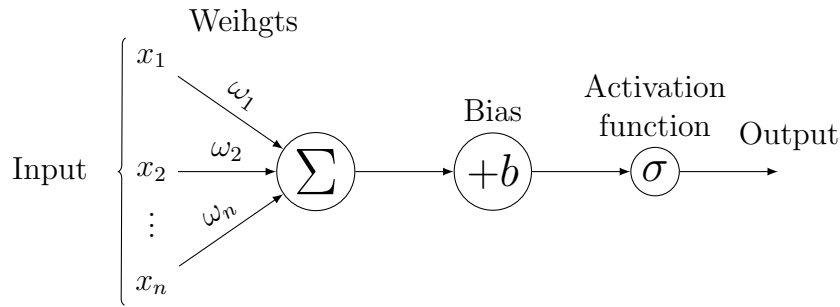
where  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  und  $n \in \mathbb{N}$  is the so-called **activation function**.

Further, we call  $x \in \mathbb{R}^n$  the **input**,  $\omega \in \mathbb{R}^n$  the **weights**, and  $b \in \mathbb{R}$  the **bias** of the neuron. The result  $f(x; \omega, b)$  of a neuron is typically referred to as **output**. For example, when learning to play Pacman, a neuron's output might approximate the Q-value of an action, and the input could be a vector of features derived from the state. Figure 2.6 shows a visual representation of the structure of a neuron.

If we choose  $m = 1$  and  $\Theta = \mathbb{R}^n \times \mathbb{R}$ , then a neuron satisfies the general form  $f : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}^m$  of a neural network. To make it clear that the parameters are split into weights and biases, they are often separated from the input  $x$  with a semicolon instead of just a comma.

The motivation behind the definition of neurons is that the weights  $\omega_i$  “learn” how important each input  $x_i$  is to the neuron. The activation function  $\sigma$  repre-





**Figure 2.6.:** The inside of a single neuron.

sents the neuron’s “firing”. Last, the bias  $b$  ensures that a neuron’s input can take values throughout  $\mathbb{R}$ , even if it is constrained a priori.

The parameters  $(\omega, b)$  are adjusted during training. Therefore, the activation function is the crucial factor in the choice of a neuron. Pinkus [1999] showed that using non-polynomial activation functions is sufficient for neural networks to be able to approximate arbitrary continuous functions in theory. However, in practice, different activation functions are used depending on the application.

One of the most common activation functions, which the DQN also uses, is the **rectified linear Unit (ReLU)** function:

$$\sigma(x) = \max(0, x).$$

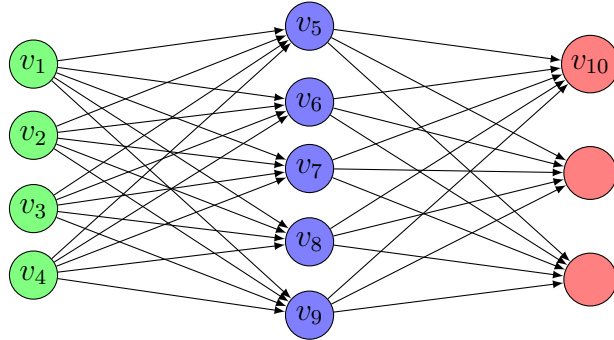
**The Architectural Graph.** To combine neurons into a neural network, we need the concept of a directed graph (cf. [Haykin, 2009] page 15). A **directed graph** consists of a finite set  $V$ , called **vertices**, and a set of **edges**  $E \subset V \times V$ . We say an edge  $(v_1, v_2) \in E$  is **directed** from  $v_1$  to  $v_2$ .

With this concept, we can precisely describe the connections of different neurons.

Let  $V$  be a set of neurons, then  $(v_1, v_2) \in V \times V$  describes that the output of neuron  $v_1$  is used as part of the input of neuron  $v_2$ . Thus, the output of  $v_2$  is given by  $v_2(\dots v_1(x; \theta_1) \dots; \theta_2)$ .

Let  $(V, E)$  be a directed graph whose nodes  $V$  consist only of neurons. We call the composition of all neurons from  $V$  according to the links given by  $E$  in the aforementioned way a **(artificial) neural network**. We will refer to the graph  $(V, E)$  in this thesis as the **architectural graph** of the neural network since it defines the network’s architecture. For simplicity, there is often no distinction between a neural network and its architectural graph.

Figure 2.7 shows an example of an architectural graph for a neural network.



**Figure 2.7.:** Visualization of a neural network by an architectural graph.

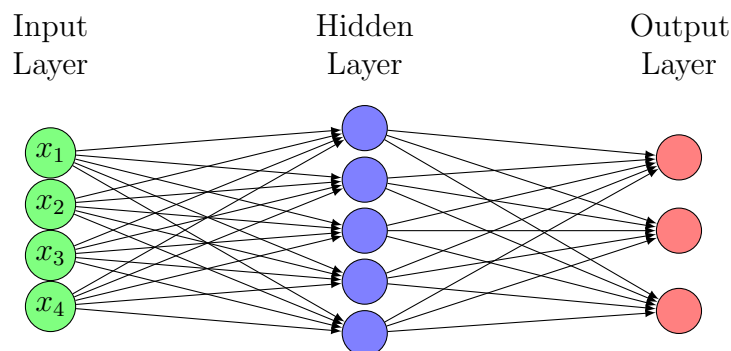
In this example, the output of  $v_{10}$  is given by

$$v_{10} \left( (v_5((v_1(x; \theta_1), \dots, v_4(x; \theta_4)); \theta_5), \dots, v_9(\dots; \theta_9)); \theta_{10} \right).$$

**Layers.** Since neural networks often consist of millions of neurons, it is difficult to convey the structure of a network only by neurons. For this reason, when describing neural networks, one often uses network sub-structures that occur in several networks and are represented by functions that can be understood more quickly.

Motivated by representations such as Figure 2.7, in which neurons are arranged in rows or layers, such a sub-network is called a **layer** (cf. [Goodfellow et al., 2016] page 168f, [Goldberg, 2017] page 41f). Thus, when we refer to a neural network as a layer, we want to emphasize that the network is part of a larger neural network. In this sense, a neural network can be thought of as a composition of layers that are connected according to a directed graph.

In the following, we adhere to [Goodfellow et al., 2016] Page 168f. In a network composed of several layers, it is common to call the last layer the **output layer** and to consider the input as its own layer, the **input layer**. All other layers are called **hidden layers**. An example of how an architectural graph can be organized into layers is shown in Figure 2.8. The intuition behind such architectures is that each layer progressively learns features that are increasingly specific to the given task. Consider the DQN architecture used for learning ALE games such as Pacman, as depicted in Figure 2.5, which includes five layers. Here, the motivation is that the initial layers focus on identifying basic elements within the frames of Pacman, such as edges, while the subsequent layers capture more complex features, like Pacman’s position. This structured approach to learning, where basic features are discerned first and more intricate ones later, led to the idea that the hidden layers’ outputs effectively create a **latent space** within the network, representing embedded states.



**Figure 2.8.:** An example of the arrangement of the architectural graph in layers.

The neurons within a given layer are often called the layer’s **units**. To compare neural networks, the number of units of a layer, i.e., the dimension of its output, is called the **width** of the layer. The number of layers in a network is called the **depth** of the network. This nomenclature inspired the notion of *deep neural network* for neural networks with more than one layer. This is also the origin of the term deep reinforcement learning for RL algorithms that utilize neural networks.

For a generally valid theory of neural networks, the definition of neural networks as a composition of layers is problematic. It is not uniformly definable which structures count as layers since almost any function can be used as a layer (cf. [Goodfellow et al., 2016] page 376). Thus, notions such as the depth of a network are highly dependent on the convention of the person using them (cf. [Goodfellow et al., 2016] page 7).

In practice, however, this way of thinking about neural networks is very useful. Precisely because a layer does not have to fulfill clearly defined templates, it is possible to test a wide variety of functions. Thus, in some applications, layers are used that contain functions that cannot be represented by the neurons defined in this thesis.

**Fully Connected Layers.** The simplest layers for neural networks are so-called **fully connected** or **dense** layers, where all neurons are executed in parallel. They correspond to a vector  $(f_1, \dots, f_n)$ , where each component is a neuron of the form  $f_i = \sigma(\omega_i \cdot x + b_i)$ . Here, one uses the same activation function  $\sigma$  for all neurons. The weights  $\omega_i$  and biases  $b_i$  of the individual neurons can then be combined into a matrix  $W$  and a vector  $b$ , respectively. Taken together, they

are a function  $f : \mathbb{R}^n \times (\mathbb{R}^{m \times n} \times \mathbb{R}^m) \rightarrow \mathbb{R}^m$  with

$$f(x; W, b) = \sigma(Wx + b),$$

where  $\sigma$  describes the component-wise application of an activation function  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ . As with single neurons, we call the two parameters the **weight** matrix  $W \in \mathbb{R}^{m \times n}$  and the **bias** vector  $b \in \mathbb{R}^m$ .

We have already seen the architectural graph of fully connected layers in Figure 2.8. As an example application, we saw that the DQN for solving ALE games like Pacman 2.5 uses fully connected layers as the last two layers. Their role is to connect the learned high-level features, such as Pacman's position, and map them to the final Q-values.

**Convolutional Layers.** In contrast to fully connected layers, there are also layers where each neuron is connected to only a subset of its input. This allows these neurons to learn to describe patches of the input. By covering the entire input with such patches, the network can thus recognize patterns learned on the patches within the input. This process can be used, for example, to identify Pacman within the in-game screen of an ALE state.

Since this method is inspired by the mathematical convolution operation, such layers are called **convolutional**.

To define convolutional neural layers, we first need to formalize what we mean by a subset of the input. Here, we write the input space as  $\mathbb{R}^{n_1 \times n_2 \times \dots \times n_k}$  since convolutional networks are often applied to input in matrix or tensor form – e.g., on images where each pixel corresponds to an entry of a matrix,

Let  $m < \prod_{j=1}^k n_j$ . A **window** or **patch** is a projection  $p : \mathbb{R}^{n_1 \times n_2 \times \dots \times n_k} \rightarrow \mathbb{R}^m$  onto specific components, that is, there exists an injective  $\pi : \{1, \dots, m\} \hookrightarrow \{1, \dots, n_1\} \times \{1, \dots, n_2\} \times \dots \times \{1, \dots, n_k\}$ , so that

$$p(x) = (x_{\pi(1)}, \dots, x_{\pi(m)}).$$

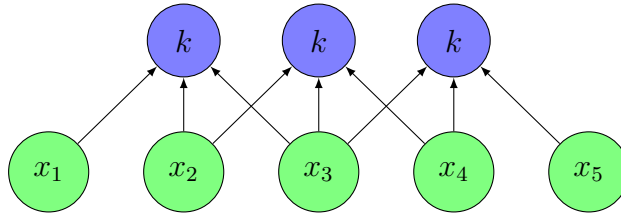
We will look at concrete examples of patches later in this section.

Another feature of convolutional networks is that multiple neurons share parameters. The same neuron is applied to all selected patches.

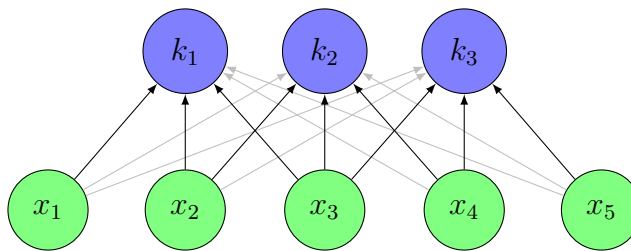
A **Convolutional Layer** consists of a finite set  $P = \{p_1, \dots, p_s\}$  of patches  $p_i : \mathbb{R}^{n_1 \times n_2 \times \dots \times n_k} \rightarrow \mathbb{R}^m$  and a set  $\{k_1, \dots, k_r\}$  of neurons  $k_i : \mathbb{R}^m \rightarrow \mathbb{R}$ , which all use the same activation function and are called **filters** or **kernels**. The network as a function  $\mathbb{R}^{n_1 \times n_2 \times \dots \times n_k} \rightarrow \mathbb{R}^s \times \mathbb{R}^r$  is then given by

$$f(x, \theta)_{ij} = k_j(p_i(x), \theta_j).$$

Figure 2.9 shows the architectural graph of a simplified **Convolutional Neural Network (CNN)** with 3 patches and only a single filter  $k$ .



**Figure 2.9.:** The architectural graph of a highly simplified convolutional network with only a single filter  $k$ . The filter is represented by three different neurons, all sharing the same parameters.



**Figure 2.10.:** The architectural graph of the convolutional network in Figure 2.9 interpreted as a fully connected layer. The kernel  $k$  is split into three individual neurons  $k_i$  where the black arrows represent the learned weights of the kernel  $k$  while the gray arrows represent zero weights.

We can see that it looks similar to the architectural graph of a fully connected layer, but not all neurons are connected, and some neurons share their weights. If we look solely at fully trained networks (i.e., without any further parameter adjustments), we can write convolutional layers as fully connected layers in the following way: Each time a kernel is applied to the input, we consider it to be an individual neuron, where the weights connecting the kernel and the corresponding patch in the input are the learned weights of the kernel, and all other weights are zero. The bias and activation function are the same as the bias and activation function of the kernel. Figure 2.10 illustrates how the convolutional network shown in Figure 2.9 can be viewed as a fully connected layer. This may not be useful in practice, but it will help to unify mathematical statements about neural networks later in this thesis.

The choice of patches for a convolutional layer depends strongly on the type of input data. In natural language processing, for example, sentences are often represented as vectors in  $\mathbb{R}^n$  where each entry corresponds to a word. In this case, the patches within the sentence are usually chosen to be  $m$  consecutive words. A more detailed description can be found in [Goldberg, 2017] chapter 13.1.

In this work, we focus on visual input since the states generated by the ALE consist of visual frames representing the game screen. As mentioned above, such visual inputs are often represented as matrices, where each entry corresponds to a pixel with several channels. More specifically, the inputs are elements of  $\mathbb{R}^{n \times m \times c}$  where  $n$  and  $m$  are the height and width of the input, respectively, and  $c$  is the number of channels (e.g., four channels for the temporal frames of the ALE). For this type of data, the most common convolutional architecture is called **2D convolution**. For a 2D convolution, one chooses the patches as  $k_1 \times k_2 \times c$  subtensors where  $k_1$  and  $k_2$  are called the height and width of the convolution filters, respectively. These patches are moved across the input with a **stride**  $(s_1, s_2)$ . That is, the patch that produces the output in the  $i$ -th row and the  $j$ -th column is a sub-tensor of the form

$$p_{i,j} = \begin{array}{|c|c|c|c|} \hline & x_{is_1, is_2, c} & \cdots & x_{is_1, is_2+k_2, c} \\ \hline x_{is_1, js_2, 1} & \cdots & x_{is_1, js_2+k_2, 1} & \\ \hline \vdots & \ddots & \vdots & \\ \hline x_{is_1+k_1, js_2, 1} & \cdots & x_{is_1+k_1, js_2+k_2, 1} & \\ \hline \end{array} \begin{array}{l} \\ \\ \\ +k_2, c \end{array}$$

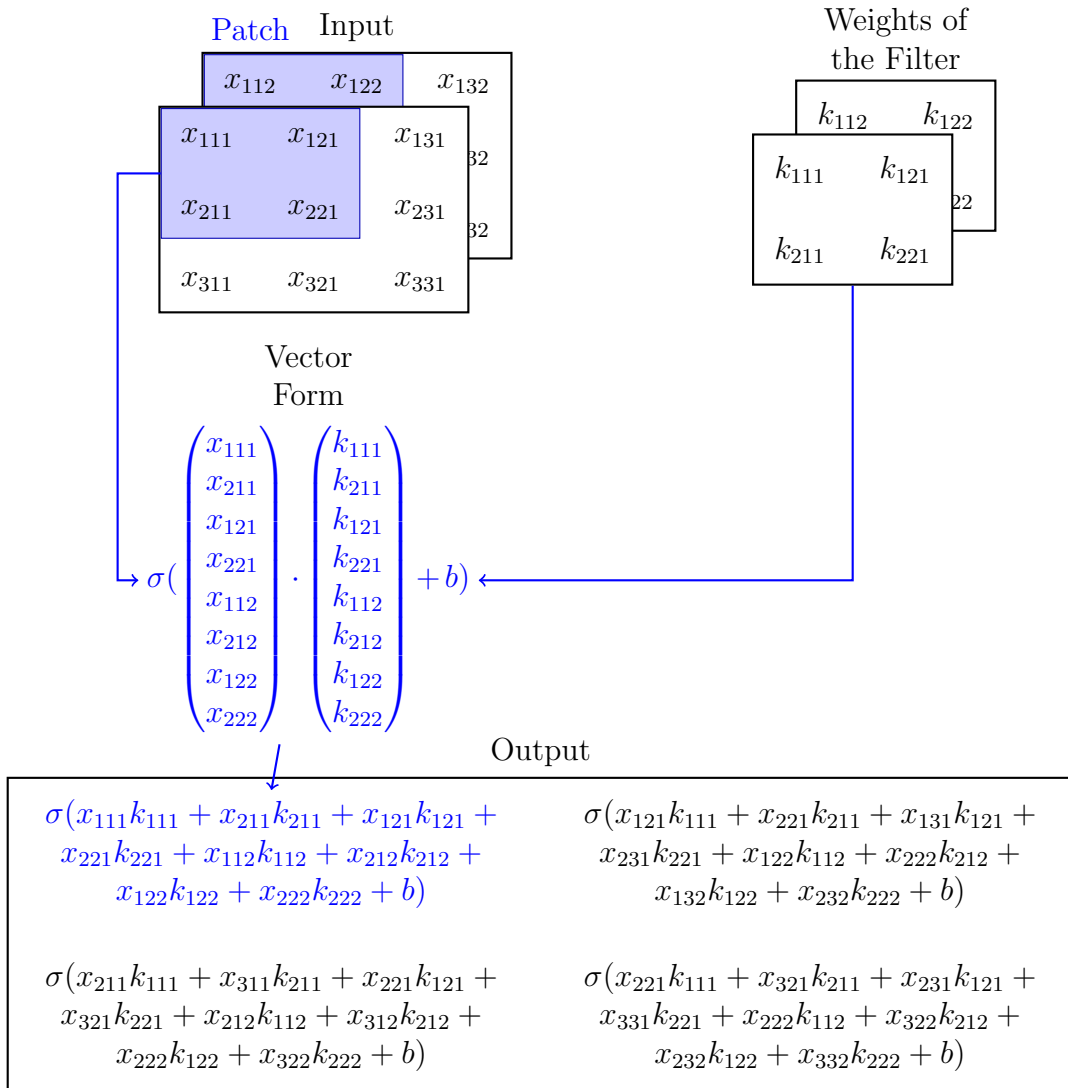
for  $i, j \in \mathbb{N}_0$  with  $is_1 + k_1 \leq n$  and  $js_2 + k_2 \leq m$ . To use this tensor as input to filter neurons, it is converted to vector form by writing its columns one below the other. This vector form quickly becomes unwieldy. Therefore, the weights of the filters of a 2D convolution are often also written in tensor form. To better understand this abstract procedure, Figure 2.11 contains an example calculation of a 2D convolution with only one filter.

If there is more than one filter, the output of a convolutional layer contains a separate channel for each filter.

The final concept used during the forward pass of the DQN is padding. Here, the input is padded with zeros so that the filters are able to cover the edges of the input states even if they would not normally fit. The DQN uses a padding that results in an output that has the same height and width as the input as long as the stride is  $(1, 1)$ . This type of padding is called **half-padding** or **same-padding**. To see how this padding works, we will only look at the padding in the height direction. The width is done analogously. Assume a 2D-convolution with input height  $n$ , kernel height  $k_1$  and stride  $(s_1, s_2)$ , then the number of zeros  $p$  to be padded according to same-padding is

$$p = \begin{cases} \max(k_1 - s_1, 0) & \text{if } n \bmod s_1 = 0 \\ \max(k_1 - (n \bmod s_1), 0) & \text{else} \end{cases}$$

Half of this padding  $p/2$ , rounded down, is appended at the beginning of the input (above the input in the case of the height dimension) and the other half at the end (below the input).



**Figure 2.11.:** Example for a 2D convolution with a single filter  $k$  with bias  $b$  and activation function  $\sigma$ , and stride  $(1, 1)$ .

## 2.3. Training the DQN

In this section, we will look at how we can use the ideas from Q-Learning in Section 2.1.3 to train the neural network behind the DQN introduced in Section 2.2.1.

### 2.3.1. The DQN Loss Function

To train the neural network behind the DQN, we have to define a **loss function**  $L : \Theta \rightarrow \mathbb{R}$  which shows us how much the current Q-values  $Q(s, a; \theta)$  diverge from the optimal Q-values  $Q^*(s, a)$ . For this, we want to use the **mean squared error** between our current Q-values and the optimal Q-values:

$$L(\theta) = \mathbb{E}_{s \in \mathcal{S}, a \in \mathcal{A}} \left( Q^*(s, a) - Q(s, a; \theta) \right)^2$$

However, since we work in a reinforcement learning setting (Section 2.1), we do not know the optimal policy  $\pi^*$  or optimal Q-value function  $Q^*$  for a given state-action pair  $s \in \mathcal{S}, a \in \mathcal{A}$ .

Thus, the DQN utilizes the Q-learning idea of iteratively approaching the optimal Q-value function  $Q^*$  through the Bellman optimality equation (see Equation 2.1). It uses intermediate targets

$$Y_i^{DQN}(s, a, s') = r(s, a, s') + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta_i^-), \quad (2.2)$$

where the so-called **target parameters**  $\theta_i^-$  are parameters that have been learned in previous iterations. The target parameters are not trained but simply get replaced by the current learned parameter  $\theta$  every  $\tau$  learning steps. This procedure gives rise to a sequence of loss functions

$$\begin{aligned} L_i(\theta) &= \mathbb{E}_{s, a} \left( \left( \sum_{s' \in \mathcal{S}} P(s'|s, a) Y_i^{DQN}(s, a, s') \right) - Q(s, a; \theta) \right)^2 \\ &= \sum_{s \in \mathcal{S}, a \in \mathcal{A}} P_\pi(s, a) \left( \left( \sum_{s' \in \mathcal{S}} P(s'|s, a) Y_i^{DQN}(s, a, s') \right) - Q(s, a; \theta) \right)^2, \end{aligned} \quad (2.3)$$

where  $P_\pi(s, a)$  describes the probability of encountering the state  $s \in \mathcal{S}$  and action  $a \in \mathcal{A}$  when following the strategy  $\pi$  given by the current Q-values.

At a high level, the DQN trains the network parameters  $\theta$  by minimizing the sequence of loss functions  $L_i(\theta)$  via gradient descent.



**Gradient Descent.** Gradient descent is a method for finding local minima of real-valued functions. It was first proposed by Cauchy [1847].

This method uses the fact that one can think of a partially differentiable function  $L : \Theta \rightarrow \mathbb{R}$  as a graph over  $\Theta \subset \mathbb{R}^n$ , where  $L(\theta)$  describes the *height* over  $\theta \in \Theta$ . The slope or rate of change of this graph at the point  $\theta \in \Theta$ , in a direction  $v \in \mathbb{R}^n$ , is then given by the directional derivative  $D_v L(\theta) = \nabla L(\theta) \cdot v$ . In other words,  $D_v L(\theta)$  gives us the rate of change for the function  $L$  when we move the parameters  $\theta$  in a given direction  $v$ . To find a local minimum of  $L$  and thus minimize the average difference from the target  $Y_i^{DQN}$ , we look for the direction  $v$  for which  $L$  decreases the most. This is the case if the direction  $v$  corresponds to the negative gradient  $-\nabla L(\theta)$ . Therefore, we update our parameters by slightly moving them in the direction  $-\nabla L(\theta)$ . To this end, we choose a so-called **learning rate**  $\alpha > 0$  and calculate the updated parameters  $\tilde{\theta}$  as follows:

$$\tilde{\theta} = \theta - \alpha \nabla L(\theta).$$

This ensures that the new  $L(\tilde{\theta})$  is smaller than  $L(\theta)$  in each iteration, as long as we do not cross a local minima.

### 2.3.2. Experience Replay

To train the DQN via gradient descent using the loss functions defined in Equation 2.3, we would need a list of all possible states  $s \in \mathcal{S}$  and actions  $a \in \mathcal{A}$ . Furthermore, we would need to know the probability  $P_\pi(s, a)$  of encountering each state-action pair under the current strategy and the transition probabilities  $P(s'|s, a)$ . None of this is readily available to the agent in most RL environments.

Therefore, calculating this loss function is not feasible in practice. Instead, Mnih et al. [2015] use **experience replay**. They create a replay memory data set  $D$  by sampling experience tuples  $(s_t, a_t, s_{t+1})$  consisting of a state  $s_t$ , an action  $a_t$ , and the state  $s_{t+1}$  that followed that state-action pair. Initially,  $D$  is filled by randomly selecting actions within the environment until  $D$  reaches a predefined minimum size  $N$ . Then, they let the agent run in the environment with a  $\epsilon$ -greedy strategy with respect to the currently learned Q-value function  $Q(s, a; \theta)$ . A  **$\epsilon$ -greedy** strategy does not always choose actions greedily but also chooses random actions with a chance of  $\epsilon \in [0, 1]$  to facilitate exploration. After each time step, the experience tuple  $(s_t, a_t, s_{t+1})$  is added to the replay memory  $D$ , and a step of **stochastic gradient descent** is performed on  $D$ . In stochastic gradient descent, the loss is not calculated for the entire data set  $D$ , but only for a randomly chosen subset  $B$ , a so-called **batch** or **mini-batch**. Since we know all the states and actions in  $D$ , and we also store the transition

to the next state for all those state-action tuples, we can calculate the DQN loss functions (Equation 2.3) for  $B \subset D$  in the following way

$$L_i(\theta) = \frac{1}{|B|} \sum_{(s_t, a_t, s_{t+1}) \in B} \left( Y_i^{DQN}(s_t, a_t, s_{t+1}) - Q(s_t, a_t; \theta) \right)^2. \quad (2.4)$$

When the size of the replay memory exceeds a predefined maximum size  $M$ , the oldest experience is removed from the memory.

### 2.3.3. Double DQN

This thesis uses an improved training method for the DQN called **double DQN** [Van Hasselt et al., 2016].

When calculating the targets  $Y_i^{DQN}$  of the DQN, both the expected future action  $a_{t+1} = \operatorname{argmax}_{a \in \mathcal{A}} Q(s_{t+1}, a; \theta_i^-)$  as well as the Q-value for this action  $Q(s_{t+1}, a_{t+1}; \theta_i^-)$  are determined by the target parameters  $\theta_i^-$ . If the learned parameters  $\theta$  diverge too much from the target parameters  $\theta_i^-$ , the agent might choose very different actions than expected by the target  $Y_i^{DQN}$ . To avoid this problem, double DQN selects the expected future action based on the currently learned parameters  $\theta$  and only calculates the Q-value based on the target parameters  $\theta_i^-$ . That is, double DQN replaces the targets  $Y_i^{DQN}$  with

$$Y_i^{DoubleDQN}(s, a, s') = r(s, a, s') + \gamma Q(s', \operatorname{argmax}_{a' \in \mathcal{A}} Q(s', a'; \theta); \theta_i^-).$$

Apart from replacing  $Y_i^{DQN}$  with  $Y_i^{DoubleDQN}$ , the training process remains unchanged.

### 2.3.4. Alternative to the DQN: Actor-Critic Algorithms

In addition to the DQN, some sections of this dissertation will also use actor-critic DRL algorithms.

The most prominent **actor-critic** DRL algorithm, A3C (Asynchronous Advantage Actor-Critic), was proposed by Mnih et al. [2016]. This method eliminates the need for experience replay and target parameters by asynchronously running multiple agents in parallel. Each of these agents maintains its own parameters, updating the global parameters every  $k$  steps and subsequently synchronizing their parameters with the global parameters. To simplify the notation and unify it with the previous sections, we will assume  $k = 1$  in this section.

The A3C algorithm learns two outputs in tandem. Similar to the DQN, it trains a value function  $V(s; \theta_V)$  with parameters  $\theta_V$ . As was the case with

$Q^*(s, a)$  for the DQN, A3C approximates the optimal value function  $V^*(s)$  by utilizing the bellman equation. The algorithm minimizes the following loss for individual samples  $s_t, a_t, s_{t+1}$ :

$$L_V(\theta_V) = (r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}; \theta_V) - V(s_t; \theta_V))^2$$

This loss can be viewed as a modified version of Equation 2.4, where  $V(S)$  replaces  $Q(s, a)$  and the target parameters are omitted.

The biggest difference to the DQN is that actor-critic methods directly train the policy  $\pi(a, s; \theta)$  with trainable parameters  $\theta$ . Here,  $\pi(a, s; \theta)$  describes a stochastic policy that returns the probability of choosing action  $a$  in state  $s$ . This probability should be high if the agent expects a promising reward in the future, both in the short and long term. To this end, A3C uses the following loss function for individual samples  $s_t, a_t, s_{t+1}$ :

$$L(\theta) = -[r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}; \theta_V) - V(s_t; \theta_V)] \log \pi(a_t, s_t; \theta)$$

Here, the negative sign at the beginning ensures that the rest of the formula is maximized. The term  $[r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}; \theta_V) - V(s_t; \theta_V)]$  describes how good the chosen action was. It combines the immediate reward  $r(s_t, a_t, s_{t+1})$  with the advantage of transitioning from state  $s_t$  to state  $s_{t+1}$  – as measured by the difference in the state value function  $V(s, \theta_V)$ . For example, if Pacman moved closer to a pellet, the action was valuable even if it did not result in an immediate reward. The only part of  $L(\theta)$  that depends on  $\theta$  is  $\pi(a_t, s_t; \theta)$ . Therefore, the loss pushes  $\pi(a_t, s_t; \theta)$  to be high when  $a_t$  was beneficial and low when  $a_t$  was less effective or detrimental.

To train  $V(s; \theta_V)$  and  $\pi(a, s; \theta)$ , Mnih et al. [2016] use the same network architecture as the DQN (Section 2.2.1). However, they use distinct output layers for  $V(s; \theta_V)$  and  $\pi(a, s; \theta)$ . Therefore, most parameters in  $\theta$  and  $\theta_V$  are shared, except for the weights of the respective output layers.

While A3C by Mnih et al. [2016] asynchronously runs several agents in parallel, Wang et al. [2016a] proposed an actor-critic variant (Actor Critic with Experience Replay, ACER) that works on a single agent by utilizing experience replay similar to the DQN (Section 2.3.2).

## 2.4. Conclusion

This chapter explored the basic notation and foundational principles of DRL, which we will use throughout this thesis. As a specific example of a DRL algorithm, it introduced the DQN and its application to the game of Pacman within the ALE. This detailed examination lays the groundwork for the use of DQN agents as the primary objects of investigation throughout this dissertation. Understanding how these DQN agents perceive their environment, make

decisions, and learn over time is crucial for the subsequent exploration of XAI techniques intended to make their operations transparent and comprehensible. As we move forward, the insights from this chapter will guide our efforts to develop and apply XAI methods to DRL with visual input.

## 3. Explainable AI

Besides RL, this thesis focuses on the field of Explainable AI (XAI), which is dedicated to enhancing the explainability of artificial intelligence systems. XAI seeks to shed light on how AI models arrive at their decisions, making them more comprehensible to humans.

Many works on XAI concentrate on explaining classifiers that make decisions in isolation without affecting other decisions of the classifier. Thus, they do not fully address the problem of explaining behavior in sequential decision-making settings, where an agent’s actions, rewards, and effect on the state of the world are interconnected. Nonetheless, these established XAI methods can still be adapted for use with reinforcement learning (RL) agents, as we will see in Part II. Additionally, numerous methods for explaining RL agents have been developed drawing from XAI techniques for classification models.

This chapter will introduce key general XAI ideas that are relevant to Explainable RL and XAI methods for classifier models that have served as the basis for explainable RL methods.

### 3.1. Terminology

In recent years, there has been a plethora of work on defining XAI terminology and concepts [Lipton, 2018; Tomsett et al., 2018; Adadi and Berrada, 2018; Miller, 2019; Arrieta et al., 2020; Molnar, 2022]. To prevent potential confusion or misinterpretation, this section defines how the different terminologies are used in this thesis.

The first concept we need to define is **explainability** since it is crucial to XAI. Following Miller [2019] and Molnar [2022], this dissertation uses the terms interpretability and explainability as synonyms that refer to the “degree to which an observer can understand the cause of a decision”.

The second core concept we need to define is an **explanation**. Adopting the definition by Tomsett et al. [2018], this thesis uses the term **explanation** to describe “the information provided by a system to outline the cause and reason for a decision or output for a performed task”.

Now, we can look at what an XAI system entails. As outlined by Gunning and Aha [2019], an **XAI system** consists of an explainable model and an explanation interface. The **explainable model** is responsible for providing decisions

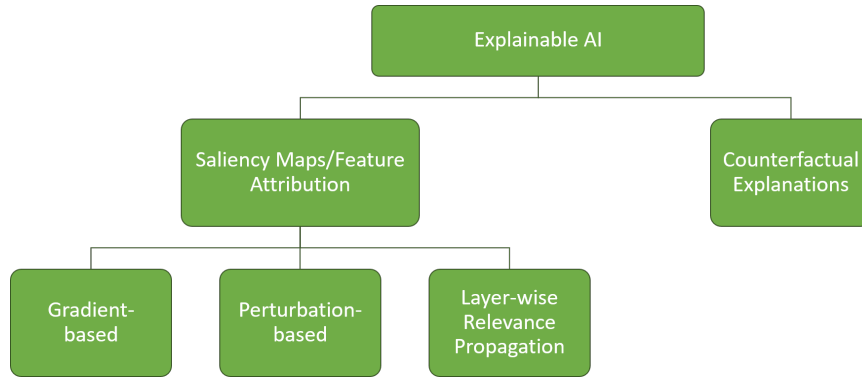
along with insights into the model’s reasoning process for those decisions. The **explanation interface** communicates this information to the user.

This dissertation focuses on the exploration of explainable models. In general, there are two approaches to creating explainable models [Molnar, 2022]. The first approach is to build the model in such a way that it is **intrinsically explainable**, for example, by using a simple model architecture. In the same vein, some methods only make certain parts of the model intrinsically explainable. We refer to those as **intrinsic explanation methods**. In contrast to the intrinsic approach to explainability, the second approach is the use of **post-hoc explanation methods** that are employed after the model has been trained.

Besides this distinction, Molnar [2022] and Adadi and Berrada [2018] introduce two additional categorizations for explainable models:

**Scope.** We can divide explanations into **local explanations**, which provide information regarding the AI’s decision-making in a particular instance, and **global explanations**, which attempt to convey the overall behavior of the model.

**Model Dependency.** Finally, we can divide XAI methods into **model agnostic** methods that can be applied to arbitrary black-box AI models and **model specific** methods that require specific model architectures and need access to the inner-workings of the model.



**Figure 3.1.:** An overview of the XAI methods for classifiers that we will explore in this thesis.

## 3.2. Explanation Methods for Classifiers

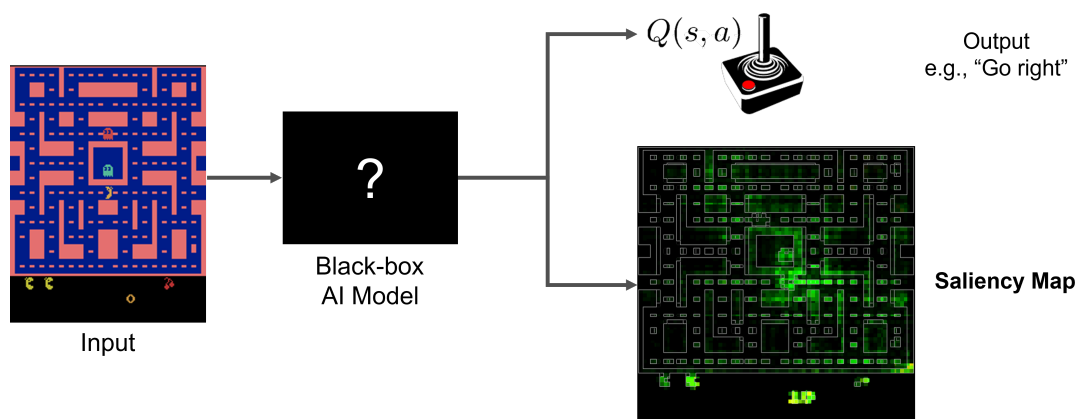
There is a plethora of different methods to explain classification models in the literature. This thesis focuses on two common types of explanation methods for classifiers that also have been used for explaining DRL agents, as we will see in Part II: Feature attribution (or saliency maps) and counterfactual explanations. Both of these two methods are primarily used as local explanation methods that explain a fully trained agent post-hoc. Figure 3.1 shows an overview of the methods we will look at in this section.

### 3.2.1. Saliency Maps or Feature Attribution

One of the most common XAI methods is called **feature attribution** or feature relevance [Arrieta et al., 2020]. It refers to the process of determining the relevance or importance of individual input features for the decision of an AI model. Feature Attribution is mostly used as a local post-hoc explanation method. Here, we assume a fully trained agent  $\pi$  and a given input state  $s$ . A feature attribution method now assigns a relevance score  $R(s_i, \pi)$  to each feature  $s_i$  of the state  $s$ . The value  $R(s_i, \pi)$  measures how relevant or important the feature  $s_i$  was for the decision of the agent  $\pi$  in the state  $s$ . If it is clear which agent is meant, we will omit  $\pi$  and only write  $R(s_i)$ .

For visual input, the relevance information is often displayed as **saliency maps**, which are heatmaps that visualize the relevance of each pixel for a particular decision of the agent. Since the Atari agents used in this thesis rely on visual input, we will use the term saliency map method even if the very same algorithm can be applied to non-visual input data. Figure 3.2 illustrates the idea behind saliency maps.

In the remainder of this section, we will look at different saliency map gener-

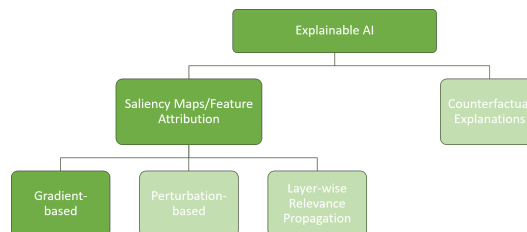


**Figure 3.2.:** The idea behind saliency maps or feature attribution: instead of just returning a prediction, the AI model also provides a saliency map that visualizes how important the input features were for the prediction. In this example, the green heatmap shows how relevant each pixel was for the AI – the brighter a pixel is, the more relevant it was.

ation methods proposed for image classifiers.

### 3.2.1.1. Gradient-based Saliency Maps

Gradient-based saliency map generation methods [Simonyan et al., 2013; Springenberg et al., 2014; Sundararajan et al., 2017; Selvaraju et al., 2020] utilize the derivative with respect to the input to estimate how much a small change in this input’s value would change the prediction. This section describes different gradient-based methods and is based on text from our paper:



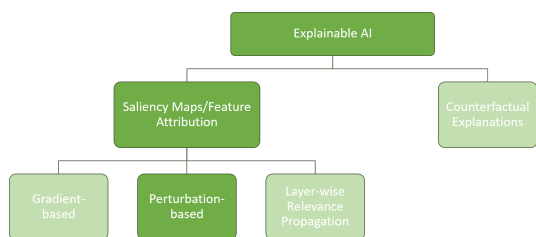
**Tobias Huber**, Dominik Schiller, and Elisabeth André [2019]. “Enhancing Explainability of Deep Reinforcement Learning Through Selective Layer-Wise Relevance Propagation”. In: *KI 2019: Advances in Artificial Intelligence*. Springer International Publishing, pp. 188–202

One of the first methods used to measure the relevance of pixels of visual input data is to see how much a change in that pixel impacts the prediction of a neural network. If a pixel is relevant for the decision of the model, then even small changes in the pixel will greatly impact the output of the model. This local rate of change with respect to certain inputs of the network can be



calculated by using partial derivatives. Simonyan et al. [2013], for example, use the derivative of an image classifier with respect to an input pixel to determine the relevance of that input pixel. To get this derivative, they use the backpropagation algorithm, which is also used during the training of the neural network. The deconvolution [Zeiler and Fergus, 2014] and guided backpropagation [Springenberg et al., 2014] approaches are based on the same theory but use modified versions of the backpropagation algorithm to get relevance values for the input pixels of image classifiers. Another similar approach is Grad-CAM [Selvaraju et al., 2020]. This approach was introduced for CNN-based image classifiers and uses partial derivatives of the fully connected part of a CNN with respect to the output of the last convolutional layer to identify regions inside the input, which were relevant for the specific prediction of the CNN. Guided backpropagation and Grad-CAM can be combined by computing the component-wise product of the attention maps created by the different approaches. The result is called guided Grad-CAM and creates a fine-granular but class-specific saliency map [Selvaraju et al., 2020].

### 3.2.1.2. Perturbation-based Saliency Map Approaches



Perturbation-based methods ([Zeiler and Fergus, 2014; Ribeiro et al., 2016; Petsiuk et al., 2018; Greydanus et al., 2018; Puri et al., 2020]) perturb areas inside the input and measure how much this changes the model’s prediction. The idea behind this is to introduce uncertainty to the perturbed area

and to see how much the model is influenced by the loss of information in that area. Perturbation-based methods often come with the advantage of being independent of the model’s structure but with the drawback of not being as precise as some model-specific methods. This section describes different perturbation-based saliency map generation methods and is based on text from our paper:

**Tobias Huber**, Benedikt Limmer, and Elisabeth André [2022]. “Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents”. In: *Frontiers in Artificial Intelligence* 5. ISSN: 2624-8212. DOI: 10.3389/fraai.2022.903875

The basic saliency map generation process is the same for all perturbation-based approaches. Although three out of the five methods presented in this section were originally proposed for image classifiers, we employ a consistent DRL notation (Section 2.1) to formalize all the methods. This notation clarifies how they can be applied to the DQN agents in this thesis. Let  $\pi$  be the agent

that takes a visual input state  $s$  and maps it to a Q-value  $Q(s, a)$  for each possible action. To ease notation, we use  $Q(s)$  to describe the Q-value of the action that should be analyzed. Most often, this is the action with the highest Q-value for the unperturbed input  $s$  since this is the action that a fully trained agent would choose for  $s$ . A visual input state  $s$  with height  $H$  and width  $W$  can be defined as a mapping  $s : \Lambda_s \rightarrow \mathbb{R}^c$  of each pixel  $\lambda \in \Lambda_s = \{1, \dots, H\} \times \{1, \dots, W\}$  to  $c$  channels (e.g.  $c = 4$  for the Atari environment which uses the channels to store the last 4 frames). To determine the relevance  $R(\lambda)$  of each pixel  $\lambda$  for the prediction of the agent, perturbation-based approaches feed perturbed versions  $s'$  of  $s$  to the agent and then compare the resulting output values with the original results.

Based on this general approach, different perturbation-based approaches are defined by two methods: First, a perturbation method that describes how the perturbed input states  $s'$  are created. Second, a relevance method  $R(\lambda)$  that calculates the relevance of each pixel based on the perturbed outputs  $Q(s', a)$  and the original outputs  $Q(s, a)$ .

We will now take a look at popular perturbation-based saliency map approaches and how they implement these two methods.

**Occlusion Sensitivity.** This approach is the most basic perturbation-based saliency map approach and was first proposed by Zeiler and Fergus [2014] for image classifiers. It creates perturbed states  $s'$  by shifting a  $n \times n$  patch across the original state  $s$  and occluding this patch by setting all the pixels within to a certain color (e.g., black or grey). An example is shown in Figure 3.3.

The relevance  $R(\lambda)$  of each pixel  $\lambda$  inside the patch is then computed based on the agent’s confidence after the perturbation

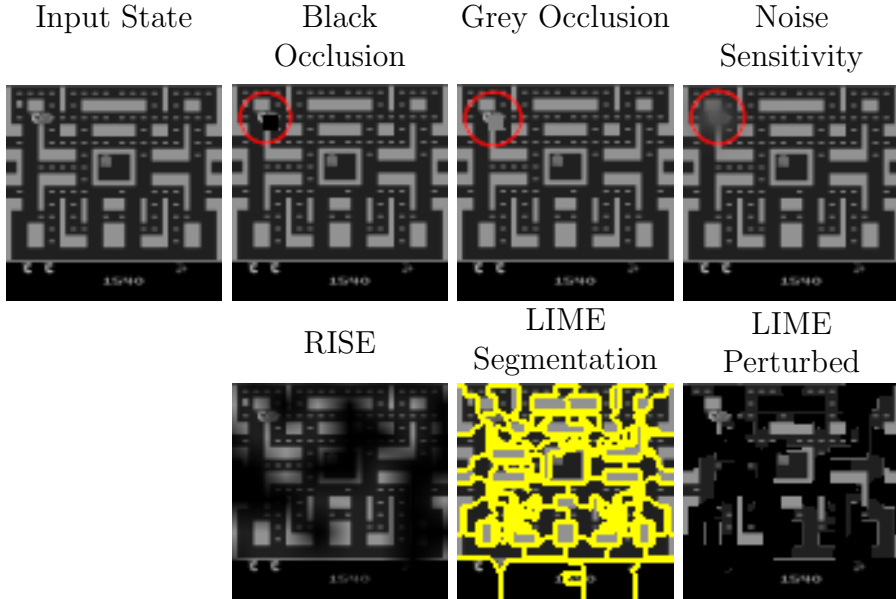
$$R(\lambda) = 1 - Q(s'). \quad (3.1)$$

Since the original source does not go into details about the algorithm, we use the *tf-explain* implementation as reference<sup>1</sup>. As long as the saliency maps are normalized, this is equivalent to  $Q(s) - Q(s')$  since all values in the saliency map are shifted by the same constant  $Q(s) - 1$ .

**Noise Sensitivity.** Greydanus et al. [2018] proposed noise sensitivity specifically for DRL agents with visual input. Instead of completely occluding patches of the state, this approach adds noise to the state  $s$  by applying a Gaussian blur to a circle with radius *rad* around a pixel  $\lambda$  (see Figure 3.3). The modified state  $s'(\lambda)$  is then used to compute the relevance of the covered circle by comparing the agent’s logit units  $\pi(s)$ . For the DQN agents in this thesis,  $\pi(s)$  is the vector

---

<sup>1</sup>Available under: <https://github.com/sicara/tf-explain>



**Figure 3.3.:** An example of the different types of perturbation used by perturbation-based saliency map approaches. The parameters are chosen in such a way that the idea of the perturbation can be easily identified. For Occlusion and Noise, the disturbed area is marked with a red circle.

of all Q-values  $Q(s, a)$  for each possible action:

$$R(\lambda) = \frac{1}{2} \|\pi(s) - \pi(s'(\lambda))\|^2 \quad (3.2)$$

This is done for every  $radth$  pixel, resulting in a temporary saliency map smaller than the input. For the final saliency map, the result is up-sampled using bilinear interpolation.

**Randomized Input Sampling for Explanation (RISE).** RISE was developed by Petsiuk et al. [2018] for image classifiers. This approach uses a set of  $N$  randomly generated masks  $\{M_1, \dots, M_N\}$  for perturbation. To this end, temporary  $n \times n$  masks are created by setting each element to 1 with a probability  $p$  and 0 otherwise. These temporary masks are upsampled to the size of the input state using bilinear interpolation. The states are perturbed by element-wise multiplication with those masks  $s \odot M_i$  (see Figure 3.3). The relevance of each pixel  $\lambda$  is given by

$$R(\lambda) = \frac{1}{p \cdot N} \sum_{i=1}^N Q(s \odot M_i) \cdot M_i(\lambda), \quad (3.3)$$

where  $M_i(\lambda)$  denotes the value of the pixel  $\lambda$  in the  $i$ th mask.

**Local Interpretable Model-agnostic Explanations (LIME).** Ribeiro et al. [2016] proposed LIME and tested it on image classifiers. LIME uses image segmentation algorithms, like *SLIC*, *Quickshift*, and *Felzenszwalb*, to divide the input state into superpixels (i.e., groups of pixels that share similar visual properties such as color). Then, a dataset of  $N$  perturbed samples in the neighborhood of the input state is created. For each of those samples  $s'$ , a different combination of superpixels is “deleted” by setting all pixels within the superpixels to a certain value (in this thesis, we use 0). An example for a segmentation of a state  $s$  and a corresponding perturbed state  $s'$  can be seen in Figure 3.3. Using this dataset, an interpretable surrogate model is trained to predict the agent’s decision based on the presence of superpixels. This surrogate model can for example be trained with gradient descent. During training, the samples are weighted based on their proximity to the original input state. Finally, analyzing the weights of the trained surrogate model provides a relevance value  $R(\lambda)$  for each superpixel, which is assigned to all pixels  $\lambda$  within this superpixel.

**Specific and Relevant Feature Attribution (SARFA).** SARFA by Puri et al. [2020] is another algorithm proposed explicitly for DRL agents. This approach does not use a specific perturbation method. Puri et al. test noise perturbation [Greydanus et al., 2018] for Atari games and occlusion [Zeiler and Fergus, 2014] for other domains. Given a perturbed state  $s'$ , SARFA measures the information specific to the action  $a'$ , which should be analyzed, by utilizing a softmax normalization  $P(s, a') := \frac{\exp(Q(s, a'))}{\sum_{a \in \mathcal{A}} \exp(Q(s, a))}$  and calculating

$$\Delta p = P(s, a') - P(s', a'). \quad (3.4)$$

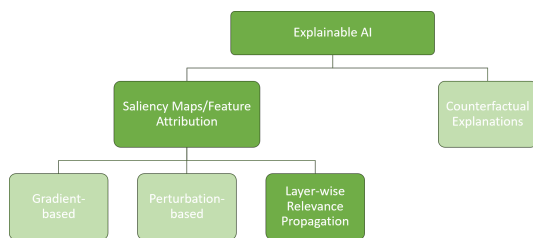
To only measure information that is relevant to  $a'$  and not relevant to other actions  $a \neq a'$  (i.e., the perturbation should not affect actions other than  $a'$ ), they additionally calculate

$$K = \frac{1}{1 + KL(P_{rem}(s, a'), P_{rem}(s', a'))}, \quad (3.5)$$

where  $KL$  is the Kullback–Leibler divergence and the vector  $P_{rem}(s, a') := \left( \frac{\exp(Q(s, a'))}{\sum_{a \neq a'} \exp(Q(s, a))} \right) \forall a \neq a'$  is the softmax over all outputs *except* the chosen action  $a'$ . The final relevance for each pixel that is perturbed in the state  $s'$  is then given by the harmonic mean of  $\Delta p$  and  $K$ :

$$R(\lambda) = \frac{2K \Delta p}{K + \Delta p} \quad (3.6)$$

### 3.2.1.3. Layer-Wise Relevance Propagation



In contrast to the aforementioned methods for generating saliency maps, Bach et al. [2015] proposed to use the intermediate activations of the neurons during the forward pass to estimate the contribution of each input pixel to the prediction of an image classifier. This section describes their approach and is

based on text from our paper:

**Tobias Huber**, Dominik Schiller, and Elisabeth André [2019]. “Enhancing Explainability of Deep Reinforcement Learning Through Selective Layer-Wise Relevance Propagation”. In: *KI 2019: Advances in Artificial Intelligence*. Springer International Publishing, pp. 188–202

Instead of calculating how much a change in an input pixel would impact the prediction, Bach et al. investigate the contribution of the input pixels to prediction. For this purpose, they not only describe a single specific algorithm but introduce a general concept, which they call Layer-wise Relevance Propagation (LRP). This concept has two advantageous properties that the aforementioned saliency map approaches lack. The first is the conservation property, which says that the sum of all relevance values generated by LRP is equal to the value of the prediction. This ascertains that the relevance values reflect the certainty of the prediction and facilitates the comparison of saliency maps for different states. The second property is positivity, which states that all relevance values are non-negative. This ascertains that the generated saliency maps do not contain contradictory evidence [Montavon et al., 2018]. Some gradient-based approaches achieve positivity by squaring the partial derivatives, but this only masks the negativity for the viewer. Finally, LRP is computationally efficient compared to perturbation-based methods and can even be more efficient than gradient-based methods in cases where the inference has to be run without GPU since LRP can reuse the values of the forward pass.

**Basic Algorithm.** As mentioned above, LRP does not describe a specific algorithm but a concept that can be applied to any classifier  $f$  that fulfills the following two requirements. First,  $f$  has to be decomposable into several layers of computation, where each layer can be modeled as a vector of real-valued functions. Secondly, the first layer has to be the input  $x$  of the classifier containing, for example, the input pixels of an image, and the last layer has to be the real-valued prediction of the classifier  $f(x)$ . Any DRL agent fulfills those requirements if we consider the inputs  $x$  to be the input states  $s$  and  $f(x)$  to be

the output value  $Q(s, a')$  that corresponds to the action  $a'$  we want to analyze.

For a given input  $x$ , the goal of any method following the LRP concept is to assign relevance values  $R_j^l$  to each computational unit  $j$  of each layer of computation  $l$ , in such a way that  $R_j^l$  measures the local contribution of the unit  $j$  to the prediction  $f(x)$ . A method of calculating those relevance values  $R_j^l$  is said to follow the LRP concept if it sets the relevance value of the output unit to be the prediction  $f(x)$  and calculates all other relevance values by defining

$$R_j^l := \sum_{k \in \{j \text{ is input for neuron } k\}} R_{j \leftarrow k}^{l+1}, \quad (3.7)$$

for **messages**  $R_{j \leftarrow k}^{l+1}$ , such that

$$R_k^{l+1} = \sum_{j \in \{j \text{ is input for neuron } k\}} R_{j \leftarrow k}^{l+1}. \quad (3.8)$$

In this way, an LRP variant is determined by choosing messages  $R_{j \leftarrow k}^{l+1}$ . Through Definition 3.7, it is then possible to calculate all relevance values  $R_j^l$  in a backward pass, starting from the prediction  $f(x)$  and going towards the input layer. Furthermore, Equation 3.8 gives rise to

$$\begin{aligned} \sum_k R_k^{l+1} &= \sum_k \sum_{j \in \{j \text{ is input for neuron } k\}} R_{j \leftarrow k}^{l+1} \\ &= \sum_j \sum_{k \in \{j \text{ is input for neuron } k\}} R_{j \leftarrow k}^{l+1} = \sum_j R_j^l. \end{aligned}$$

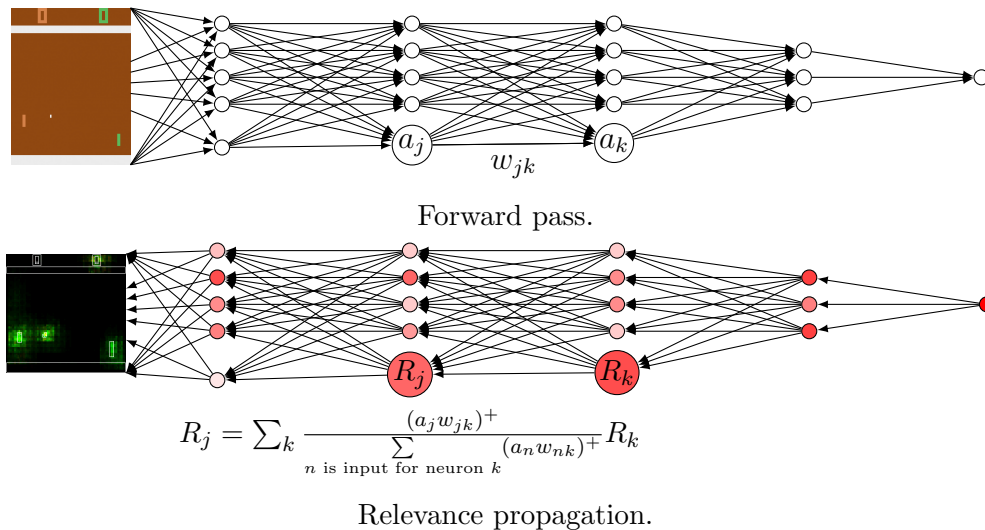
This ensures that the relevance values of each layer  $l$  are a linear decomposition of the prediction

$$f(x) = \dots = \sum_{j=1}^{\dim(l)} R_j^l = \dots = \sum_{j=1}^{\dim(input)} R_j^{input}.$$

Such a linear decomposition is easier to interpret than the original classifier because we can think of positive values  $R_j^l$  to contribute evidence in favor of the decision of the classifier and of negative relevance values to contribute evidence against the decision.

**The  $z^+$ -Rule.** To see how specific LRP variants look in practice, we will look at one of the most basic LRP messages, the  $z^+$ -rule.

Let  $f$  be a CNN with a mixture of convolutional layers  $conv_i$  and fully connected layers  $fc_i$  like the DQN described in Section 2.2.1. Following the LRP



**Figure 3.4.:** Visualization of the LRP  $z^+$ -rule on a 5-layer CNN like the DQN. Each node symbolizes a neuron with activation  $a_j$  during the forward pass, and the red hue illustrates the relevance  $R_j$  of each neuron for the prediction. Notably, the last fully connected layer (the output layer) is depicted solely by the chosen action’s neuron, as the others aren’t involved in LRP. To simplify the visualization, there is no differentiation between fully connected and convolutional layers.

notation, we denote the relevance value of the  $j$ -th neuron in the layer  $l$  with  $R_j^l$ .

As described above, we have to define messages  $R_{j \leftarrow k}^{l, l+1}$  for any two consecutive Layers  $l, l+1$  to determine an LRP variant. To simplify the notation, we assume that both  $l$  and  $l+1$  are fully connected layers  $fc_i$ . The convolutional case works analogously, as we have seen in Section 2.2.2 that a trained convolutional layer can be written as a fully connected layer. For an input  $x$ , we write  $fc_i(x)$  for the output of the layer  $fc_i$  during the forward pass that calculates  $f(x)$ .  $R_{j \leftarrow k}^{l, l+1}$  should measure the contribution of the  $j$ -th neuron of  $fc_{i-1}$  to the  $k$ -th neuron of  $fc_i$ , therefore we have to look at the calculation of  $fc_i(x)_k$ . The fully connected layer  $fc_i$  uses a weight matrix  $W_i$ , a bias vector  $b_i$  and an activation function  $\sigma_i$  as parameters for its output (See Section 2.2.2). Let  $W_i^k$  be the  $k$ -th row of  $W_i$  and  $b_i^k$  the  $k$ -th entry of  $b_i$ . Then the activation of the  $k$ -th neuron in  $fc_i(x)$  is

$$\sigma_i(W_i^k \cdot fc_{i-1}(x) + b_i^k),$$

where  $\cdot$  denotes the dot product and  $fc_0$  is the flattened input.

Usually, the ReLU function  $\sigma(x) = \max(0, x)$  is used as the activation function  $\sigma_i$  in the DQN architecture. Bach et al. [2015] argue that any monotonous

increasing function  $\sigma$  with  $\sigma(0) = 0$ , like the ReLU function, conserves the relevance of the dot product  $W_i^k \cdot fc_{i-1}(x)$ . Newer LRP variants, like the one used by Montavon et al. [2018], also omit the bias when defining  $R_{j \leftarrow k}^{l, l+1}$ . With those two assumptions the relevance of each neuron of  $fc_{i-1}$  to  $fc_i(x)_k$  is the same as their contribution to the dot product

$$W_i^k \cdot fc_{i-1}(x) = \sum_j w_{jk} fc_{i-1}(x)_j.$$

This is a linear decomposition, so we can use  $w_{jk} fc_{i-1}(x)_j$  to measure the contribution of the  $j$ -th neuron of  $fc_{i-1}$ .

Since we want to find the parts of the input that contributed evidence in favor of the decision of the DQN agent, we restrict ourselves to the positive parts of that sum. That is, we set

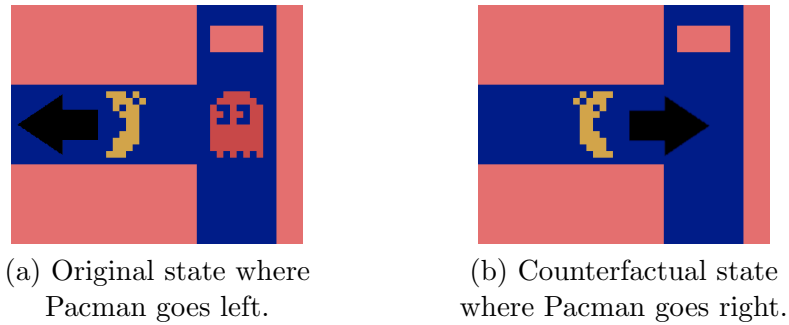
$$z_{jk}^+ := \begin{cases} w_{jk} fc_{i-1}(x)_j & \text{if } w_{jk} fc_{i-1}(x)_j > 0 \\ 0 & \text{if } w_{jk} fc_{i-1}(x)_j \leq 0 \end{cases}.$$

With this, we define the messages as

$$R_{j \leftarrow k}^{l, l+1} := \frac{z_{jk}^+}{\sum_{n \in \{n \text{ is input for neuron } k\}} z_{nk}^+} R_k^{l+1}. \quad (3.9)$$

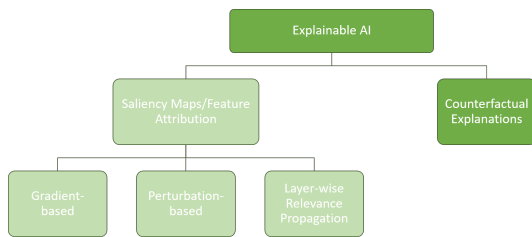
This method is called the  $z^+$ -rule (**without bias**) and satisfies the LRP Equation 3.8. Figure 3.4 shows a visualization of the  $z^+$ -rule on a 5-layer CNN like the DQN.





**Figure 3.5.:** Example for a counterfactual explanation: In the original situation (a), the agent does not take the fastest path to the pill in the top right corner. It is unclear if the agent is afraid of the ghost or does not recognize the shortest path. The counterfactual state (b) shows that the agent would have taken the fastest path to the pill if the ghost was not there. This indicates that the agent is afraid of the ghost.

### 3.2.2. Counterfactual Explanations

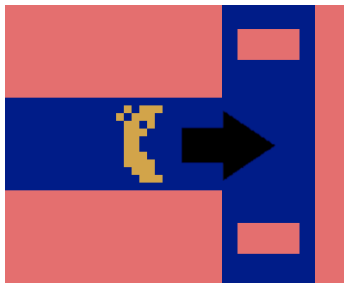


Another prominent local explanation paradigm to make the decisions of black-box AI agents transparent and comprehensible are so-called **counterfactual explanations**. This section describes these kinds of explanations. It extends text from our previous publication:

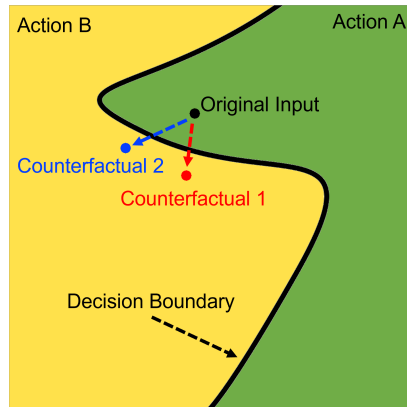
**Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. IFAAMAS, 10 pages

By providing an alternative reality where the AI would have made a different decision, counterfactual explanations follow a rather human way of describing decisions [Miller, 2019; Byrne, 2019]. For example, if a person would have to explain why a warehouse robot took a detour instead of directly moving to its desired target, they would probably give an explanation similar to *If there was no production worker in the way, the robot would have moved straight to its target* - and, by doing so, give a counterfactual explanation of the warehouse robot’s behavior. Figure 3.5 shows a similar situation from the Atari game Pacman (see Section 2.1.2.1 for the rules of the game).

Several desirable properties of counterfactual explanations have been high-



**Figure 3.6.:** A second counterfactual state for the situation in Figure 3.5 containing an additional pill that is not necessary to change the agent’s decision. Through this additional modification of the original image, an observer cannot determine whether the agent was afraid of the ghost or is now convinced by the additional pill.



**Figure 3.7.:** An illustration of the idea of counterfactual explanation. The figure shows two counterfactual instances that slightly cross the decision boundary of an RL agent.

lighted in the literature. One widely supported idea is to minimize the changes required to cross the AI’s decision boundary [Wachter et al., 2017; Mothilal et al., 2020]. Altering as little as possible makes it easier to detect relevant changes (see Figure 3.6). Furthermore, it helps to create plausible counterfactuals that do not deviate too far from realistic scenarios. However, recent work [Keane et al., 2021] has challenged this principle of minimal change. It is often unclear what it means for a change to be minimal. In Pacman, for instance, it is not clear if adding a normal pill is a smaller change than adding a ghost, which contains more pixels than a pill. Addressing this problem, another desirable property for counterfactual explanation approaches is the ability to generate multiple, diverse counterfactuals that enable exploration of the decision boundary [Wachter et al., 2017; Keane et al., 2021]. Figure 3.7 illustrates the idea behind desirable counterfactual explanations.

In the remainder of this section, we will look at two families of methods for generating counterfactual explanations for classifier models.

**Optimization-based Counterfactual Explanations.** The first family of approaches to generating counterfactual explanations defines them as **optimization problems**. These approaches formulate objective functions that describe the desired properties of the counterfactual explanations for a given input and the desired AI decision. This objective function can then be solved case-by-case using optimization methods such as gradient descent (see Section 2.3).

Optimization-based counterfactual methods have been applied to tabular data [Wachter et al., 2017; Mothilal et al., 2020] and images [Looveren and Klaise, 2021]. However, a major limitation of these optimization-based approaches is their dependence on complete optimization processes to generate each individual counterfactual explanation. Running full optimization processes is often not feasible at runtime.

**GAN-based Counterfactual Explanations.** The second family of approaches to generating counterfactual explanations uses **generative models** like Generative Adversarial Networks (GANs). GANs consist of two neural networks that are trained in tandem: a generator and a discriminator. The generator is trained to generate images that look like images from a given target domain. The discriminator aims to distinguish between images generated by the generator and original images from the target domain.

Nemirovsky et al. [2022] and Zhao [2020] use a modified GAN architecture to train a generator that takes an image as input and generates a residual that is added to the input image to change the decision of an image classifier. To this end, they propose novel loss functions that include the classifier to ensure that the counterfactuals are classified as intended. The losses also contain terms that ensure that the residuals are minimal. One drawback of these approaches is that they use the classifier in the loss function and thus require access to the gradient of the classifier. Therefore, they are not fully model-agnostic. However, Nemirovsky et al. [2022] also propose an alternative loss function that is applicable to black-box classifiers without access to the gradient.

Instead of adding a residual to the original image, other approaches train GANs that take the original image as input and directly generate a complete counterfactual image. To do this, Mertes et al. [2022a] train a GAN to transfer between the domains given by the labeled classes from the classifier’s training dataset. Additionally, their loss function includes the classifier, which is to be explained. Including the classifiers ensures that the generated counterfactuals are actually classified as the desired class. Finally, their loss function requires the GAN to recreate the original image to enforce the minimal change constraint for counterfactual explanations. While Mertes et al. [2022a] can only handle binary classifiers, Matsui et al. [2022] propose a similar loss function but use a different GAN architecture that is able to create counterfactuals for more than two classes. Singla et al. [2023] also propose a similar approach but adapt it to account for models with continuous outputs instead of discrete classes.

All of the GAN-based approaches described above have been tested on image classifiers. For images, the slow runtime of the optimization-based counterfactual generation approaches mentioned above is particularly problematic. In contrast, the GAN-based counterfactual explanation approaches fully train their

GAN models before runtime. During runtime, they can generate counterfactuals with a single forward pass of the GANs. However, training the GANs requires a training set that is often not available for RL agents. Furthermore, most of the proposed loss functions require the gradient of the classifier and are, thus, not model-agnostic.

### 3.3. Evaluating XAI

Having explored various methods for making AI systems understandable in Section 3.2, we now shift our focus to evaluating the effectiveness of these methods in this section. In general, evaluation metrics for XAI approaches can be separated into two broad categories: Human user studies and computational measurements [Mohseni et al., 2021b].

Both categories are crucial for making the behavior of deep reinforcement learning agents explainable. On the one hand, it is important that the explanations actually reflect the agents’ reasoning. This is best evaluated with computational metrics. Furthermore, computational metrics provide a cost-effective way to evaluate different explanation methods before investing in more resource-intensive user studies. On the other hand, only user studies can reveal whether the explanations are clear, useful, and truly make these systems easier for the user to understand. Moreover, the long-term adoption of an XAI system by users depends on their satisfaction with the explanatory interface.

The section begins with an exploration of computational metrics in Subsection 3.3.1, followed by an examination of metrics for human user studies in Subsection 3.3.2. The section extends text from our publications:

**Tobias Huber**, Benedikt Limmer, and Elisabeth André [2022]. “Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents”. In: *Frontiers in Artificial Intelligence* 5. ISSN: 2624-8212. DOI: 10.3389/frai.2022.903875

and

**Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

The discussion will be limited to the evaluation of XAI methods for classifiers introduced in Section 3.2. An examination of XAI evaluation specifically targeting RL agents can be found in Chapter 5.

### 3.3.1. Computational Metrics

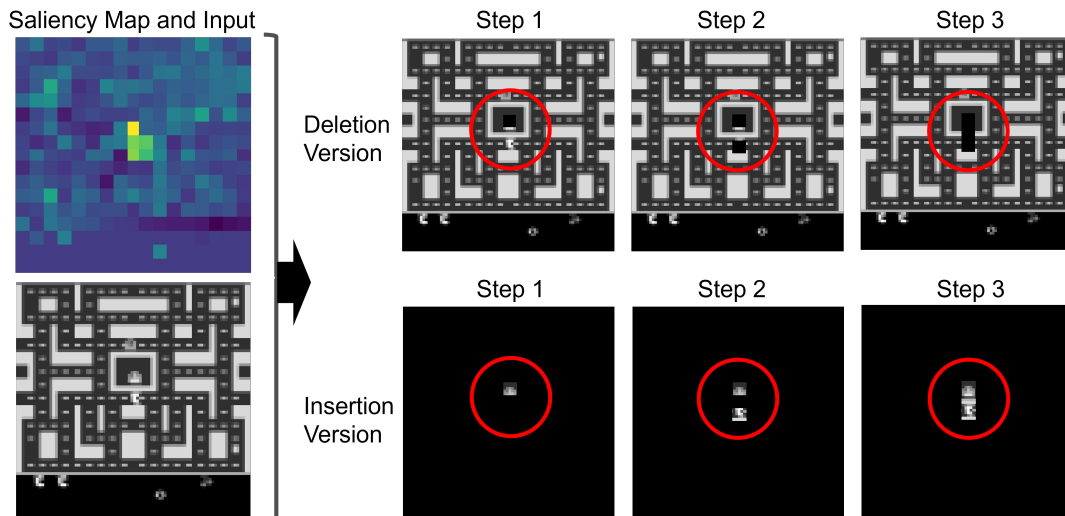
Computational metrics offer a way to evaluate explanations through quantifiable measures that can be calculated automatically, without needing human intervention at runtime.

Some computational metrics are intended to serve as proxies for measuring **subjective criteria** desired by human users. In the case of counterfactual explanations (Section 3.2.2), this is, for example, done by measuring how similar the counterfactuals are to the original state [Pawelczyk et al., 2021; Keane et al., 2021; Mothilal et al., 2020]. This ensures that the counterfactuals describe the smallest possible modification that would change the AI’s prediction. For saliency maps (Section 3.2.1), some work computationally compares the generated saliency maps to prerecorded human visual attention [Schiller et al., 2020; Mohseni et al., 2021a]. This comparison aims to check if the attentional patterns of the saliency maps resemble those of humans. The intuition is that users subjectively prefer saliency maps that reflect human attention patterns. However, this comparison does not objectively measure whether the saliency maps focus on the correct information that was important to the AI. Furthermore, this methodology requires human users to record their visual attention. These recordings can then be used to evaluate various different saliency maps automatically. As such, this methodology can be seen as a hybrid between computational metrics and user studies.

More important, however, is the use of computational metrics to determine **objectively** whether explanations reflect the AI’s internal reasoning. For counterfactual explanations, this is usually done by evaluating their **validity**, i.e., the degree to which the counterfactuals actually evoke the target prediction for the AI [Mertes et al., 2022a; Mothilal et al., 2020]. In the case of saliency maps, we want to know whether the most relevant input features according to the saliency map are actually the most relevant features for the AI. This metric is often referred to as the saliency map’s **fidelity** [Mohseni et al., 2021b] or **faithfulness** [Tomsett et al., 2020].

#### 3.3.1.1. Evaluating Saliency Map Fidelity

There is a growing body of work on computationally evaluating the fidelity of saliency maps for image classification models. The most common measurement is **input degradation**. Here, the input of the model is gradually perturbed, starting with the most relevant input features according to the saliency map. For visual input, this is either done by perturbing individual pixels per step [Petsiuk et al., 2018; Ancona et al., 2018] or by perturbing patches of the image in each step [Samek et al., 2017; Kindermans et al., 2018; Schulz et al., 2020]. If the saliency map matches the model’s reasoning, then the model’s confidence



**Figure 3.8.:** Representation of the different variations of the input degradation metric. Based on a given saliency map, the metric either perturbs features from the input (top row) or inserts them into an empty input (bottom row) step by step. The red circles are added in this thesis to highlight the areas that have changed. At each step, the confidence of the AI model is measured to see how much the features influence the AI’s decision.

should fall quickly. In addition to perturbing features, some newer approaches also propose an insertion metric where they start with fully perturbed inputs and gradually insert relevant features [Ancona et al., 2018; Petsiuk et al., 2018; Schulz et al., 2020]. Figure 3.8 shows a representation of how these different variations of the input degradation metric work. Recently, Tomsett et al. [2020] demonstrated that input degradation can be unreliable and is sensitive to implementation details like the type of perturbation. They conclude that researchers should employ several versions of this metric and try to understand potential reasons for unreliability.

A different technique to computationally evaluate saliency maps for classification models is to compare them with ground-truth saliency maps on modified datasets [Yang and Kim, 2019; Zhou et al., 2022]. Here, a natural dataset is manipulated by adding artificial features that a model has to focus on to classify the dataset perfectly. Now, saliency maps for a perfect model on the manipulated dataset can be evaluated based on how well they localize the artificial features. However, we cannot be 100% certain that the perfect model learned to focus on the artificial features. Furthermore, it is not obvious how this method could be applied to DRL agents. First, in a reinforcement learning setting, there is no easily available dataset that can be manipulated. Secondly, DRL agents do not directly classify which objects or features are contained in an image.

Therefore, it is not clear how the long-term decision-making of DRL agents is supposed to react to artificial features.

Another prominent computational measurement for saliency maps for image classification models are the so-called **sanity checks** proposed by Adebayo et al. [2018]. These tests measure whether the saliency maps are dependent on the learned parameters of the model’s neural network. One method for this is gradually randomizing the layers of the neural network and measuring how much this changes the saliency maps. If the saliency maps are faithful to what the network learned, then they should change considerably for each randomized layer. Examples of this can be seen later in Figure 9.5.

Adebayo et al. conducted sanity checks for various gradient-based approaches and Sixt et al. [2020] additionally tested modified propagation methods. Both groups found that some approaches did not really depend on the parameters of the network and, therefore, cannot faithfully reflect the model’s internal reasoning. As far as I know, before this thesis, no work has verified whether different types of perturbation-based saliency maps depend on the network’s learned parameters for any kind of model even though this is one of the most popular saliency map approaches.

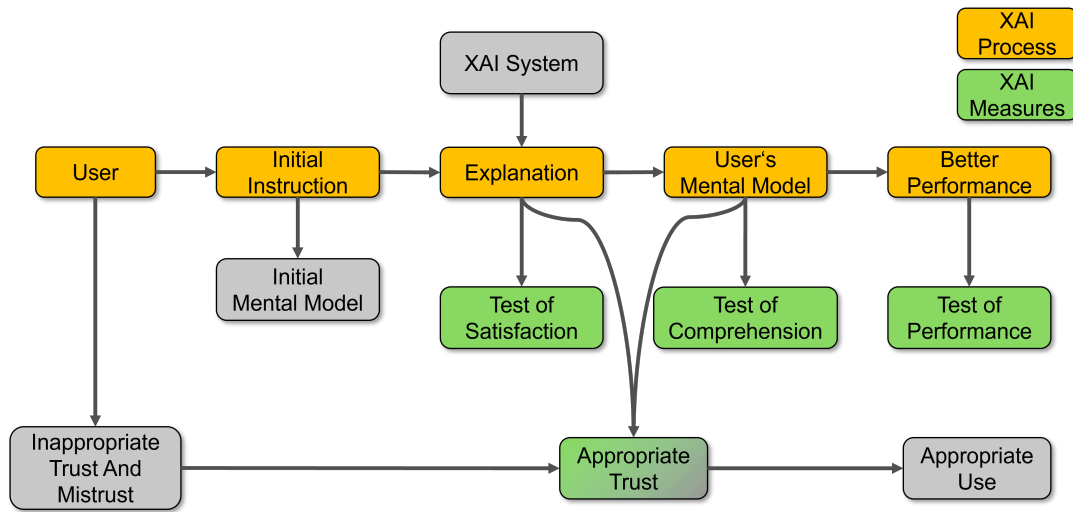
### 3.3.2. Human User Studies

One critical challenge in evaluating XAI is the lack of consensus on how to computationally evaluate XAI [Tomsett et al., 2020]. Currently, there is no definitive method for discerning the reasoning processes of black box models, such as neural networks. Consequently, we lack ground truth explanations against which to benchmark the explanations generated by these models. Without this ground truth, computational metrics can only provide different ways to approximate the fidelity of XAI methods.

This underscores the need to approach the evaluation of XAI not just through computational metrics but also through human user studies. Furthermore, user studies are necessary to avoid explanations that are technically sound but not user-friendly [Miller et al., 2017]. This is especially important for explanations that target end users without a machine learning background.

First, to understand how we can design user studies to evaluate XAI, we look at the process of explanation in the context of XAI, as outlined by Hoffman et al. [2018] and Gunning and Aha [2019] (Figure 3.9).

After receiving initial instructions about the AI system, the user forms an initial mental model of how the AI operates. Here, the **mental model** is the cognitive representation that the user has about a complex system [Halasz and Moran, 1983; Norman, 2014], in our case, the AI model or the RL agent. For example, an observer of an RL agent trained to play Pacman might form a mental model that says: “The Pacman agent looks at the area in front of Pacman



**Figure 3.9.:** The explanation process as outlined by Hoffman et al. [2018] and Gunning and Aha [2019].

and moves toward regular pills to eat them.” Note that the mental model need not be correct. In our example, DRL agents for Pacman typically do not just look at the area in front of Pacman, and depending on the reward function, they have goals other than regular pills. Humans automatically form mental models of intelligent agents based on their behavior [Anjomshoae et al., 2019]. These mental models help users understand and explain an agent’s behavior.

After the initial instructions, the user does not have enough information to trust the AI system appropriately and may even mistrust it. Here, the term **appropriate trust** is based on the work of Lee and See [2004], who present a conceptual “trust in automation” framework. They define appropriate trust as a well-calibrated trust that matches the true capabilities of a technical system.

While interacting with the AI, the user receives explanations from the XAI system. Through these explanations, the user refines their mental model of the AI. Ideally, this leads to a better mental model of how the AI works and a more appropriate level of trust. This improved mental model enhances the user’s performance. At the same time, appropriate trust leads to more appropriate use of the AI model.

This model of the process of explanation shows us four different metrics through which we can measure the effectiveness of XAI systems:

**Explanation Satisfaction.** The first metric is how subjectively satisfied the users are with the explanations. To measure this satisfaction, Hoffman et al. [2018] proposed a satisfaction scale with the following items:



1. From the explanation, I understand how the [software, algorithm, tool] works.
2. This explanation of how the [software, algorithm, tool] works is satisfying.
3. This explanation of how the [software, algorithm, tool] works has sufficient detail.
4. This explanation of how the [software, algorithm, tool] works seems complete.
5. This explanation of how the [software, algorithm, tool] works tells me how to use it.
6. This explanation of how the [software, algorithm, tool] works is useful to my goals.
7. This explanation of the [software, algorithm, tool] shows me how accurate the [software, algorithm, tool] is.
8. This explanation lets me judge when I should trust and not trust the [software, algorithm, tool]

The participants rate those items on a 5-point Likert scale from “I disagree strongly” to “I agree strongly”. For XAI on image classifiers, variants of the explanation satisfaction scale have been used by Mertes et al. [2022a] and Mertes et al. [2022b]. Other studies used custom questions to evaluate the users’ subjective satisfaction with the explanations [Weitz et al., 2019b; Weitz et al., 2021].

**Mental Models.** Second, we can measure the users’ mental model of the AI. The examination of users’ mental models and their correctness helps to verify whether XAI has been successfully applied [Rutjes et al., 2019; Arrieta et al., 2020]. An example of a direct method for eliciting a user’s mental model are *teach-back interviews* in which the user describes their own understanding of the system [Hoffman et al., 2018]. However, these interviews are very time-consuming. Therefore, mental model assessments are frequently conducted using indirect methods, such as proxy tasks like the *prediction task*. In such prediction tasks, users are asked to predict the AI’s decision based solely on the input and explanations provided. A well-formed mental model of the AI should enable users to accurately predict its decisions, indicating successful understanding facilitated by the XAI technique. Alqaraawi et al. [2020] and Selvaraju et al. [2020] found that participants who saw saliency maps were able to predict the decision of an image classification model better than participants who did not see them. However, the participants were still only correct in about 60% of the

cases, and Alqaraawi et al. proposed to look beyond instance-level explanations in the future. Moreover, in another prediction task experiment for image classifiers conducted by Mertes et al. [2022a], counterfactual explanations (Section 3.2.2) were more useful than LRP and LIME saliency maps (Section 3.2.1).

**Appropriate Trust.** Third, we can measure how calibrated the user’s trust in the AI models is Lee and See [2004] – i.e., does the user place greater trust in models that perform better? For example, Selvaraju et al. [2020] measure appropriate trust by presenting participants with decisions and explanations from two distinct image classifiers. In all instances that are presented to the participants, the decisions of the AI models are identical – only the explanations differ. However, one model generalizes better to the entire dataset than the other model. In this way, the participants’ trust in the individual models can be evaluated to determine whether the participants’ trust is aligned with the accuracy of the models.

**Performance.** The fourth metric to evaluate the effectiveness of XAI in user studies is user performance. Given that enhancing user performance is one of the primary objectives of XAI, this metric is arguably the most crucial. However, assessing user performance during real-world tasks is challenging and may not always be feasible in a rigorous scientific manner. Some previous work has measured user performance in simulated tasks. For instance, in a study by Buçinca et al. [2020], participants had to identify the percentage of fat in various meals with the assistance of an XAI system. Similarly, Bansal et al. investigated the effect of XAI on user performance in several AI-assisted decision-making tasks [Bansal et al., 2019; Bansal et al., 2021].

## 3.4. Conclusion

This chapter has provided the necessary XAI background for the contributions of this dissertation.

- It introduced the basic terminology and concepts of XAI.
- It explored established XAI methods for classifiers, setting the stage for adapting these methods to meet the unique challenges posed by DRL. In particular, this chapter discussed saliency map methods, including the perturbation-based methods we will evaluate for DRL in Chapter 9, and LRP, for which this dissertation will introduce a DRL-specific variant in 6. Additionally, we explored counterfactual explanation methods, providing the groundwork for the novel DRL-specific counterfactual generation method that Chapter 7 will propose.

- It provided a comprehensive foundation for evaluating XAI methods. By discussing both computational and user-centered metrics for assessing explanation effectiveness, this chapter provided the background for the computational evaluations in Chapters 6, 7, and 9 as well as the user studies in Part V.

## **Part II.**

# **Related Work - Explainable Reinforcement Learning**

Chapter 3 laid the foundation by introducing general XAI concepts and presenting examples of explanation methods and XAI metrics proposed for image classifiers. Building on this foundation, this part will discuss related work that focuses specifically on explaining (deep) RL agents. With the growth of work on XAI in recent years, Explainable Reinforcement Learning (XRL) solidified itself as a distinct subfield [Heuillet et al., 2021; Alharin et al., 2020; Puiutta and Veith, 2020]. This part will show how this dissertation fits into and extends the XRL research field. As with Chapter 3, this part will begin by exploring explanation methods for DRL in Chapter 4 and then cover the evaluation of XRL in Chapter 5.

## 4. Explanation Methods for DRL

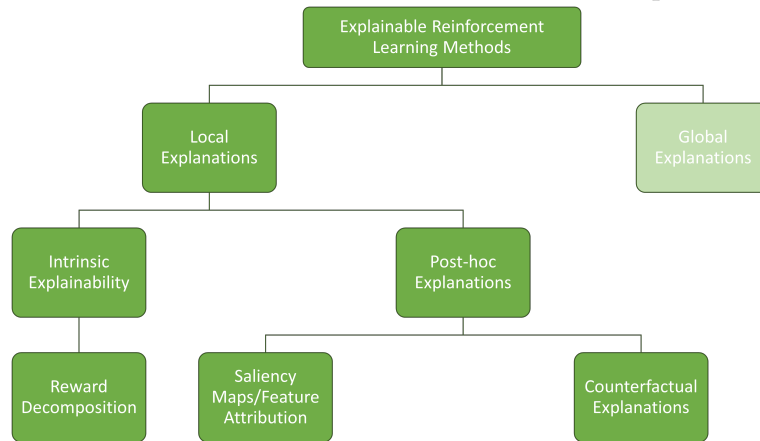
This chapter explores different methods that are used to explain (deep) RL agents. To this end, Section 4.1 starts by looking at local explanation methods, which explain individual decisions of the agent. Then, in Section 4.2, this chapter covers global explanation methods for DRL, which explain the agent’s overall behavior. To provide an overview, Table 4.1 lists all the explanation methods for DRL that this thesis will cover and how I categorize them in the XAI terminology (see Section 3.1).

**Table 4.1.:** Categorization of the XRL methods covered in this thesis.

Method	Global or Local	Intrinsic or Post-hoc
Saliency Maps	Local	Post-hoc
Counterfactual Explanations	Local	Post-hoc
Reward Decomposition	Local	Intrinsic
Explainable Surrogate Models	Global (can be used locally)	Post-hoc
Intrinsically Explainable Agent Architectures	Global (can be used locally)	Intrinsic
Example-based Policy Explanation	Global	Post-hoc

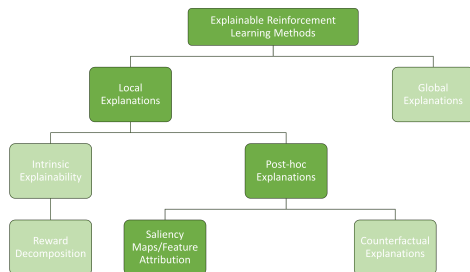
## 4.1. Local Explanations of Agent Behavior

Local explanation approaches in the context of DRL aim to elucidate specific decisions of the agent – e.g., why the agent turned right in a specific Pacman state. Figure 4.1 shows the local XRL methods we will explore in this thesis.



**Figure 4.1.:** An overview of the local XRL methods we will explore in this thesis.

### 4.1.1. Saliency Maps for DRL



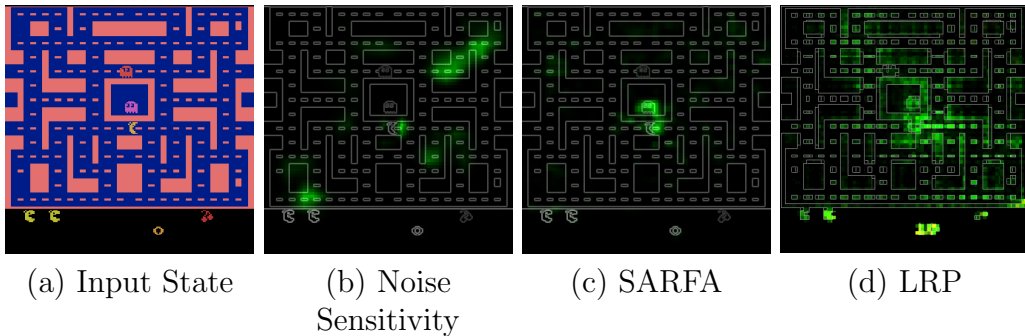
Similar to image classification, the most common explanation methods for DRL are saliency maps that highlight which input information was most relevant to the agent’s decision. In Section 3.2.1, we covered methods to generate saliency maps for image classifiers. This section looks at implementations of saliency maps that focus on DRL. It

extends text from our publication:

**Tobias Huber**, Dominik Schiller, and Elisabeth André [2019]. “Enhancing Explainability of Deep Reinforcement Learning Through Selective Layer-Wise Relevance Propagation”. In: *KI 2019: Advances in Artificial Intelligence*. Springer International Publishing, pp. 188–202

Because many DRL algorithms utilize similar CNNs as image classifiers, it is possible to apply the methods we covered in section 3.2.1 directly to DRL agents.

For **gradient-based saliency maps** (Section 3.2.1.1), Zahavy et al. [2016]



**Figure 4.2.:** The left image (a) shows a Pacman screen. Images (b) and (c) show perturbation-based saliency maps for this state generated by the noise sensitivity approach of Greydanus et al. [2018] (b) and the SARFA approach of Puri et al. [2020] (c). Image (d) shows an LRP  $z^+$ -rule saliency map for a state similar to (a).

and Wang et al. [2016b] utilize the gradient of the network with respect to the input pixels to explain DQN agents (Section 2.2), similar to what Simonyan et al. [2013] did for image classifiers. Weitkamp et al. [2019] tested Grad-CAM on an actor-critic DRL agent.

Lapuschkin et al. [2019] used **LRP** (Section 3.2.1.3) to create saliency maps for DQN agents. Similar to the gradient-based methods mentioned above, they applied LRP without adapting it to the specific challenges of DRL.

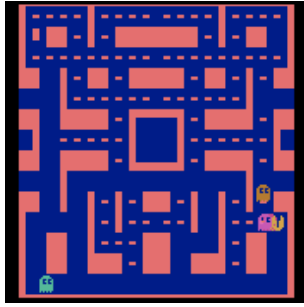
Prior to this dissertation, the only saliency map variants tailored to the specific challenges of explaining DRL agents were **perturbation-based saliency maps** (Section 3.2.1.2). As mentioned in Section 3.2.1.2, Greydanus et al. [2018] and Puri et al. [2020] already targeted DRL agents with their model-agnostic perturbation-based saliency maps. Furthermore, Iyer et al. [2018] proposed a novel mixture of an intrinsically explainable model and perturbation-based saliency maps. Their approach uses template matching to identify objects in each input image and uses this information as additional input channels to train the DRL agent. Given an agent trained in this way, the relevance of an identified object can be measured by comparing the prediction of the input image containing that object with the prediction for the same input image without that specific object.

A drawback of perturbation-based saliency maps is that they are computationally expensive. This often means that they cannot be generated at runtime while the agent is interacting with the environment. Furthermore, while the perturbation-based saliency maps of Greydanus et al. [2018] and Puri et al. [2020] have the advantage of being model agnostic, they depend on various hyperparameters. These hyperparameters must be tuned carefully to avoid the risk of saliency maps that do not faithfully represent the agent’s internal reasoning

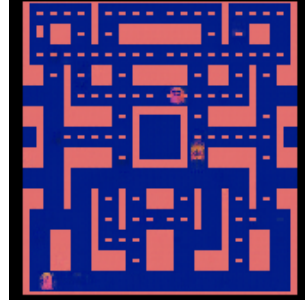


(see Section 3.3.1).

Figure 4.2 shows example saliency maps for the methods used by Greydanus et al. [2018], Puri et al. [2020], and Lapuschkin et al. [2019]. Chapters 6 and 9 will show additional examples.



Original State where Pacman moves down.

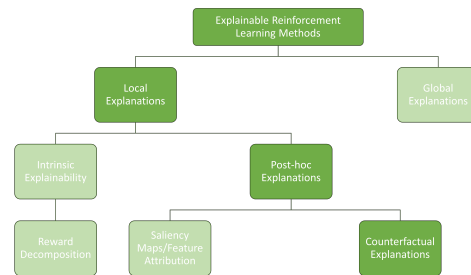


CSE Counterfactual State: Pacman is supposed to move up.

**Figure 4.3.:** An example of a counterfactual explanation generated with the CSE approach by Olson et al. [2021]. Interestingly, the counterfactual completely removes Pacman from the maze. This may be a consequence of the fact that CSE uses an action-invariant latent space. If Pacman is not in the state, it does not matter which action the agent chooses.

#### 4.1.2. Counterfactual Explanations for DRL

RL is often used to create counterfactual explanations for other models (for example, in [Chen et al., 2021]). However, to the best of my knowledge, there is only one previous work on generating visual counterfactual explanations for RL agents [Olson et al., 2021]. This section describes this approach by Olson et al. [2021]. The section is based on text from our publication:



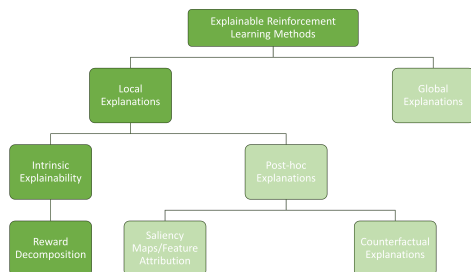
**Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. IFAAMAS, 10 pages

Olson et al. [2021] train a neural network  $E$  that creates an action-invariant latent representation of the agent’s latent space. This is achieved by adversarially training  $E$  in tandem with a discriminator  $D$ , where  $D$  tries to predict the agent’s action and  $E$  aims to make the decision of  $D$  as uncertain as possible. In addition, they train a generative model  $G$  to replicate states  $s$  based on the action-invariant latent representation  $E(s)$  and the agent’s action probability distribution  $\pi(s)$  for this state. By providing  $G$  with a counterfactual action

distribution  $\pi(s)'$ , they obtain a state that is similar to  $s$  but brings the agent’s action distribution closer to the desired counterfactual distribution. However, Olson et al. argue that an arbitrary counterfactual action distribution does not represent a realistic agent output and thus leads to unrealistic counterfactual states. To avoid this, they train an additional neural network that reduces the dimension of the agent’s latent space. This low dimensional latent space is used to perform gradient descent towards a realistic agent output that resembles the desired counterfactual action. Olson et al. refer to their approach as Counterfactual State Explanations (CSE). Therefore, we will also refer to it as CSE in this thesis.

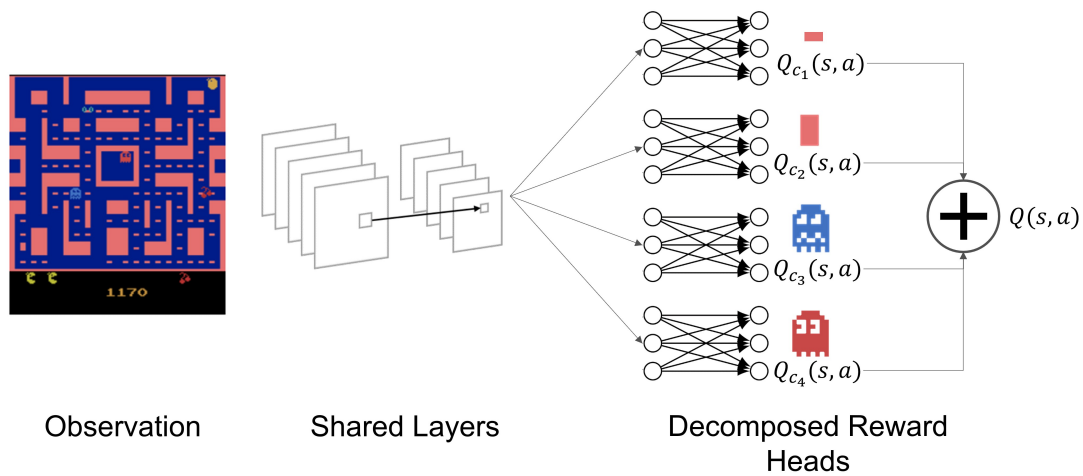
The CSE approach is fairly complex and requires extensive access to the agent’s inner workings. Furthermore, as Olson et al. mention themselves, the loss function of the generator  $G$  does not directly force the resulting state  $G(E(s), \pi(s)')$  to be classified as the counterfactual action distribution  $\pi(s)'$ . This is only learned indirectly by replicating states based on the action-invariant latent space and the desired action distribution  $\pi(s)'$ . As we will show in Section 7.2, this does not seem to be enough to change the agent’s decision correctly. Figure 4.3 shows an example of a counterfactual explanation generated by CSE that illustrates this problem. To solve those problems, we formulate a simpler counterfactual generation method that uses the counterfactual actions in a more direct way.

### 4.1.3. Intrinsic Explanation Methods



Intrinsic explanation methods are built into the agent’s underlying decision model to make it more explainable. Most intrinsic approaches to explainability use intrinsically explainable architectures that cover the global model of the agent. Section 4.2 will go into more detail about such methods.

However, there are also intrinsic approaches that only explain local decisions. One such example, which we will focus on in this thesis, is **reward decomposition**.



**Figure 4.4.:** A schematic of reward decomposition using the hybrid reward architecture.

#### 4.1.3.1. Reward Decomposition

Using the Hierarchical Reward Architecture (HRA) as an example, this subsection introduces reward decomposition. The text is based on our publication:

Yael Septon, **Tobias Huber**, Elisabeth André, and Ofra Amir [2023]. “Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection - 21st International Conference*. Vol. 13955. Lecture Notes in Computer Science. Springer, pp. 320–332. DOI: 10.1007/978-3-031-37616-0\_27

Van Seijen et al. [2017] proposed the HRA model. HRA takes as input a decomposed reward function and learns a separate Q-function for each reward component. In a game like Pacman (see Section 2.1.2.1 for the game rules), such reward components could, for instance, correspond to dying or eating pills. Because each component typically only depends on a subset of all features, the corresponding Q-function can be approximated more easily by a low-dimensional representation, enabling more effective learning.

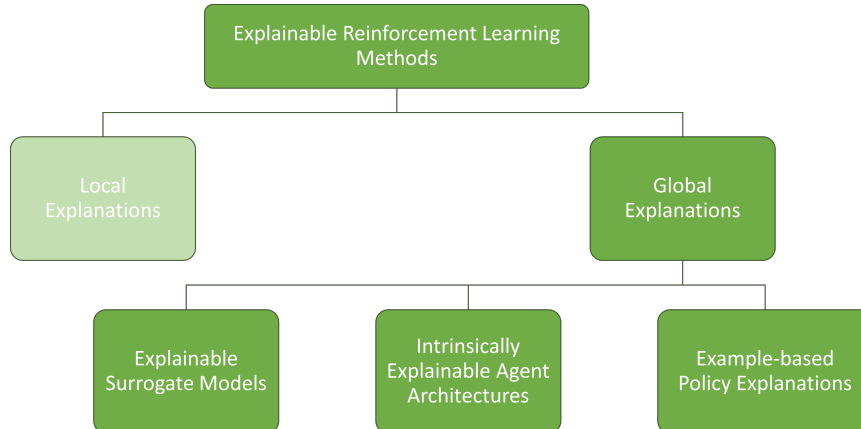
This can be incorporated in the MDP formulation by specifying a set of reward components  $C$  and decomposing the reward function  $r$  into  $|C|$  reward functions  $r_c(s, a, s')$ . The objective for the HRA agent remains the same as for traditional Q-learning: to optimize the overall reward function  $r(s, a, s') := \sum_{c \in C} r_c(s, a, s')$ . HRA achieves this by training several Q-functions  $Q_c(s, a)$  that only account for rewards related to their component  $c$ . If the under-

lying agent uses deep neural networks, the different Q-functions  $Q_c(s, a)$  can share multiple lower-level layers of the neural network. In this case, the collection of Q-functions that each have one type of reward can be viewed as a single agent with multiple *heads*, such that each head calculates the action-values of the current state under its own reward function. For choosing an action for the next step, the HRA agent uses the sum of these individual Q-functions:  $Q_{HRA}(s, a) := \sum_{c \in C} Q_c(s, a)$ . For the update  $Y_i^{DoubleDQN}$  (Section 2.3) each head is used individually. That is, we set the target for  $Q_c$  to  $Y_{i,c}^{DoubleDQN}(s, a, s') := r_c(s, a, s') + \gamma Q_c(s', \operatorname{argmax}_{a' \in \mathcal{A}} Q_c(s', a'; \theta), \theta_i^-)$ . Figure 4.4 shows a schematic of the HRA architecture.

HRA was originally proposed to make the learning process more efficient. However, Juozapaitis et al. [2019] and Erwig et al. [2018] suggested the use of *Reward Decomposition (RD)* as a local explanation method. Traditional Q-values do not give any insight into the positive and negative factors contributing to the agent’s decision since the individual reward components are mixed into a single reward scalar. Showing the individual Q-values  $Q_c(s, a)$  for each reward component  $c$  can explicitly reveal which rewards an agent expects from different actions. Figure 8.3 will show an example of how the individual Q-values can be displayed to users as bar graphs. A user study exploring the usefulness of different local RL explainability methods showed that reward decomposition contributed to people’s understanding of agent behavior [Anderson et al., 2019].

## 4.2. Global explanations of agent behavior

Global explanations for RL agents attempt to describe the high-level policy of an agent. This section will first give a brief overview of common global explanation methods for DRL to put this work in context. Then, we will discuss strategy summarization, which is the global explanation method this thesis focuses on. Figure 4.5 shows the global XRL methods we will explore in this section.



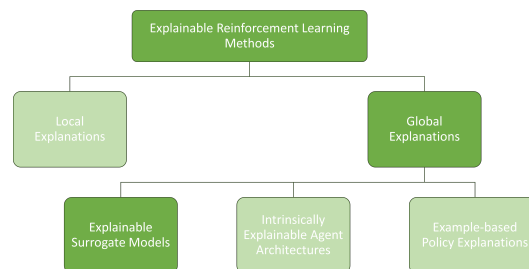
**Figure 4.5.:** An overview of the global XRL methods that we will explore in this thesis.

### 4.2.1. Explainable Surrogate Models

The most common global explanation method for DRL agents is to distill the policy of the black-box agent into an intrinsically explainable surrogate model that approximates the policy of the black-box agent. Here, the decisions of the black-box RL agent are sampled to generate training data for the surrogate model. Analyzing the in-

trinsically explainable surrogate model provides insight into the global policy of the original agent, but it can also be used to create local explanations for specific decisions. Typically, surrogate models are not used to choose actions but only to analyze the agent’s strategy or to generate explanations. However, they can also replace the black-box agent. In this case, they can be considered a hybrid between post-hoc methods and intrinsically explainable models.

A common group of intrinsically explainable architectures are reasonably small **decision trees**. Bastani et al. [2018] sample informative state-action pairs from the trajectories of a black-box RL agent. Based on this dataset, they



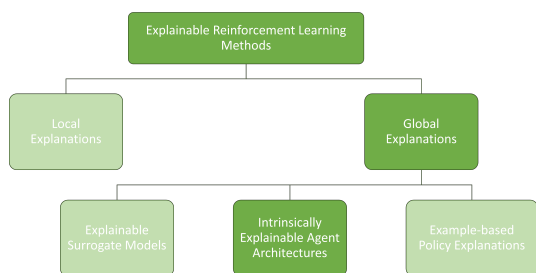
train a decision tree to approximate which discrete action the agent takes in a given state. Liu et al. [2019] use decision trees containing linear functions to approximate the Q-values of value-based DRL agents (see Section 2.1.3). The linear functions also allow them to utilize gradient descent during the training of their surrogate models. Similarly, Coppens et al. [2019] also use trees that contain linear functions but use them to approximate the agent’s policy directly. Bewley and Lawry [2021] aim to train surrogate decision trees that capture the temporal dynamics of the environment by including the difference between successive states in the training.

Besides decision trees, **graphs** are another popular explainable surrogate model. Topin and Veloso [2019] extract what they call policy graphs from a trained DRL agent. They search for abstract states that encompass a set of environment states where the agent chooses the same action and expects a similar outcome. Examining how the agent transitions between these abstract states provides a good understanding of its policy. Bewley et al. [2022] build on the policy graph method by extracting them during the RL agent’s training to reveal how the agent’s policy evolves over time.

Madumal et al. [2020] propose structural causal graphs. These causal graphs are trained together with the RL agent. They approximate the causal relationships between environmental variables and how the agent’s actions affect those variables. Thus, this structural graph can be used to infer causal explanations for the agent’s policy.

A drawback of surrogate-based global explanations is that training intrinsically explainable surrogate models for high-dimensional state spaces, such as visual states, poses considerable challenges. All of the examples above were tested on feature-based state spaces.

#### 4.2.2. Intrinsically Explainable Agent Architectures



Rather than relying on surrogate models to approximate black-box RL agents, other approaches employ RL to directly train agents whose architectures are inherently explainable. In contrast to the previous section, these models are used for both the action selection and the explanation generation.

Similar to intrinsically explainable surrogate models, decision trees are also used directly as an explainable architecture for the agent. Silva et al. [2020] propose the use of differential decision trees and show how to train them with deep reinforcement learning algorithms while keeping their size at an explainable level. Topin et al. [2021] introduce

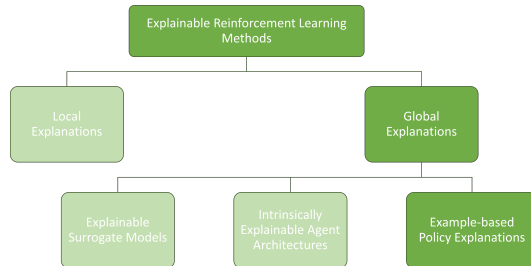
a method to extend normal MDPs so that any DRL policy trained on the extended MDP can be converted into a decision tree for the original MDP. They do this by adding states, which describe bounded value ranges of the original states, and information-gathering actions that compare input features to a given value. In this way, the agent can learn to test which value range the features currently belong to. Using these additions, DRL agents can mimic decision tree nodes by checking which value range the current state belongs to.

Mathematical expressions can also serve as an intrinsically explainable policy. Landajuela et al. [2021] train a neural network to generate a policy given by a mathematical expression. This mathematical expression consists of “operators, input variables, and constants” that compute an action for a given state (e.g.,  $a = s_1 + 0.5s_2$ ). When utilized to execute multiple episodes, this expression yields a reward for the network. To optimize this reward, Landajuela et al. use reinforcement learning.

As with explainable surrogate models, training intrinsically explainable RL agents for complex environments poses substantial challenges and has been limited to relatively simple non-visual scenarios. The aforementioned methods were all tested on feature-based state spaces.

### 4.2.3. Example-based Policy Explanations

The final category of global explanation methods for DRL agents discussed in this thesis is **example-based policy explanation**. The main idea here is to illustrate the agent’s strategy to the user through representative or informative examples that demonstrate how the agent behaves in different scenarios. In contrast to the previous two categories of global XRL, this approach does not attempt to explain the whole logic of the model underlying the agent, which is how Molnar [2022] and Adadi and Berrada [2018] define global explainability for general AI agents. Instead, the objective is to convey the global policy that the RL agent has learned. This category of global XRL aligns with what Amitai and Amir [2023] refer to as explanation by demonstration. It is also similar to the “example-based explanation” approaches described by Adadi and Berrada [2018]. However, Adadi and Berrada [2018] also subsume in this category approaches that explain local decisions by presenting similar examples (e.g., counterfactual explanations). The example-based policy explanation approaches described in this section aim exclusively at explaining the global policy without the need for the examples to resemble a specific local state. Finding suitable examples is not as complex





as creating symbolic descriptions, making example-based policy explanations feasible for the complex visual environments and agent architectures common in DRL.

**T-SNE.** One way of demonstrating an agent’s policy through examples is the use of t-SNE (t-distributed Stochastic Neighbor Embedding) [Maaten and Hinton, 2008] on latent space representations (Section 2.2.2) of a DRL agent. T-SNE visualizes high-dimensional data by projecting it to a 2D or 3D representation. During this process, the proximity of data points is preserved – data points that are close together in the original high-dimensional space remain close in the low-dimensional representation. Several publications apply t-SNE to latent space representations – generated by the agent – of states from different episodes [Jaderberg et al., 2019; Jaunet et al., 2020; Mnih et al., 2015; Such et al., 2019; Zahavy et al., 2016]. This process creates a 2D visualization where states that are close to each other have similar representations in the agent’s latent space. Exploring examples from this visualization gives the user an understanding of how the agent divides the state space. To facilitate the analysis of this 2D state-space visualization, the data points are often colored according to additional useful information. Mnih et al. [2015] and Zahavy et al. [2016] use color to represent the agent’s state value  $V(s)$ . Other examples of color-coding visualize handcrafted values such as specific game situations or the agent’s position [Zahavy et al., 2016; Jaderberg et al., 2019]. While t-SNE visualizations are a useful debugging tool for machine learning experts, they are less suitable for users without a machine learning background.

#### 4.2.3.1. Strategy Summarization and HIGHLIGHTS

**Strategy Summarization.** An approach to explanation-based policy explanation that is more suitable for general users is **strategy summarization** [Amir et al., 2019].

Strategy summarization addresses the following problem: given a trained agent  $\pi$ , we search for a set of state-action pairs that are representative of the agent’s strategy. The motivation is that users lack the time to watch several full episodes of the agent to get a good understanding of its strategy. Instead, viewing only the state-action pairs in the summary should be sufficient to convey the agent’s global strategy to the user.

Several methods have been proposed for selecting the set of state-action pairs to present in a summary. Some methods use ideas from machine teaching, where a trained RL agent “teaches” its strategy to an untrained agent. The intuition is that examples that help transfer knowledge from the trained RL agent to another agent will also help users understand the agent’s strategy. Huang et al. [2019] create summaries containing states that would help other agents infer the

original agent’s reward function. Similarly, Lage et al. [2019] select the state-action pairs that would help other agents imitate the original agent’s actions on arbitrary states. An alternative strategy summarization approach uses heuristics to identify “interesting” situations based on the distribution of the agent’s output values – in our case, the Q-values. The HIGHLIGHTS algorithm [Amir and Amir, 2018], which is used in the experiments in this dissertation and will be discussed in more detail in the following paragraph, falls into this category. Huang et al. [2018] developed a similar approach in parallel. Finally, Sequeira and Gervasio [2020] add other criteria, such as the frequency of occurrence of a situation, in addition to the distribution of Q-values.

Strategy summarization methods are mostly based on the output of the agent. In addition, they are straightforward to display, as they simply show the state-action pairs within the summary. Therefore, they are applicable to a wide range of DRL agents, including agents with high-dimensional visual state spaces. Their simple presentation also makes them accessible to a wide range of users. However, a notable limitation of strategy summaries is that they rely on the user to interpret the displayed state-action pairs. This limitation is made worse by the fact that they do not provide any local information about the agents’ reasoning in each state. To address this issue, this thesis proposes a combined approach in Chapter 8 that integrates comprehensive strategy summaries with local explanations (see Chapter 4.1), shedding light on agents’ reasoning processes in specific situations.

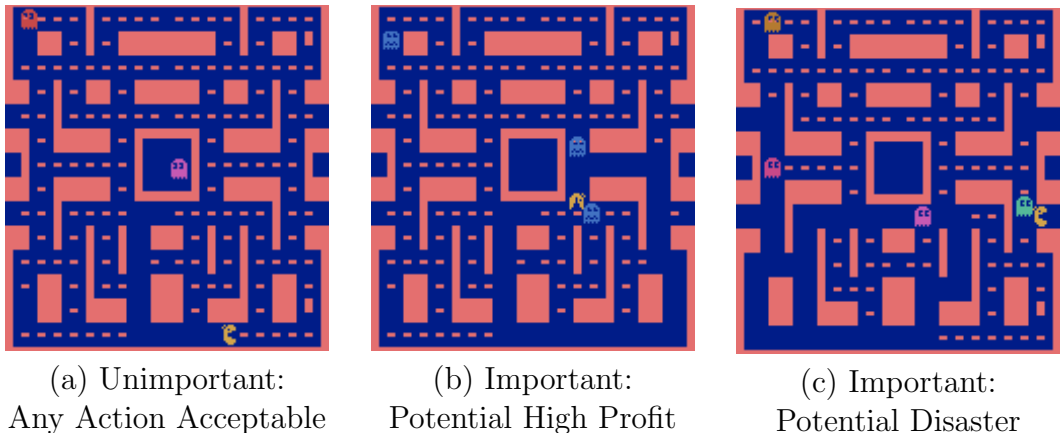
To this end, the subsequent paragraph delves into HIGHLIGHTS, the specific summarization algorithm used in this dissertation.

**HIGHLIGHTS.** This paragraph introduces the HIGHLIGHTS-DIV variant of the HIGHLIGHTS algorithm proposed by Amir and Amir [2018]. Since this dissertation exclusively uses HIGHLIGHTS-DIV, we will refer to it as HIGHLIGHTS for simplicity. The HIGHLIGHTS algorithm [Amir and Amir, 2018] selects state-action pairs for the summary by defining an *importance* metric:

$$I(s) := \max_a Q^\pi(s, a) - \min_a Q^\pi(s, a) \quad (4.1)$$

The intuition behind this metric is that a state is considered *important* by the agent if the agent expects high profits from the best action (Figure 4.6 (b)) or fears disastrous consequences from the worst action (Figure 4.6 (c)). If the agent is indifferent to the choice of action, all Q-values are in a similar range, and  $I(s)$  is low (Figure 4.6 (a)).

The importance metric  $I(s)$  was originally proposed by Torrey and Taylor [2013] to identify the most suitable teaching opportunities for a trained RL agent to impart knowledge to an untrained agent. Similar to strategy summarization, they assume that the teacher agent has only a limited budget  $k$  of situations to



**Figure 4.6.:** Examples of the HIGHLIGHTS state selection. (a) shows a state that would not be included in a HIGHLIGHTS summary since it is unimportant to the agent. Pacman cannot move up or down because walls block its path. Moving right or left does not change its future reward much. (b) and (c) show states that are suitable for a HIGHLIGHTS summary, as they could be interesting to the strategy of a Pacman agent. (b) shows a promising state for the agent. If Pacman moves to the right, it will receive a high reward for eating the blue ghost. If Pacman moves left or up, it will miss this reward. (c) shows a state where the wrong action has fatal consequences. If Pacman moves left, it will be eaten by the ghost and lose one life. Appendix D.4 contains additional examples of HIGHLIGHTS for five different agents.

demonstrate to the student agent. Amir and Amir [2018] transfer the idea of Torrey and Taylor [2013] to the problem of explaining the strategy of a black-box agent to a human user.

To this end, Amir and Amir [2018] propose an algorithm that generates a set  $T = \{t_1, \dots, t_k\}$  of trajectories that summarize the agent’s strategy, where each trajectory  $t_j$  consists of a sequence of  $l$  state-action pairs  $t_j = \langle (s_i, a_i), \dots, (s_{i+l-1}, a_{i+l-1}) \rangle$ . They use trajectories instead of individual state-action pairs to provide context for the important state-action pairs. For example, consider a situation where Pacman is in front of a ghost similar to Figure 4.6 (c). If we do not know what action the agent takes after this frame, we do not know whether Pacman successfully avoids the ghost.

Algorithm 1 shows the pseudocode for the HIGHLIGHTS algorithm proposed by Amir and Amir [2018] without the HIGHLIGHTS-DIV variant. The algorithm can be executed online while the agent interacts with the environment.

The algorithm runs for the number of episodes given by *numSimulations* (lines 1, 5-7, and 24). At each time step, the agent  $\pi$  chooses an action  $a$  for the state  $s$  (line 8), and the importance  $I(s)$  is calculated (line 14). The current

---

**Algorithm 1:** The HIGHLIGHTS algorithm. Replicated from Amir and Amir [2018].

---

**Input:**  $\pi, k, l, numSimulations, intervalSize, statesAfter$

**Output:**  $T$

```

1 runs = 0
2  $T \leftarrow PriorityQueue(k, importanceComparator)$ 
3  $t \leftarrow$  empty list
4  $c = 0$ 
5 while ( $runs < numSimulations$ ) do
6    $sim = InitializeSimulation()$ 
7   while ( $!sim.ended()$ ) do
8      $(s, a) \leftarrow sim.advanceState(\pi)$ 
9     if ( $|t| == l$ ) then
10      |  $t.remove()$ 
11       $t.add((s, a))$ 
12      if ( $c > 0$ ) then
13        |  $c = c - 1$ 
14       $I_s \leftarrow computeImportance(\pi, s)$ 
15      if ( $intervalSize - c == statesAfter$ ) then
16        |  $lastSummaryTrajectory.setTrajectory(t)$ 
17      if ( $(|T| < k)$  or ( $I_s > minImportance(T)$ )) and ( $c == 0$ ) then
18        | if  $|T| == k$  then
19          |  $T.pop()$ 
20        |  $st \leftarrow new summaryTrajectory(I_s)$ 
21        |  $T.add(st)$ 
22        |  $lastSummaryTrajectory \leftarrow st$ 
23        |  $c = intervalSize$ 
24     $runs = runs + 1$ 

```

---

trajectory of the last  $l$  state-action pairs is constantly stored and updated in the list  $t$  (lines 3 and 9-11). The algorithm stores the current summary as a priority queue  $T$  sorted by the importance  $I(s)$  (line 2). The *intervalSize* specifies the minimum number of states between two trajectories in the summary. This is implemented with the parameter  $c$  in lines 4, 12-13, 17, and 23. If the summary  $T$  is not yet full or the importance  $I(s)$  of the current state is higher than the lowest importance in the summary, the current trajectory is added to the summary (lines 17-22). Here, the algorithm uses a *summaryTrajectory* object that stores the trajectory along with the importance  $I(s)$  to compare against future importance values. Finally, the last trajectory added to the summary (*lastSummaryTrajectory*) is extended by *statesAfter* states to include the context after the important state-action pair for which it was added to the summary (lines 15-16). The algorithm depends on five hyperparameters, which Amir and Amir [2018] chose as follows:  $k=5$ ,  $l=40$ ,  $numSimulations=50$ ,  $intervalSize=50$ ,  $statesAfter=10$

**HIGHLIGHTS-DIV** changes the algorithm at lines 17-20. **HIGHLIGHTS-DIV** does not compare the current state’s importance  $I(s)$  with the lowest importance of the summary  $T$ . Instead, it searches for the state  $s'$  in the summary  $T$  that is most similar to  $s$ . Then it compares  $I(s)$  with  $I(s')$  and replaces the trajectory around  $s'$  with the trajectory around  $s$  if  $I(s') < I(s)$ . This adjustment allows less interesting states to remain in the summary if they are different enough from other states. Thus, **HIGHLIGHTS-DIV** increases the diversity of the summary and conveys more information within the same budget. However, **HIGHLIGHTS-DIV** assumes we can define a similarity metric on the state space  $S$ . The Euclidean distance is a natural candidate for such a metric for visual input.

### 4.3. Conclusion

This chapter reviewed relevant related work on the development of explanation methods for DRL. We uncovered several gaps and challenges that this dissertation will address.

- Before this thesis, the development of saliency maps tailored to DRL has primarily focused on perturbation-based variants. Although useful, these approaches are often computationally inefficient, which limits their application in real-time scenarios. Alternatively, Gradient-based and LRP saliency maps have been applied to DRL directly without specific adjustments to the challenges of XRL. To fill this gap, Chapter 6 will introduce a DRL-specific variant of LRP.
- Due to their model-agnostic nature, perturbation-based saliency maps are

advantageous when the agent’s internal model is inaccessible. However, their reliance on hyper-parameters can lead to saliency maps that do not accurately reflect the agent’s internal reasoning. In Chapter 9, this thesis will demonstrate a methodology to tune the hyper-parameters of perturbation-based saliency maps for DRL.

- Prior to this thesis, there was only a single method for creating counterfactual explanations for DRL agents with visual input. This approach requires extensive access to the agent’s internal model and only implicitly accounts for the agent’s actions. Thus, its applicability to diverse DRL agents is limited, and it often results in counterfactuals that inadequately represent the agent’s actions. To overcome these limitations, Chapter 7 will introduce a model-agnostic counterfactual explanation approach that directly incorporates the agent’s actions.
- A substantial portion of the related work on global explanations for DRL focuses on creating intrinsically explainable agents – either by training them from scratch with RL or by distilling trained black-box agents into explainable models. However, such approaches are not feasible for high-dimensional state spaces, such as those involving visual inputs. In contrast, example-based global explanations are feasible for visual domains but do not provide any explanations about local decisions. Chapter 8 bridges this gap by presenting a combination of global example-based strategy summaries with local explanations. Thereby, it offers a comprehensive explanation framework suitable for complex visual environments.

## 5. Evaluation of Explanation Methods for RL Agents

This chapter discusses related works on evaluating XRL. It extends text from our publications:

**Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

*and*

**Tobias Huber**, Benedikt Limmer, and Elisabeth André [2022]. “Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents”. In: *Frontiers in Artificial Intelligence* 5. ISSN: 2624-8212. DOI: 10.3389/frai.2022.903875

As with the evaluation of general XAI methods in Section 3.3, this chapter divides the evaluation of XRL into human user studies and computational metrics.

### 5.1. Human User Studies

So far, DRL agents have primarily been evaluated with user studies. As mentioned in Section 1.1, DRL agents are particularly challenging for users to understand compared to classifier models. DRL agents may learn strategies that are unexpected for the user but still optimal for the reward function. In addition, users need to understand the temporal interactions between different actions. This section provides an overview of XRL user studies sorted by their evaluation metrics. See Section 3.3.2 for a definition of the metrics.

**Mental Models.** A common method to get insights into the participants’ mental model of RL agents are prediction tasks, similar to the ones for image classifiers by Alqaraawi et al. [2020] and Selvaraju et al. [2020] (see Section 3.3.2). Iyer et al. [2018] used an action prediction task to evaluate the effect of saliency maps (Section 4.1.1) on the participants’ mental model of the agents but found

no clear benefit of saliency maps. Huang et al. [2019] and Lage et al. [2019] asked participants to predict what actions an agent would take based on strategy summaries optimized for policy reconstructions (Section 4.2.3). Their results show that summary methods that better match people’s computational models lead to improved action prediction but that people may use different models in different contexts. Madumal et al. [2020] showed that their intrinsically explainable RL method significantly helped participants in a prediction task.

Other studies used debugging tasks aimed at machine learning practitioners to see if the participants’ mental models are good enough to identify faulty agents. Greydanus et al. [2018] showed that their perturbation-based saliency maps (Section 3.2.1.2) helped participants to identify overfit policies. Similarly, Olson et al. [2021] demonstrated that their counterfactual explanations (Section 4.1.2) helped participants to identify a faulty agent that did not observe the input correctly.

Finally, some studies used agent understanding tasks to get a more nuanced view into the participants’ mental model of the agent. Anderson et al. [2019] asked their participants to describe the strategy of an RL agent textually. Based on these descriptions, they used summative content analysis [Hsieh and Shannon, 2005] to assess the participants’ mental models of the agent. They compared saliency maps, reward decomposition (Section 4.1.3.1), and a combination of both methods. Their results show significant positive effects for reward decomposition and the combined approach and a marginally significant ( $p = 0.086$ ) effect in favor of saliency maps. However, they only used a single agent in their evaluation. Sequeira and Gervasio [2020] extended this method by using custom observation spaces that can be manipulated to train several agents with qualitatively different policies. They showed that strategy summaries generated by a variety of interestingness criteria improved people’s ability to identify regions of the state space in which each agent spends more time and regions of the state space in which each agent requires additional training [Sequeira and Gervasio, 2020]. However, a drawback of their evaluation is that the custom observation spaces require profound modifications to the RL agents’ architecture and environment. The studies in this thesis improve on this by solely changing the reward function to obtain agents with qualitatively different policies. Sequeira and Gervasio also adjust the reward, but only for one of their agents.

**Appropriate Trust.** Importance-based strategy summaries (e.g., HIGHLIGHTS) were shown to improve people’s ability to identify the better-performing agent in an agent comparison task [Amir and Amir, 2018] and their ability to decide whether to trust an agent in specific world states [Huang et al., 2018]. Both of these studies create different agents by varying the training duration. While the resulting agents differ in their final scores, they all follow similar



strategies. Sequeira and Gervasio [2020] took this further by showing that their interestingness-based strategy summaries help participants to appropriately trust agents with qualitatively distinct policies (see previous paragraph). However, Sequeira and Gervasio only measured the participants’ perceived trust by asking them to rate their trust in each agent. They did not provide an incentive for a greater allocation of trust to superior agents. The studies in this thesis address this limitation by measuring demonstrated trust through an agent comparison task. Participants have to choose an agent to play on their behalf, and if they choose the better agent, they receive a bonus payment.

**Performance.** Puri et al. [2020] showed that their SARFA saliency maps (see Section 3.2.1.2) can help participants solve chess puzzles by highlighting which pieces were relevant for an agent’s solution for these puzzles. Similarly, Tabrez et al. [2022] demonstrated that saliency map explanations can support participants in an AR-based minesweeper environment.

**Explanation Satisfaction.** Madumal et al. [2020] showed that their intrinsically explainable RL agents significantly improved explanation satisfaction, which they measured through the scale proposed by Hoffman et al. [2018] (see Section 3.3.2 for the full scale). Tabrez et al. [2022] found promising results for saliency maps in their subjective questions, which were sampled from different established questionnaires, including the explanation satisfaction scale [Hoffman et al., 2018]. Both Madumal et al. and Tabrez et al. also measure the participants’ subjectively perceived trust in the agents with Tabrez et al. finding positive results for their explanations. However, neither study measures appropriate trust as they do not compare the participants’ trust with the performance of the RL agents.

## 5.2. Computational Metrics

Exclusively relying on user studies might only measure how convincing the explanations are but not how much they reflect the agent’s internal reasoning. Therefore, it is important to additionally evaluate explanations through computational measurements [Mohseni et al., 2021b]. Such measurements also provide an easy way to collect preliminary data before recruiting users for a user study.

In Section 3.3.1, we already looked at computational evaluation for general XAI. However, XRL presents additional challenges related to the long-term decision-making of DRL agents. Each action of a DRL agent is potentially influenced by delayed rewards that the agent expects in the future. Therefore, evaluating explanations for these actions must take into account how the actions fit into the agent’s overall strategy. For example, value-based DRL agents learn

both the value of each action and the value of the state in the current strategy (Section 2.1.3). A computational evaluation of XRL must account for both of these value estimates.

Despite this additional challenge, there is very little work on computationally evaluating the fidelity of post-hoc explanation methods, such as saliency maps, for DRL agents. Puri et al. [2020] recorded which chess pieces human experts identified as important in a set of chess puzzles. This allows them to computationally compare these pieces to the pieces that saliency maps identify as relevant for an agent. However, this does not measure the saliency maps' fidelity to the agent's reasoning – it only measures whether the saliency maps coincide with human reasoning. Atrey et al. [2020] conduct experiments to verify hypotheses that are generated from observing saliency maps. However, both the formulation of hypotheses as well as their verification rely on manual inspection of the saliency maps. Therefore, this method requires extensive human effort. Moreover, it is not certain whether an erroneous hypothesis has been formulated because the saliency maps are faulty and do not reflect the agent's reasoning, or because the human observers misinterpreted the saliency maps.

### 5.3. Conclusion

To summarize, this dissertation extends existing work on evaluating explanations for DRL in three ways:

- Regarding user studies, this dissertation presents the first user studies that evaluate the combined and individual benefits of global and local explanations for DRL. While Anderson et al. [2019] also investigated a combined explanation approach, they integrated two local explanation methods.
- Additionally, before this dissertation, no user study for XRL encompassed a holistic evaluation of appropriate trust, agent understanding, and subjective explanation satisfaction. Madumal et al. [2020] and Sequeira and Gervasio [2020] used similar metrics, but Sequeira and Gervasio did not assess subjective explanation satisfaction and Madumal et al. only measured perceived trust and not appropriate trust.
- Prior to this dissertation, there was no fully automated computational saliency map evaluation that specifically targeted XRL. The sanity check in Chapter 6 and the thorough computational evaluation in Chapter 9 can be seen as the first computational evaluation to benchmark different saliency map approaches for DRL agents.

## **Part III.**

# **Approaches for Explaining DRL Agents**

## 6. LRP-Argmax: Selective Saliency Maps for DRL Agents

As we have seen in Section 4.1.1, the generation of saliency maps, which highlight the areas in the input that were relevant for the agents’ decision-making process, is a common approach to explaining the actions of DRL agents. While such saliency map algorithms are already well established and were even used to understand and improve how we transfer knowledge from one classification model to another [Schiller et al., 2019; Schiller et al., 2020; Prajod et al., 2021], they are usually developed with experienced machine learning practitioners in mind. This can make the generated explanations difficult to interpret for beginners or users who are unrelated to the field of machine learning. Weitz et al. [2019a], for example, found that traditional saliency maps are too fine-granular for humans to easily detect relevant features for the classification. In a recent meta-study, Miller [2019] explored the explanation process between humans to derive new design paradigms for explainable artificial intelligence algorithms that can help to make such methods more accessible to non-expert users. One major finding of this study was that people usually prefer selected explanations that focus on specific evidence instead of showing every possible cause of a decision. Based on this insight, this chapter adjusts an existing saliency map approach to be more focused on the parts of the input that are most relevant for the decision-making process of a system. The chapter is based on the publication:

**Tobias Huber**, Dominik Schiller, and Elisabeth André [2019]. “Enhancing Explainability of Deep Reinforcement Learning Through Selective Layer-Wise Relevance Propagation”. In: *KI 2019: Advances in Artificial Intelligence*. Springer International Publishing, pp. 188–202

With some additions from:

**Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

We base our approach on LRP, which we discussed in detail in Section 3.2.1.3. In contrast to most other approaches, LRP offers the benefit of conserving the confidence value of the prediction throughout its process [Montavon et al., 2018].

Thus, the intensity of the saliency maps reflects the agent’s confidence, which can help to decide whether to trust the agent. Furthermore, there is a natural way of excluding negative relevance values with LRP [Montavon et al., 2018]. In some use cases, such contradictory evidence can be confusing for users. For example, when the saliency maps are shown as a video, the user does not have time to compare positive and negative relevance values.

Our adjustment uses an *argmax* function to follow only the most contributing neurons of each convolutional layer, which enables us to filter out the most relevant information. Therefore, we can create selective and more focused saliency maps while maintaining the advantageous properties of LRP mentioned above.

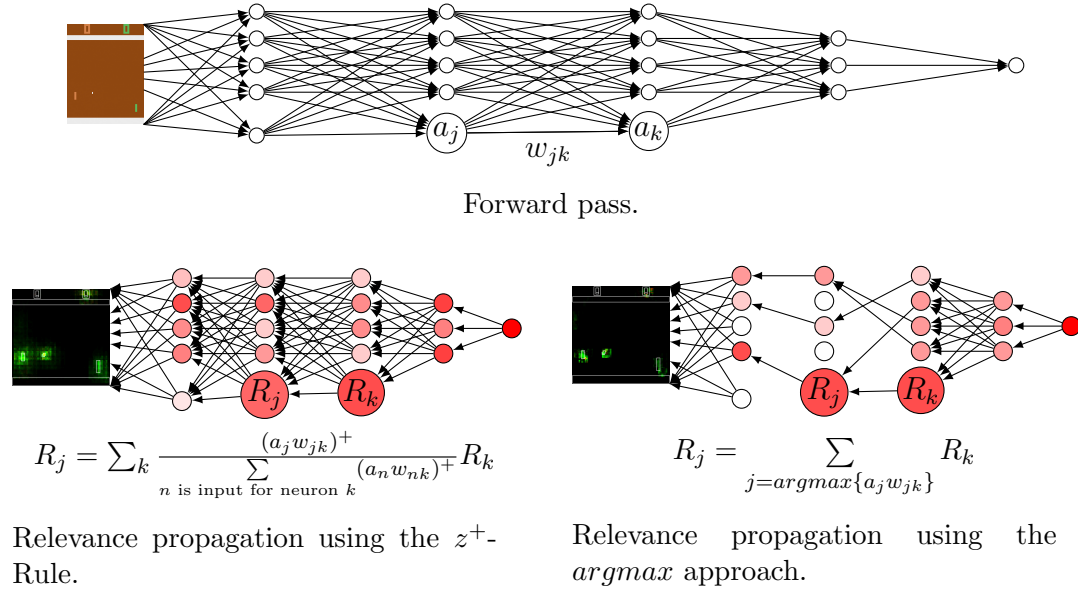
Modern DQN variants, like the rainbow algorithm [Hessel et al., 2018], employ dueling DQN systems that use two separate estimators to measure the value of the current state and the advantage of each action the agent can take in that state. Since the dueling DQN approach considerably alters the neural network architecture of the original DQN (see Section 2.2), it requires its own LRP variant. To test our approach with state-of-the-art dueling DQN algorithms, we introduce a slightly adapted version of LRP that can handle the dueling DQN architecture without losing its advantageous properties. Since no other improvement of the DQN algorithm considerably changes the underlying neural network architecture, this extension allows us to use LRP on any DQN-based DRL algorithm without any further adjustments.

We test our approach on three Atari 2600 games (see Section 2.1.2) of varying complexity using the OpenAi gym and baselines libraries [Brockman et al., 2016; Dhariwal et al., 2017].

## 6.1. An argmax approach to LRP

In this section, we introduce our adjustment to the LRP variant called  $z^+$ -rule, which is described in Section 3.2.1.3. Recent work indicates that DRL agents focus on certain objects within the visual input [Iyer et al., 2018; Goel et al., 2018]. With our approach, we aim to generate saliency maps that reflect this property by focusing on the most relevant parts of the input instead of giving too many details. For this purpose, we propose to use an *argmax* function to find the most contributing neurons in each convolutional layer.

This idea is inspired by Mopuri et al. [2019], who generated saliency maps for neural networks solely based on the positions of neurons that provide evidence in favor of the prediction. During this process, they follow only the most contributing neurons in each convolutional layer. Our method adds relevance values to the positions of those neurons and, therefore, expands the approach of Mopuri et al. by an additional dimension of information. Since those relevance values follow the LRP concept, they also possess the advantageous properties



**Figure 6.1.:** A visualization of how our  $\operatorname{argmax}$  approach differs from the  $z^+$  Rule.

of the LRP concept, like the conservation of the prediction value.

As we have seen in the background Section 3.2.1.3, an LRP method is defined by its messages  $R_{j \leftarrow k}^{l, l+1}$  which propagate the relevance from a layer  $l + 1$  to the preceding layer  $l$ . If  $l + 1$  is a fully connected layer  $fc_i$  of the DQN (see Section 2.2.1 for our notation of the DQN architecture), we use the same messages that are used in the  $z^+$ -rule. In the case that  $l$  and  $l + 1$  are convolutional layers  $conv_{i-1}$  and  $conv_i$ , we propose new messages based on the  $\operatorname{argmax}$  function. To define those messages, we analyze how the activation of a neuron  $conv_i(x)_k$  was calculated during the forward pass. Let  $W$  and  $A$  denote the weight kernel and part of  $conv_{i-1}(x)$  respectively that were used to calculate  $conv_i(x)_k$  during the forward pass. If we write  $W$  and  $A$  in an appropriate vector form, we get

$$conv_i(x)_k = \sigma\left(\sum_j w_j a_j + b\right),$$

where  $\sigma$  denotes the activation function of  $conv_i$  and  $b$  the bias corresponding to  $W$ . Analogously to the  $z^+$ -rule, we assume that the activation function and the bias can be neglected when determining the relevance values of the inputs  $a_j$ . We propose to use an  $\operatorname{argmax}$  function to find the most relevant input neurons by defining the messages in the following way

$$R_{j \leftarrow k}^{l, l+1} := \begin{cases} R_k^{l+1} & \text{if } j = \operatorname{argmax}\{w_j a_j\} \\ 0 & \text{if not.} \end{cases}$$

This definition satisfies the LRP condition given by Equation 3.8 because the only non-vanishing summand of the sum

$$\sum_{j \in \{j \text{ is input for neuron } k\}} R_{j \leftarrow k}^{l, l+1}$$

is  $R_k^{l+1}$ .

If we use the same *argmax* approach to propagate relevance values from  $conv_1$  to the input, which we denote with  $conv_0$ , then we get very sparse saliency maps where only a few neurons are highlighted. If we highlight the entire areas of the input  $conv_0$  that were used to calculate relevant neurons of  $conv_1$ , then we lose information about the relevance values inside those areas. Therefore, we draw inspiration from the guided Grad-CAM approach introduced by [Selvaraju et al., 2020]. Guided Grad-CAM uses one thorough relevance analysis for the neurons of the last convolutional layer to get relevant areas for the specific prediction and another thorough relevance calculation for the input pixels to get fine granular relevance values inside those areas. We already did a thorough analysis of the neurons of the last convolutional layer by using the  $z^+$ -rule on the fully connected layers. By following the most relevant neurons through the convolutional layers, we keep track of the input areas that contributed the most to those values. Mimicking the second thorough analysis of the Guided Grad-CAM approach, we propose to use the  $z^+$ -rule to propagate relevance values from  $conv_1$  to  $conv_0$ . This generates fine granular relevance values inside the areas identified by following the most contributing neurons and ascertains that those relevance values follow the LRP concept.

Figure 6.1 visualizes the differences between our *argmax* approach and the  $z^+$ -rule. An open-source implementation of our algorithm that builds upon the iNNvestigate framework [Alber et al., 2019] can be found here: [https://github.com/HuTobias/LRP\\_argmax](https://github.com/HuTobias/LRP_argmax).

## 6.2. LRP on Dueling Q-Networks

The dueling Q-network is a neural network architecture first introduced by Wang et al. [2016b] as an improvement of the neural network architecture used in the DQN algorithm (see Section 2.2). Because it is only changing the architecture of the neural network, it is independent of the training algorithm. Therefore, it can easily be combined with other improvements to the DQN algorithm. This can be seen in the rainbow algorithm, the current state-of-the-art version of the DQN [Hessel et al., 2018], which combines many different improvements of the DQN algorithm. We chose Dueling DQN because the LRP concept only depends on the neural network architecture. Therefore, applying LRP to the

Dueling DQN architecture suffices to apply LRP on all currently used versions of the DQN algorithm.

Instead of using a single fully connected network after the convolutional part of the DQN, the Dueling DQN architecture uses two fully connected networks  $A$  and  $S$ , both of which use the output of the last convolutional layer as input. These two fully connected networks share the same architecture apart from their output layer. For an input state  $s$ , the state value network  $S$  has only one single output neuron  $S(s)$  that measures the value of the state  $s$ . The network  $A$  has an output neuron  $A(s, a)$  for each action  $a$ , describing the advantage of choosing the action  $a$  in the state  $s$ . The Q-value (the prediction of the whole model) for an input state  $s$  and an action  $a$  is then calculated by

$$Q(s, a) = S(s) + A(s, a) - \frac{1}{N} \sum_{i=1}^N A(s, a_i), \quad (6.1)$$

where  $N$  denotes the number of available actions  $a_i$ .

One way of using LRP on this architecture would be to use LRP methods on each of the networks  $S$  and  $A$  separately, but then we would lose the conservation property because the relevance values would not add up to  $Q(s, a)$ . Therefore, we have to define a way to propagate the relevance value of the output  $Q(s, a)$  to  $S(s)$  and  $A(s, a)$ . Because Equation 6.1 is already a linear decomposition, the main question is how we handle the summand  $-\frac{1}{N} \sum_{i=1}^N A(s, a_i)$ . For this we follow the original thought process of Wang et al. [2016b], who treat  $(A(s, a) - \frac{1}{N} \sum_{i=1}^N A(s, a_i))$  as the modified contribution of  $A(s, a)$  to  $Q(s, a)$ . Analogously to the  $z^+$ -rule, we only propagate those values if they are positive since we want to exclusively highlight evidence in favor of the chosen action  $a$ . That is, we set

$$S(s)^+ := \max(0, S(s)) \quad (6.2)$$

$$A(s, a)^+ := \max(0, A(s, a) - \frac{1}{N} \sum_{i=1}^N A(s, a_i)). \quad (6.3)$$

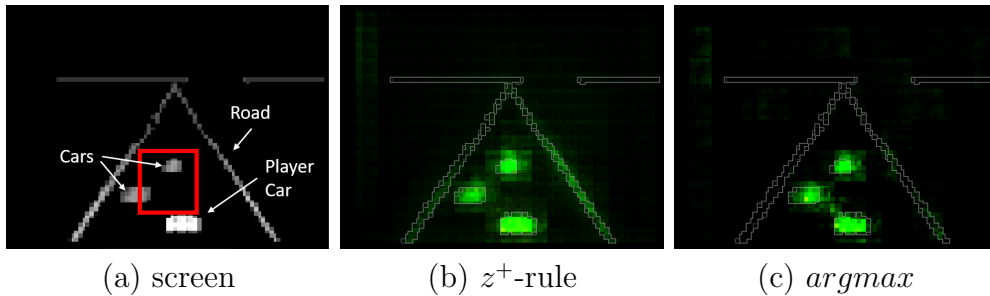
If we would use these values as LRP messages, then the LRP Equation 3.8 would not hold if either of  $S(s)$  or  $A(s, a)$  are negative. Therefore we set the LRP messages analogously to the  $z^+$ -rule as:

$$R_{S(s) \leftarrow Q(s, a)} := \frac{S(s)^+}{S(s)^+ + A(s, a)^+} Q(s, a) \quad (6.4)$$

$$R_{A(s, a) \leftarrow Q(s, a)} := \frac{A(s, a)^+}{S(s)^+ + A(s, a)^+} Q(s, a). \quad (6.5)$$

If both  $S(s)$  and  $A(s, a)$  are negative, then there is no evidence in favor of the prediction. Consequently, it is justified that we do not propagate any relevance values in this case.



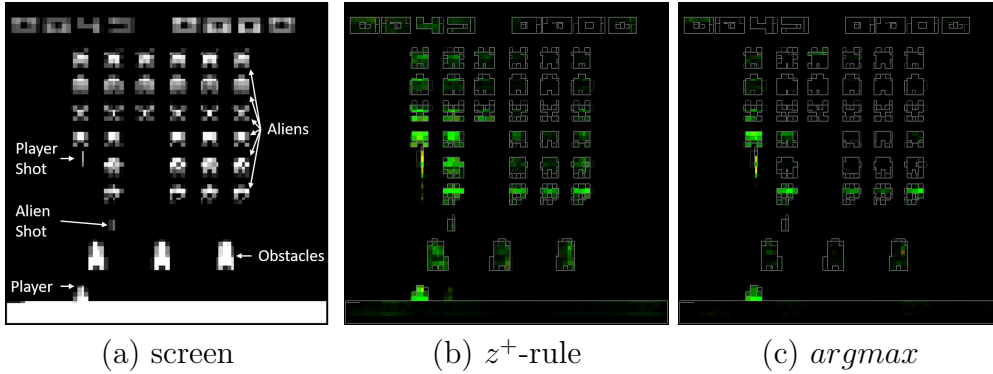


**Figure 6.2.:** A comparison of action advantage analysis: The left image (a) shows a screen from the Atari game Enduro with additional descriptions. The red area was identified as relevant by gradient-based saliency maps in [Wang et al., 2016b]. While the  $z^+$ -rule (b) highlights the cars and the edge of the road, even though it is not important in this situation, our *argmax* approach (c) selects only the relevant cars.

### 6.3. Illustration of the Selectivity of the *argmax*-rule

In order to verify that our *argmax* approach, described in Section 6.1, creates more selective saliency maps than the  $r^+$ -rule (see Section 3.2.1.3), we tested our approach on three different Atari 2600 games and present the results of those experiments in this section. For all games, we trained an agent using the DQN implementation of the OpenAi baselines framework [Dhariwal et al., 2017]. Since this implementation utilizes the Dueling DQN architecture [Wang et al., 2016b], we used the approach described in Section 6.2 to apply LRP to this architecture.

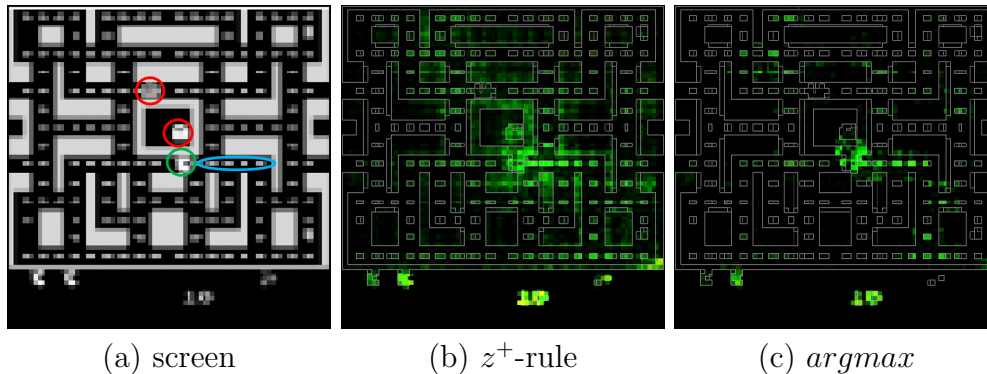
We keep track of which relevance values correspond to the state value and the action advantage values and differentiate them by coloring them red and green, respectively. This allows us to compare our saliency maps with the ones generated by gradient-based methods in [Wang et al., 2016b] for a Dueling DQN agent trained on the Atari game Enduro. In this simple driving game, the Player controls a car and has to avoid hitting other cars while overtaking as many of them as possible. The left image of Figure 6.2 shows a screen from this game in the preprocessed form that the agent received. The area that was identified as relevant for the action advantage value in similar game-states by the gradient-based saliency maps in [Wang et al., 2016b] is marked in red. To facilitate readability, we added descriptions of the important game objects and cut off the lower part of the screen, which only contains the score. The middle and right images show saliency maps generated by the  $z^+$ -rule and our *argmax* approach, respectively, for the game state shown in the left image. All three saliency maps identified the area in front of the player’s car as the most relevant



**Figure 6.3.:** The first image (a) shows a screen of the Atari game Space Invaders with additional descriptions. The saliency map created for this game-state by the  $z^+$ -rule (b) highlights most of the aliens and all the obstacles, while our *argmax* approach (c) focuses on the first row of aliens which the agent can actually hit.

area. The gradient-based saliency map in [Wang et al., 2016b] focused strongly on this region but was not fine-grained enough to select individual cars. The  $z^+$ -rule, on the other hand, emphasizes all the relevant cars but does not focus on the area in front of the agent. Instead, it also highlights the general course of the road, which is not particularly important in this situation. Our *argmax* approach is the most selective and only highlights the relevant cars inside the area that was also identified by the gradient-based approach.

The second game we trained our agent on is called Space Invaders. In this game, the agent controls a cannon, which can move horizontally along the bottom of the screen, and has to destroy descending waves of aliens. Additionally, the player needs to evade incoming projectiles fired from the aliens or take cover behind three floating obstacles. In contrast to purely reactive games like Enduro, Space Invaders requires the agent to develop long-term strategies, as it has to determine an order in which it destroys the aliens in each wave and also has to decide when to hide behind obstacles. While this does not necessarily imply that the game is harder for an agent to learn, analyzing the trained model might lead to a better understanding of an optimal strategy to solve this game. Figure 6.3 shows a comparison of the two different saliency map approaches for a specific game state of space invaders. Both the  $z^+$ -rule, as well as the *argmax* approach, show that the agent mostly considers the aliens positioned on the outline of the grid as relevant. However, the *argmax* approach does so more clearly by only highlighting aliens on the outline of the grid. This selection makes sense since the other enemies cannot be hit by the agent. Our selective *argmax*-rule further shows that the agent is not paying attention to the obstacles. Given a certain performance level of our model, this suggests that they might not be a

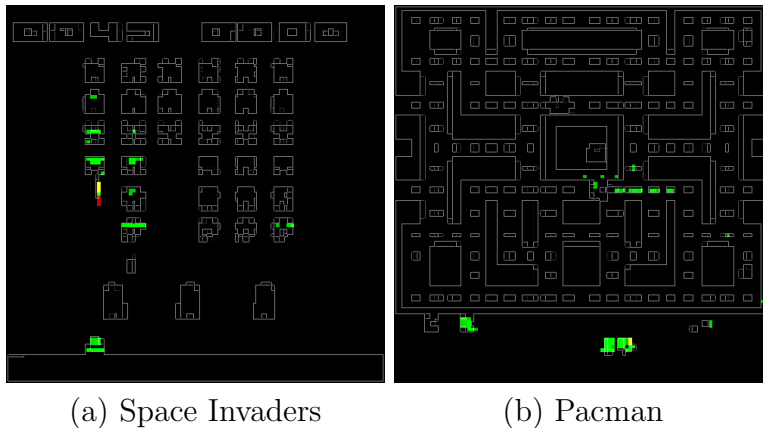


**Figure 6.4.:** The left image (a) shows a screen of Pacman. The player (green circle) has to collect pellets (blue area) while avoiding ghosts (red circles). The saliency map created for this game state by the  $z^+$ -rule (b) highlights a huge area as relevant, while our *argmax* approach (c) focuses on the vicinity of the player.

necessary component of an optimal strategy for Space Invaders. In this way, our selective saliency maps enable us not only to find errors in our model but also to pass on the learned knowledge to human players.

The last game we used to verify our approach is Pacman, where the player has to navigate through a maze and collect pellets while avoiding enemy ghosts (see Section 2.1.2.1 for a more detailed description). Because this game contains many important objects and gives the agent a huge variety of possible strategies, DQN agents struggle in this environment and perform worse than the average human player (see [Mnih et al., 2015]). Explainable AI methods are especially desirable in environments like this, where the agent is struggling, because they help us understand where the agent has difficulties. The saliency maps created by the  $z^+$ -rule (Figure 6.4 (b)) reflect the complexity of Pacman by showing that the agent tries to look at nearly all of the objects in the game. This information might be helpful to optimize the DRL agent, but it also distracts from the areas that influenced the agents’ decision the most. Figure 6.4 (c) shows that the saliency map created by the *argmax* approach is more focused on the vicinity of the agent and makes it clearer what the agent is focusing on the most. Figure 6.4 further illustrates that a fine-granular saliency map in the vicinity of the agent is necessary to see that the agent will most likely decide on moving to the right as his next action.

For the sake of completeness, we want to mention that a similar selective effect can be obtained by using the  $z^+$ -rule and implementing some kind of threshold, for example, only showing the highest 1% of all relevance values. However, this approach comes with its own set of challenges. While a threshold might be suited for one environment, it might be too high or low for other environments,



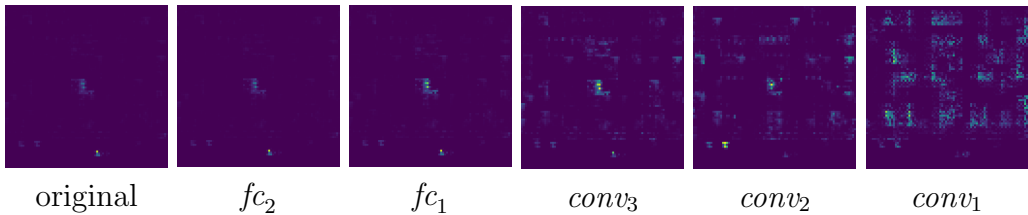
**Figure 6.5.:** Only showing the top 42 relevance values created by the  $z^+$ -rule produces a saliency map (a) which is similar to the one created by our *argmax*-approach for Space Invaders in Figure 6.3(c). Using the same threshold for Pacman (b), we lose some relevant information since, in contrast to 6.4(c), the position of the player is no longer highlighted.

presenting too much or too little information (see, for example, Figure 6.5). Our proposed approach is independent of the environment, which eliminates the need to empirically determine a specific threshold for each new problem. Furthermore, the conservation property of LRP is lost by simply removing relevance values. Therefore, the generated saliency maps are not proportional to the prediction, which makes it harder to compare different saliency maps.

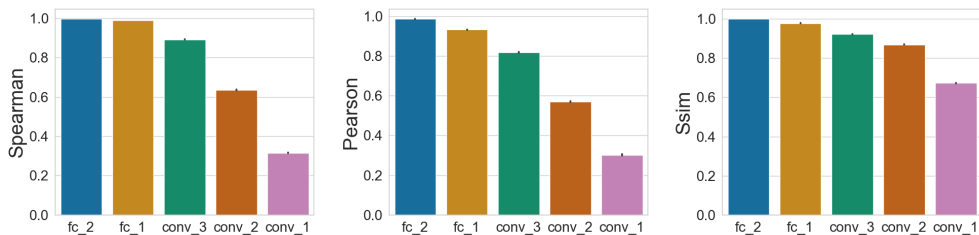
In total, our experiments have shown that our approach can be used on three games, each of which presents a different challenge, and that it generates informative saliency maps that are more selective than the ones generated by the  $z^+$ -rule.

## 6.4. Sanity Checks

As we have seen in Section 3.3.1, it is not yet possible to verify whether a saliency map algorithm perfectly reflects what a model learned. However, a basic prerequisite for this is that the saliency maps depend on the weights learned by the model. To verify this, Adebayo et al. [2018] proposed sanity checks that cascadingly randomize each layer of the network, starting with the output layer. If the saliency maps depend on the learned weights, then this will lead to increasingly different visualizations. Sixt et al. [2020] applied the sanity checks to several LRP variants, but they have never been used on our *argmax*-rule.



**Figure 6.6.:** Example for how the LRP-argmax saliency maps change when the network’s layers are randomized cascadingly, beginning with output layer  $fc_2$ .



**Figure 6.7.:** The average similarities between saliency maps for the fully trained agent and agents where the layers have been randomized cascadingly, starting with the last layer  $fc_2$ . The values are based on a stream of 1000 actions in the Atari 2600 Pacman game.

Therefore, we implemented the sanity checks<sup>1</sup> for our *argmax*-rule and test it on the regular Pacman agents described in Section 10.1. An example of these tests for a single state is shown in Figure 6.6.

To measure how similar two saliency maps are, we use three different metrics proposed by Adebayo et al. [2018]: Spearman rank correlation, structural similarity (SSIM), and Pearson correlation of the histogram of oriented gradients. To account for a possible change of sign in the saliency maps, we adopt an approach by Sixt et al. [2020] and use the maximum similarity of the original and the inverted saliency map. That means that for two saliency maps  $S, S' \in \mathbb{R}^{m \times n \times c}$  and a similarity measurement  $sim : \mathbb{R}^{m \times n \times c} \times \mathbb{R}^{m \times n \times c} \rightarrow \mathbb{R}$  we calculate the actual similarity with

$$\max(sim(S, S'), sim(\mathbf{1} - S, S')) \quad (6.6)$$

where  $\mathbf{1} \in \mathbb{R}^{m \times n \times c}$  is filled with 1s. Figure 6.7 shows the average similarities per randomized layer for a gameplay stream of 1000 states.

The relatively high values for the structural similarity (SSIM) can be explained by the high amount of intersecting zeros in all saliency maps. Apart from that, we see the same trends already observed by Sixt et al. [2020] and Adebayo

<sup>1</sup>The code we used for the sanity checks can be found here: [https://github.com/HuTobias/HIGHLIGHTS-LRP/tree/master/sanity\\_checks](https://github.com/HuTobias/HIGHLIGHTS-LRP/tree/master/sanity_checks)

et al. [2018]: The saliency maps do analyze the learned weights, but the fully connected layers are not sufficiently analyzed. As a consequence, the saliency maps are not class discriminatory. However, class discriminatory saliency maps often come with other drawbacks like being noisy [Sixt et al., 2020] or not analyzing all layers [Selvaraju et al., 2020].

## 6.5. Conclusion

In this chapter, we presented two adjustments to the LRP concept that enable compatibility with state-of-the-art deep reinforcement learning approaches and increase the selectivity of the generated saliency maps while maintaining all desired properties of the original algorithm. For one, we have shown a way to use LRP on the Dueling DQN architecture, which makes it possible to use LRP on all current versions of the DQN algorithm. Secondly, we introduced an adjustment to an existing LRP variant, which generates saliency maps that focus more on the important objects inside the input image.

We tested our approach on three different Atari 2600 games and verified that the saliency maps generated by our system are more selective than the ones created by existing LRP methods while still including the information expected from visual explanations. Since this selectiveness is an important property of inter-human explanations, we argue that our approach might prove beneficial when it comes to explaining the actions of a trained agent to people without a machine-learning background.

## 7. GANterfactual-RL: Counterfactual Explanations for RL Agents with Visual Input

Counterfactual explanations are a common tool to explain artificial intelligence models (see Section 3.2.2). For Reinforcement Learning (RL) agents, they answer “Why not?” or “What if?” questions by illustrating what minimal change to a state is needed such that an agent chooses a different action. In other machine learning domains, such as image classification, counterfactual explanations are already frequently used (see Section 3.2.2).

However, generating counterfactual explanations for RL agents with visual input is especially challenging. Because of the large visual state spaces, counterfactual generation approaches that utilize optimization at runtime are often too slow, and because DRL agent actions are part of an overarching policy, the counterfactuals must account for the long-term consequences of the changes to the original state. Furthermore, for RL agents, there is no direct counterpart to the training datasets used by supervised models. Therefore, counterfactual explanation approaches for supervised models that utilize the training data cannot be applied to RL agents without adjustment [Wells and Bednarz, 2021].

Due to the difficulties mentioned above, there is only one approach that focuses on creating counterfactual explanations for deep RL agents with visual input (see Section 4.1.2). This Counterfactual State Explanation (CSE) approach by Olson et al. [2021] utilizes a complex combination of models where the final generator is only indirectly trained to change the action.

This chapter proposes a novel method for generating counterfactual explanations for RL agents with visual input. It is based on our publication:

**Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS. IFAAMAS*, 10 pages

We formulate the counterfactual generation problem as a domain transfer problem where the domains are represented by sets of states that lead the agent to different actions. Our approach is fully model-agnostic, easier to train than the CSE approach presented by Olson et al., and includes the counterfactual actions more directly into the training routine by solving an action-to-action domain transfer problem. We evaluate our approach with computational metrics (e.g., how often do the counterfactuals change the agent’s decision) using the ALE (Section 2.1.2). We also conducted a user study that will be described in Chapter 12.

As such, the contributions of this chapter are as follows:

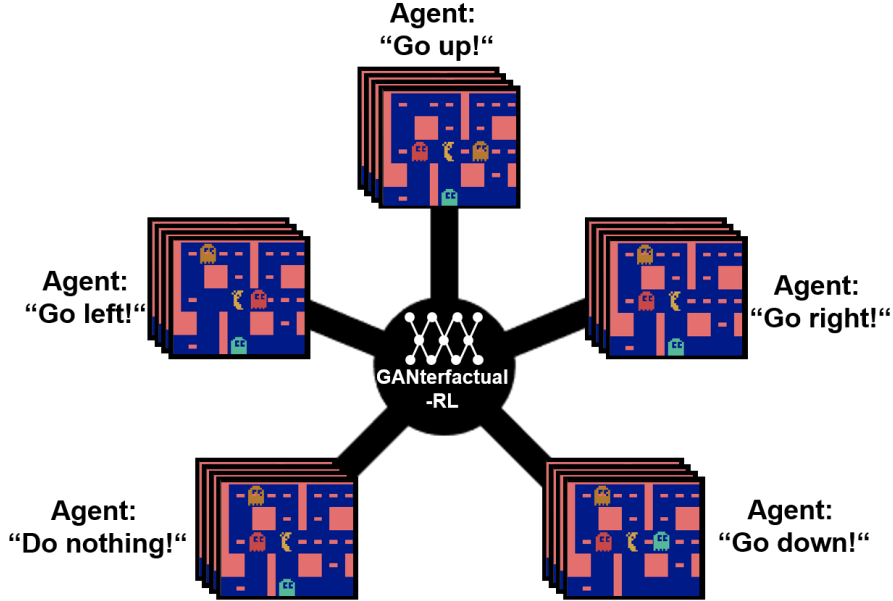
- We formulate a novel, model-agnostic approach for generating counterfactual explanations for RL agents.
- We demonstrate that our approach outperforms the previous method in several computational metrics.

## 7.1. Approach

### 7.1.1. The GANterfactual-RL Approach

As we have seen in Section 2.1.1, RL agents are usually employed in an MDP which consists of states  $s \in \mathcal{S}$ , actions  $a \in \mathcal{A}$ , and rewards  $r$ . Given a state  $s$ , the goal of an RL agent  $\pi : \mathcal{S} \rightarrow \mathcal{A}$  is to choose an action  $\pi(s)$  that maximizes its cumulative future rewards. To explain such an agent, the objective of a counterfactual explanation approach for RL agents is defined as follows (for more details, see Section 3.2.2): Given an original state  $s$  and a counterfactual action  $a'$ , we want a counterfactual state  $s'$  that makes the agent choose the counterfactual action  $\pi(s') = a'$ . Hereby, the original state  $s$  should be altered as little as possible. On an abstract level, the action  $\pi(s)$  that the agent chooses for a state  $s$  can be seen as a top-level feature that describes a combination of several underlying features that the agent considers to be relevant for its decision. Thus, the counterfactual state  $s'$  should only change the features that are relevant to the agent’s decision while maintaining all other features not relevant to the decision. This is similar to image-to-image translation, where features that are relevant for a certain image domain should be transformed into features leading to another image domain, while all other features have to be maintained (e.g., the background should remain constant when transforming horses to zebras). Taken together, we can formulate the generation of counterfactual states for RL agents as a domain transfer problem similar to image-to-image translation: The agent’s action space  $\mathcal{A}$  defines the different domains  $\mathcal{A}_i = \{s \in \mathcal{S} | \pi(s) = a_i\}$ ,





**Figure 7.1.:** Schematic of our counterfactual generation approach. We formulate the problem as domain transfer, where each domain represents an action. States are assigned to domains based on the action that the agent chooses for them.

where each state belongs to the domain that corresponds to the action that the agent chooses for this state (see Figure 7.1).

To solve the reformulated domain transfer problem, we base our system on the StarGAN architecture [Choi et al., 2018] since RL agents usually use more than two actions. The StarGAN architecture incorporates multiple loss components that can be reformulated to be applicable to the RL domain. The first component, the so-called adversarial loss, leads the network to produce highly realistic states that look like states from the original environment. Reformulated for the task of generating RL states, we define it as follows (following Choi et al. [2018] we use a Wasserstein objective with gradient penalty):

$$\begin{aligned} \mathcal{L}_{adv} = & \mathbb{E}_s [D_{src}(s)] - \mathbb{E}_{s,a'} [D_{src}(G(s, a'))] \\ & - \lambda_{gp} \mathbb{E}_{\hat{s}} [(\|\nabla_{\hat{s}} D_{src}(\hat{s})\|_2 - 1)^2], \end{aligned}$$

where  $D_{src}$  is the StarGAN’s *discriminator* network and  $G$  its *generator* network. The second loss component, which is specific to the StarGAN architecture, guides the generator network to produce states that lead to the desired counterfactual actions. It consists of two sub-objectives, one that is applied while the network is fed with original (*real*) states from the training set (Equation 7.1), and the other while the network is generating counterfactual states (Equation

7.2):

$$\mathcal{L}_{cls}^a = \mathbb{E}_{s,a}[-\log D_{cls}(a|s)], \quad (7.1)$$

$$\mathcal{L}_{cls}^{a'} = \mathbb{E}_{s,a'}[-\log D_{cls}(a'|G(s, a'))], \quad (7.2)$$

where  $D_{cls}$  refers to the StarGAN discriminator’s classification output, which learns to approximate the action that the agent is performing in a particular state. Further, as counterfactual states should be as close to the original states as possible, a *Reconstruction Loss* is used. This loss forces the network to only change features that are relevant to the agent’s choice of action:

$$\mathcal{L}_{rec} = \mathbb{E}_{s,a,a'}[\|s - G(G(s, a'), a)\|_1]$$

Taken together, the whole loss of the StarGAN architecture, reformulated for RL counterfactual explanations, is defined as follows:

$$\begin{aligned} \mathcal{L}_D &= -\mathcal{L}_{adv} + \lambda_{cls} \mathcal{L}_{cls}^a \\ \mathcal{L}_G &= \mathcal{L}_{adv} + \lambda_{cls} \mathcal{L}_{cls}^{a'} + \lambda_{rec} \mathcal{L}_{rec} \end{aligned}$$

where  $\lambda_{cls}$  and  $\lambda_{rec}$  are weights controlling the corresponding loss component’s relevance. Since our approach utilizes a GAN architecture to generate counterfactuals for RL agents, we refer to it as GANterfactual-RL.

### 7.1.2. Dataset Generation

As described above, our GANterfactual-RL approach relies on training data in the form of state-action pairs. Olson et al. [2021] train their CSE approach on state-action pairs generated by concurrently running an MDP with a trained agent (see Section 4.1.2). This strategy is simple but allows for little control over the training data, which can lead to the following complications:

- Frames extracted from a running MDP contain a temporal pattern since consecutive states typically have a high correlation. Such correlations and patterns can lead to bias and sub-optimal convergence during training.
- For episodic MDPs, there is a high probability of reaching the same state throughout several episodes. This is amplified by the fact that RL agents often learn to execute only a few optimal trajectories. This results in duplicate samples that are effectively over-sampled during training.
- RL agents generally do not execute each action equally frequently since most environments contain actions that are useful more often than others. This leads to an imbalanced amount of training samples per domain.

To mitigate the aforementioned issues, we propose to generate datasets as follows: Data is gathered by running a trained agent in an MDP. Each state corresponds to one dataset sample and is labeled with the action that the agent chooses to execute in this state. An  $\epsilon$ -greedy policy ( $\epsilon=0.2$  in our case) is used to increase the diversity of states reached over multiple episodes. State-action pairs with an explored (randomly chosen) action are not added to the dataset. After the data is gathered, duplicates are removed. Then, a class balancing technique (under-sampling in our case) is used to account for over- or underrepresented actions. Finally, the dataset is split into a training set, a test set, and potentially a validation set.

Most of these techniques are commonly used in other application domains of machine learning. However, to the best of our knowledge, this is the first work to generate and preprocess datasets for generating counterfactual explanations for RL agents.

### 7.1.3. Application to the Atari Domain

**Environment.** The environments we use for our experiments are the Atari 2600 games Pacman and Space Invaders, included in the ALE (see Section 2.1.2). However, we crop the raw input frames so that only the actual playing field remains. This removes components such as the score and life indicators that would allow participants of the user study, which will be described in Chapter 12, to easily see which agent receives higher scores. After that, we use the same preprocessing as described in Section 2.1.2. Two steps from this preprocessing are particularly important for us. First, the frames are gray-scaled and downsized. Second, in addition to the current frame, the agent receives the last three preprocessed frames as input. This allows the agent to detect temporal relations. The ALE actions normally correspond to the meaningful actions achieved with an Atari 2600 controller (e.g., six actions for Space Invaders). Since we wanted to use our Pacman agents in a user study (see Chapter 12), we removed four redundant actions (e.g., *Up & Right*) whose effect differs between situations and is therefore hard to convey to participants. This left us with five actions for Pacman (*Do nothing, Up, Down, Left, Right*).

**Agent Training.** To evaluate participants' ability to differentiate between alternative agents and analyze their strategies, we modified the reward function of three Pacman agents. This is a more natural method of obtaining different agents compared to withholding information from the agent as Olson et al. [2021] did. Furthermore, it results in agents that behave qualitatively differently. Therefore, participants in the user study, which will be described in Section 12, have to actually analyze the agents' strategies instead of simply looking for objects that the agents ignore.

- **Blue-Ghost Agent:** This agent was trained using the default reward function of the ALE, where blue ghosts get the highest reward.
- **Power Pill Agent:** This agent only received positive rewards for eating power pills.
- **Fear-Ghost Agent:** This agent got a small positive reward of 1 for every step in which it did not die to ghosts.

For training the first two Pacman agents, we use the DQN algorithm (see Section 2.2). Each agent was trained for 5 Million steps. The *fear-ghosts agent* was trained using the ACER algorithm [Wang et al., 2016a] for 10M steps. At the end of the training period, the best-performing policy is restored. For all three agents, we build upon the OpenAI baselines [Dhariwal et al., 2017] repository. For Space Invaders, we used the two Asynchronous Advantage Actor-Critic (A3C) agents trained by Olson et al. [2021]. For the training details, we refer to their paper. One agent is trained normally, while the other agent is flawed and does not see the laser cannon at the bottom of the screen.

**GANterfactual-RL on Atari.** To generate human-understandable counterfactual explanations for our Atari agents, the generated counterfactual states should represent the frames that humans see during gameplay. That means we cannot train our GANterfactual-RL model on the preprocessed and stacked frames that the Atari agents use. Instead, we train it on the cropped RGB frames before preprocessing. The only preprocessing we still use on those frames is the countermeasure against flickering objects in Atari games, which was proposed by Mnih et al. [2015] and is described in Section 2.1.2. While generating the dataset, we only save the most recent of the four stacked frames for each state  $s$ . This frame generally influences the agent’s decision the most. For feeding the counterfactual frame back into the agent (e.g., to evaluate the approach), we stack it four times and then apply preprocessing.

Our implementation details can be found in Appendix D.1. The full code is available online.<sup>1</sup>

## 7.2. Computational Evaluation

### 7.2.1. Used Metrics

We evaluate our approach using the metrics *validity* (or *success rate*), *proximity* (or *cost*), *sparsity*, and *generation time*. We consider these metrics to be the

---

<sup>1</sup><https://github.com/hcmlab/GANterfactual-RL>

most suitable and widely used metrics for image-based counterfactual explanations [Chen et al., 2021; Pawelczyk et al., 2021; Keane et al., 2021; Mothilal et al., 2020].

**Validity** captures the rate of CounterFactuals (CFs) that actually evoke the targeted action when fed to the agent. With  $N_T$  being true CFs (correctly changing the agent’s action),  $N_F$  being false CFs, and  $N$  the total amount of evaluated CFs, this metric is defined as:

$$Validity = \frac{N_T}{N_T + N_F} = \frac{N_T}{N}$$

**Proximity** measures the similarity between an original state image and its CF via the  $L1$ -norm. We normalize the metric to measure the proximity in the range  $[0, 1]$ .

$$Proximity(s, G) = 1 - \frac{1}{255 \cdot \mathbb{S}} \|s - G(s, a)\|_1$$

where  $s$  is the original state image,  $G(s, a)$  is the generated CF for an arbitrary target action domain  $a$  and  $\mathbb{S}$  is the domain of color values of  $s$  ( $\mathbb{S} = 3 \cdot Width \cdot Height$  for RGB-encoded images). The normalization with  $255 \cdot \mathbb{S}$  assumes an 8-bit color encoding with color values in range  $[0, 255]$ . High proximity values are desirable since they indicate small adjustments to the original state.

**Sparsity** quantifies the number of unmodified pixel values between an original state image and its CF via the  $L0$ -norm (a pseudo-norm that counts the number of non-zero entries of a vector/matrix). The sparsity is normalized to the range  $[0, 1]$  as well.

$$Sparsity(s, G) = 1 - \frac{1}{\mathbb{S}} \|s - G(s, a)\|_0$$

A completely altered image has a sparsity of 0, and an unmodified image has a sparsity of 1. High sparsity values are thus desirable.

**Generation time** determines the time it takes to generate one CF with a trained generator, not including pre- or post-processing.

## 7.2.2. Computational Results

The computational results for the three Pacman agents are shown in Table 7.1 and the results for the two Space Invaders agents in Table 7.2. Appendix D.1 describes the training details for the GANterfactual-RL and CSE models used in the evaluation. To create evaluation test sets for the Pacman agents, we took the fully cleaned training datasets (section 7.1.2) and reserved 10% of each action for the test sets. To show the contribution of our proposed dataset generation, we additionally trained a GANterfactual-RL model for the *blue-ghost agent* without the steps proposed in Section 7.1.2 and evaluated it on the test set from the clean dataset. This dropped the validity to 0.45 and sparsity

**Table 7.1.:** Computational evaluation results for the Pacman agents. *Proximity, sparsity and generation time* are specified by *mean  $\pm$  standard deviation*.

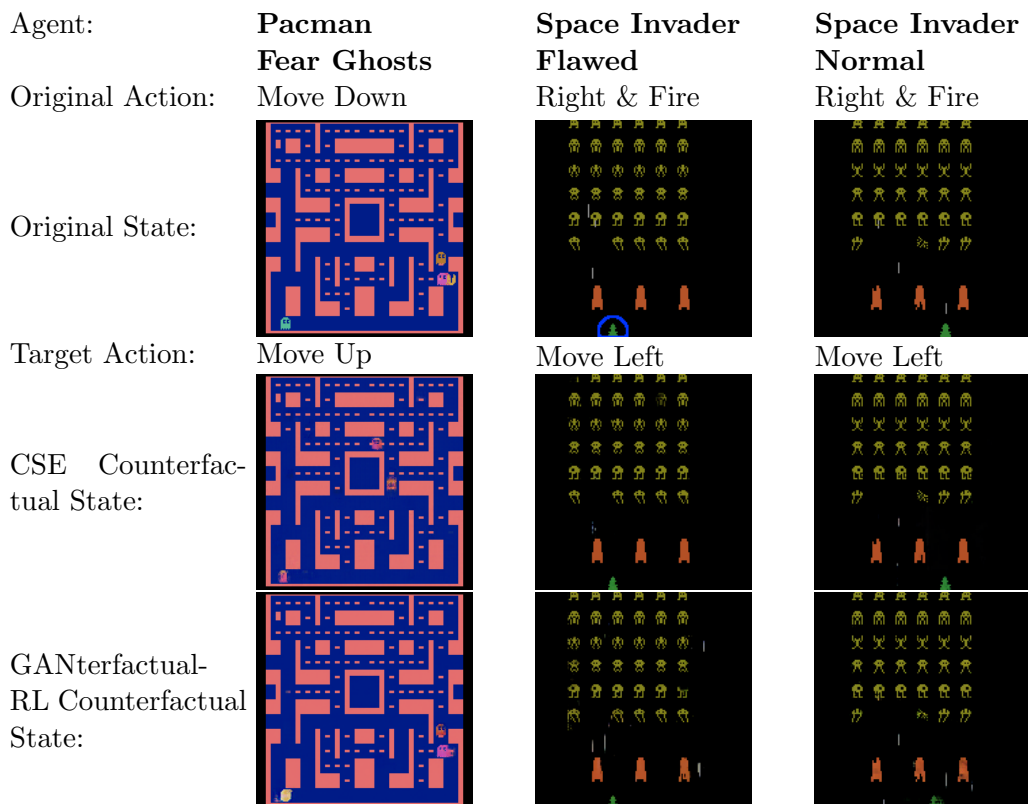
<i>Approach</i>	<i>Validity</i> ( $\uparrow$ )	<i>Proximity</i> ( $\uparrow$ )	<i>Sparsity</i> ( $\uparrow$ )	<i>Gen. Time [s]</i> ( $\downarrow$ )
<i>Blue-Ghost Agent</i>				
Ours	0.59	$0.997 \pm 0.001$	$0.73 \pm 0.02$	$0.011 \pm 0.012$
CSE	0.28	$0.992 \pm 0.002$	$0.33 \pm 0.03$	$0.085 \pm 0.021$
<i>Power-Pill Agent</i>				
Ours	0.49	$0.997 \pm 0.001$	$0.70 \pm 0.02$	$0.011 \pm 0.008$
CSE	0.20	$0.993 \pm 0.002$	$0.32 \pm 0.02$	$0.566 \pm 0.731$
<i>Fear-Ghost Agent</i>				
Ours	0.46	$0.995 \pm 0.001$	$0.45 \pm 0.01$	$0.013 \pm 0.014$
CSE	0.20	$0.992 \pm 0.002$	$0.32 \pm 0.04$	$0.020 \pm 0.017$

**Table 7.2.:** Computational evaluation results for the Space Invaders agents. *Proximity, sparsity and generation time* are specified by *mean  $\pm$  standard deviation*.

<i>Approach</i>	<i>Validity</i> ( $\uparrow$ )	<i>Proximity</i> ( $\uparrow$ )	<i>Sparsity</i> ( $\uparrow$ )	<i>Gen. Time [s]</i> ( $\downarrow$ )
<i>Normal Agent</i>				
Ours	0.70	$0.998 \pm 0.002$	$0.97 \pm 0.02$	$0.011 \pm 0.013$
CSE	0.18	$0.995 \pm 0.003$	$0.89 \pm 0.05$	$6.180 \pm 9.727$
<i>Flawed Agent</i>				
Ours	0.53	$0.998 \pm 0.002$	$0.96 \pm 0.01$	$0.011 \pm 0.015$
CSE	0.17	$0.995 \pm 0.004$	$0.94 \pm 0.01$	$0.020 \pm 0.035$

to  $0.50 \pm 0.01$  while the other values stayed comparable. To be more comparable to the results by Olson et al. [2021], we do not remove duplicates from the Space Invaders datasets and do not apply class balancing. Here, we create the test set by sampling 500 states for each action and removing all duplicates of these states from the training set. Our GANterfactual-RL approach outperforms the CSE counterfactuals in every single metric.

Figure 7.2 shows example counterfactuals generated for the Pacman *fear-ghosts agent* and the two Space Invaders agents. Additional examples for all our agents can be seen in Appendix D.4.



**Figure 7.2.:** Example counterfactual states. Our approach does not change the *Laser Cannon* (marked in blue) for the flawed agent, who does not see it, but changes it for the normal agent.

### 7.3. Discussion

Our computational evaluation shows that our proposed approach is correctly changing the agents' actions in 46% to 70% of the cases, depending on the agent. While this is not perfect, one has to consider that this is not a binary task but that the agents have five or six different actions. Furthermore, CSE [Olson et al., 2021], the only previous method that focuses on generating counterfactual explanations for RL agents, only successfully changed the agent's decision in 17% to 28% of the cases. We can think of two reasons for the low validity values for the CSE approach. First, they only incorporate the agent's action in their loss functions related to the latent space. The generation of the final pixels did not include constraints to faithfully ensure that a specific action was taken by the agent. Second, their loss functions for the latent space focus on creating action-invariant states. Olson et al. [2021] showed that their CSE approach was useful for differentiating between a normal agent and a flawed agent. We think this is due to the fact that CSE is good at generating action-invariant

states. This can help to identify the object that the flawed agent did not see since irrelevant objects are not changed for action-invariant states. We found that our approach also does not change the irrelevant object for the flawed agent (illustrated in Figure 7.2). This demonstrates that the counterfactuals generated by our approach are similarly effective for identifying the flawed agent. Looking at the distance between the original and the counterfactual states in pixel-space, we see that counterfactual states generated by our GANterfactual-RL approach, on average, have less distance to the original states and change fewer pixel values compared to the counterfactuals generated by the previous CSE method by Olson et al. [2021]. This indicates that our GANterfactual-RL method is better at achieving the goal of finding the smallest possible modification of the original state to change the agent’s decision. Since our method only requires a single forward pass to generate a counterfactual state, it is faster than the CSE method, which relies on potentially time-consuming gradient descent for the counterfactual generation.

## 7.4. Conclusion

In this chapter, we formulated a novel method for generating counterfactual explanations for RL agents. This GANterfactual-RL method is fully model-agnostic, which we demonstrate by applying it to three RL algorithms, two actor-critic methods, and one deep Q-learning method. Using computational metrics, we show that our proposed method is better at correctly changing the agent’s decision while modifying less of the original input and taking less time than the only previous method that focuses on generating visual counterfactuals for RL.

While there is still room for improvement, we can confidently say that our approach was the state-of-the-art for counterfactual explanations for RL agents with visual input at the time of its publication.



## 8. Combining Local and Global Explanations for Agent Behavior

Explainable reinforcement learning methods can roughly be divided into local explanations that analyze specific decisions of the agents (see Section 4.1) and global explanations that convey the general strategy of the agents (Section 4.2). In this chapter, we explore the combination of global and local explanations describing RL agent policies. Parts of this chapter build upon three of our previous publications:

**Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

Yael Septon, **Tobias Huber**, Elisabeth André, and Ofra Amir [2023]. “Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection - 21st International Conference*. Vol. 13955. Lecture Notes in Computer Science. Springer, pp. 320–332. DOI: 10.1007/978-3-031-37616-0\_27

**Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS. IFAAMAS*, 10 pages

The motivation for integrating the two approaches is their complementary nature: while local explanations can help users understand what information the agent attends to in specific situations, they do not provide any information about its behavior in different contexts. This is reinforced by a previous study conducted by Alqaraawi et al. [2020], who evaluated local explanations and came to the conclusion that sole instance-level explanations are not sufficient and should be augmented with global information. Conversely, for global explanations, we have seen in Section 4.2 that showing users representative examples of the agent’s actions provides them with a sense of the overall strategy

of the agent. However, this approach does not provide any explanations as to what information the agent considered when choosing how to act in a certain situation.

Zahavy et al. [2016] proposed a combination of example-based t-SNE explanations and saliency maps to address this problem. However, they focused on the global information obtained through t-SNE and did not evaluate the combined and individual benefits of the two approaches. Furthermore, as we have seen in Section 4.2.3, t-SNE visualizations are best suited for users with a machine learning background.

In this dissertation, we combine the complementary benefits of local and global explanations by integrating local explanations with global strategy summaries. Specifically, we adapt the HIGHLIGHTS algorithm for generating strategy summaries (Section 4.2.3.1) such that it can be applied to deep learning settings, and integrate it with several local explanation methods. To demonstrate that our method is easily adaptable to a variety of local explanation algorithms, we use three different local explanation methods. In particular, we use the post-hoc saliency map and counterfactual explanation methods described in chapters 6 and 7, and the intrinsic reward decomposition approach described in Section 4.1.3.1. We combine these approaches with HIGHLIGHTS by adding the local explanations, which provide information about the agent’s reasoning in specific states, to the summary generated by HIGHLIGHTS.

As such, this chapter makes the following two contributions:

- It proposes a joint local and global explanation approach for RL agents by integrating local explanations and HIGHLIGHTS summaries.
- It demonstrates that the HIGHLIGHTS algorithm, which was so far only used on classic reinforcement learning, can be applied to deep reinforcement learning agents with slight adjustments.

The remainder of this chapter is structured as follows: first, we will describe the combination of saliency maps and HIGHLIGHTS in the form of videos. Then, we will look into combining reward decomposition and counterfactual explanations, respectively, with HIGHLIGHTS in an interactive image-based interface since these local methods are not suitable for videos.

## 8.1. Combining Saliency Maps and HIGHLIGHTS as Video

As the first local explanation method, we opted to combine HIGHLIGHTS with saliency maps since this integration seemed the most natural to us. Saliency maps are typically displayed as heatmaps that are overlaid on the original state.

This makes it straightforward to add the saliency maps to the videos generated by HIGHLIGHTS. The approaches are also similar in nature, as saliency maps reveal which pixels were important locally, while HIGHLIGHTS shows what states were important globally.

Regarding the choice of a specific saliency map method, we chose the LRP-argmax approach described in Chapter 6. This choice was based on two main reasons. First, LRP-argmax is designed to selectively identify the most crucial information, an essential feature for video presentations where each frame only appears for a short duration. Second, LRP offers beneficial properties, such as preserving the agent’s confidence levels (in our case, Q-values), as described in Section 3.2.1.3.

**Generating Gameplay Streams and Saliency Maps.** To test our approach, we use DQN agents (see Section 2.2) that are trained to play the game Pacman (see Section 2.1.2.1).

Deep neural networks increase the time that the agent needs for each prediction compared to the traditional Q-learning agents used by Amir and Amir [2018]. Furthermore, the LRP analysis of the agent’s decision requires additional computation time. Therefore, we do not create the HIGHLIGHTS summaries online. Instead, we record a stream of 10,000 steps for each agent and use them to create our summaries. These streams also increase the reproducibility of our experiments<sup>1</sup>.

Since the Atari 2600 version of Pacman does not respond to input for the first 250 frames (empirically tested) after the game starts, we exclude those frames from the streams. Furthermore, we force the agent to repeat the *do nothing* action for a random amount of steps between 0 and 30 until it is allowed to choose actions based on its policy. This method introduces randomness into the deterministic Pacman game and is also used during training by the DQN algorithm (see Section 2.1.2).

The saliency maps are created using the LRP-argmax algorithm described in Section 6.1 and saved separately. They will be added to the states in a later step.

**Adjustments to HIGHLIGHTS.** For the summaries, we make several adjustments to the HIGHLIGHTS algorithm described in Section 4.2.3.1 to adapt it to the DQN settings. First, we change the way importance is calculated. Instead of using Equation 4.1, which calculates the importance by comparing the highest with the lowest Q-value, we use the difference between the highest and second-highest Q-values. Let *second-highest* be the operation that finds the

---

<sup>1</sup>Since the streams are fairly big, we did not upload them. They are available upon request from the author.

second-highest value in a set, then this can be written as:

$$I(s) = \max_a Q_{(s,a)}^\pi - \text{second-highest}_a Q_{(s,a)}^\pi \quad (8.1)$$

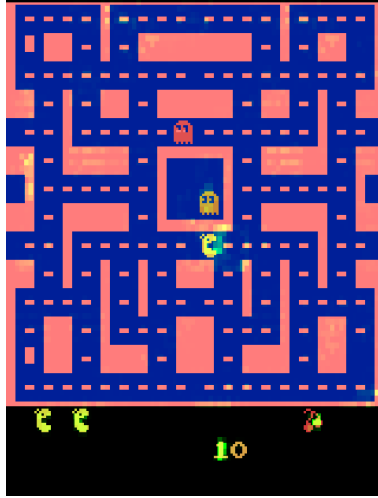
While examining the gap between the best and worst actions worked well in a simpler Pacman environment in which there were only four possible actions, it did not generalize well to the Atari environment, where there is a larger number of actions. One possible explanation for this is that some of the nine actions of the Pacman environment overlap. For example, *left* and *up-left* can be used interchangeably in many states. Therefore, the agent might ignore some of the actions completely. To verify this, we examined the frequency of choosing each action and found that two of the three agents we trained were clearly biased against certain actions.<sup>2</sup> Therefore, some Q-values are largely uninformed by exploration and might have arbitrarily low values, making the worst Q-value non-informative. For the diversity computation in HIGHLIGHTS, we use Euclidean distance over the  $84 \times 84 \times 4$  input states.

Since we pre-generate a stream of 10,000 states, we implement an offline version of HIGHLIGHTS that selects the states for the summary retrospectively from the generated stream. The procedure begins by sorting the states based on their importance scores (Equation 8.1) and adding them to the summary according to this ordering. To reduce the number of overlapping trajectories, we compare each new state with all states in the current summary and corresponding context states (this is equivalent to the HIGHLIGHTS-DIV part of HIGHLIGHTS described in Section 4.2.3.1). To find a suitable threshold that determines when a state is too similar to the states that were already selected for the summary, we randomly pick a subset of 1,000 random states from the recorded stream and calculate the similarity between each pair of states in this set. Then, we set the threshold to be a percentile of the distribution of those similarity values. We empirically found (by manually examining a sample of states) that using a threshold of 3% led to no obvious duplicate trajectories for any of the agents.

**Video Generation.** The videos we generate from the states chosen by the summary show 30 frames per second. To emphasize that demonstrations show different trajectories, they are separated by a black screen that appears for 1 second (inspired by the fade-out effect used in [Sequeira and Gervasio, 2020]). To prevent participants in our user study (Chapter 10) from using the in-game score to gauge how good an agent is, we mask the bottom half of the screen with black pixels. In pilot studies, participants complained that the videos were

---

<sup>2</sup>The results can be seen in [https://github.com/HuTobias/HIGHLIGHTS-LRP/tree/master/action\\_checks](https://github.com/HuTobias/HIGHLIGHTS-LRP/tree/master/action_checks)



**Figure 8.1.:** An example frame from the videos that combine HIGHLIGHTS and saliency maps. The Pacman screens selected by HIGHLIGHTS are overlaid with green saliency maps that show the most relevant pixels for the agent. During our user study in Chapter 10, the bottom of the screen is masked with black pixels to prevent participants from assessing the agents’ capabilities based on their score or remaining lives.

flickering too much. One of the reasons for this is that the Atari 2600 implementation of Pacman does not show every object in every frame to save computing power. Since we showed all frames consecutively, these objects appeared to be blinking and distracted the viewers. To combat this problem, we do not display the current frame  $f_i$ . Instead we display  $\max(f_i, f_{i-1})$ , the maximum of each pixel over the current frame  $f_i$  and the preceding frame  $f_{i-1}$ . This is similar to what Mnih et al. [2015] do during their preprocessing (see Section 2.1.2). While this procedure introduces some artifacts (e.g., red pellets showing through blue ghosts), it considerably reduces the flickering.

Another measure we take against this flickering is to interpolate between the different saliency maps instead of showing a completely different saliency map for each frame. Let  $f_1$  to  $f_4$  be the four stacked frames of an input state (see Section 2.1.2) and let  $s_1$  to  $s_4$  be the saliency maps for each of these frames that analyze the agent’s decision in this state. For  $i < 4$ , the action that Pacman will take after frame  $f_i$  is not related to the saliency map  $s_i$  since the agent only decides on a new action every four frames and is still repeating the action that he decided on based on the last state (composed of the four frames before  $f_1$ ). Therefore, we show the saliency map  $s_4$  over the frame  $f_4$ , and for the other frames ( $i < 4$ ) we interpolate between the last shown saliency map and  $s_4$ .

Before this interpolation, we normalize the saliency maps to have a maximum of 1 and a minimum of 0. We do this across all four frames of the states

$s_1, \dots, s_4$  to avoid losing information that might be transported in the magnitude of relevance values between the frames.

Finally, we add the interpolated saliency maps to the green channel of the original screen frame. Figure 8.1 shows an example frame. Our complete implementation can be found here: <https://github.com/HuTobias/HIGHLIGHTS-LRP>

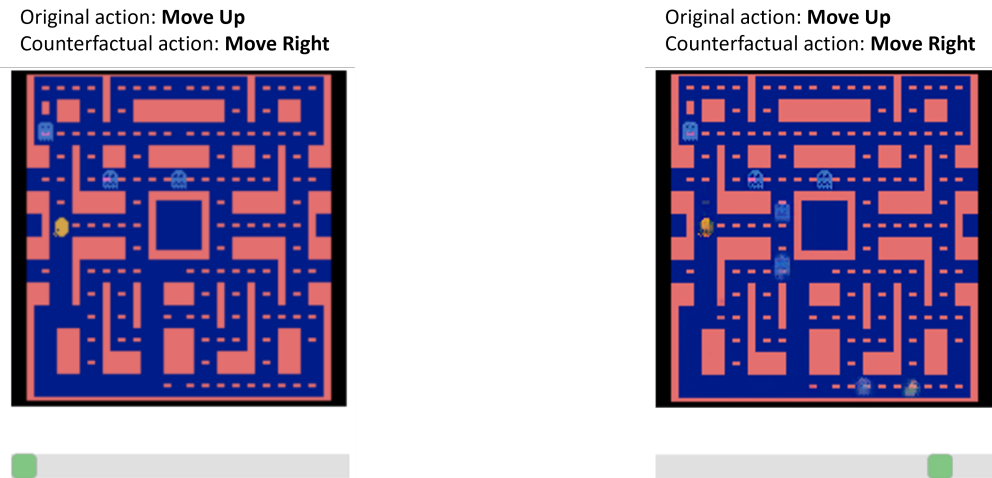
## 8.2. Combining HIGHLIGHTS and Local Explanations as Interactive Images

While saliency maps are one of the most common and successful explanation methods for neural-network-based DRL agents, they also have drawbacks. For example, saliency maps only show *what* information was important and not *why* that information was important.

Therefore, we want to extend our combined local and global explanation approach to other local explanation methods that might alleviate some of the drawbacks of saliency maps. In a user study by Mertes et al. [2022a], for example, counterfactual explanations were better than saliency maps at showing users *why* certain regions were relevant to a classifier.

However, many local explanation methods, such as counterfactuals, cannot be easily integrated into HIGHLIGHTS videos since they require more time for user interpretation (e.g., users have to comprehend what has changed between the original and the counterfactual state). To accommodate such local explanations, we propose an interactive summary based on HIGHLIGHTS states. To this end, we show only the most important states according to HIGHLIGHTS without adding their context states. Apart from that, we use HIGHLIGHTS based on streams exactly as described in the previous Section 8.1. Depending on the length  $k$  of the summary, this gives us  $k$  states that we can display individually, e.g., as images. While this removes the context information from the HIGHLIGHTS summaries, it reduces the cognitive load and allows more freedom in designing the presentation of the local explanations. To mimic videos where users can switch between different scenes, we design the summaries interactively by allowing the user to freely switch between the individual states (shown by the buttons in Figure 8.3).

In the following two subsections, we will see how such an interactive summary can look like with local counterfactual explanations and reward decomposition.

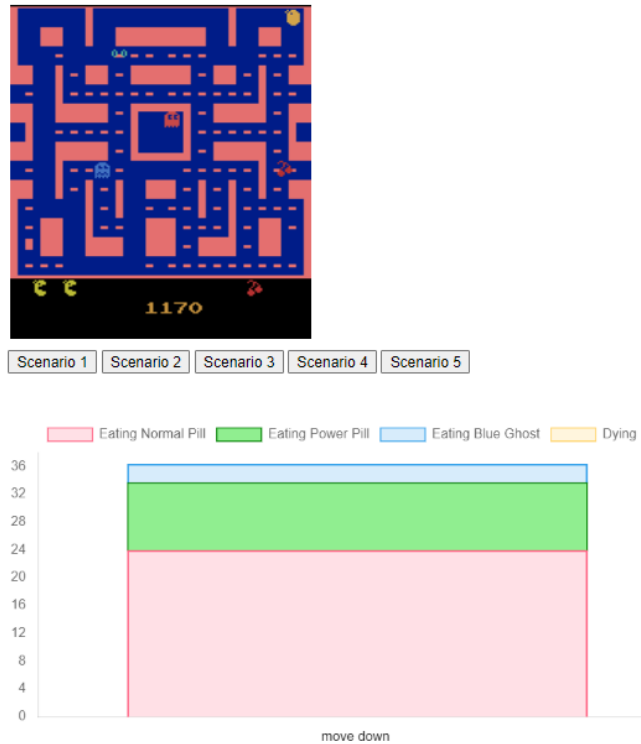


**Figure 8.2.:** Example for the presentation of counterfactual explanations for an individual HIGHLIGHTS state. As the slider beneath the state is moved, a smooth transition occurs, linearly interpolating from the original state (on the left) to the counterfactual state. The image to the right demonstrates the state being approximately 90% transitioned towards the counterfactual scenario.

### 8.2.1. Integrating Counterfactual Explanations and HIGHLIGHTS

As a first step towards interactive summaries that accommodate complex local explanations, we start by describing the integration of HIGHLIGHTS-DIV with counterfactual explanations, such as the ones generated by the GANterfactual-RL approach presented in the previous Chapter 7. The presentation of the counterfactual explanations is designed as follows. For each HIGHLIGHTS state, we generate a single counterfactual state. We were concerned that too many counterfactual states would cause too much cognitive load. To generate meaningful counterfactual instances, we limited the counterfactual action to turning around in a corridor and randomly selecting a new direction at an intersection (do not turn around). This was done because of the way that the MsPacman version of Pacman is implemented: actions that do nothing or move directly into a wall are ignored.

Inspired by Mertes et al. [2022a], the generated counterfactual states are presented by a slider under each state. Moving the slider from left to right linearly interpolates the original state to the counterfactual state (*per-pixel interpolation*). The original and counterfactual actions are written above the state. Figure 8.2 shows an example of the presentation of counterfactual explanations for a single HIGHLIGHTS state.



**Figure 8.3.:** A screenshot of the combination of strategy summaries with reward decomposition on the game Pacman. The upper part of the image shows a specific state extracted from an agent’s behavior. The bottom part shows the reward bars corresponding to the state shown above. The x-axis displays the chosen action, and the y-axis shows the Q-value. In this case, it can be observed that “eating normal pill” is the largest reward component affecting the behavior of this agent in this state. Users can switch to different states by selecting a scenario from the list. The states (scenarios) are chosen based on the strategy summary method (e.g., HIGHLIGHTS).

### 8.2.2. Integrating Reward Decomposition and HIGHLIGHTS

Both saliency maps and counterfactual explanations are post-hoc explanations that aim to explain a black-box agent. For our final combination, we want to explore an intrinsic explanation method that increases the explainability of DRL agents. To this end, we augment HIGHLIGHTS with reward decomposition (Section 4.1.3.1). We chose to combine these two types of explanations because we believe they complement each other. Reward decomposition reflects the intentions of an agent, while HIGHLIGHTS gives a broader perspective on the agent’s decisions. For each state that was chosen by the HIGHLIGHTS algorithm, we create reward decomposition bars that depict the decomposed Q-



values for actions in the chosen state (see Figure 8.3). Similar to counterfactuals, the reward decomposition bars vary for each state and need to be interpreted by the user. Therefore, when integrating the two methods, we used HIGHLIGHTS to extract the important states but displayed them using static images rather than videos. As described at the beginning of this section, the users can freely switch between different states.

Understanding an agent’s strategy requires users to consider several states and their corresponding reward decomposition bars simultaneously. Therefore, to reduce the cognitive load, we only present the reward decomposition bar for the action that the agent chose in this state.

### **8.3. Discussion and Conclusion**

This chapter is a first step toward the development of combined explanation methods for reinforcement learning (RL) agents that provide users with both global information regarding the agent’s strategy as well as local information regarding its decision-making in specific world-states. To this end, we presented a joint global and local explanation method, building on strategy summaries (HIGHLIGHTS) and different local explanation methods. We show that this method is easily adaptable to various local explanation algorithms by applying it to two post-hoc explanation methods (LRP-argmax saliency maps and GANterfactual-RL counterfactuals) and one intrinsic explanation method (reward decomposition). A user study evaluation of our combined explanation approach, together with a thorough discussion of how users interact with our proposed approach, will follow in Part V.

**Part IV.**

**Computational Evaluation of XRL  
Approaches**

# 9. Benchmarking Perturbation-Based Saliency Maps for DRL Agents

Enhancing the explainability of DRL agents necessitates a dual focus on both developing and evaluating explanation methods, as discussed in Section 3.3 and Chapter 5. The previous three chapters presented novel explanation techniques for DRL agents. This chapter shifts focus to the computational evaluation of such explanation techniques. In particular, it benchmarks different perturbation-based saliency maps (see Section 3.2.1.2). As we have seen in Section 4.1.1, perturbation-based saliency maps are one of the most prevalent explanation methods for DRL agents. The major advantage of perturbation-based approaches is their model agnosticism – they can be applied to any kind of RL agent since they only use the in- and outputs of the agent.

If saliency maps are used to analyze DRL agents in high-risk applications, it is crucial that we can rely on the information provided by the saliency map. That is, the most relevant pixels, according to the saliency map, should actually be the most relevant pixels for the agent’s strategy. This is what we defined as the *fidelity* of a saliency map method in Section 3.3. The need for evaluating the fidelity of saliency maps was further demonstrated by Adebayo et al. [2018]. They proposed sanity checks that showed that for some saliency map approaches, there is no strong dependence between the learned parameters of image classifiers and the saliency maps that analyze their underlying neural network. Surprisingly, there are no computational evaluations that assess and compare the fidelity of different saliency maps for DRL agents. This is despite the fact that DRL agents are more challenging to analyze than classification models [Heuillet et al., 2021]. As mentioned in Section 1.1, the decisions of a DRL agent are not isolated but are part of an overarching policy and might be influenced by delayed rewards, which may not be discernible in the current state. This makes it even more challenging to verify whether a saliency map matches the internal reasoning behind a DRL agent’s action selection. In the prominent family of value-based DRL algorithms (Section 2.1.3), for example, the output values do not only describe the expected future reward after choosing each action. They also encode the estimated value of the input state for the current policy. This ambiguity is often ignored when saliency maps are applied

to analyze the decisions of value-based DRL agents.

To address this research gap, this chapter presents our previous work:

**Tobias Huber**, Benedikt Limmer, and Elisabeth André [2022]. “Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents”. In: *Frontiers in Artificial Intelligence* 5. ISSN: 2624-8212. DOI: 10.3389/frai.2022.903875

To the best of our knowledge, this work conducted the first computational fidelity evaluation of different saliency maps for DRL agents. In particular, it makes the following contributions. By focusing on five perturbation-based saliency map approaches, the work gives an overview of which approaches should be used by practitioners who do not have full access to their DRL agent’s model. One drawback of perturbation-based saliency maps is that they depend on a choice of parameters for the saliency map approaches. To ensure that all of the algorithms tested in this chapter perform reasonably well, we present a novel methodology to fine-tune the parameters of perturbation-based saliency maps for DRL agents. Furthermore, we propose a way to separately measure how well a saliency map captures an agent’s respective action and state value estimation. We demonstrate that the performance of saliency map approaches differs considerably when measuring state values compared to action values.

As test-bed for our evaluation, we use the Atari 2600 environment (Section 2.1.2). As metrics, we use the sanity checks proposed by Adebayo et al. [2018] and an insertion metric that measures if the most relevant pixels, according to the saliency map, actually affect the agent’s decision (see Section 3.3.1.1). As far as we know, this is the first time that sanity checks are done for different perturbation-based saliency maps for any kind of model.

## 9.1. Test-bed

The test-bed in our computational evaluation is the Atari Learning Environment (see Section 2.1.2). Four DRL agents were trained on the games Pacman (see Section 2.1.2.1 for details), Space Invaders, Frostbite, and Breakout. As training algorithm, we used the OpenAI Baselines [Dhariwal et al., 2017] implementation of the Deep Q-Network (DQN) (Section 2.2). We chose the DQN because it is the most basic DRL architecture, which many other DRL agents build upon. The games were selected because the DQN performs very well on Breakout and Space Invaders but performs badly on Frostbite and Pacman. This choice allows us to investigate scenarios in which the DRL agents achieve optimal performance and situations in which they are prone to errors. All experiments were done on the same machine with an Nvidia GeForce GTX TITAN X GPU to ensure comparability of the results. Our code is available online.<sup>1</sup>

---

<sup>1</sup><https://github.com/belimmer/PerturbationSaliencyEvaluation>

## 9.2. Evaluated Saliency Map Approaches

As saliency map approaches, we chose Occlusion Sensitivity [Zeiler and Fergus, 2014] since it is the first and most basic perturbation-based saliency map approach. Furthermore, we use LIME [Ribeiro et al., 2016] and RISE [Petsiuk et al., 2018], which are two of the most popular perturbation-based saliency maps in general. Finally, we chose two approaches that were specifically proposed for DRL: Noise Sensitivity [Greydanus et al., 2018] and SARFA [Puri et al., 2020]. Details about those five algorithms can be found in Section 3.2.1.2.

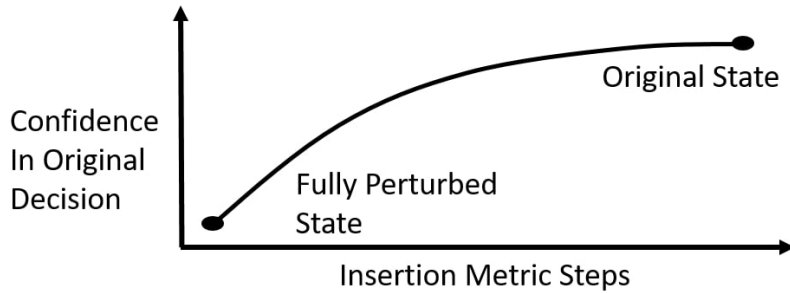
## 9.3. Metrics

We evaluate the generated saliency maps using two different computational metrics: Sanity checks and an insertion metric (see Section 3.3.1.1).

### 9.3.1. Sanity Checks

The sanity checks proposed by Adebayo et al. [2018] measure the dependence between the saliency maps and the parameters learned by the neural network of the agent. To this end, the parameters of each layer in the network are randomized in a cascading manner, starting with the output layer. Every time a new layer is randomized, a saliency map for this version of the agent is created. The resulting saliency maps are then compared to the saliency map for the original network, using three different similarity metrics (Spearman rank correlation, Structural Similarity (SSIM), and Pearson correlation of the Histogram of Oriented Gradients (HOGs)). If the saliency maps depend on the learned parameters of the agent, then the saliency maps for the randomized models should vastly differ from the ones of the original model. Following Sixt et al. [2020], we account for saliency maps that differ only in sign by additionally computing the similarity metrics between the original saliency map and a version of each saliency map for the randomized models that was multiplied by  $-1$ . For each randomized model, we use the maximum of the similarity values with and without the  $-1$  multiplication. For our tests, we calculate the sanity checks for 1000 states of each game.

Analogous to Adebayo et al. [2018], we calibrate the similarity metrics (Spearman rank correlation, SSIM, and Pearson correlation of the HOGs) such that high similarity values actually indicate similar saliency maps. Following Adebayo et al. [2018], we do this by calculating the similarity of 100 pairs of randomly generated saliency maps (Uniform and Gaussian). Since randomly sampled saliency maps should be very different on average, the mean of these similarities should be low. Using an SSIM window size of 7 and a HOG function with



**Figure 9.1.:** A schematic representation of the insertion metric curve.

(3, 3) pixels per cell, two randomly sampled saliency maps with uniform distribution had mean similarity values (0.0087, 0.0136, 0.0096) and two random saliency maps with Gaussian distribution had mean similarity (0.0093, 0.0374, 0.0087).

### 9.3.2. Insertion Metric

If a saliency map is faithful to the agent, then the most relevant pixels should have the highest impact on the agent’s decision. To test this property, we use an insertion metric similar to Petsiuk et al. [2018]. We do not use a deletion metric since we feel that it is too similar to the way that perturbation-based saliency maps are created. The insertion metric starts with a fully perturbed state. How this perturbation is done will be discussed in Section 9.3.2.1. In each step, 84 perturbed pixels (approx. 1.2% of the full state) are uncovered, starting with the most relevant pixels according to the saliency map. For LIME, the superpixels are sorted by their relevance, but the order of pixels within superpixels is randomized. The partly uncovered state is then fed to the agent, and its output for the action that the saliency map analyzes is measured. If the saliency map correctly highlights the most important pixels for this action, then the agent’s output corresponding to this action should increase quickly for each partly uncovered image. Plotting the agent’s output in each step of the insertion metric results in an insertion metric curve (Figure 9.1). If the output increases quickly, then the area under the insertion curve is high. Therefore, the Area Under the insertion metric Curve (AUC) is used to represent the result of the insertion metric for a single state. Before we can apply the insertion metric to our DRL agents, we have to decide how to perturb the input and which output value we measure in each step.

#### 9.3.2.1. How to Perturb the Input

Tomsett et al. [2020] found that the choice of perturbation during the insertion metric has a high impact on the result of the metric. To be more robust against

this influence, we use two different perturbations: black occlusion and uniform random perturbation in the range  $[0, 1]$ . Black is similar to the background color in most Atari games and, therefore, acts as “deleting” features from the state. Uniform random perturbation performed well for Tomsett et al. [2020].

### 9.3.2.2. Which Output to Measure

Next, we have to decide which output we want to measure during the insertion metric. This comes with two further challenges.

First, the output Q-values of value-based reinforcement learning algorithms like the DQN do not directly describe the agent’s confidence in particular actions. Instead, they approximate the value of the current state in combination with each action (see Section 2.1.3). To disentangle this ambiguity, we propose to use two different sub-metrics. One measures how well the saliency map identifies features relevant to the state value, and the other measures the same for the action value. For the state value, we suggest using the Q-value  $Q(s, a)$  of the action that the saliency map is analyzing. For the action value, we propose an estimation of the advantage as used by Wang et al. [2016b]:

$$A(s, a) = Q(s, a) - \frac{\sum_{a \in \mathcal{A}} Q(s, a)}{|\mathcal{A}|} \quad (9.1)$$

The second challenge is that a reliable metric should not be distorted by outliers. For our Pacman agent, for example, we observed states with Q-values around 1 and other states with Q-values around 50. To reduce the effect of outlier states, we tested different methods of normalizing the agent’s output during the insertion metric. The first normalization method we tested was inspired by Sixt et al. [2020] and forces each insertion curve to start at 0 and finish at 1. This is achieved by applying  $f(x) = \frac{x-b}{t-b}$  to each insertion step result, where  $b$  is the output of the fully perturbed state and  $t$  is the output of the original state. As the second method, we only divided each insertion step by the output of the original state  $t$ . In this way, all insertion curves finish at the value of 1.

To identify which normalization method works best, we used 28 different variants of Occlusion Sensitivity saliency maps. The variants were obtained by varying the occlusion patch size between 4 and 10, using gray or black occlusion, and using the raw Q-values or adding a softmax layer for the relevance calculation (Eq. 3.1).<sup>2</sup> For each variant, we calculated differently normalized insertion metrics over 1000 states of the Pacman environment for each of our two insertion metric perturbation methods. Tomsett et al. [2020] suggest using

---

<sup>2</sup>To see how these variants performed in our final insertion metric tests, refer to Section 9.4.3.

**Table 9.1.:** The minimum and maximum SD when evaluating 28 different parameter combinations of Occlusion Sensitivity saliency maps with an insertion metric using different normalization functions.

Normalization Function	Minimum SD	Maximum SD
Measuring Q-values		
No Normalization	5.16	10.17
$f(x) = \frac{x}{t}$	1.14	2.06
$f(x) = \frac{x-b}{t-b}$	10.33	48.56
Measuring Advantage		
No Normalization	0.84	1.42
$f(x) = \frac{x}{t}$	1.99	3.78
$f(x) = \frac{x-b}{t-b}$	9.45	165.20

a low Standard Deviation (SD) as an early indicator for reliable saliency map metrics. Therefore, we chose the normalization method that resulted in the lowest SD of the area under the insertion curve across the 1000 states and both perturbation methods<sup>3</sup>. For each normalization method, Table 9.1 shows the highest and lowest SD among the 28 different Occlusion Sensitivity variants.

Interestingly, the full normalization to curves between 0 and 1 resulted in the highest SD. We think that this comes from the fact that our agents sometimes assign higher values to the fully perturbed state than to the original state. In these cases,  $t - b$  is negative, and applying  $f(x)$  inverts the insertion curve. For the advantage, we obtained the lowest SD if we did not use any normalization. The Q-values got the lowest SD when we divided each insertion metric step by the result of the original state.

### 9.3.2.3. Final Setting

For our final evaluation of the different saliency map methods, we use 1000 states of each of the four Atari games. For each of those states, we calculated the insertion metric in four different variants: measuring the advantage of the chosen action with random and black perturbation, and measuring the normalized Q-value with random and black perturbation.

<sup>3</sup>Let  $\mu$  be the mean of the 2000 resulting AUC values  $x_i$  then the SD is given by  $\sqrt{\frac{\sum_i (x_i - \mu)^2}{2000}}$ .



## 9.4. Parameter Tuning

One of the biggest drawbacks of perturbation-based saliency map approaches is that they depend on a choice of parameters as can be seen in Section 3.2.1.2. Before we can run our final experiments, we have to find suitable parameters. Section 9.4.3 will list the specific parameters that we tuned for each saliency map approach. This tuning is often done by manually adjusting the parameters until the resulting saliency maps look reasonable. However, tuning the parameters in this way does not guarantee that the saliency maps match the agent’s internal reasoning. To obtain a fidelity benchmark for saliency maps, we computationally tune the parameters to perform well in the insertion metric. We do not tune the parameters for the sanity checks since sanity checks do not measure how well a saliency map approach performs. Instead, they identify which approaches do not work at all. To tune the parameters for our final tests, we need to decide on two things: how we combine the results from the four different insertion metric variants and which states we test the parameters on.

### 9.4.1. Combining Insertion Metric Results

To combine the results of the random and black insertion metric variants (see Section 9.3.2.1), we measure the mean of the area under the insertion curve over both the black and the random perturbation insertion metric. For our evaluation, we would also like to find parameters that are able to analyze both the agent’s action value and state value estimation. To this end, we standardize the mean AUC results of the aforementioned tests for the advantage and Q-values measurements respectively<sup>4</sup>. The sum of these standardized values is then used as a single value that measures the performance of the parameters. Parameters such as patch size have a strong influence on the run-time of the saliency map approaches. Therefore, to ensure comparability between approaches and to run our final experiment in a reasonable time, we do not select the top parameters. Instead, we use the best parameters that take up to three seconds to compute a single saliency map.

### 9.4.2. Choosing a Test Set

As a test set for our parameter tuning, it is not feasible to use the full stream of 1000 states that we want to use in our final evaluation of the different saliency maps. LIME and RISE, in particular, have long computation times and a large number of possible parameter combinations (We will provide more information on the run-time of each saliency map approach in Section 9.5.3). This would

---

<sup>4</sup> Let  $\mu$  and  $\sigma$  be the mean and standard deviation of the mean AUC results  $x_i$  then the standardized results  $z_i$  are given by  $z_i = \frac{x_i - \mu}{\sigma}$ .

make the run-time of the parameter test explode. Therefore, we need to find a suitable subset of states that represent as many states as possible. Since there are no test- or validation-sets in reinforcement learning, we have to choose these subsets from the full stream of gameplay.

As potential candidates, we tested 22 different subsets consisting of 10 states each. Ten of these subsets were randomly selected. The other 12 subsets were selected by different variants of the HIGHLIGHTS algorithm. HIGHLIGHTS selects a diverse set of states that give a good overview of the agent’s policy (Section 4.2.3.1). Hereby, it utilizes a diversity threshold that makes sure that the selected states are not too similar to each other. For this diversity threshold, we tested the 10, 20, 25, 28, 30, 32, 33, 35, and 40 percentile of the similarity values of the full 1000 states stream<sup>5</sup>. To get even more diverse sets of states, we additionally tested two novel variations of HIGHLIGHTS: one variant where we used 5 of the most important and 5 of the least important states for the agents’ strategy and one variant where we sorted all 1000 states by importance and chose every 100th state to obtain states of all importance levels.

To compare how well these subsets represent the full stream of gameplay, we calculated the combined insertion metric results, as described above, for the full 1000 states of Pacman using 28 different parameter combinations of Occlusion Sensitivity. The particular parameters were chosen since they are fast to compute. Based on these results, we obtained a “ground truth” for how those 28 parameters for Occlusion Sensitivity should be ranked. Now, a subset of states is suited for searching parameters if the parameter ranking obtained by the subset is similar to the ranking obtained by the full 1000 states. To calculate the similarity of different rankings, we used both Spearman’s and Kendall rank correlation coefficients. While this does not give conclusive evidence, it gives a good estimation of which states do and do not work. The highest correlation to the ranking obtained by the full stream was achieved by the 30 percentile HIGHLIGHTS variant. For the action value, the Spearman’s rank correlation was 0.96, and the Kendall rank correlation was 0.85. For the state value, the Spearman’s rank correlation was 0.95, and the Kendall rank correlation was 0.81. The correlations for the other subsets can be seen in our repository.<sup>6</sup> It is important to note that HIGHLIGHTS only performed well when the diversity threshold was very high. When the threshold was low, the HIGHLIGHTS states performed worse than the random ones. We got the best results when the threshold was so high that increasing the threshold resulted in subsets with less than ten states since the algorithm could not find any more states that could be added to the subset.

---

<sup>5</sup>Here the  $n$  percentile is the value  $\frac{n}{100}$  of the way from the minimum of the similarity values to the maximum.

<sup>6</sup><https://github.com/belimmer/PerturbationSaliencyEvaluation>

**Table 9.2.:** Best parameters for Occlusion Sensitivity. The final parameters are marked in bold.

AUC	Patch Size	Color	Softmax	Time
6.76	1	Black	No	10.94
3.44	1	Gray	No	11.09
3.42	1	Black	Yes	11.51
3.03	1	Gray	Yes	11.50
<b>2.33</b>	<b>2</b>	<b>Black</b>	<b>No</b>	<b>2.80</b>
0.93	2	Black	Yes	2.88
0.32	3	Black	No	1.26
0.26	2	Gray	No	2.83
0.04	2	Gray	Yes	2.87
-0.06	4	Black	No	0.69

### 9.4.3. Used Saliency Map Parameters

Using the combined insertion metric results and the test set described above, we tested a total of 4918 parameter combinations across all five saliency map methods. The full results of our tests for each method can be viewed in our repository. <sup>7</sup>

**Occlusion Sensitivity.** For Occlusion Sensitivity, we tested patches of size 1 to 10, black and gray occlusion color, and whether applying a softmax layer to the output Q-values before creating the saliency map improves results. The top ten results are shown in Table 9.2.

**Noise Sensitivity and SARFA.** For Noise Sensitivity, we tested circles with a radius of 1 to 10. The top ten parameters are shown in Table 9.3 (a). SARFA was not introduced with a specific perturbation method. Analogous to Puri et al., we test blurred circles of radius 1 to 10 as used in Noise Sensitivity. Additionally, we also use circles that are occluded with black color. The top ten results are shown in Table 9.3 (b).

**RISE.** For RISE we tested 500, 1000, ..., 3000 masks of size 4 to 24. The probability  $p$  with which each pixel is occluded varied between 0.1 and 0.9 in steps of 0.1. Analogous to Occlusion Sensitivity, we also investigated whether it makes sense to add a softmax layer after the output during the saliency map creation. The top five results are shown in Table 9.4.

<sup>7</sup><https://github.com/belimmer/PerturbationSaliencyEvaluation>

**Table 9.3.:** Best parameters for Noise Sensitivity (a) and SARFA (b). The final parameters are marked in bold.

(a)			(b)			
AUC	Radius	Time	AUC	Radius	Perturbation	Time
3.08	2	5.79	7.03	1	Black	12.05
2.21	1	22.84	<b>1.46</b>	<b>2</b>	<b>Black</b>	<b>3.00</b>
<b>0.94</b>	<b>3</b>	<b>2.62</b>	1.09	1	Blur	23.70
0.68	9	0.38	0.57	8	Blur	0.46
0.48	10	0.31	0.55	2	Blur	6.12
-0.05	4	1.48	0.49	9	Blur	0.39
-0.49	8	0.44	0.40	10	Blur	0.32
-0.84	5	0.99	0.27	3	Black	1.40
-2.07	6	0.68	0.08	3	Blur	2.77
-3.94	7	0.51	0.01	5	Blur	1.06

**Table 9.4.:** Best parameters for RISE. The final parameters are marked in bold.

AUC	$p$	Mask Size	Masks	Softmax	Time
3.21	0.8	11	3000	Yes	5.09
3.04	0.7	13	3000	No	4.76
2.99	0.9	24	2500	Yes	3.98
2.94	0.8	4	3000	No	4.66
:	Skipping 9 parameters that took more then 3 seconds.				
<b>2.66</b>	<b>0.5</b>	<b>8</b>	<b>1000</b>	<b>No</b>	<b>1.54</b>

**Table 9.5.:** Best parameters for LIME with Felzenszwalb segmentation. The final parameters are marked in bold.

AUC	Scale	Sigma	Minimum Size	Num Samples	Time
4.35	21	0.5	0	3000	10.73
3.58	21	0.75	2	3000	7.38
3.53	1	1.0	0	2000	22,03
3.29	21	0.5	0	2500	8.95
:		Skipping 14 parameters that took more then 3 seconds.			
<b>2.55</b>	<b>21</b>	<b>0.5</b>	<b>4</b>	<b>1000</b>	<b>1.71</b>

**Table 9.6.:** Best parameters for LIME with SLIC segmentation. The final parameters are marked in bold.

AUC	Num Segments	Compactness	Sigma	Num Samples	Time
3.99	200	10.0	1.0	3000	3.13
<b>3.86</b>	<b>200</b>	<b>10.0</b>	<b>0.25</b>	<b>2000</b>	<b>2.08</b>
3.48	200	10.0	0.0	3000	3.11
3.46	200	0.001	0.25	3000	2.36
3.44	200	10.0	0.5	1000	1.06

**LIME.** For LIME, we tested the three most common Segmentation techniques *SLIC*, *Quickshift*, and *Felzenszwalb* and varied the number of samples on which the local interpretable model is trained. For the number of samples, we took the default number of samples (1000) and increased it in steps of 500 up to 3000. To determine which parameter ranges we should explore for each segmentation algorithm, we performed preliminary tests where we visually checked which parameters resulted in different segmentation. For Felzenszwalb segmentation we used a scale factor of 1,21,...,101, a minimum component size from 1 to 8 and Gaussian smoothing kernels with width  $\sigma$  of 0,0.25,...,1. The top results are shown in Table 9.5. For SLIC we tested 40,60 to 240 segments, a compactness factor of 0.001,0.01,...,10 and Gaussian smoothing kernels with width  $\sigma$  of 0,0.25,...,1. The top five parameter combinations can be seen in Table 9.6. Finally, we tested Quickshift with a color ratio of 0.0,0.33,0.66, and 0.99, a kernel size from 1 to 6 and a max distance of  $kernelsize * i$ , where  $i$  goes from 1 to 4. The top results are shown in Table 9.7.

**Table 9.7.:** Best parameters for LIME with Quickshift segmentation. The final parameters are marked in bold.

AUC	Kernel Size	Max Distance	Ratio	Num Samples	Time
6.24	1	1	0.0	3000	11.38
4.97	1	1	0.0	2500	9.57
4.80	1	2	0.0	2500	4.46
4.50	1	2	0.0	3000	5.39
<b>4.23</b>	<b>1</b>	<b>2</b>	<b>0.0</b>	<b>1500</b>	<b>2.75</b>

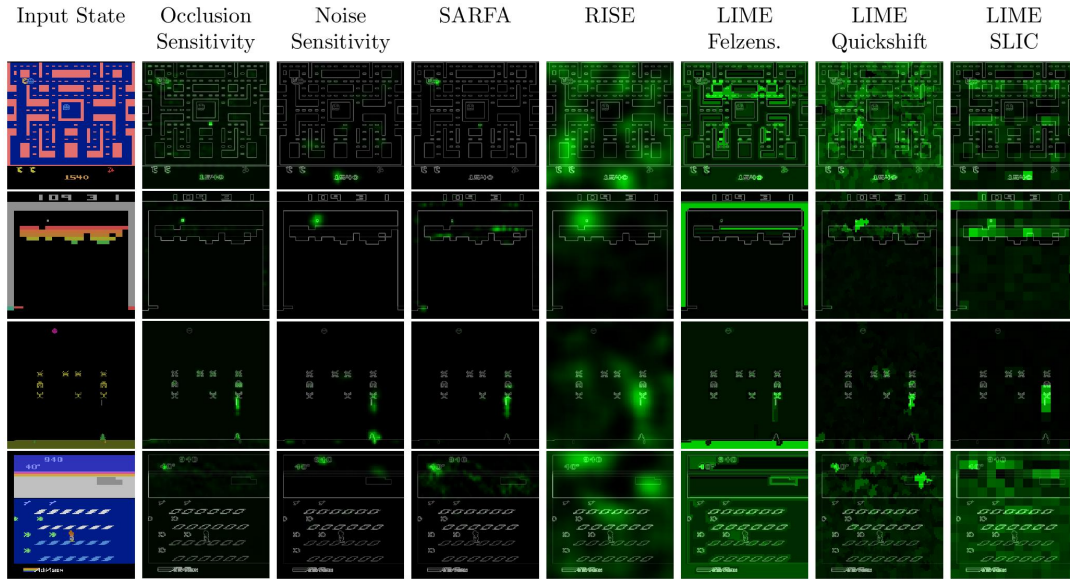
## 9.5. Results

Figure 9.2 shows example saliency maps for all four games used in our experiments. To prevent cherry-picking of particularly convincing states, the states are chosen by the HIGHLIGHTS algorithm (see Section 4.2.3.1).

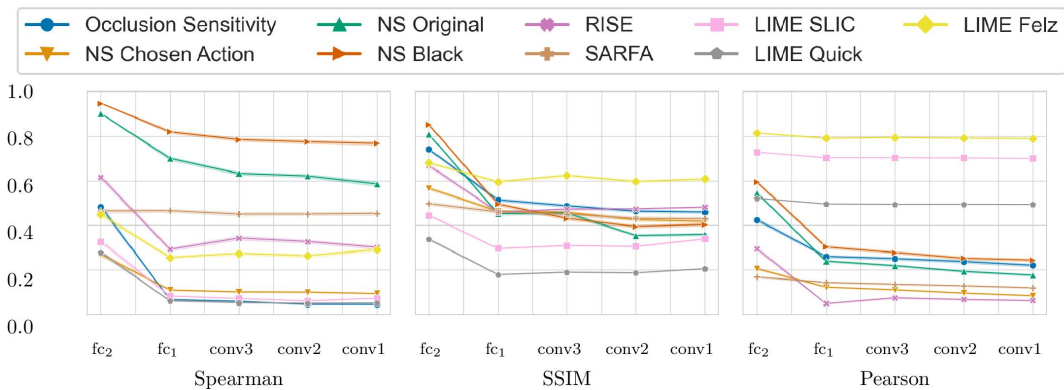
### 9.5.1. Sanity Checks

The combined results of the sanity checks test are shown in Figure 9.3. The results for each individual game can be seen in Figure 9.4. The lower the scores, the higher the dependence on the agents’ learned parameters. Notably, LIME has a very high Pearson correlation of HOGs. Furthermore, the original Noise Sensitivity has a low dependence on the parameters of the output layer compared to Occlusion Sensitivity. Since those two approaches are very similar in theory, we implemented two modifications of Noise Sensitivity to investigate the reason for this difference in parameter dependence. First, *Noise Sensitivity Black* occludes the circles in the Noise Sensitivity approach with black color instead of blurring them. Second, *Noise Sensitivity Chosen Action* changes the way that the relevance of each pixel is calculated from the original equation (Eq. 3.2), which takes all actions into account, to the one used by Occlusion Sensitivity (Eq. 3.1), which focuses on the chosen action. We did not test a combination of black circles and the Occlusion Sensitivity relevance calculation since that would be equivalent to Occlusion Sensitivity with circles instead of squares. While the black occlusion did not really change the sanity check results, the change in the relevance calculation immensely increased the dependence on the learned parameters.

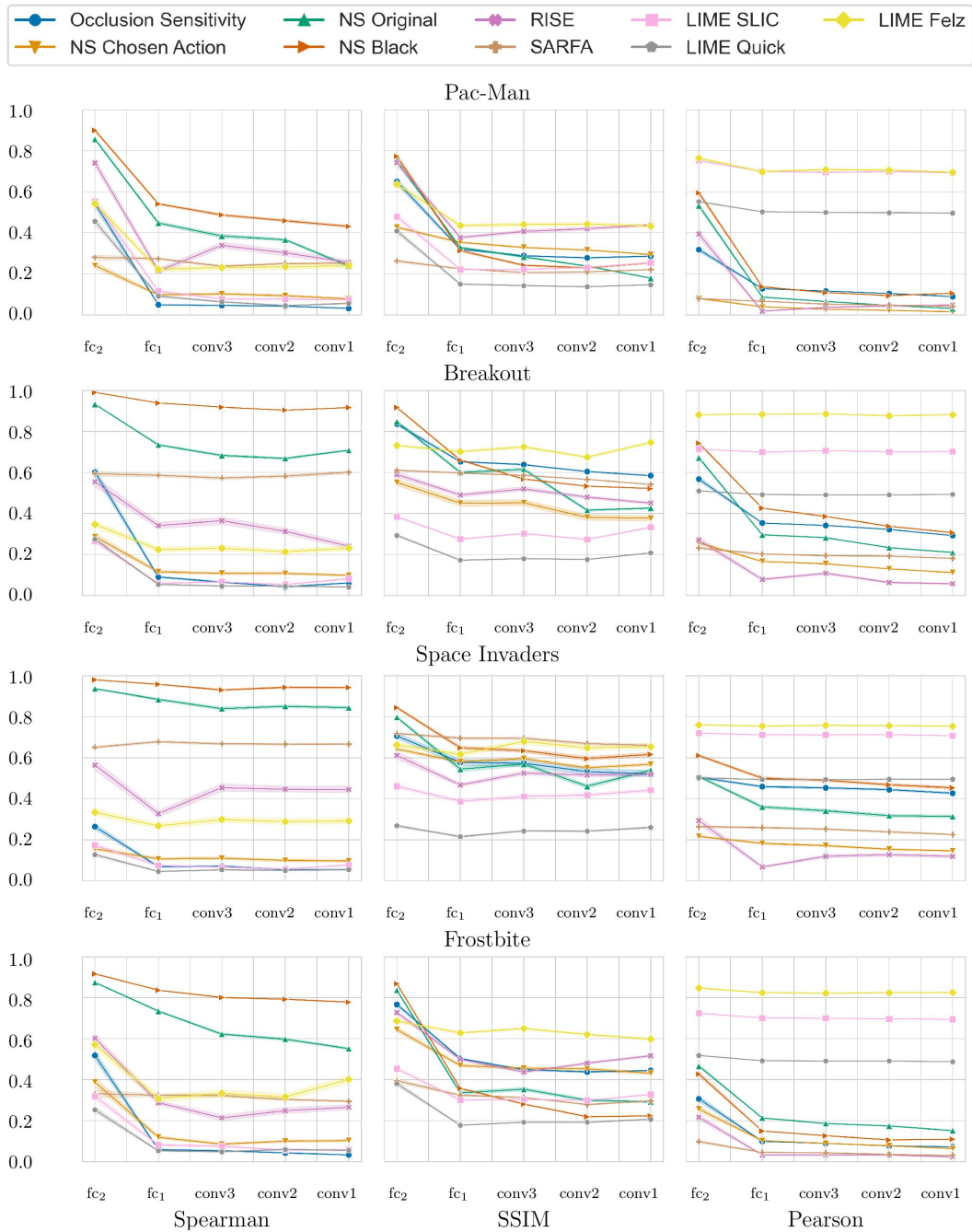
An example of the different saliency maps during a single run of the sanity check can be seen in Figure 9.5.



**Figure 9.2.:** Example saliency maps for the games we tested. From top to bottom: Pacman, Breakout, Space Invaders, and Frostbite. For better visibility, the saliency maps are displayed in green color over a simplified version of the states. The higher the intensity of the green color, the higher the relevance of the corresponding pixel for the agent’s decision.

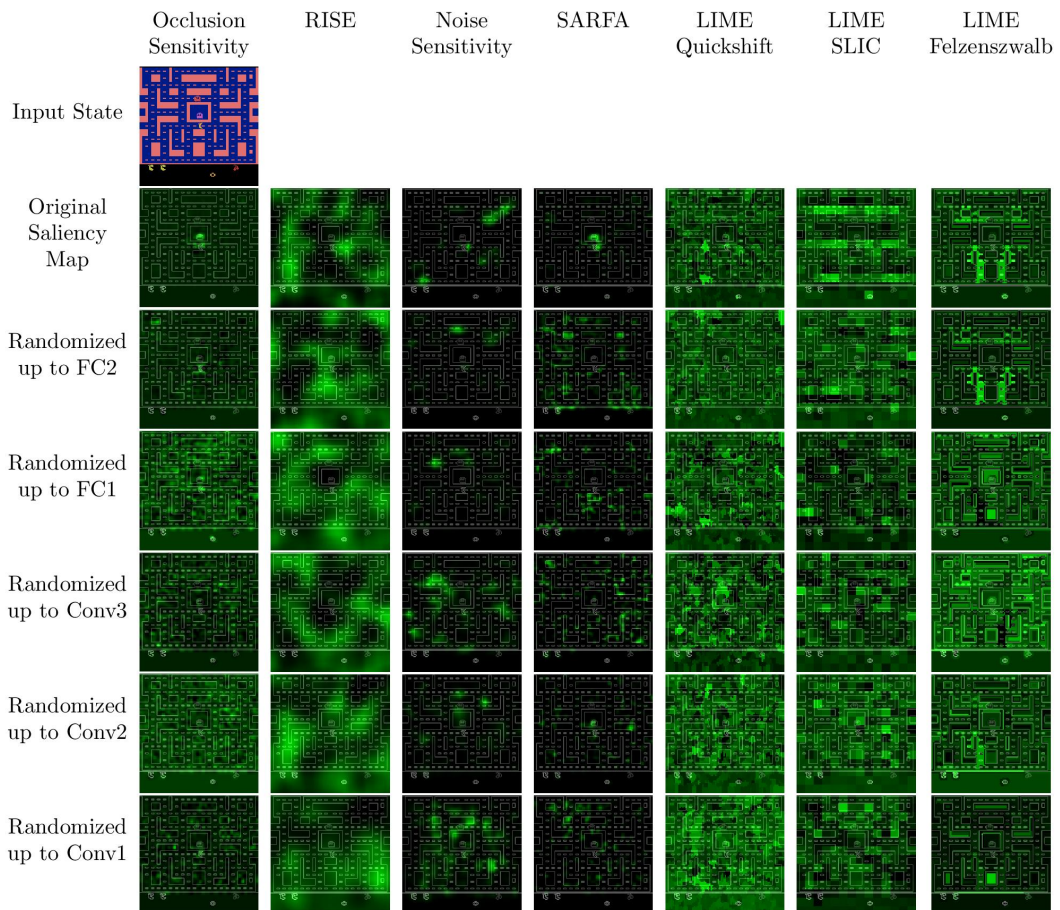


**Figure 9.3.:** Results of the sanity checks for the different saliency map approaches ( $NS$  is noise Sensitivity). Measured for 1000 states of each of the four tested games. Starting from the left, each mark represents an additional randomized layer starting with the output layer. The y-axis shows the average similarity values (Spearman rank correlation, SSIM, Pearson correlation of the HOGs). High values indicate a low parameter dependence. The translucent error bands show the 99% CI but are barely visible due to low variance in the results.



**Figure 9.4.:** Results of the sanity checks for each individual game for the different saliency map approaches (*NS* is noise Sensitivity). Measured for 1000 states of each of the four tested games. Starting from the left, each mark represents an additional randomized layer starting with the output layer. The y-axis shows the average similarity values (Spearman rank correlation, SSIM, Pearson correlation of the HOGs). High values indicate a low parameter dependence. The translucent error bands show the 99% CI.





**Figure 9.5.:** Example saliency maps for the parameter randomization sanity check. From top to bottom, each row after the first is generated for agents with cascadingly randomized layers, starting with the output layer.

### 9.5.2. Insertion Metric

Tables 9.8 and 9.9 report the sample mean and SD of the insertion metric results for 1000 states of each game and each saliency map approach.<sup>8</sup> To get a baseline performance, we also calculated the insertion metric with uniformly sampled random saliency maps. For some games and sub-metrics, the mean area under the insertion curve is negative. This is due to the fact that some agents assign high negative Q-values and advantages to the fully perturbed state. For most games, RISE has the best results for measuring the raw Q-values on random perturbation. However, the results for measuring advantage with random perturbation are poor for all approaches. For Frostbite and Space Invaders, and measuring the advantage with random perturbation, the random saliency maps even performed better than all other approaches. For the other two games, RISE has the highest values. When using black color perturbation during the insertion metric, Occlusion Sensitivity obtained very good results for measuring the state value, and SARFA worked best for the advantage. However, their results for random perturbation were very poor. From our parameter tuning, we knew that this depended on the color of perturbation used during the saliency map generation. Therefore, we additionally tested Occlusion Sensitivity with gray color and SARFA with noise perturbation as used by Noise Sensitivity. The other parameters remained unchanged. Table 9.10 shows the results of those additional tests. Notably, Occlusion Sensitivity got the highest Q-value random insertion results in Pacman, Frostbite, and Space Invaders. SARFA got the best advantage results for random insertion for Pacman and Frostbite, only slightly losing to Occlusion Sensitivity with gray color in Space Invaders. The performance of both approaches on black perturbation fell to a level similar to the random baseline. The exception to most observations described above is Breakout. Here, the LIME variants performed the best across most metrics. SLIC segmentation, in particular, achieves at least the second-highest score in each metric. It is worth noting that this game also has the highest SD values.

---

<sup>8</sup>For each game and metric the sample mean of the 1000 insertion metric results  $x_i$  is calculated by  $\mu = \frac{\sum_i x_i}{1000}$  and the SD by  $\sqrt{\frac{\sum_i (x_i - \mu)^2}{1000}}$ .

**Table 9.8.:** Part 1 of the sample mean and SD of the insertion metric curve for 1000 states of each game, see Table 9.9 for part 2. *Q-val* and *Adv* measure the change of the normalized Q-value and advantage respectively. *Rand* and *black* use random and black perturbation, respectively, during the insertion metric.

Metric	Occlusion	Noise	SARFA	RISE	Baseline
Pacman:					
Q-val rand	0.54±1.3	0.75±0.7	0.76±1.2	<b>1.1±2.0</b>	0.85±1.5
Adv rand	-0.52±1.2	-0.03±0.8	-0.74±1.3	<b>-0.01±1.1</b>	-0.22±1.0
Q-val black	<b>3.08±3.2</b>	0.66±0.8	0.83±1.8	1.01±1.8	0.53±0.8
Adv black	1.23±1.6	0.15±0.3	<b>1.7±0.8</b>	0.21±0.4	0.06±0.3
Breakout:					
Q-val rand	-0.72±2.5	-1.01±3.0	-3.19±3.9	-0.97±2.7	-2.21±2.9
Adv rand	-0.42±4.7	-1.52±8.4	-0.92±8.4	<b>0.85±6.1</b>	-0.76±5.8
Q-val black	3.16±4.2	3.04±4.2	1.97±2.0	3.39±4.2	2.13±3.1
Adv black	0.02±0.5	0.19±0.6	0.53±1.1	0.29±0.6	0.07±0.2
Frostbite:					
Q-val rand	0.56±1.0	0.83±1.0	0.73±1.0	<b>0.92±1.1</b>	0.88±1.1
Adv rand	0.31±1.1	0.38±1.2	0.2±1.2	0.35±0.9	<b>0.4±1.2</b>
Q-val black	<b>5.65±3.1</b>	0.58±0.2	1.53±1.6	2.4±1.7	0.51±0.4
Adv black	0.59±0.9	0.2±0.2	<b>1.22±0.9</b>	0.25±0.3	0.16±0.2
Space Invaders:					
Q-val rand	-0.7±0.6	-0.6±0.6	-0.8±0.6	<b>-0.39±0.4</b>	-1.1±0.8
Adv rand	0.76±3.5	0.83±3.4	0.79±3.7	0.66±2.8	<b>0.89±4.2</b>
Q-val black	1.01±0.2	0.73±0.1	0.74±0.2	0.89±0.1	0.56±0.1
Adv black	0.28±0.4	0.26±0.3	<b>0.59±0.4</b>	0.21±0.2	0.13±0.2

**Table 9.9.:** Part 2 of the sample mean and SD of the insertion metric curve for 1000 states of each game, see Table 9.8 for part 1. *Q-val* and *Adv* measure the change of the normalized Q-value and advantage respectively. *Rand* and *black* use random and black perturbation, respectively, during the insertion metric.

Metric	LIME Felz.	LIME Quick.	LIME SLIC	Baseline
Pacman:				
Q-val rand	0.46±0.7	0.67±1.1	0.62±1.1	0.85±1.5
Adv rand	-0.43±1.2	-0.44±1.0	-0.36±1.1	-0.22±1.0
Q-val black	2.83±5.3	2.49±4.7	2.47±4.4	0.53±0.8
Adv black	0.64±0.7	0.94±0.5	0.67±0.5	0.06±0.3
Breakout:				
Q-val rand	-0.98±2.7	<b>-0.48±4.1</b>	-0.53±3.2	-2.21±2.9
Adv rand	-0.7±6.5	-0.54±5.4	-0.05±4.8	-0.76±5.8
Q-val black	<b>7.48±9.6</b>	5.8±8.7	6.12±9.7	2.13±3.1
Adv black	0.24±0.6	0.24±0.4	<b>0.71±1.4</b>	0.07±0.2
Frostbite:				
Q-val rand	0.75±0.9	0.37±1.0	0.36±1.0	0.88±1.1
Adv rand	0.2±0.9	0.24±1.3	0.23±1.3	<b>0.4±1.2</b>
Q-val black	2.71±2.4	5.12±4.1	3.25±2.5	0.51±0.4
Adv black	0.26±0.3	0.28±0.4	0.26±0.3	0.16±0.2
Space				
Invaders:				
Q-val rand	-1.12±0.9	-0.81±0.7	-0.88±0.7	-1.1±0.8
Adv rand	0.87±4.3	0.76±3.7	0.87±3.6	<b>0.89±4.2</b>
Q-val black	1.02±0.2	1.08±0.2	<b>1.11±0.3</b>	0.56±0.1
Adv black	0.24±0.2	0.25±0.2	0.29±0.2	0.13±0.2

**Table 9.10.:** The sample mean and SD of the insertion metric curve for our additional experiments with different perturbations for Occlusion Sensitivity and SARFA. *Q-val* and *Adv* measure the change of the normalized Q-value and advantage, respectively. *Rand* and *black* use random and black perturbation, respectively, during the insertion metric. The bold values beat the highest values for the respective metric in our original experiment.

Metric	Occlusion gray	SARFA blur
Pacman:		
Q-val rand	<b>2.98±3.5</b>	1.0±2.2
Adv rand	0.44±1.8	<b>1.12±1.0</b>
Q-val black	0.32±0.2	0.62±1.3
Adv black	-0.13±0.3	0.23±0.4
Breakout:		
Q-val rand	-0.83±2.6	-0.8±3.4
Adv rand	0.4±5.3	-0.4±5.8
Q-val black	1.99±2.5	3.0±3.9
Adv black	0.11±0.5	0.21±0.7
Frostbite:		
Q-val rand	<b>3.54±2.3</b>	1.13±1.2
Adv rand	0.66±1.1	<b>0.73±1.2</b>
Q-val black	0.5±0.5	0.58±0.3
Adv black	0.16±0.2	0.3±0.3
Space Invaders:		
Q-val rand	<b>0.07±0.7</b>	-0.75±0.7
Adv rand	<b>1.04±3.5</b>	1.02±3.7
Q-val black	0.48±0.2	0.66±0.2
Adv black	0.12±0.3	0.44±0.4

### 9.5.3. Run-time Analysis

The run-time of an algorithm can be an important aspect when choosing between different approaches. We computed the mean time it took each algorithm to create a single saliency map using the *timeit* python library. To get a feeling of how this is affected by different parameters of the saliency map approaches, we measured the time during our parameter tuning process where each parameter combination was used on 10 different states (see Section 9.4 for the full results).

The fastest approach was Occlusion Sensitivity, which uses simple color occlusions followed by the more complex blur perturbation of SARFA and Noise Sensitivity. However, this was strongly dependent on the size of the perturbation patches and circles respectively. Using a patch size or radius of 1, these approaches were among the slowest, with a mean run-time of around 22s for the blur perturbation and approximately 11s for the black occlusion variant. However, increasing the patch size and radius to 2 already drastically reduced the run-time. For RISE, the run-time mainly depends on the number of masks. With 3000 masks, the run-time was always close to 5s per saliency map. However, compared to the aforementioned saliency map approaches, the run-time only decreased slowly when the number of masks decreased. Thus, the average and the fastest run-time were much slower for RISE than for SARFA, and Occlusion and Noise Sensitivity. The slowest approach we tested was LIME. However, this was strongly influenced by the number of segments that the segmentation functions generated and the number of learning steps for the locally interpretable classifier. For SLIC, which creates relatively big segments, LIME was quite fast, with a maximum run-time of 3.87s with the slowest parameters. In contrast, the run-time for Felzenswalb easily exploded and reached a maximum of 33.64s per saliency map. Quickshift was in the middle of those two approaches with a maximum run-time of 12.50s, which did not decrease as quickly as the run-time of Occlusion and Noise Sensitivity, and SARFA.

## 9.6. Discussion

### 9.6.1. Sanity Checks

The results of our sanity checks show that most of the perturbation-based saliency map approaches tested in this chapter are dependent on the learned parameters of the agent’s neural network. Their dependence on the learned parameters is generally comparable to the best gradient-based approaches tested by Adebayo et al. [2018] and the best modified propagation approaches tested in Sixt et al. [2020]. The only exceptions to this are Noise Sensitivity and LIME.

Noise Sensitivity showed little dependence on the parameters of the output

layer (Figure 9.3). Since the output layer has the highest impact on the actual decision of a network, it is crucial that a faithful saliency map depends on the weights learned in this layer. Our results empirically show that replacing the original equation of Noise Sensitivity to calculate the relevance of each pixel with the equation used by Occlusion Sensitivity greatly increases the parameter dependence. We think that this is due to the fact that the original equation takes all actions into account and, therefore, measures a general increase in entropy within the activations of the output layer. In contrast, Occlusion Sensitivity only measures the action that is actually analyzed and, therefore, captures a more specific change in the output layer activation. Recently, Puri et al. [2020] also criticized that the saliency maps by Greydanus et al. [2018] take all actions into account. The results of our sanity checks provide the first computational evidence for this critique.

LIME performed well in the sanity check measurements using SSIM and Spearman correlation. Only the Pearson correlation of the HOGs was very high between LIME saliency maps for the trained and randomized agents. However, the reason for this is not necessarily a low dependence on the agent’s learned weights. More likely, it is due to the fact that all LIME saliency maps for a given state work with the same superpixels. Since every pixel inside a superpixel has the same value, there are hard edges between the superpixels. These edges are captured by the HOGs and result in high values of the Pearson correlation of the HOGs.

### 9.6.2. Insertion Metric

During our parameter tuning, we tried our best to find parameters that result in saliency maps that work for both black and random perturbation and capture both the agent’s action value as well as state value estimation. Despite these efforts, no saliency map approach performed well across all sub-metrics. The best results for measuring the state value were obtained by Occlusion Sensitivity, and the best results for the action value were obtained by SARFA. This distinction is illustrated by the fact that no SARFA saliency map for Pacman, which we looked at, identified the in-game score as relevant (e.g., Figure 9.2). The score is a good indicator of the value of the current state and is frequently highlighted by all other approaches we tested. However, based on the rules of the game, it is not necessary to know the score to choose the correct action in a given Pacman state.

Additionally, the saliency maps’ fidelity depended on the type of perturbation. The area under the insertion curve with black perturbation was the highest when the saliency map approaches used black occlusion. To mitigate this effect, some saliency map approaches utilize blurring during their perturbation. Surprisingly, this was also sensitive to the perturbation type of the insertion metric

in our tests. Similar to gray occlusion, blurring performed best for the random perturbation insertion metric and did not do well on black perturbation. The closest thing to a saliency map approach that fits all sub-metrics was RISE. However, the results here were considerably worse than the results for Occlusion Sensitivity and SARFA with parameters that fit the respective sub-metric, especially when analyzing the action value estimation.

These results do not necessarily mean that the evaluated saliency map methods are not suited to explain DRL agents. However, they demonstrate that none of the approaches answers the general question: “What was the most relevant input region for the agent’s decision?”. Instead, they answer more specific questions depending on the type of perturbation and whether the state or action value is analyzed. For example, SARFA with black perturbation for Pacman is suited to answer the question: “The presence of which objects was relevant for the agent’s choice of action”. Since the black background color of Pacman acts as deleting objects, and SARFA measures the action advantage. In contrast, Occlusion Sensitivity with black color would answer the same question with regard to the agent’s evaluation of the current state. Based on this, we advise future researchers to clearly define what question they want to investigate. Depending on that question, a fitting saliency map method can be chosen.

Our parameter tuning experiments also showed that the fitting saliency map method can strongly depend on single parameters of the saliency map methods. Therefore, we encourage future researchers to conduct systematic parameter searches fitting their questions similar to the one described in this chapter. Manually adjusting the parameters until the resulting saliency maps look reasonable might lead to saliency maps that look convincing but do not match the agent’s internal reasoning.

### 9.6.3. Limitations

We used four different variants of the insertion metric to get a good estimate of saliency map approaches’ fidelity in different situations. Between those variants, we already found distinct differences. This fact reinforces the findings by Tomsett et al. [2020] that current fidelity metrics for saliency maps can be very sensitive to the specifics of their implementation. For value-based RL in particular, we extend the results of Tomsett et al. by demonstrating that there are also considerable differences between metrics that measure the action value and metrics that measure the state value. However, it can not be ruled out that other fidelity metric variants might result in even more insights. To ease future evaluations and parameter searches, a great challenge for XAI research will be the development of more general fidelity metrics for saliency maps.

Another potential limitation of our results is that recent work indicates that simply displaying saliency maps to end-users might not be suited as a final



explanation (see Chapter 10 and [Danesh et al., 2021]). However, saliency maps are still often used as primary components of more sophisticated explanation frameworks (e.g., [Danesh et al., 2021]). We argue that it is even more crucial to evaluate the fidelity of saliency maps in situations where their information is used as an integral component of more complex explanation mechanisms.

We only used DRL agents with visual input in our evaluation since this is the most common application for saliency maps. It is possible to apply saliency map methods to DRL agents with other input domains as used in 3D locomotion tasks [Todorov et al., 2012], queueing network controls [Dai and Gluzman, 2022], and recommendation systems [Zhao et al., 2021]. In this context, the saliency map methods are often referred to as Feature Attribution methods. This raises the question of whether our results extend to Feature Attribution methods in non-visual domains. Since visual image manipulations (e.g., image segmentation and Gaussian noise) do not make sense in non-visual input domains, Feature Attribution methods use different input perturbations in non-visual domains. Apart from that, the saliency map methods discussed in this work can be directly applied to any agent with discrete action space. Continuous action spaces require further adjustments. Therefore, our findings that are not related to the input perturbation should still apply to DRL agents with discrete action spaces in non-visual domains. This includes the difference between analyzing the agent’s action value and state value estimation, as well as the parameter independence of relevance calculation of Noise Sensitivity.

## 9.7. Conclusion

This chapter compared five different perturbation-based saliency map approaches, measuring their dependence on the agent’s parameters and their fidelity to the agent’s reasoning. Our main findings are:

- Most of the approaches tested in this work do depend on the agent’s learned parameters. Only Noise Sensitivity showed less dependence on the learned parameters of the output layer. We empirically show that this is due to Noise Sensitivity’s original relevance calculation. Replacing this calculation with a calculation that only takes the analyzed action into account drastically increases the dependence on the parameters of the output layer.
- For value-based DRL agents, there are considerable differences between analyzing the agent’s action value and state value estimation. While this distinction is hidden within the agent’s output Q-values, future practitioners should be aware of which of the two they want to analyze and choose their saliency maps accordingly. To investigate how well saliency

maps for value-based DRL agents capture this distinction, we proposed an adjustment to existing input degradation metrics for image classifiers. In our tests, SARFA worked best to capture the action value while Occlusion Sensitivity and RISE were more suited for the state value.

- Depending on which perturbation method the approaches use, the resulting saliency maps only analyze how sensitive the agent is with regard to specific types of perturbation. While this seems obvious, it was true even for perturbation methods that utilized blurring specifically to reduce their dependence on a choice of occlusion color. In contrast to the action and state value distinction, this is not an inherent property of the DRL agents but might be seen as a flaw of current perturbation-based saliency map approaches. Our results demonstrate that there is still a need to further develop perturbation-based saliency map approaches. For now, researchers have to decide which types of perturbation are meaningful and interesting for their application. Based on this, they can choose an appropriate perturbation method – for example, by performing a parameter search similar to the one conducted in this work.

## **Part V.**

# **User Studies Evaluating the Effect of Local and Global XRL Approaches**

As Chapter 8 laid out, there are benefits to combining local and global explanations. While local explanations provide detailed information about individual decisions of an agent, they do not provide any information about its behavior in different contexts. Similarly, while global explanation methods provide a high-level view of a policy, they do not provide decision-specific insights. Due to the potential complementarity of such approaches, it is important to examine the effectiveness of combining them rather than studying each approach in isolation.

This part is based on three consecutive user studies [Huber et al., 2021b; Septon et al., 2023; Huber et al., 2023], in which we investigated the different and complementary benefits of integrating one global explanation method for DRL agents (strategy summarization) with three local explanation methods (saliency maps, reward decomposition, and counterfactual explanations).

For all those user studies, we used the ALE as test environment. The ALE is a common benchmark for state-of-the-art reinforcement learning algorithms [Bellemare et al., 2013; Dhariwal et al., 2017; Mnih et al., 2015; Wang et al., 2016b] and to test explanation methods for those algorithms [Amir and Amir, 2018; Greydanus et al., 2018; Lopuschkin et al., 2019; Weitkamp et al., 2019]. See Section 2.1.2 for details on this environment and why it is a useful benchmark. More specifically, we used the game Pacman (see Section 2.1.2.1 for a detailed description) since it is not as reaction-based as some other Atari games (e.g., Breakout or Enduro) and allows the RL agents to develop different strategies. Furthermore, no additional domain knowledge is necessary to understand Pacman, and the rules are not too complicated. This enables us to conduct studies with a wide range of participants by simply explaining the rules at the beginning of the study.

# 10. First Study: Strategy Summaries and Saliency Maps

In this chapter, we begin our evaluation of combined local and global explanation approaches for DRL agents by investigating the combination of HIGHLIGHTS summaries with LRP-argmax saliency maps described in Section 8.1. The chapter is based on the user study in our publication:

**Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

Strategy summaries (Section 4.2.3.1) show demonstrations of the agent’s behavior in a carefully selected set of world states. Saliency maps, in contrast, are used to show users what information the agent is attending to (Section 3.2.1).

As outlined in Chapter 8, the combination of saliency maps and HIGHLIGHTS is a very natural one. Firstly, the typical presentation of saliency maps as heatmap overlays lends itself to the typical presentation of strategy summaries in the form of videos. Moreover, both methods highlight important information: HIGHLIGHTS finds important states, while saliency maps identify important pixels within each state.

For the saliency map method, we chose LRP-argmax because of its selective nature, which is particularly important when the information is presented in the form of videos, and because of the advantageous properties of LRP, such as conserving the Q-value.

We evaluate the combination of global strategy summaries and LRP-argmax, as proposed in Section 8.1, in a user study in which we explore both the benefits of HIGHLIGHTS summaries and the benefits of adding saliency maps to strategy summaries. Specifically, we compare likelihood-based summaries (which select states for the summaries based on the likelihood of visiting them) and HIGHLIGHTS summaries, both with and without the addition of saliency maps. Study participants completed two types of tasks requiring the analysis of different agents trained to play the game of Pacman (Section 2.1.2.1): an agent comparison task in which they compare the performance of two agents, and an agent understanding task, in which they reflect on an agent’s strategy. We

chose those tasks to investigate whether the users trusted the right agent and to evaluate their mental models of the agents, respectively.

Our results show that participants who were shown HIGHLIGHTS summaries performed better on both tasks compared to participants who were shown likelihood-based summaries, and were also more satisfied with HIGHLIGHTS summaries. We find mixed results with respect to the benefits of adding saliency maps to summaries, which improved participants' ability to identify some aspects of the agents' strategies but, in most cases, did not lead to improved performance.

This chapter makes the following contributions:

- It evaluates the combination of global strategy summaries and local saliency maps in a user study, demonstrating the benefits of HIGHLIGHTS summaries and the potential benefits and limitations of local explanations based on saliency maps.
- It provides the first user study evaluation of LRP-based saliency maps for reinforcement learning. Despite its advantageous mathematical properties and promising user study results in classification tasks [Alqaraawi et al., 2020], there have been no user studies with LRP for RL agents prior to our study.

## 10.1. Study Design

### 10.1.1. Research Question

To evaluate our hypothesis that there are benefits to combining global strategy summaries and local saliency maps for explaining DRL agents, we conducted a user study. In this study, participants were asked to compare different agents and to reflect on the strategies of the agents based on the information they were shown. Next, we describe in detail the study design, the specific hypotheses we tested, and the metrics we used to evaluate the results.

### 10.1.2. Experimental Conditions

To evaluate the potential benefits of integrating global and local explanations and their relative importance, we assigned participants to four different conditions (summarized in Table 10.1). The first two conditions included only global information, while the remaining two conditions integrated local explanations as well.

**Table 10.1.:** The four study conditions in our first user study, comparing global strategy summaries with local saliency maps.

	Likelihood-based summaries	HIGHLIGHTS
No saliency maps	$L$	$H$
LRP saliency maps	$L+S$	$H+S$

- **Likelihood-based Summaries ( $L$ ):** The summaries in this condition consisted of states that the agent was likely to encounter during gameplay. To generate these summaries, we randomly select state-action pairs from the streams of the Pacman agents playing the game. Since each state encountered in the game had the same probability of being chosen, states that are encountered more frequently will be more likely to be included. Because of the random component of these summaries, it is possible that a single summary is, by chance, particularly good or particularly bad. Therefore, we generated ten different likelihood-based summaries and randomly assigned them to participants in this condition.
- **HIGHLIGHTS Summaries ( $H$ ):** In this condition, participants were shown summaries generated by the HIGHLIGHTS algorithm (Section 4.2.3.1). The specific implementation of this algorithm and the parameters we used for diversity are described in Section 8.1.
- **Likelihood-based Summaries+Saliency ( $L+S$ ):** These summaries included the same states as those shown in the  $L$  summaries, but each image was overlaid with a saliency map generated by the LRP-argmax algorithm (Section 6.1) as described in Section 8.1.
- **HIGHLIGHTS Summaries+Saliency ( $H+S$ ):** These summaries included the same states as those shown in the  $H$  summaries, where each image was overlaid with a saliency map generated by the LRP-argmax algorithm (Section 6.1) as described in Section 8.1.

We used a budget of  $k = 5$  for the summaries. That is, each summary included five base states chosen either based on likelihood or by HIGHLIGHTS, where for each state, we included a surrounding context window of ten states that occurred right before and after the chosen state. We used an interval size of ten states to prevent directly successive states in the summary.

The video creation and saliency map overlay process are described in detail in Section 8.1. All video summaries used in the study are available online.<sup>1</sup>

We note that we did not include a condition that shows only local explanations since, by definition, a local explanation is given for a specific state, forcing

<sup>1</sup>[https://github.com/HuTobias/HIGHLIGHTS-LRP/tree/master/Survey\\_videos](https://github.com/HuTobias/HIGHLIGHTS-LRP/tree/master/Survey_videos)

us to make some choice about which states to show (which means making a global decision). However, the  $L+S$  condition simulates a scenario where local explanations are shown for states selected based on the likelihood of the agent encountering them during gameplay.

### 10.1.3. Dependent Variables and Main Tasks

To evaluate participants' ability to differentiate between alternative agents and analyze their strategies, we trained Pacman agents that behave qualitatively differently. To this end, we modified the reward function used for training (similar to the approach used by Sequeira and Gervasio [2020]), resulting in three types of agents. We based all of those reward functions on the default ALE [Bellemare et al., 2013] reward function, which measures the increase in the in-game score between the first and last frame of a state (see Section 2.1.2).

- *Regular agent*: This agent was trained using the default reward function of the ALE
- *Power pill agent*: This agent was trained using a reward function that only assigned positive rewards to eating power pills<sup>2</sup>.
- *Fear-ghosts agent*: This agent used the default ALE reward function but was given an additional negative reward of  $-100$  when being eaten by ghosts, causing it to more strongly fear ghosts (which is implicitly learned by the other agents due to the lack of future rewards caused by being eaten).

To remove unnecessary magnitude, we divided all the rewards described above by the factor 10, such that a regular pill gives a reward of 1. Each agent was trained for 5 Million steps with the baselines [Dhariwal et al., 2017] implementation of the DQN algorithm described in Section 2.2. At the end of this training period, the best-performing policy is restored.

**Main Tasks.** We aimed to investigate three aspects related to the participants in the study: (1) the mental model of the participant about the agent, (2) participants' ability to assess agents' performance (appropriate trust), and (3) participants' satisfaction with respect to the explanations presented.

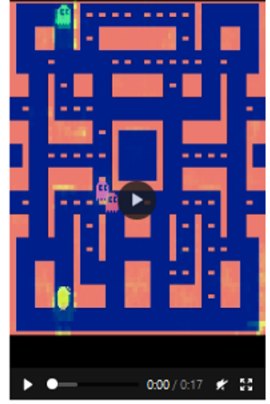
**Task 1: Eliciting Mental Models through Retrospection.** The examination of participants' mental models (see Section 3.3.2) and their correctness

---

<sup>2</sup>We achieved this by only giving the agent a reward if the increase in score was between 50 and 99. The range is necessary since Pacman is forced to eat at least one regular pill directly before it eats a power pill.





Task Description



Please briefly describe the strategy of the AI agent shown in the video above:

Based on this video, select the objects that you think were most important for the strategy of this particular agent.

Pacman 

Normal pills 

...

How confident are you that you chose the better agent?

Not at all confident        Very confident

Please briefly explain your selection:

**Figure 10.1.:** A sketch of the agent understanding task: participants were asked to analyze the behavior of each agent by providing a textual description of its strategy and identifying the objects that are most important to its decision-making. The full task can be seen in Appendix B.4.

helps to verify if explainable AI has been successfully applied [Rutjes et al., 2019; Arrieta et al., 2020]. To evaluate which mental models participants have formed about the agent’s behavior, we designed an **agent understanding task**. Here, we used a task reflection method inspired by prior studies [Anderson et al., 2019; Sequeira and Gervasio, 2020], which is recommended by Hoffman et al. [2018]. This task involved asking the participants to analyze the behavior of the three different AI agents: *Regular agent*, *Power pill agent*, and *Fear-ghosts agent*. The ordering of the agents was randomized. Specifically, participants were shown the video summary (according to the condition they were assigned to) and were asked to briefly describe the strategy of the AI agent (textual) and to select up to three objects that they thought were most important to the strategy of the agent (the possible objects were Pacman, power pills, normal pills, ghosts, blue ghosts, and cherries). They were also asked how confident they were in their responses and to justify their reasoning. Figure 10.1 shows a sketch of the agent understanding task.


**Task 2: Measuring Appropriate Trust through Agent Comparison.**

We use the term appropriate trust, based on the work of Lee and See [2004] who present a conceptual ‘trust in automation’ framework. They define appropriate trust as a well-calibrated trust that matches the true capabilities of a technical system (see also Section 3.3.2). We measure the appropriate trust using an **agent comparison task**. Here, the participants were shown summaries of two of the three agents at a time and were asked to indicate which agent performs better in the Pacman game (similar to tasks used in [Amir and Amir, 2018; Selvaraju et al., 2020]). They thus made three comparisons (*Regular agent Vs. Power pill agent*, *Regular agent Vs. Fear-ghosts agent* and *Power pill agent Vs. Fear-ghosts agent*). We do not ask the participants directly about their trust in the two agents shown. Instead, the participants have to choose one of the two agents that they would like to play on their behalf (see Figure 10.2). To objectively say which agent performed better, we computed the average in-game score of each trained agent during the simulations used for our summaries (Section 8.1). If the participants choose the correct agent, then they receive a bonus payment as described in the *Procedure* paragraph. This implicit question reveals which agent participants consider more reliable and qualified for the task. As in the retrospection task, they were asked to indicate their level of confidence and to provide a textual justification for their decision. The ordering of the three agent comparisons was randomized.

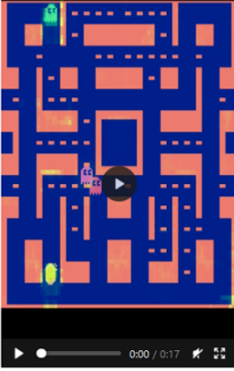
**Explanation satisfaction questions.** Miller et al. [2017] argue that the end users’ impressions about the agent should be queried and included in the evaluations of the explainable AI methods. This would ensure that the developed explanation methods are comprehensible not only to ML experts but also to end-users. We address this concern in our study by measuring participants’ subjective satisfaction. To this end, we used **explanation satisfaction ques-**

Task Description

Agent A



Agent B



Which of the Pacman agents do you choose to play on your behalf?

Agent A
  Agent B

How confident are you that you chose the better agent?

Not at all confident        Very confident

Please briefly explain your selection:

**Figure 10.2.:** A sketch of the agent comparison task: participants were asked to choose which agent they would like to play on their behalf (i.e., identify the better-performing agent) according to the two summary videos. The full task can be seen in Appendix B.4.

**tions** adapted from the questionnaire proposed by Hoffman et al. [2018] (see Section 3.3.2 for the original questionnaire). We did this separately for the agent understanding task (immediately after completing the three retrospection tasks) and for the agent comparison task (after completing the three comparisons), as we hypothesized there may be differences in the usefulness of the summaries for these two different types of tasks. Specifically, participants were asked the following questions using a 5-point Likert scale:

1. From watching the videos of the AI agents, I got an idea of the agents' strategies.
2. The videos showing the AI agents play contain sufficient detail about the agents' behavior.
3. The videos showing the AI agents play contain irrelevant details.
4. The videos showing the AI agents play were useful for *the task*. (only shown in groups  $L$  and  $H$ )
5. The gameplay scenarios shown in the videos were useful for *the task*. (only shown in groups  $L+S$  and  $H+S$ )
6. The green highlighting in the videos was useful for *the task*. (only shown in groups  $L+S$  and  $H+S$ )

We substituted *the task* with either *analyzing the agents' behavior* or *choosing the agent that performs better*, depending on the task they had just completed.

**Analysis.** We analyze the main hypotheses using the non-parametric Mann-Whitney test [McKnight and Najab, 2010], as our dependent variables are not normally distributed. We report effect sizes using rank biserial correlation [Tomczak and Tomczak, 2014]. Additionally, we report the mean values and the 95% confidence interval (CI) computed using the bootstrap method. In all plots, the error bars correspond to the 95% confidence intervals.

For evaluating the retrospection task, we use a scoring system where two of the authors involved in the training of the agents assigned a score to each item for each agent before the study started (see Appendix B.3 for details). For example, for the *Power pill agent*, which was only rewarded when it ate a Power pill, selecting the Power pill or Pacman increased the score by 1 point, and including any other item reduced the score by 1 point. Furthermore, selecting more than three items resulted in a score of zero since the participants were told to select a maximum of three items.

Inspired by Anderson et al. [2019], we use summative content analysis [Hsieh and Shannon, 2005] to evaluate participants' textual responses. An independent

coder (not one of the authors) classified responses to the questions “Please briefly describe the strategy of the AI agent shown in the video above” in the agent understanding task, and the question “Please briefly explain how you came to your selection” in both the agent understanding task and the agent comparison task. Each question was asked three times (once for each agent description or agent comparison), resulting in 402 answers per question. For the first question, the coder identified 67 different concepts in the answers. For example, the answer “The strategy of this Pacman agents seems to be to mainly avoid the ghosts as it eats the normal pills on the screen. Although it can be seen eating a power pill, the clip still does not show Pacman seeking out and eating the ghosts” was coded to “prioritizing normal pills”, “avoiding ghosts” and “do not care about blue ghosts”. We aggregated those concepts into 16 groups by combining similar concepts like “eating normal pills” and “prioritizing normal pills”.

To evaluate the correctness of participants’ answers, we implemented a simple scoring system. For each agent and for each answer group, we decided whether it is correct, irrelevant, or wrong, based on predefined ‘ground-truth’ answers that two of the authors, who were involved in the training of the agents, wrote for each agent before the study started. The exact groups and their assigned scores can be found in Appendix B.3 and the open-sourced code.

The answers to the second question regarding participants’ justifications of their responses were classified into six categories (the answer could be based on the game rules, the saliency maps, the gameplay, participants’ interpretation, and two categories for unjustified or unrelated justifications, which we grouped into one “unjustified” category) and an additional seventh category for the agent comparison task, that encoded that the user could not decide between the two agents and guessed.

We note that the classifications assigned by the coder are not mutually exclusive.

#### 10.1.4. Hypotheses

Overall, we hypothesized that HIGHLIGHTS summaries will be more useful than likelihood-based summaries in both the retrospection and agent comparison tasks and that adding saliency maps will further improve participants’ performance. More specifically, we state the following hypotheses:

- H1: For both tasks, participants shown summaries generated by HIGHLIGHTS will perform better than participants shown likelihood-based summaries. That is, performance in  $H$  will be better than performance in  $L$  and performance in  $H+S$  will be better than performance in  $L+S$ . We expect HIGHLIGHTS summaries to be more useful as they demonstrate

the agent’s behavior in more meaningful states, which should help both in identifying which agent performs better (in line with prior findings [Amir and Amir, 2018; Huang et al., 2018]), as well as in determining whether an agent is capable of performing well in certain scenarios [Huang et al., 2018]. We expect similar effects in terms of participants’ explanation satisfaction in each task.

- H2: For both tasks, adding saliency maps will improve participant’s performance and satisfaction. That is, we expect the performance in  $L+S$  will be better than in  $L$  and similarly, that performance in  $H+S$  will be better than in  $H$ . Here, too, we expect similar effects in terms of participants’ explanation satisfaction in each task. We expect this to be the case as the saliency maps allow people to see not only what actions the agent chooses but also what information it attends to. Previous studies also found positive effects of saliency maps on participants’ mental models [Anderson et al., 2019; Alqaraawi et al., 2020] and on their ability to choose the better-performing prediction model [Selvaraju et al., 2020]
- H3: The effect of the summary generation method on satisfaction and performance will be greater than that of the inclusion of saliency maps in the agent comparison task. That is, we expect that global information will be more crucial for identifying the better-performing agent, as it explicitly demonstrates how the agents act.
- H4: The effect of adding saliency maps on satisfaction and performance will be stronger than that of the summary generation method in the agent understanding task. Since saliency maps explicitly show what information the agent attends to, we hypothesize it will contribute more to identifying the agent’s strategy. However, this is complicated by the fact that likelihood-based summaries might not include interesting scenarios, making saliency maps less helpful in this case. Therefore, our more specific hypotheses are:
  - H4.1: Participants in the saliency conditions will be more likely to identify Pacman, the main source of information for our agents, as an important object.
  - H4.2: Participants in the HIGHLIGHTS conditions will be more likely to identify objects that relate to agent goals, such as power pills and blue ghosts. Therefore, they will also more accurately describe the agents’ strategies.

### 10.1.5. Procedure and Compensation

Participants were first asked to answer demographic questions (age and gender) and questions regarding their experience with Pacman and their views on AI. Then, they were shown a tutorial explaining the rules of the game Pacman and were asked to play the game to familiarize themselves with it. To verify that participants understood the rules, they were asked to complete a quiz and were only allowed to proceed with the survey after answering all questions correctly. After completing the quiz, they were given information and another quiz regarding the Pacman agent video summaries. In conditions  $L+S$  and  $H+S$ , this also included an explanation and a quiz about saliency maps. Then, they proceeded to the main experimental tasks. See Appendix B.4 for the complete questionnaire. Participants were compensated as follows: they received a \$4 base payment and an additional bonus of 10 cents for each correct answer. The study protocol was approved by the Institutional review board at the Technion.

### 10.1.6. Participants

We recruited participants through Amazon Mechanical Turk ( $N = 133$ ). The majority of participants were between the ages of 25 and 44, 47 females). Participation was limited to people from the US, UK, or Canada (to ensure a sufficient English level) with a task approval rate greater than 97%. Each participant was randomly assigned to one of the four conditions ( $L:33$ ,  $H:33$ ,  $L+S:34$ ,  $H+S:33$ ). Since saliency maps are not designed for color-blind people, the participants were also asked if they were color-blind and stopped from participating if they were.

To make sure that the participants involved in our analysis did, in fact, watch the videos of the agents, we recorded whether they clicked play on each video in addition to how often each video was paused. We did not force them to watch the videos to filter out participants who would have just pressed play to avoid the forcing mechanism. Since we saw from the raw data that some participants only stopped watching videos after the agent understanding task, we checked each task separately. As a heuristic to measure how attentively a user watched the videos of a task, we took the sum of pauses of the videos in this task, where watching a video until the end was recorded as a pause and not clicking play was counted as  $-1$  pause. Based on this heuristic, we removed all participants from the agent understanding task who did not have at least three pauses (5 participants) and all participants from agent comparison task who did not have at least six pauses (11 participants). The number of necessary pauses in each task is equal to the number of videos in this task.

**Table 10.2.:** Summary of all significance tests (calculated with Mann-Whitney tests). The \* denotes statistically significant differences and † denotes a p-value < 0.1.

Task	Variable	Effect of strategy summarization:		Effect of saliency maps:	
		$H > L$	$H+S > L+S$	$L+S > L$	$H+S > H$
agent understanding task	score	0.008*	$3.3e - 05^*$	0.965	0.514
	satisfaction	0.021*	0.035*	0.677	0.710
	text score				0.088†
agent comparison task	score	0.014*	0.180	0.062†	0.307
	satisfaction	0.147	0.235	0.627	0.833

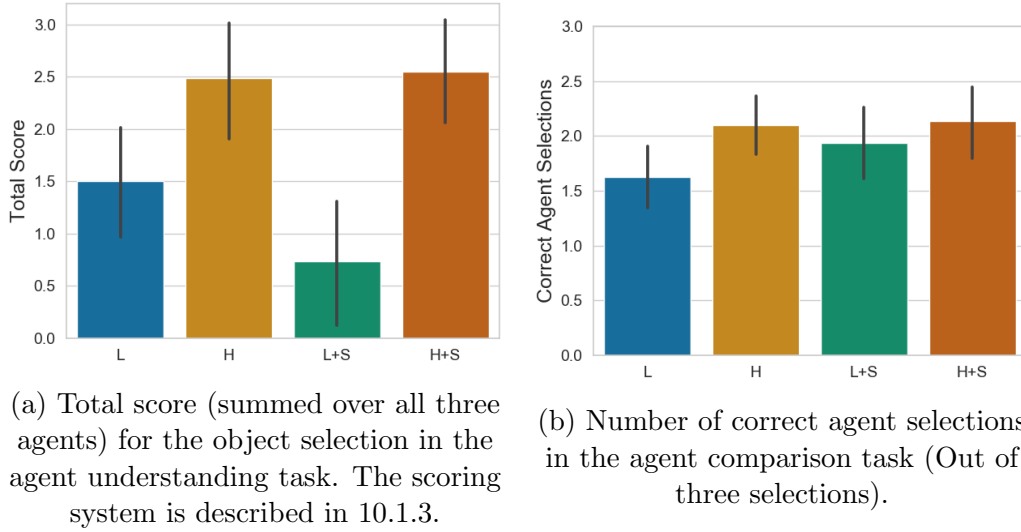
**AI and Pacman Experience.** We verify that participants in different conditions did not differ much in their AI experience and views and in their experience with the game Pacman. To this end, we asked them when they played Pacman for the last time. Across all four conditions, the majority of participants answered: “I played Pacman more than 5 years ago”. After receiving a short description of what AI is (using a formulation based on [Russell and Norvig, 2016]), 104 participants stated that they had experience with AI. The exact kind of experience ranged from “I know AI from the media” (78 participants) to “I do research on AI related topics” (14 participants). On average, the users had a positive attitude towards AI (mean of 3.95 on a 5-point Likert scale). There were no meaningful differences between the conditions (see Appendix B.1 for more details).

## 10.2. Results

This section reports the results of our first study. First, it assesses the main hypotheses of that study (H1–H4) (results summarized in Table 10.2). Then, it provides a descriptive analysis of additional variables such as participants’ confidence and analysis of mistakes.

**(H1) Participants shown HIGHLIGHTS summaries performed better than participants shown likelihood-based summaries.** Participants’ correctness rates for the agent comparison task are shown in Figure 10.3(b). These results support H1, which states that HIGHLIGHTS summaries will lead to improved performance in both the agent comparison task and the agent understanding task. The exact definition of performance per task is described in more detail in Section 10.1.3. Specifically, in the agent comparison task we find that participants in condition  $H$  significantly outperformed participants in condition



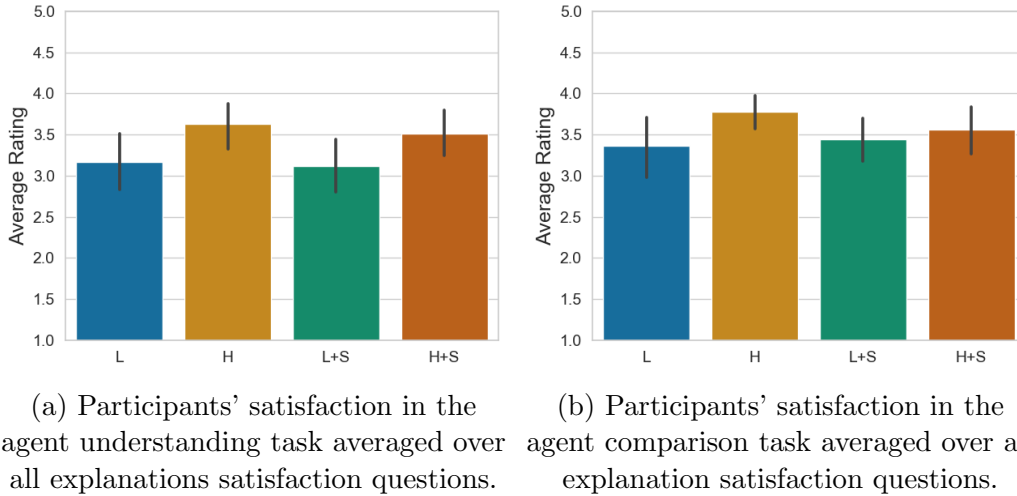


**Figure 10.3.:** Comparison of participants’ average performance in each task, by condition. Participants in the HIGHLIGHTS conditions  $H$  and  $H+S$  outperformed the likelihood-based conditions  $L$  and  $L+S$ . Saliency maps only had a slight positive effect when added to likelihood-based summaries in the agent comparison task.

$L$  ( $H$ : mean=2.1, 95% CI=[1.83, 2.33],  $L$ : mean= 1.63, 95% CI=[1.34, 1.91], Mann-Whitney test  $U=334.5$ ,  $p = 0.014$ ,  $r_{rb}=0.3$ )<sup>3</sup>. While participants in the  $H+S$  condition achieved higher mean correctness rates than participants in the  $L+S$  condition, this difference is not statistically significant ( $H+S$ : mean=0.71, 95% CI=[0.6, 0.82],  $L+S$ : mean=0.65, 95% CI=[0.54, 0.75], Mann-Whitney test  $U=391$ ,  $p = 0.180$ ,  $r_{rb}=0.13$ ). Similarly, participants’ average explanation satisfaction ratings, shown in Figure 10.4(b), indicate that participants in condition  $H$  were more satisfied with the videos they received than the other participants. However, this difference is not significant (see Table 10.2).

With respect to participants’ performance during the agent understanding task, we find even stronger results (Figure 10.3(a)) than in the agent comparison task, further supporting H1. Here too, participants in condition  $H$  obtained a higher score in the object selection sub-task than participants in condition  $L$  ( $H$ : mean=2.5, 95% CI=[1.89, 3.03],  $L$ : mean=1.5, 95% CI=[0.92, 2.06], Mann-Whitney test  $U=346.5$ ,  $p = 0.008$ ,  $r_{rb}=0.34$ ) and participants in the  $H+S$  condition received a higher score than participants in the  $L+S$  condition ( $H+S$ : mean=2.55, 95% CI=[2.02, 3.06],  $L+S$ : mean=0.73, 95% CI=[0.13, 1.31], Mann-Whitney test  $U=206.5$ ,  $p = 0.00003$ ,  $r_{rb}=0.58$ ). We found analogous significant differences in participants’ explanation satisfaction during the agent un-

<sup>3</sup>Here 95% CI is the 95% confidence interval and  $r_{rb}$  is Rank biserial correlation.

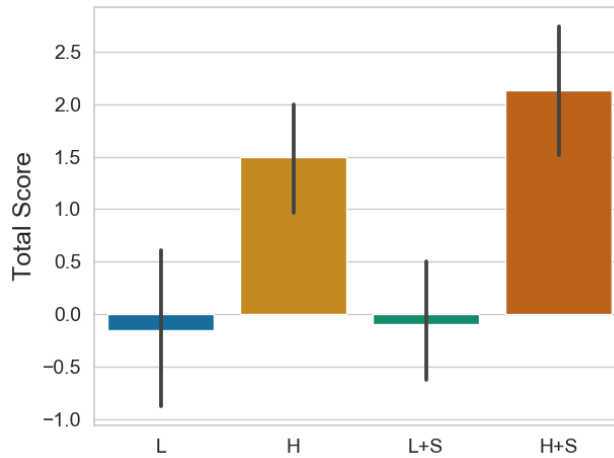


**Figure 10.4.:** Comparison of participants' average explanation satisfaction in each task, by condition. Each participant rated their agreement with several statements adapted from the explanation satisfaction questions proposed by Hoffman et al. [2018] on a 5-point Likert scale (see Section 10.1.3). The participant's final rating was averaged over all those ratings, reversing the rating of the negative statements. Overall, participants in the HIGHLIGHTS conditions  $H$  and  $H+S$  rated the explanations highest.

derstanding task (Figure 10.4(a)). Here, participants in condition  $H$  were more satisfied than participants in condition  $L$  ( $H$ : mean=3.63, 95% CI=[3.35, 3.88],  $L$ : mean=3.17, 95% CI=[2.82, 3.5], Mann-Whitney test  $U=373.0$ ,  $p = 0.021$ ,  $r_{rb}=0.29$ ) and participants in the  $H+S$  condition were more satisfied than participants in the  $L+S$  condition ( $H+S$ : mean=3.52, 95% CI=[3.25, 3.78],  $L+S$ : mean=3.12, 95% CI=[2.81, 3.43], Mann-Whitney test  $U=364.5$ ,  $p = 0.035$ ,  $r_{rb} 0.27$ ).

**(H2) Adding saliency maps improved performance in some areas depending on the task.** There were no significant differences supporting our second hypothesis H2, which predicted that adding saliency maps would improve participants' performance in both tasks. Nevertheless, we report two positive effects of saliency maps that are only marginally<sup>4</sup> significant and which might guide future research in this area. For the agent comparison task, we find that the saliency maps only improved performance when added to likelihood-based summaries ( $L$ : mean=0.54, 95% CI=[0.45, 0.64],  $L+S$ : mean=0.65, 95% CI=[0.54, 0.75], Mann-Whitney test  $U=390.5$ ,  $p = 0.062$ ,  $r_{rb}=0.21$ ). Figure 10.3(a) shows that

<sup>4</sup>In accordance with convention [Vogt, 2005], we use *marginally significant* to describe  $0.05 \leq p < 0.1$

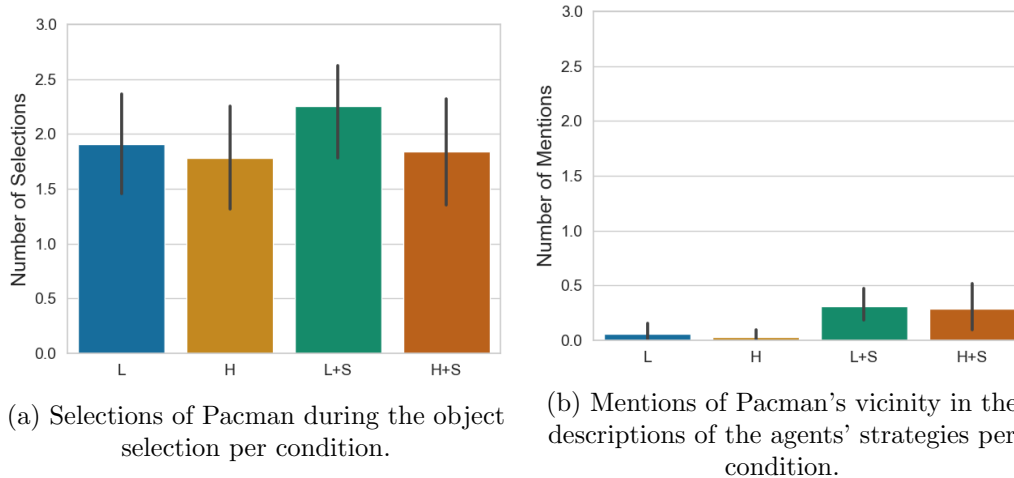


**Figure 10.5.:** Participants’ total score for their textual descriptions of the agents’ strategies during the agent comparison task (summed over all three agents). The scoring function is described in 10.1.3. The descriptions of participants in the HIGHLIGHTS conditions  $H$  and  $H+S$  received a higher score than those of participants in the likelihood-based conditions. The addition of saliency maps ( $H+S$ ) slightly improved this effect further.

the saliency maps did not help participants identify the most important objects in the agent understanding task. However, the summative content analysis of participants’ textual descriptions of the agents’ strategies, shown in Figure 10.5, indicates that saliency maps helped participants to correctly describe how the agents use those objects. The descriptions of the agents’ strategies written by participants in condition  $H+S$  received a higher score than the ones by participants in condition  $H$  ( $H$ : mean=1.50, 95% CI=[0.97, 2.0],  $H+S$ : mean=2.13, 95% CI=[1.55, 2.71], Mann-Whitney test  $U=400$ ,  $p = 0.088$ ,  $r_{rb}=0.195$ ).

**(H3 + H4) The effect of the summary generation method was greater than that of adding saliency maps.** We hypothesized that the summary generation method would affect the performance of participants more than the addition of saliency maps in the agent comparison task (H3) and that the saliency maps will have a greater effect than the summary method in the agent understanding task (H4). The study results support H3: we found that participants shown HIGHLIGHTS summaries significantly outperformed participants shown likelihood-based summaries in the agent comparison task while adding saliency maps only improved performance for the likelihood-based summaries, and to a lesser extent.

For selecting the most important objects for the agent’s strategy in the agent understanding task, the addition of saliency maps did not improve perfor-



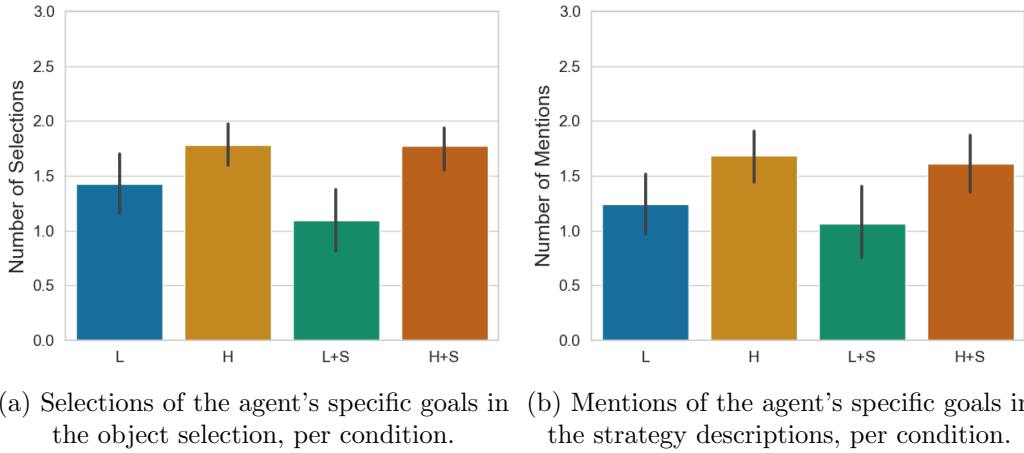
**Figure 10.6.:** The average number of times that participants correctly selected Pacman during the object selection (a) or referred to its vicinity in their textual descriptions (b) of the agents' strategies (sum over all three agents). The results indicate that saliency maps help the participants to identify what information the agents use.

mance, while HIGHLIGHTS summaries did improve performance compared to the likelihood-based summaries. Therefore, we reject H4, even though the results shown in Figure 10.5 indicate that saliency maps improved the textual descriptions of the agent's strategy written by participants in  $H+S$  compared to  $H$ .

In line with Hypothesis H4.1, Figure 10.6 indicates that the improvement of the descriptions of the agents' strategies mainly stems from participants in the saliency groups  $L+S$  and  $H+S$  identifying that the agent mostly paid attention to the vicinity of Pacman. This effect was not as strong in the object selection question since it did not capture the participants' reasoning.

Sub-Hypothesis H4.2 stated that strategy summarization would help participants identify the goals of the agents. The results shown in Figure 10.7 support this hypothesis since participants in the HIGHLIGHTS conditions  $H$  and  $H+S$  identified the correct goals of the agent more often.

**Participants with AI expertise benefited more from saliency maps.** To investigate the effect of AI expertise, we looked at participants with advanced AI experience. In total, 19 participants stated that they completed at least one course on AI or did research in AI (the latter being more common). These 19 participants were divided among conditions as follows: 5 per condition, except for condition  $L+S$ , which had only 4 (see Appendix B.1 for more information). For the agent understanding task (Figure 10.8 a)), their results were in line with

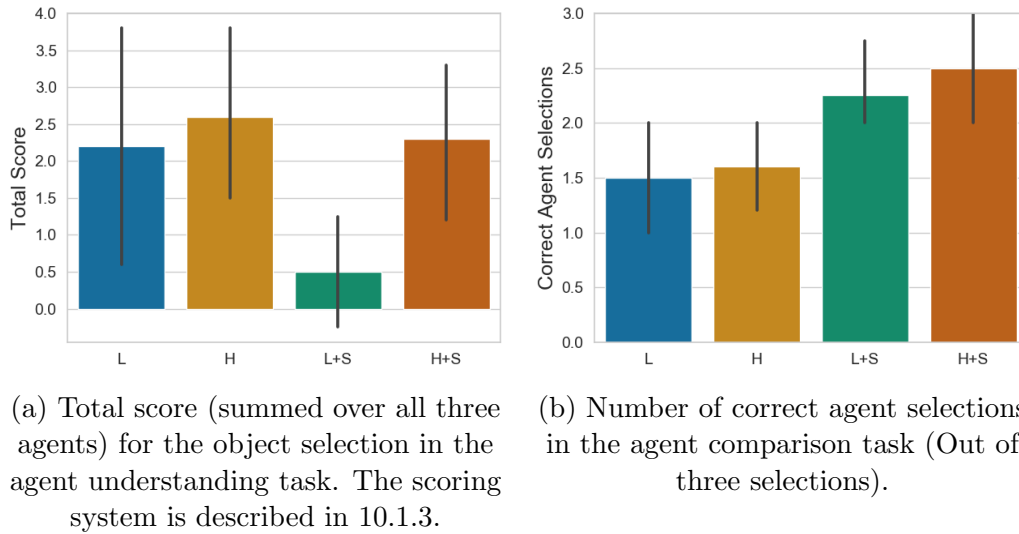


**Figure 10.7.:** The number of times that participants identified the agent's specific goal in the object selection (a) and strategy description (b) components of the agent understanding task. The results are in line with Hypothesis H4.2 that strategy summarization helps to identify the agents' goals.

the results of the general population, albeit a little bit higher in condition  $L$ . For the agent comparison task (Figure 10.8 b)), two participants did not watch the videos, leaving 17 participants with advanced AI experience (4 per condition except for 5 in the  $H$  condition). Compared to the general population (Figure 10.3 b)), the participants with AI expertise performed better in conditions  $L+S$  (mean = 2.25) and  $H+S$  (mean = 2.5) but worse in condition  $H$  (mean=1.6). Moreover, the participants with AI expertise actually performed better in the saliency map conditions  $L+S$  and  $H+S$  than in the conditions without saliency maps ( $L$  and  $H$ ). Since these observations were not part of our initial hypotheses and we do not have a sufficient number of participants with AI expertise, we did not investigate them further.

**Participants' Justifications.** Across all groups, most participants mainly based their justifications on the agents' gameplay (Figure B.9). In the saliency conditions, most participants did not mention the saliency maps in their justifications. On average, less than one out of three justifications in  $H+S$  and in  $L+S$  referred to the green highlighting during the agent understanding task. During the agent comparison task, even fewer participants mentioned them (see Figure B.10 for more details).

Another interesting point we found in participants' justifications during the agent understanding task is that participants in condition  $H$  gave more unjustified explanations than any other condition ( $H$ : mean=0.66, compared to the second-highest condition  $L+S$ : mean=0.38 ). This is just an observation and



**Figure 10.8.:** The average per task performance of participants with AI expertise, by condition.

did not repeat in the agent comparison task, but it might be interesting to investigate further in the future. The values for all conditions can be seen in Figure B.11.

**Participants’ Confidence and Viewing Dynamics.** In addition to the main metrics used in our study, we further measured participants’ confidence (in particular, whether they were more confident when they answered correctly) and their viewing dynamics of the summaries (time and number of pauses). However, apart from a slight positive effect for the participants in condition *H*, there were no interesting differences in the three aforementioned variables (see Figure B.6 to B.8 and Appendix B.2 for additional details).

### 10.3. Discussion & Future Work

With the increasing use of RL agents in high-stakes domains, there is a growing need to develop and understand methods for describing the behavior of these agents to their human users. In this chapter, we explored the combination of global information describing agent behavior, in the form of strategy summaries, with local information in the form of saliency maps. To this end, we augmented HIGHLIGHTS summaries (Section 4.2.3.1), which select important and diverse states (adapted to DQN agents), with saliency maps generated using the LRP-argmax algorithm (Chapter 6).

We implemented the combined approach in the Atari Pacman environment

and evaluated the separate and joint benefits of showing users global and local information about the agent. We used two types of tasks: an agent understanding task about the agent’s strategy and an agent comparison task.

**Strategy Summarization.** The results of this study reinforce prior findings [Amir and Amir, 2018] showing that summaries generated by HIGHLIGHTS lead to significantly improved performance of participants in the agent comparison task compared to likelihood-based summaries, and show that this result generalizes to RL agents based on neural networks. Furthermore, we show that HIGHLIGHTS summaries were more useful for analyzing agent strategies and were preferred by participants. Overall, in our study, the choice of states that were shown to participants was more important than the inclusion of local explanations in the form of saliency maps.

**Limitations of Saliency Maps.** With respect to the addition of saliency maps, we found mixed results. In contrast to previous studies about saliency maps for image classification tasks, which found weak positive effects for saliency maps [Alqaraawi et al., 2020; Selvaraju et al., 2020], there were no significant differences between the saliency and non-saliency conditions in our study. When examining participants’ answer justifications, we observed that most participants did not mention utilizing the saliency maps, which may provide a partial explanation for their lack of contribution to participants’ performance. Especially in the agent comparison task, participants seldom mentioned the saliency maps even though there was a marginally significant difference between the performance of participants in condition  $L$  and in condition  $L+S$ . Participants’ comments also reflect their dissatisfaction with saliency maps, e.g., “I do not believe that the green highlighting was useful or relevant” and “The green highlights didn’t seem to help much”. This suggests that saliency maps in their current form may not be accessible enough to the average user.

Based on the comments from the participants and in-depth feedback we received in pilot studies, we note some possible accessibility barriers. First, when saliency maps are shown as part of a video, it may be difficult for users to keep track of the agent’s attention, compared to displays of static saliency maps, as done in previous user studies [Selvaraju et al., 2020; Anderson et al., 2019] [Alqaraawi et al., 2020]. For instance, one participant reported that “[i]t wasn’t so easy to see the green area, it needed to be bigger or more prominent to be of more use.” We tried to take measures against this by using a selective saliency map generation algorithm (LRP-argmax) and interpolating between selected saliency maps (see Section 8.1) to reduce the amount of information, as well as allowing participants to pause the video at any time. However, this does not seem to have been enough.

Second, participants were not accustomed to interpreting saliency maps, which can be non-intuitive to non-experts. One participant even commented that “[he/she] feel[s] as though this came with somewhat of a learning curve”. In our pilot studies, we noticed that people who were familiar with reinforcement learning or deep learning could more easily interpret saliency maps than those who were not. For example, some participants said that they thought the agent was good when its attention was spread to different areas because they inferred it considered more information, while in fact, the agent was attending to different regions because it did not yet learn what the important information is. Similarly, one study participant commented: “...I don’t know if I would prefer an AI that ‘looked’ around more at the board, or focused more in a small area to accomplish a task”. Similar effects can be observed when examining the participants with AI expertise in our main study. Participants with AI expertise in the saliency conditions  $L+S$  and  $H+S$  received higher scores in the agent comparison task than the general population, while the participants with AI expertise in the  $H$  and  $L$  conditions received comparable or even lower scores than the general population. Even though this is only a small group, it further indicates that saliency maps, in the form of raw heat maps, might be better suited for debugging purposes than for actual end-users. It is possible that prior studies, which used raw saliency maps for interpreting image classification [Alqaraawi et al., 2020; Selvaraju et al., 2020], did not encounter this problem due to the more intuitive nature of the task. Interpreting a visual highlighting for image classification only requires identifying objects that contributed to the classification, while in RL, there is an added layer of complexity as interpretation also requires making inferences regarding how the highlighted regions affect the agent’s long-term sequential decision-making policy.

Finally, while the sanity checks reported in Section 6.4 showed that our saliency maps do analyze what the network learned, they were also found to be indifferent to specific actions. Since prior studies have shown that users find class discriminatory explanations more useful for understanding agents’ decisions [Goudet et al., 2018; Lopez-Paz et al., 2017; Byrne, 2019], the lack of discrimination between certain actions can be detrimental to the usefulness of saliency maps.

**Potential of Saliency Maps.** Regarding the potential of saliency maps, we made encouraging observations. Even though saliency maps did not significantly increase participants’ scores in the simple object selection part of the agent understanding task, they did result in improved scores in the textual strategy description. The difference between our HIGHLIGHTS conditions  $H+S$  and  $H$  is similar to the one observed by Anderson et al. [2019] ( $p=0.086$  compared to our  $p=0.088$ ), who also evaluated participants’ mental models for RL agents



utilizing a strategy description task. The poor result of condition  $L+S$  can be explained by the fact that Anderson et al. manually chose meaningful states, which we only did with our global explanation method in the HIGHLIGHTS conditions.

A possible reason for the difference between the object selection and the strategy description sub-tasks is the higher complexity of strategy descriptions. It requires participants to not only identify the correct objects but also to describe how they are used. Under this assumption, the increased performance of participants in condition  $H+S$  suggests that saliency maps were useful for putting the objects in the correct context. For example, participants’ textual descriptions showed that, while the non-saliency groups know that Pacman is important (most likely based on the fact that it is important for them as players), they did not identify it as a central source of information for the agent.

Second, we observed in the agent comparison task that saliency maps alone improved participants’ ability to place appropriate trust into different agents when comparing conditions  $L$  and  $L+S$ . The performance in condition  $L+S$  was comparable to the performance of participants in the HIGHLIGHTS conditions,  $H$  and  $H+S$ . This indicates that there is valuable information for this kind of task within saliency maps. The lack of improvement of condition  $H+S$  compared to  $H$  might be explained by the accessibility issues of saliency maps mentioned earlier. When presented with strategy summaries, participants may have had less reason to rely on the non-intuitive saliency maps.

**Combination of Local and Global Explanations.** It is important to note that the positive effects of saliency maps in the agent understanding task were only visible in the HIGHLIGHTS condition  $H+S$ , reinforcing our claim that the choice of states is crucial for explaining RL agents. Therefore, even if the limitations of saliency maps mentioned above are addressed, the potential benefits might only be visible and likely reinforced by a combination with strategy summarization techniques. We note that studies that evaluate local explanations typically implicitly make a global decision about which states to present local explanations for [Anderson et al., 2019; Madumal et al., 2020]. Our results suggest that this implicit choice may have a substantial impact on participants’ understanding of agent behavior.

In the agent understanding task, we observed that local explanations in the form of saliency maps were useful for identifying what objects the agent attends to (see Figure 10.6), while strategy summaries were more useful for identifying the agent’s goals (see Figure 10.7). This was reflected by participants’ utterances such as: “The agent seemed to be paying attention to the area directly in front of it and partly to the areas directly to each side.” and “Pacman wanted those ghosts! His goal was to move as fast as he could towards them.” and suggests

that the two approaches are indeed complementary. The local saliency maps contribute to users’ understanding of the agents’ *attention*, as they reflect the information the agent attends to, while strategy summaries contribute to users’ understanding of the agent’s *intentions*, as they reflect how the agent acts.

Taken together, our results suggest that there is potential for a combined explanation framework in the future if the accessibility issues of saliency maps are addressed.

**Study Limitations.** Our study has several limitations. First, we used a single domain in our user study. However, other recent work has used strategy summaries similar in spirit to HIGHLIGHTS in another domain [Sequeira and Gervasio, 2020], and several works have used saliency maps in other domains (e.g., several Atari games, including Pong and Space Invaders, were used by Greydanus et al. [2018]).

Second, while our combined explanation approach is easily adaptable to other global explanation methods, which choose an informative subset of states, and local methods, which highlight relevant information in those states, our study only explored one combination of a particular global explanation method and a particular local explanation method. We chose the HIGHLIGHTS summary method since strategy summary approaches that are based on policy reconstruction require making various assumptions about people’s computational models [Lage et al., 2019]. We chose saliency maps as a local method both because it is visual and thus can be integrated with a visual summary and also because other methods typically require additional models or assumptions (e.g., causal explanations [Madumal et al., 2020] require a causal graph of the domain). The specific choice of the LRP-argmax algorithm was motivated by its selectivity, which reduces the amount of information that participants have to process. The accessibility problems of saliency maps we identified were mainly related to the presentation of the information. This indicates that simply highlighting how relevant parts of the input are for the prediction of an agent is insufficient, even when based on other saliency map algorithms.

Finally, it is not yet possible to obtain actual ground truth about an agent’s reasoning for the agent understanding task. To approximate this, we trained three agents with different optimization goals (i.e., reward functions) and checked whether the participants were able to differentiate between them, similar to [Sequeira and Gervasio, 2020]. We note that it is possible that the agents do not pay attention to the objects that correspond to their goals (e.g., the power pills) within the visual input in a way a human would. In fact, our results indicate that the agents are not paying direct attention to their goals since the saliency maps did not help users identify the differences in the goals. Creating agent variations in ways that do not depend on different optimization goals might be

better suited to show the potential benefits of saliency maps.

## 10.4. Conclusion

To evaluate the combination of global strategy summaries and local saliency maps proposed in Chapter 8, as well as the contribution of each explanation type, we conducted a user study. Hereby, we examined participants' mental models through an agent understanding task and used an agent comparison task to investigate whether their trust was appropriate given the agents' capabilities.

Regarding the usefulness of *global strategy summaries*, our results show that HIGHLIGHTS summaries (1) help to establish appropriate trust in agents based on neural networks (extending prior results about classic RL agents [Amir and Amir, 2018]) and (2) improve participants' mental models of those agents.

The evaluation of *local explanations* in the form of LRP saliency maps reveals strengths as well as weaknesses. On the one hand, our analysis shows that reinforcement learning comes with additional usability challenges not present in previously evaluated image classification tasks. First, presenting saliency maps on videos instead of static images [Anderson et al., 2019; Alqaraawi et al., 2020] overwhelms users with a lot of information in a short amount of time and increases the risk of overlooking crucial information. Second, compared to more intuitive image classification tasks [Alqaraawi et al., 2020; Selvaraju et al., 2020], the average user lacks the experience to correctly infer how the highlighted regions affect the agent's long-term sequential decision-making.

On the other hand, the results indicate that saliency maps have the potential to (1) extend users' mental models beyond strategy summaries by providing insight into what information the agent used and (2) improve users' ability to choose the better agent even with likelihood-based summaries.

Taken together, the results support a combination of local and global explanations, since participants in the combined explanation condition received the highest scores during our survey. However, our evaluation suggests that simply highlighting pixels that are relevant for the agent's decision is insufficient for RL agents and that more work is needed to increase the accessibility of saliency maps.

# 11. Second Study: Strategy Summaries and Reward Decomposition

This chapter evaluates the benefits of integrating reward decomposition (Section 4.1.3.1), a local explanation method that exposes which components of the reward function influenced a specific decision, and HIGHLIGHTS (Section 4.2.3.1), a global explanation method that shows a summary of the agent’s behavior in decisive states. It is based on the Pacman user study in our publication:

Yael Septon, **Tobias Huber**, Elisabeth André, and Ofra Amir [2023]. “Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection - 21st International Conference*. Vol. 13955. Lecture Notes in Computer Science. Springer, pp. 320–332. DOI: 10.1007/978-3-031-37616-0\_27

Our first user study (Chapter 10) investigated the combination of local and global explanation methods by integrating strategy summaries with saliency maps. This study showed that the combination of local and global explanations is promising. However, the saliency maps we used as local explanations were lacking. One potential reason for this is that saliency maps are a post-hoc explanation technique that is created after the RL agents are fully trained. Recent literature by Rudin [2019] and our previous work described in Chapter 9 suggest that such post-hoc explanations do not always faithfully reflect the agent’s learned decision model. Therefore, we explore the use of reward decomposition, an intrinsic explanation method built into the agent’s decision model, as the local explanation method in this chapter.

We conducted a user study in which participants were randomly assigned to one of four different conditions that vary in the combination of global and local information: (1) being presented or not presented with a local explanation (reward decomposition), and (2) being presented with a global explanation in the form of a HIGHLIGHTS policy summary (Section 4.2.3.1) or being presented with frequent states the agent encounters (a baseline for not providing global

explanations). Similar to the first study (Chapter 10), we trained agents that varied in their priorities by modifying the reward function. Participants were asked to determine the priorities of these agents based on the explanations shown in their assigned study condition.

## 11.1. Study Design

### 11.1.1. Research Question and Hypothesis

In this study, we evaluate the benefits of integrating HIGHLIGHTS (Section 4.2.3.1) with reward decomposition (Section 4.1.3.1) as well as their respective contributions to users’ understanding of agents’ behavior. To this end, we asked participants to evaluate the preferences of different agents. We hypothesized that the combined explanations would best support participants’ ability to correctly identify agents’ preferences and that both the local and global explanations would be better than the baseline information.

### 11.1.2. Dependent Variables and Main Task

Similar to the agent understanding task in the first study (Section 10.1), the participant’s task was to assess the preferences of three different agents.

To this end, we trained three qualitatively different Pacman (Section 2.1.2.1) agents using the HRA reward decomposition method described in Section 4.1.3.1. All three agents use the DQN network architecture (Section 2.2). We implemented reward decomposition by sharing the convolutional layers but training individual fully connected layers for each reward component.<sup>1</sup> Based on the rules of Pacman, we used four different reward components for the RL agent controlling Pacman. Similar to the first user study (Chapter 10), the reward components are based on the ALE reward function (Section 2.1.2) but divided by 10 to remove unnecessary magnitude. The agent receives a reward of 1 for eating normal pills (NP) and a reward of 5 for eating Power Pills (PP). Additionally, after eating a PP, the ghosts turn blue, and Pacman can eat them. The agent receives a reward of 20, 40, 80, or 160 for each blue ghost (BG) it eats successively. Finally, the agent receives a reward of -10 for dying. To get agents with distinct strategies, we used different weights for the reward components (see Table 11.1). Each agent was trained for 5 million steps. In the Pacman environment, the values of the individual rewards do not directly correlate to the agents’ preferences. For example, the labyrinth contains a huge amount of normal pills compared to power pills and ghosts. Therefore, the agent with no

---

<sup>1</sup>Our implementation can be found online: [https://github.com/hcmlab/baselines/tree/reward\\_decomposition](https://github.com/hcmlab/baselines/tree/reward_decomposition)

**Table 11.1.:** How each of the reward components was weighted for our reward decomposition Pacman agents.

	NP weights	PP weights	BG weights	Dying
Normal Pill Agent	1	1	1	1
Power Pill Agent	0.01	1	0.01	0.01
Blue Ghost Agent	0.1	0.1	10	0.01

specific reward component weights focuses very strongly on normal pills even though the reward value for individual normal pills is the lowest. To determine what the agents preferred, we observed the Q-values and actions of each fully trained agent for several full games before running the experiment. In total, we trained three different Pacman agents with the following preferences:

- Normal Pill Agent - Highest preference for eating normal pills, next eating power pills, and lastly eating blue ghosts
- Power Pill Agent - Highest preference for eating power pills, next eating normal pills and eating blue ghost has the same preference
- Blue Ghost Agent - Highest preference for eating blue ghosts, next eating normal pills, lastly eating power pills.

Participants were asked to rank which of each pair of reward components (e.g., eating power pills vs. eating normal pills) the agent prioritizes or whether it is indifferent between the two options. If participants have a correct mental model of the agents' strategy, they should be able to rank the different reward components according to the agents' priorities. To avoid learning effects, the ordering of the agents was randomized.

Participants were then asked to rate their confidence in each of their answers on a Likert scale from 1 ("not confident at all") to 5 ("very confident") and to describe their reasoning in a free-text response. Lastly, participants rated their agreement on a 7-point Likert scale with the following items adapted from the explanation satisfaction questionnaire proposed by Hoffman et al. [2018] (see Section 3.3.2 for the original questionnaire):

1. The videos\graphs helped me recognize agent strategies
2. The videos\graphs contain sufficient detail for recognizing agent strategies
3. The videos\graphs contain irrelevant details
4. The videos\graphs were useful for the tasks I had to do in this survey

**Table 11.2.:** The four study conditions in our second user study comparing global strategy summaries with reward decomposition.

	Likelihood-based summaries	HIGHLIGHTS
No Reward Decomposition	$L$	$H$
Reward Decomposition	$L+RD$	$H+RD$

5. The specific scenarios shown in the videos\images were useful for the tasks I had to do in this survey.

### 11.1.3. Experimental Conditions

Extending our first study (Chapter 10), we wanted to evaluate the impact of combining global HIGHLIGHTS summaries and local reward decomposition explanations, as well as the effect of each method individually. To this end, we assigned participants randomly to one of four different conditions (see Table 11.2). We used the same likelihood-based baseline condition  $L$  (but with five instead of ten different summaries per agent) and HIGHLIGHTS condition  $H$  as in the first study. To extract these policy summaries, we ran the trained agents for 1,000 episodes after the training and recorded the traces. In addition to the purely global conditions  $L$  and  $H$ , we added two new conditions:

- **HIGHLIGHTS Summaries + Reward Decomposition ( $H+RD$ ):** In this condition, we used the interactive presentation proposed in Section 8.2.2. Since interpreting reward decomposition takes some time, we did not show videos. Instead, participants were only shown the most “important” state of each trajectory. This means that they did not get the context of that state as the video summaries provide. However, the chosen states were the same states that appeared in the middle of the videos in condition  $H$ . Each chosen state was shown using an image alongside a bar plot that represents the Q-values of the different reward components for the agents’ chosen action (see Figure 8.3).
- **Likelihood-based Summaries + Reward Decomposition ( $L+RD$ ):** This condition was the same as condition  $H+RD$ , but the shown states were taken from the summaries in condition  $L$  instead of  $H$ .

For the conditions without local explanation ( $L$  and  $H$ ), the presentation was similar to Figure 8.3, but the reward bars were omitted, and each scenario showed a short video.

#### 11.1.4. Procedure

After a consent form and demographic questions, participants were given an explanation regarding the environment (Pacman). Second, they were given a brief explanation about reinforcement learning and, specifically, about Q-values in layperson vocabulary. In particular, they were told that the agents are maximizing their future total score by taking into account both immediate points as well as points for future actions. Lastly, participants were told about the task and, depending on the condition, were given information about the type of explanation they will see, including an example explanation. At the end of each instruction phase, the participants were asked to complete a quiz and were only allowed to proceed after answering all questions correctly. After the instructions, the participants analyzed the three different agents, as described above, and then completed the explanation satisfaction scale. Participants were compensated as follows: they received a \$3 base payment and a 30-cent bonus for identifying the preferences of each of the agents correctly. The full study setup can be seen in Appendix C.1.

#### 11.1.5. Participants

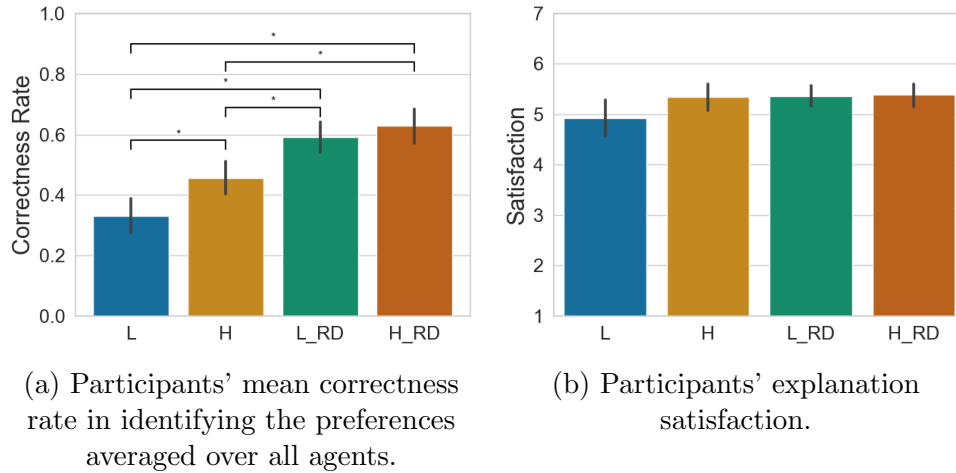
We recruited participants through Amazon Mechanical Turk (N = 164). We excluded participants who did not answer the attention question correctly, as well as participants who completed the survey in less than two standard deviations from the mean completion time in their condition. After screening, we had 159 participants (mean age = 36 years, 88 females, all from the US, UK, or Canada).

### 11.2. Results

We measured participants' ability to assess the agents' preferences by calculating the mean fraction of correct reward component comparisons, i.e., their correctness rate, for each condition (see Figure 11.1 (a)). We tested our hypotheses using the non-parametric, one-sided Mann-Whitney  $U$  test. Only when comparing the individual explanation conditions  $H$  and  $L+RD$  we used a two-sided test since we did not have a hypothesis as to which method would be better.

**We found that reward decomposition improved the participants' ability to assess the agents' preferences.** The combination of  $L+RD$  significantly improved participants' performance compared to  $L$  or  $H$  ( $U=1187$  and  $U=1176$  respectively,  $p<0.001$  for both). Participants in the  $H+RD$  condition performed better compared to  $L$  and  $H$  ( $U=1286$  and  $U=1332$  respectively,  $p<0.001$  for both). Some of the participants explicitly referred to reward decomposition as being helpful for the task, e.g., "In each of the scenarios the





**Figure 11.1.:** Main results in our second study comparing HIGHLIGHTS summaries (H) and Reward Decomposition (RD). The error bars show the 95% CI.

graph clearly shows the preference for normal pills followed by eating power pills. Eating ghosts was a minor section on the graph.”

**The HIGHLIGHTS summaries contributed to the participants’ mental model of the agents.** There was a significant difference between condition *H* and condition *L* ( $U=935$ ,  $p=0.002$ ). In some explanations given by participants, it seemed that the HIGHLIGHTS summary displayed information that was useful for inferring preference, e.g., one participant wrote, “the pacman would go for a power pill, eat it and turn around” when explaining their answers about the *power pill agent* preferences.

**The combined explanation did not outperform reward decomposition alone.** There were no significant differences between  $H+RD$  and  $L+RD$  when aggregated across all agents. However, our results indicate that the combination of  $H+RD$  helped assess the agent’s preferences when the difference between the reward types was minor. For the *blue-ghost agent*, there was only a small difference between the Q-values for the blue ghosts and normal pills. Participants in conditions *H* ( $M=0.3$ , 95%  $CI=(0.2, 0.5)$ ) and  $H+RD$  ( $M=0.31$ , 95%  $CI=(0.17, 0.46)$ ) were better at correctly identifying the blue ghost as more important than the participants in conditions  $L+RD$  ( $M=0.2$ , 95%  $CI=(0.17, 0.46)$ ) and *L* ( $M=0.12$ , 95%  $CI=(0, 0.23)$ ). This indicates that even though our overall results do not show that the combination of  $H+RD$  is significantly better than  $L+RD$ , there were cases in which the addition of HIGHLIGHTS helped.

**In general, while the participants’ objective performance was better with RD compared to video-based policy summaries, this did not lead to an increase in subjective measures.** There were no substantial

differences in the confidence values between conditions. The satisfaction values of participants were above neutral in general (see Figure 11.1 (b)). Here, the explanation conditions  $L+RD$ ,  $H+RD$ , and  $H$  had higher mean satisfaction ratings ( $M$  between 5.34 and 5.39) than the baseline condition (condition  $L$  with  $M=4.93$ ). However, this was not significant.

### 11.3. Discussion

In the previous Chapter 10, HIGHLIGHTS summaries were integrated with saliency maps, but the user study showed that saliency maps did not provide much benefit to the users' understanding of agent behavior. We hypothesized that reward decomposition may be more beneficial for several reasons. First, saliency maps describe what features of the state the agent pays attention to, but it is often hard to infer how this information affects the agent's decisions, especially for laypeople. Reward decomposition has the benefit of explicitly describing what values the agent expects to get in a way that reflects its preferences for different reward components. Moreover, saliency maps are a post-hoc method and may not be faithful to the underlying model (see Chapter 9 and [Rudin, 2019]) while reward decomposition values are learned through the agent's training and reflect its true decision-making policy. Another difference between the integration of global and local information in this chapter and the one used in Chapter 10 is the use of static images rather than videos. We chose this approach based on the findings from the study in Chapter 10, which identified the use of videos as one possible limitation, as the local information is harder to discern when looking at dynamic videos.

Our study replicated the result by Amir and Amir [2018] that HIGHLIGHTS summaries have benefits compared to likelihood-based summaries. However, in contrast to the results in Chapter 10, participants in the local reward decomposition condition  $L+RD$  performed on a similar level as the combined condition  $H+RD$  and better than the HIGHLIGHTS condition  $H$ . A possible explanation for this limited contribution of HIGHLIGHTS in our study is that the experimental task may have been particularly suited to reward decomposition. Since reward decomposition was already highly effective in conveying agent preferences, the selection of states for the summary was less important.

### 11.4. Conclusion

This chapter evaluated the approach proposed in Section 8.2.2, which integrates HIGHLIGHTS, a global policy summary, with local reward decomposition. We conducted a user study to evaluate the contribution of this approach to people's

ability to analyze agent preferences. Our results show that reward decomposition was particularly helpful for this task and that HIGHLIGHTS also led to improvement in participants' performance, but only in certain situations.

The fact that the intrinsic reward decomposition method in our work outperforms the post-hoc saliency maps used in a similar experiment in Chapter 10 empirically reaffirms the recommendation by Rudin [2019] to use intrinsic explanation methods whenever possible. Furthermore, based on the difference between our study and the one in Chapter 10, where we showed local saliency maps on videos instead of static states, future combinations of local explanations and global policy summaries should provide the local explanation on static states. This allows users to discern the information within the explanation.

Another notable finding is that the use of different explanation methods did not result in substantial differences in subjective measures like explanation satisfaction. This finding emphasizes the importance of using objective performance measures for XAI while also showing the need for future work on how we can increase the usability of explanatory systems.

## 12. Third Study: Counterfactual Explanations

In the previous two chapters, we investigated the combination of global HIGHLIGHTS summaries with local saliency maps (Chapter 10) and reward decomposition (Chapter 11). In both studies, the combination of local and global explanations achieved the highest results, indicating that there is benefit to such a combination. However, in the first study, the local saliency maps only had a limited contribution compared to the global HIGHLIGHTS. In stark contrast, in the second study, the local reward decomposition had a much bigger impact than the global HIGHLIGHTS summaries. One of the main differences between saliency maps and reward decomposition is that the former is a post-hoc explanation method while the latter is built into the agent intrinsically.

The drawback of intrinsic methods is that they are not always applicable. For instance, reward decomposition only works in domains where the reward can be decomposed into individual components. Therefore, we want to explore the combination of HIGHLIGHTS with other promising post-hoc explanation techniques. To this end, this chapter evaluates the combination of strategy summaries with counterfactual explanations, which was proposed in Section 8.2.1. This chapter is based on the user study in our publication:

**Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS. IFAAMAS*, 10 pages

For the particular counterfactual explanation methods, we compare the Counterfactual State Explanation (*CSE*, see Section 4.1.2) approach by Olson et al. [2021] and the GANterfactual-RL approach proposed in Chapter 7.

Olson et al. [2021] showed that their *CSE* approach can be applied to a variety of RL environments and helps users identify a flawed agent. With the help of their counterfactual explanations, users were able to differentiate between a normal RL agent for the Atari game Space Invaders and a flawed agent that did not see a specific in-game object. For this task, it is sufficient for the counterfactual explanation to not change the particular object at all while other objects frequently change. This clearly communicates that the unchanged object

is irrelevant and ignored by the agent, implying that it is not seen at all.

However, for counterfactual explanations to be employed more widely, they also have to be useful for more complex tasks. As we have seen in Section 3.3.2, one of the main goals of a good explanation is to refine the user’s mental model of the agent. For RL agents, this includes understanding what strategy and intentions an agent pursues. Another critical goal for explanations is that they should help users to calibrate their trust in different agents. For RL agents, this entails that users should be able to choose fitting agents for specific problems, which is more complex than simply identifying defective agents. The two aforementioned challenges require counterfactual explanations to not only convey *what* objects need to change but also *how* the objects need to be altered to change the agents’ policy.

The study in this chapter presents participants with different kinds of counterfactual explanations and investigates whether this helps them to understand the strategies of Pacman agents. Furthermore, we investigate if the counterfactuals help them to calibrate their trust so they can choose fitting agents for specific tasks (surviving or receiving points).

As such, the contributions of this chapter are as follows:

- It evaluates the GANterfactual-RL approach proposed in Chapter 7 and the combination of HIGHLIGHTS and counterfactual explanations proposed in Section 8.2.1 in a user study.
- This user study shows, for the first time, that counterfactual explanations can help to understand the strategies of RL agents.
- The study also identifies current deficiencies of counterfactual explanations for RL agents that point the way for future work.

## 12.1. Study Design

### 12.1.1. Research Question and Hypothesis

The research question for our study was which counterfactual explanations help users to understand the strategies of RL agents and help them to choose fitting agents for a specific task. We hypothesized that our GANterfactual-RL method is more useful than the CSE method and is more useful than a presentation of the original states without counterfactuals. Further, we thought that the counterfactuals generated by the CSE approach might mislead participants due to the low validity of the generated counterfactual explanations (see Section 7.2). Therefore, we hypothesized that only providing the original states is more useful than adding CSE counterfactuals.

## 12.1.2. Dependent Variables and Main Tasks

**Agent Understanding Task.** To measure whether participants understand the strategies of different agents and build a correct mental model of them, we used an agent understanding task similar to the ones used in the studies described in the previous two chapters 10 and 11. Here, participants were presented with five states and the actions that the agent chose in these states. This was done for each of the three Pacman agents described in Section 7.1.3 (one agent at a time). The states were selected by the HIGHLIGHTS algorithm (Section 4.2.3.1). To this end, we let each trained agent play for an additional 50 games and chose the most important states according to HIGHLIGHTS. The resulting states show gameplay that is typical for the agent without the need to manually select states that might be biased toward our approach. Based on these states (and additional explanations depending on the condition), participants had to select up to two in-game objects that were most important for the agent’s strategy from a list of objects (Pacman, normal pills, power pills, ghosts, blue ghosts, or cherries). As described in Section 7.1.3, each agent strongly focuses on a different single in-game object depending on their reward function (e.g., the *fear-ghosts agent* focuses on normal ghosts). If the participants select this object and none of the other objects, they receive a point. The only exception is Pacman. Every agent heavily relies on the position of Pacman as a source of information. Therefore, participants receive the point whether they select Pacman or not.

**Agent Comparison Task.** To measure how well the participants’ trust is calibrated, we used an agent comparison task similar to the one used in Chapter 10 to evaluate the combination of HIGHLIGHTS and saliency maps. In this task, we implicitly measure if the participants’ trust is appropriate by asking them, for each possible pair of the three Pacman agents, which agent they would like to play on their behalf to obtain certain goals. Since a single agent can be good for one goal but bad for another, this requires a deeper analysis than the distinction between a normal and a defective agent. For each pair, the participants are shown their own descriptions of each agent from the agent understanding task and the same states and explanations that they saw during the agent understanding task. Then, they have to decide which agent should play on their behalf to achieve more points and which agent should play on their behalf to survive longer. We know the ground truth for this by measuring the agents’ average score and amount of steps for the 50 games used to find the HIGHLIGHTS states (see Appendix D.3 for the exact numbers). The amount of steps that the *blue-ghost agent* and the *power pill agent* survive is so close that we do not include this specific comparison in the evaluation.

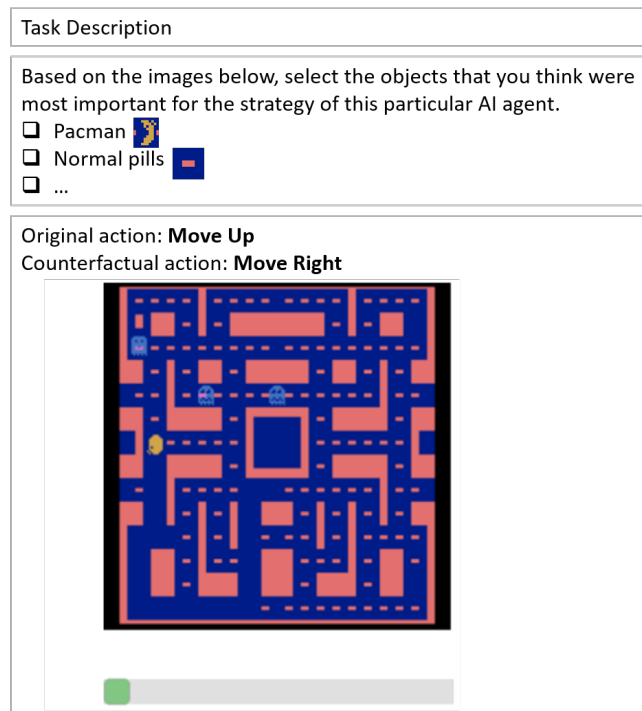
**Explanation Satisfaction.** To measure the participant’s subjective satisfaction, we use statements adapted from the Explanation Satisfaction Scale by Hoffman et al. [2018]. Our specific questions can be seen in the appendix Figure D.18. Participants have to rate their agreement with each statement on a 5-point Likert scale. Participants’ final rating was averaged over all those ratings, reversing the rating of negative statements. We do this once after the agent understanding task and once after the agent comparison task in case there are satisfaction differences between the tasks.

### 12.1.3. Conditions and Explanation Presentation

We used three independent conditions. The first was a *Control* condition without counterfactual explanations. Since we use HIGHLIGHTS states as the baseline, this condition is equivalent to condition *H* from the previous two studies. In addition, we used two conditions where the states during the agent understanding task and the agent comparison task are accompanied by counterfactual explanations. In the *CSE* condition, the counterfactuals are generated by the approach from Olson et al. [2021], and in the *GANterfactual-RL* condition, the counterfactuals are generated by our proposed method (Chapter 7). Appendix D.4 shows all states and counterfactuals used in the study, and Appendix D.1 provides training details for our *CSE* and *GANterfactual-RL* models. The *CSE* and *GANterfactual-RL* conditions are similar to the condition *H+S* in Chapter 10 and *H+RD* in Chapter 11, as they combine global and local explanations. For presenting the counterfactual explanations, we used the design proposed in Section 8.2.1 – i.e., there is a slider below the HIGHLIGHTS states that participants can use to linearly interpolate the original state to the counterfactual state. Figure 12.1 shows a simplified version of the beginning of our agent understanding task.

### 12.1.4. Procedure and Compensation

After completing a consent form, participants were asked to answer demographic questions (age and gender) and questions regarding their experience with Pacman and their views on AI. Then, they were shown a tutorial explaining the rules of the game Pacman and were asked to play the game to familiarize themselves with it. To verify that participants understood the rules, they were asked to complete a quiz and were only allowed to proceed with the survey after answering all questions correctly. Afterward, participants in the counterfactual conditions received additional information and another quiz regarding the counterfactual explanations. Then, they proceeded to the agent understanding task, which was repeated three times, once for each agent. The order of the agents was randomized. After that, participants filled out the explanation satisfaction



**Figure 12.1.:** A simplified scheme of the beginning of our agent understanding task with a single example state.

scale and continued to the agent comparison task. Again, this task was repeated three times, once for each possible agent pair, and the order was randomized. Finally, participants had to complete another satisfaction scale for the agent comparison task. Participants got a compensation of 5\$ for participating in the study. As an incentive to do the tasks properly, they received a bonus payment of 10 cents for each point they got in the agent understanding task and 5 cents for each point in the agent comparison task. The complete questionnaire can be seen in Appendix D.5. We preregistered our study online.<sup>1</sup>

### 12.1.5. Participants

We recruited participants through Amazon Mechanical Turk. Participation was limited to Mechanical Turk Masters from the US, UK, or Canada (to ensure a sufficient English level) with a task approval rate greater than 95% and without color vision impairment. We conducted a power analysis with an estimated medium effect size of 0.7 based on previous similar experiments [Mertes et al., 2022a; Mertes et al., 2022b; Huber et al., 2021b]. This determined that we need 28 participants per condition to achieve a power of 0.8. and a significance

<sup>1</sup><https://aspredicted.org/m9fi5.pdf>



level of 0.05. To account for participant exclusions, we recruited 30 participants per condition. Participants were excluded if they did not look at any of the counterfactual explanations for any of the agents during the agent understanding task, if their textual answers were nonsensical, or if they took considerably less time than the average. This left us with 30 participants in the Control condition, 28 participants in the CSE condition, and 23 in the GANterfactual-RL condition.

The distribution of age, AI experience, and Pacman experience was similar between the conditions (see Appendix D.2). There was a difference in the gender distribution and the attitude towards AI between the conditions. The Control condition had 40% female participants, the CSE condition had 32%, and the GANterfactual-RL condition had 26%. The mean attitude towards AI was the highest in the GANterfactual-RL condition and the lowest in the Control condition (see Appendix D.2).

## 12.2. Results

The results for the participants' scores during the main tasks can be seen in Figure 12.2. The explanation satisfaction values are shown in Figure 12.3. In the following, we will summarize the results of our main hypotheses, which we analyzed using non-parametric one-tailed Mann-Whitney U tests.

**Counterfactuals helped participants to understand the agents' strategies.** In the agent understanding task, there was a significant difference between the Control condition ( $M=0.8$ ) and the GANterfactual-RL condition ( $M=1.65$ ),  $U=181$ ,  $p=0.001$ ,  $r=0.477$ .<sup>2</sup> Contrary to our hypothesis, the Control condition got lower scores than the CSE condition ( $M=0.8$  vs  $M=1.18$ ),  $p=0.953$ .

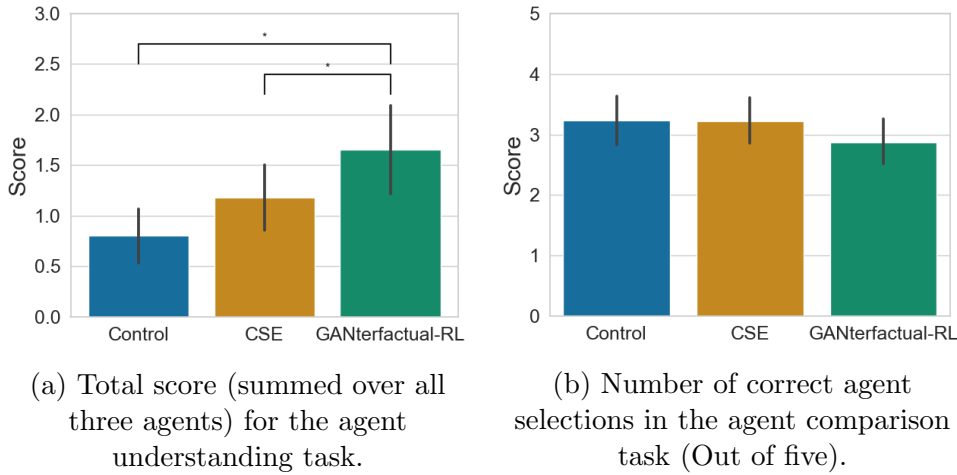
**Our GANterfactual-RL explanations were significantly more useful than the CSE approach for understanding the agents' strategies.** In the agent understanding task, the CSE condition got a mean score of 1.18, while the GANterfactual-RL condition got a mean score of 1.65 ( $U=232$ ,  $p=0.038$ ,  $r=0.2795$ ).

**The increased understanding of the agents' strategies did not result in a more calibrated trust.** Contrary to our hypothesis, there were no significant differences in the trust task (Control vs CSE:  $p=0.536$ , Control vs. GANterfactual-RL:  $p=0.852$ , CSE vs GANterfactual-RL:  $p=0.876$ ).

**Counterfactuals did not increase explanation satisfaction.** Even though participants objectively had a better understanding of the agents' strategies, they did not feel more satisfied with them. Participants in the Control condition were significantly more satisfied than participants in the CSE condition in

---

<sup>2</sup> $M$  is the mean and  $r$  is rank biserial correlation.



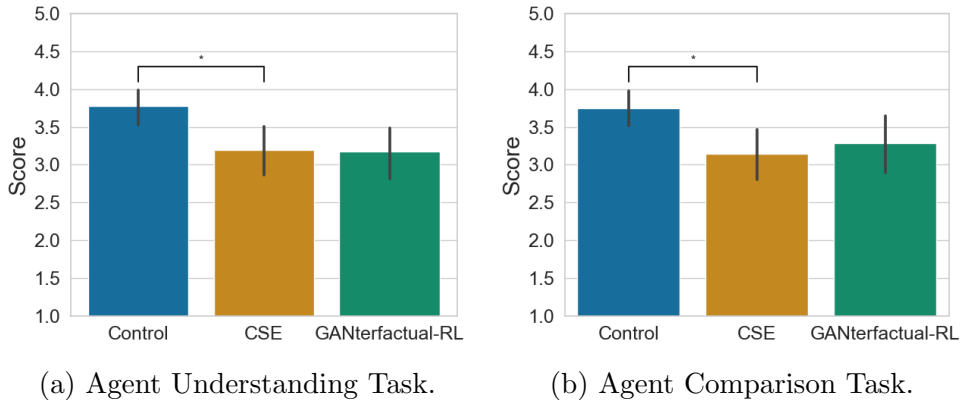
**Figure 12.2.:** Comparison of the participants’ average performance in each task, by condition. Error bars show the 95% CI.

both the agent understanding task (Control:  $M=3.77$ , CSE:  $M=3.20$ ;  $U=249$ ,  $p=0.004$ ,  $r=0.4071$ ) and the agent comparison task (Control:  $M=3.75$ , CSE:  $M=3.14$ ;  $U=267$ ,  $p=0.008$ ,  $r=0.3643$ ). Contrary to our expectations, the participants in the GANterfactual-RL condition were not more satisfied than the participants in the Control condition or the CSE condition in both the agent understanding task (Control vs. GANterfactual-RL:  $p=0.996$ , CSE vs GANterfactual-RL:  $p=0.546$ ) or the agent comparison task (Control vs. GANterfactual-RL:  $p=0.967$ , CSE vs GANterfactual-RL:  $p=0.334$ ).

### 12.3. Discussion

Our results show that counterfactual explanations help users to understand which strategies different agents pursue. In particular, our GANterfactual-RL method (see Chapter 7) was significantly more useful than both the CSE method and not providing counterfactuals. Contrary to our hypothesis, even the counterfactuals generated by the CSE method resulted in a better understanding of the agents than not providing counterfactual explanations. This demonstrates the usefulness of counterfactual explanations for RL agents even in more complex tasks than identifying defective agents.

The two studies described in Chapter 10 and 11 evaluated the usefulness of other local explanation techniques for understanding the strategies of RL agents in a similar way to our study. Chapter 10 looked at saliency map explanations and found that they did not help more than showing HIGHLIGHTS states without saliency maps. In that study, participants achieved 37% of the maximum



**Figure 12.3.:** Comparison of the participants’ average explanation satisfaction in each task, by condition.

possible score in the agent understanding task, while the participants with our counterfactual explanations obtained 50%. Chapter 11 investigated reward decomposition explanations and found that they helped participants to achieve 60% of the maximum score in the agent understanding task. However, reward decomposition is an intrinsic explanation method which the agent and the reward function have to be specifically designed for. Our counterfactual explanations resulted in only a 10% less average score even though they are post-hoc explanations that can be generated for already trained black-box agents.

Our agent comparison task showed that the increased understanding of the agent’s strategies through both counterfactual explanation methods did not help participants choose fitting agents for specific tasks. For choosing the correct agent for a given problem, it is not enough to identify the strategies of the agents. It also requires enough expertise in the environment (e.g., Pacman) to judge which strategy is better suited for the problem at hand. For example, in Pacman, humans often assume that an agent that survives longer will accumulate more points in the long run. However, this is not necessarily the case since an aggressive agent can better exploit the very high rewards of eating blue ghosts. Our results for this task are in line with the results of the agent comparison task for saliency maps in Chapter 10.

Finally, our study showed that participants subjectively were not satisfied with the counterfactual explanations even though they objectively helped them to understand the agents. This might be due to the additional cognitive load of interpreting the counterfactual explanations. The previous two studies in Chapter 10 and 11 also did not find a significant difference in user satisfaction for their local explanation techniques. Only the choice of states, which does not provide additional information, influenced the explanation satisfaction in Chapter 10. However, our study is the first to see significantly higher satisfac-

tion for the condition without local explanation compared to the two conditions with local explanations. This indicates that counterfactuals are subjectively less satisfying than saliency maps or reward decomposition. One possible explanation for this is the visual quality of the counterfactuals. Some participants from both counterfactual conditions commented that the counterfactuals had too many artifacts. One participant from the GANterfactual-RL condition, for example, wrote that *“the counterfactuals were somewhat helpful, but they would have worked better if there were fewer or no artifacts”*. Another possible reason for the low satisfaction is the presentation of the explanation. Because our study primarily aimed at investigating the benefits and drawbacks of our specific counterfactual approach, we did not use a user-friendly explanatory system where different types of explanations are provided according to the requests of the explainee.

## 12.4. Conclusion

In this chapter, we showed that adding local counterfactual explanations to global HIGHLIGHTS states significantly improved users’ understanding of the strategies of different agents in a user study. We further showed that the GANterfactual-RL method we proposed in Chapter 7 was significantly more useful for this task than the previous state-of-the-art counterfactual explanation method for DRL agents with visual input.

Our user study also identified two remaining deficiencies of counterfactual explanations. First, participants were subjectively not satisfied with the explanations, which might be due to unnatural artifacts in some counterfactuals. Second, the counterfactuals did not help them to calibrate their trust in the agents. Future work should try to improve counterfactual explanations in these directions.

# 13. Conclusions From All Three Studies

This chapter summarizes the results of Part V, which evaluated the integration of different local explanations with global strategy summaries in three user studies.

## 13.1. Potential of Combining Local and Global Explanations

Our findings show considerable potential for integrating global and local explanations for DRL agents.

Across all three studies, the combination of global and local explanations produced the best results in both the agent understanding task and the agent comparison task. The most compelling evidence of this synergy is presented in Chapter 12, where the combination of GANterfactual-RL and HIGHLIGHTS significantly outperformed all other conditions in the agent understanding task. In other cases, the combined explanation approach shared the top spot with particular individual explanations. Specifically, in Chapter 11, the combined approach was on par with the purely local reward decomposition explanations. Moreover, in Chapter 10 and the agent comparison task in Chapter 12, the exclusively global HIGHLIGHTS conditions matched the respective combined explanation approaches. Importantly, the combined explanations consistently matched or exceeded the performance of any individual explanation approach, demonstrating that there were no adverse effects of integrating global and local explanations.

## 13.2. Global Explanations Contributed to Appropriate Trust and Agent Understanding

**The global strategy summaries significantly enhanced the participants' trust calibration.** Appropriate trust was measured by our agent comparison task in Chapters 10 and 12. Here, neither the local saliency maps nor

the local counterfactual explanations improved the participants' performance beyond the levels achieved with the exclusively global HIGHLIGHTS explanations. The first study (Chapter 10) additionally included a baseline condition where states were selected based on their likelihood of occurrence. Compared to this baseline, the purely global HIGHLIGHTS condition demonstrated a significant performance improvement during the agent comparison task.

**Global strategy summaries contributed to the participants' mental models of the agents.** The HIGHLIGHTS summaries significantly outperformed the likelihood-based baseline during the agent understanding task in Chapters 10 and 11. Notably, in the first study (Chapter 10), the global explanations contributed more than the local saliency maps, which did not outperform the baseline. The first study also explored the nature of information conveyed by the explanations. The global strategy summaries were particularly effective for identifying the agent's goals.

Only the second study (Chapter 11) found a limitation of global explanations. Here, the exclusively local reward decomposition explanations outperformed the exclusively global HIGHLIGHTS summaries during the agent understanding task. We believe this result is due to the inherent utility of reward decomposition for the agent understanding task.

## 13.3. Contribution of Local Explanation

### 13.3.1. Intrinsic Explanations Outperformed Post-Hoc Explanations

The intrinsic reward decomposition explanations in the second user study (Chapter 11) substantially helped participants understand different agents. Their usefulness in this task exceeded the usefulness of the post-hoc saliency maps and post-hoc counterfactual explanations in the first and third user studies (Chapters 10 and 12). To my knowledge, this provides the first empirical evidence for XRL that **intrinsically explainable agents should be preferred over post-hoc explanations for black-box agents whenever possible**. This claim was famously formulated by Rudin [2019] for general black-box models. The claim is also supported by the results of the computational evaluation of saliency maps in Chapter 9, which demonstrated that some post-hoc saliency maps failed to reflect the agents' internal reasoning properly.

### 13.3.2. Assessing Different Post-Hoc Explanations

Intrinsically explainable agents are not always feasible. For such cases, this thesis also investigated two post-hoc explanations – saliency maps in Chapter 10 and counterfactual explanations in Chapter 12. This section provides a summary of the findings associated with these methods.

**Counterfactual Explanations Improved Agent Understanding.** As with reward decomposition, counterfactual explanations improved the participants’ agent understanding beyond global HIGHLIGHTS summaries. In contrast, saliency maps did not even improve agent understanding compared to the likelihood-based baseline. This benefit of counterfactual explanations was observed for two distinct methods of generating counterfactuals, providing evidence that the result generalizes to different methods of generating counterfactuals. Nonetheless, the choice of the counterfactual generation method still mattered. Our proposed GANterfactual-RL approach significantly outperformed the previous state-of-the-art counterfactual generation method.

**Saliency Maps as Debugging Tool.** While there were no significant differences between the saliency and non-saliency conditions in the first study (Chapter 10), they still showed potential. We found two non-significant trends indicating that saliency maps improved scores in the textual strategy description task and improved participants’ ability to place appropriate trust in different agents. Furthermore, we exploratively observed that saliency maps helped identify what information the agent attends to.

However, the saliency maps were held back by the participants’ difficulty in interpreting them correctly. On the one hand, this can be attributed to the presentation of saliency maps on videos, which made it difficult to interpret individual saliency maps. On the other hand, it requires a certain level of expertise about the task and the employed algorithms to interpret saliency maps correctly. This observation is also reflected by an exploratory result from the first study (Chapter 10) that participants with AI expertise benefited more from saliency maps.

Thus, I argue that saliency maps have the potential to be a valuable debugging tool for users with AI expertise.

## 13.4. Interactive Explanation Presentation More Effective than Videos

In Chapter 8, we proposed two different ways of presenting the local explanations – as an overlay on the HIGHLIGHTS videos and via interactive images. The study in Chapter 10 showed that participants had difficulties parsing the additional information while watching videos. In contrast, the interactive presentation in Chapters 11 and 12 significantly improved participants’ agent understanding.

Consequently, I would suggest future research to explore interactive presentations for combining local and global explanations.

## 13.5. Subjective Explanation Satisfaction Still Lacking

Regarding subjective explanation satisfaction, we only observed significant advantages during the agent understanding task of the first study (Chapter 10). Here, participants were more satisfied with global HIGHLIGHTS summaries than with the likelihood-based alternatives (with and without local saliency maps). Interestingly, during the third study (Chapter 12), participants without local explanations reported significantly higher subjective satisfaction than participants who received counterfactual explanations. This lack of subjective satisfaction contrasts the consistent observation across all our studies that at least one explanation method significantly improved the participants’ objective understanding of the agents.

One possible reason for the lack of subjective satisfaction is that we used a between-subject design in all three studies, where participants only saw their own condition. Because of this, participants without explanations were unaware of the potential information they were missing, and participants with explanations were able to focus on the shortcomings of the provided explanations.

In the future, it could be interesting to investigate this phenomenon more thoroughly – for example, through a within-subject design that allows participants to experience multiple explanations and compare them directly.

## 13.6. Generalization of Results

Since the experiments in Part V tested three of the most common local explanation methods, the results should be quite comprehensive in this direction.

However, we only tested a single global explanation method: HIGHLIGHTS strategy summaries. Our proposed combination of local and global explanations,



as described in Chapter 8, is easily adaptable to other *example-based* global explanation methods, such as T-SNE (see Section 4.2) – as demonstrated by Zahavy et al. [2016]. Nonetheless, it is unclear whether our results generalize to these methods.

Finally, this thesis used only a single environment for the user studies. In Septon et al. [2023], which is the basis for Chapter 11, we additionally conducted a second user study in a different environment: the Highway environment. In the Highway environment, the RL agent controls an autonomous vehicle and has to navigate a multi-lane highway while driving alongside other vehicles. I omitted this environment from the thesis to maintain focus and coherence. The results in the Highway environment closely match those in the Pacman environment. This consistency indicates that our results generalize well to other domains.

The only difference in the Highways environment results was that global HIGHLIGHTS summaries alone did not outperform the likelihood-based baseline. We believe this is due to the limited possible agent behaviors in the Highway environment. This finding indicates that the usefulness of global policy summaries may depend on the complexity of behaviors that agents can deploy in an environment.

**Part VI.**  
**Conclusion**

# 14. Contributions

This chapter summarizes the contributions of this dissertation, dividing them into conceptual, technical, and empirical contributions.

## 14.1. Conceptual Contributions

This thesis conceptually contributes to enhancing the explainability of DRL agents in two main ways. First is the proposal of novel explanation algorithms and approaches addressing the specific challenges of explaining DRL agents. Second, it presents new evaluation methods tailored to XRL.

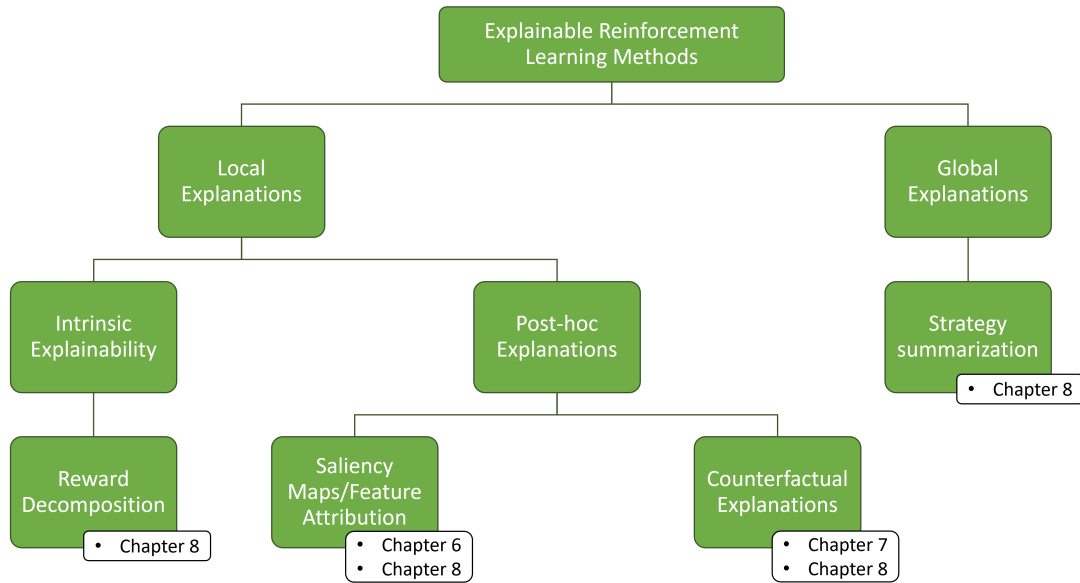
### 14.1.1. Novel Approaches to Explaining DRL Agents

This thesis contributed to several families of explanation approaches in XRL. Figure 14.1 shows an overview of how the chapters of this thesis contributed to the landscape of XRL methods as outlined in Chapter 4.

**LRP-Argmax: Selective Saliency Maps for DRL Agents.** Previous research has indicated that DRL agents primarily focus on specific objects within visual inputs. Traditional saliency map techniques, designed to highlight information relevant to an agent’s decision-making, fail to capture this aspect, as they often highlight all potential factors influencing a decision.

In Chapter 6, this dissertation introduces two DRL-focused adjustments to the existing LRP framework for creating saliency maps. First, it outlines a method for applying LRP to the Dueling DQN architecture, thereby extending LRP’s applicability to all modern iterations of the DQN algorithm. Second, it proposes the *LRP-argmax* variant, which produces saliency maps with a selective focus on pivotal areas within the visual input while retaining all desired properties of LRP. Through these adjustments, this dissertation facilitates the application of the successful LRP concept to state-of-the-art DRL algorithms.

**GANterfactual-RL: Counterfactual Explanations for RL Agents with Visual Input.** For XAI in the context of supervised learning, counterfactual explanations play an important role. They answer “Why not?” or “What if?” questions



**Figure 14.1.:** An overview of how the chapters of this thesis contributed to the landscape of XRL methods as outlined in Chapter 4.

by suggesting minimal changes required for an AI model to make a different decision. However, generating these explanations for RL agents that process visual input is particularly challenging due to the vast state spaces and the agents’ long-term decision-making policies. The vast visual state spaces make counterfactual approaches that use optimization during runtime impractically slow. Due to the agents’ long-term policies, the counterfactual explanations must consider the extended implications of any alterations to the initial state. Additionally, RL agents don’t use training datasets in the same way as supervised models, complicating the application of traditional counterfactual explanation methods.

In Chapter 7, this dissertation proposed a novel method to generate counterfactual explanations for RL agents with visual input by formulating the problem as a domain transfer, where each domain leads the agent to a different action. Its model-agnostic nature and ease of adaptation make our approach a versatile tool for explaining RL agents. Since the initial publication of our work, other researchers have already used our proposed method as the basis for new algorithms. For example, Samadi et al. [2024] augment a GAN architecture similar to ours with saliency maps to create counterfactual explanations for DRL agents. This demonstrates the value of our approach for the XAI community.

**Combining Local and Global Explanations for Agent Behavior.** The central theme of this thesis is the integration of global and local explanations for DRL

agents. As outlined in Chapter 4, local explanations analyze specific decisions of the agent, while global explanations convey its general strategy. Using either approach in isolation comes with inherent drawbacks. Local explanations provide detailed insights into particular decisions but fail to convey the agent’s reasoning in different contexts. Global explanations, in contrast, present an overarching view of the agent’s actions in a wide range of scenarios but lack details about what information the agent considers in specific situations.

Chapter 8 presents a step towards the development of combined explanation methods for DRL agents. It proposes a joint global and local explanation approach based on global strategy summaries and various local explanation methods. This approach is designed to be flexible, accommodating a range of local explanation algorithms, including post-hoc explanation methods such as saliency maps and counterfactuals, as well as intrinsic explanation methods like reward decomposition. While we only investigated a single global explanation method, Pierson et al. [2024] already extended our combined method by an additional global explanation method. This further demonstrates the applicability and flexibility of our approach.

## 14.1.2. Novel Evaluation Methods for XRL

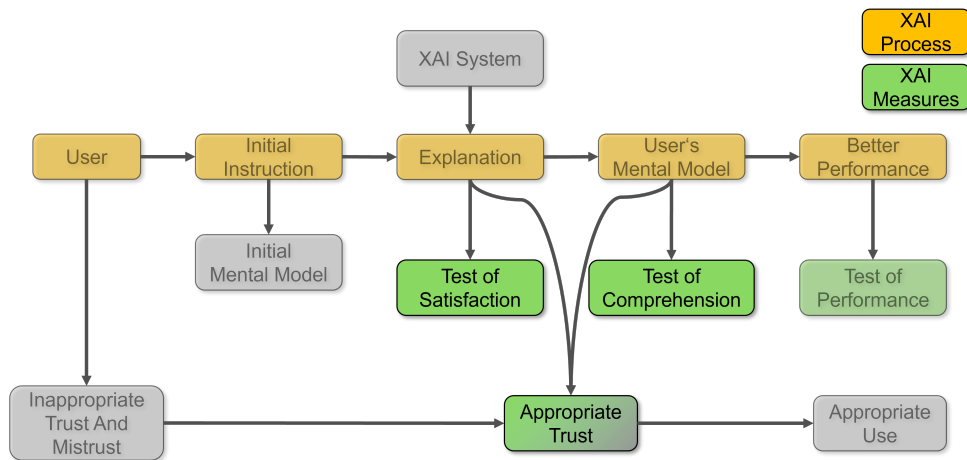
Besides introducing new methods, this thesis also contributed conceptually to the evaluation of XRL methods.

### 14.1.2.1. Advancing Computational Evaluation Methods for XRL

**Methodology to Fine-Tune Perturbation-Based Saliency Maps for Benchmarking.** In some cases, there is no comprehensive access to an agent’s internal model. Under such circumstances, it’s necessary to utilize model-agnostic explanation techniques. Particularly for saliency maps, this typically involves the use of perturbation-based methods. However, a major limitation of perturbation-based saliency maps is their reliance on a variety of adjustable parameters. Consequently, they require calibration of a wide range of parameter values before they can be tested effectively. Chapter 9 outlines a methodology for benchmarking perturbation-based saliency maps that includes a systematic approach for fine-tuning their parameters. This fine-tuning methodology improves the applicability of perturbation-based saliency maps in different contexts.

### **Individually Evaluating Action- and State-Values of Value-based RL Agents.**

Chapter 9 introduced an adaptation of an existing computational XAI metric, specifically tailoring it to value-based RL agents. The output of value-based RL agents encodes both the value of the current state as well as the expected future reward after performing each action in that state. The proposed adaptation



**Figure 14.2.:** The explanation process as proposed by Hoffman et al. [2018] and Gunning and Aha [2019]. The metrics evaluated in this dissertation are highlighted.

allows for a separate evaluation of how well saliency maps explain the agent’s action value and state value estimation. This makes it possible to determine which saliency map method is best suited to analyze the reasoning behind the agent’s evaluation of a state, and which method is best suited to explain the agent’s preference for a particular action in that state. The proposed adaptation thus facilitates a more targeted use of saliency maps for value-based DRL agents.

#### 14.1.2.2. Innovative User Study Design for XRL

Designing user studies to assess XRL remains challenging. The literature review in Section 5.1 revealed that previous studies have primarily centered on isolated elements of the explanation process, such as participants’ mental models of the agents or their subjective explanation satisfaction. In contrast, Part V of this dissertation introduced a holistic user study design for XRL. This design encompasses the evaluation of three different dimensions outlined by Hoffman et al. [2018] for effective measurement of XAI: understanding of the agent, appropriate trust, and subjective explanation satisfaction (see Figure 14.2).

To this end, we train agents with qualitatively distinct policies by altering the reward function. This approach improves on previous studies that only use a single agent, train agents with similar strategies by varying the training duration, or utilize complex manipulations of the input states. Through these differentiated agents, our studies probe into the participants’ mental model of the agents by assessing their understanding of the agents’ distinct policies. Fur-

thermore, the qualitatively different agent strategies allow the assessment of appropriate trust through pairwise agent comparisons. Here, participants are incentivized to pick the agent whose strategy aligns best with a given objective. Finally, participants are asked how satisfied they were with the explanations during each of the agent understanding task and agent comparison task.

Following the publication of our initial user study [Huber et al., 2021b], Miller [2022] highlighted our agent comparison task as a guiding example for measuring appropriate trust in the context of XAI. Subsequently, our complete study design was not only adopted but also expanded upon by Pierson et al. [2024]. The adoption of our study design in the research community demonstrates its contribution and impact on advancing the field of explainable reinforcement learning.

## 14.2. Technical Contributions

This thesis is dedicated to enhancing the explainability of DRL agents. Alongside the conceptual contributions mentioned above, it includes technical contributions that address the inherent challenge of implementing XAI approaches for intricate DRL algorithms. We implemented each of the conceptual explanation methods and evaluation methodologies presented in the previous section. These technical contributions also ensure the reproducibility of the experiments and empirical contributions presented in this thesis. To this end, we have made all our implementations accessible as open-source code. Below is a list of our repositories and what they were used for:

- For the LRP-Argmax algorithm discussed in Chapter 6, visit: [https://github.com/HuTobias/LRP\\_argmax](https://github.com/HuTobias/LRP_argmax)
- The GANterfactual-RL method and its accompanying user study, as presented in Chapters 7 and 12, are accessible at: <https://github.com/hcmlab/GANterfactual-RL>
- Our computational evaluation from Chapter 9 can be found here: <https://github.com/belimmer/PerturbationSaliencyEvaluation>
- The implementation for the user study detailed in Chapter 10 is provided at: <https://github.com/HuTobias/HIGHLIGHTS-LRP>
- Our reward decomposition implementation for the study discussed in Chapter 11 is available at: [https://github.com/hcmlab/baselines/tree/reward\\_decomposition](https://github.com/hcmlab/baselines/tree/reward_decomposition)

Our open-source code has already been used by the research community, demonstrating its contribution to the field of XRL. Notably, Pierson et al. [2024] built directly on our code from Chapter 10 for their study on integrating global and local explanations in DRL. Additionally, our GANterfactual-RL repository has served as a basis for developing new counterfactual generation methods in DRL [Samadi et al., 2024].

## 14.3. Empirical Contributions

Finally, this thesis contributes empirical insights based on extensive computational evaluations and several user studies. These findings will inform the future use and development of XAI methods for DRL agents.

### 14.3.1. Computational Evaluation of the Fidelity of Post-Hoc Explanation Methods

If post-hoc explanations for black-box RL agents are to be used in increasingly safety-critical domains, it is crucial that they reflect the agents’ internal reasoning faithfully. To this end, this thesis contributed to the computational evaluation of the fidelity of two post-hoc explanation methods: counterfactual explanations and saliency maps.

**Counterfactual Explanations.** Chapter 12 showed that the previous state-of-the-art counterfactual explanation method for DRL agents with visual input did not change the agents’ decision correctly in a majority of the cases. Our results further demonstrate that our proposed GANterfactual-RL considerably improves this drawback, modifies less of the original input, and takes less time than the only previous method. Based on this, our GANterfactual-RL approach was the state-of-the-art counterfactual explanation method for DRL agents with visual input at the time of its publication.

**First computational Evaluation of the Fidelity of Saliency Maps for DRL Agents.** Prior to this dissertation, there was no computational evaluation of the fidelity of saliency maps for DRL, as outlined in the literature review in Section 5.2. Chapter 6 and Part IV address this gap.

In Chapter 6, we evaluated our LRP-argmax approach with regard to its dependency on the agents’ parameters. Our results are consistent with earlier studies on LRP-based saliency maps for image classifiers: the saliency maps do analyze the learned weights, but the fully connected layers are not sufficiently analyzed.



Part 9 conducted a thorough evaluation of five perturbation-based saliency map techniques, examining their dependency on agent parameters and alignment with the agent’s internal reasoning. These results offer valuable insights for practitioners who rely on model-agnostic saliency maps to explain an agent’s reasoning without full access to its internal architecture.

- The tested approaches show high dependence on the agents’ learned parameters. Only one approach (Noise Sensitivity) lacked dependence on the learned parameters of the output layer. However, we show that this problem can be mitigated by a slight adjustment of the approach.
- Regarding fidelity to the agents’ reasoning, there were considerable differences between analyzing the action value and state value estimation of value-based DRL agents. Practitioners should be aware of which of the two they want to analyze and choose saliency maps that align with their objectives. In our experiment, the SARFA approach worked best to capture the action value while Occlusion Sensitivity and RISE were more suited for the state value.
- Depending on which perturbation method the approaches use, the resulting saliency maps only analyze how sensitive the agent is with regard to specific types of perturbation. This was true even for perturbation methods like blurring that aim to reduce the dependence on a choice of occlusion color, showing a need to further develop perturbation-based saliency map approaches. For now, practitioners must consider which perturbation types are relevant and meaningful for their specific applications.

### **14.3.2. User Studies Comparing the Complementary Benefits of Local and Global Explanation Methods for DRL Agents**

Part V evaluated the integration of several local explanation methods with global strategy summaries across three user studies. This section lists the key findings from these studies, which inform the design of future combined local and global XRL frameworks.

- Our findings reveal considerable potential for the combined use of global and local explanations for DRL agents. Apart from subjective satisfaction in Chapter 12, the combination consistently matched or surpassed the performance of any individual explanation approach.
- Global explanations contributed significantly to fostering appropriate trust and understanding of the agents.

- Several discoveries were made regarding local explanations:
  - This thesis presents the first empirical evidence in XRL that intrinsically explainable agents should be preferred over post-hoc explanations for black-box agents whenever possible, reinforcing similar findings for black-box classifiers.
  - Post-hoc counterfactual explanations significantly improved participants’ agent understanding. Our proposed GANterfactual-RL approach even approached the effectiveness of intrinsically explainable agents.
  - Saliency maps showed potential but were held back by the user’s difficulty interpreting them.
  - No local explanation approach significantly improved appropriate trust.
- An interactive explanation presentation based on images was more effective than video overlays in our experiments.
- Subjective explanation satisfaction was lacking despite improvements in the objective agent understanding task, indicating a disconnect between the objective utility of explanations and their subjective reception.

A more comprehensive discussion of the key findings listed above can be found in Chapter 13. Collectively, the findings emphasize the potential of integrating global and local explanations to enhance the explainability and trustworthiness of DRL agents. They reveal the strengths and limitations of different XRL approaches and point to areas that need further improvement.

# 15. Future Work

Explainable Reinforcement Learning (XRL) is still an emerging research area that has recently begun to carve out its niche within the broader field of XAI. Its emergence as a distinct subfield is relatively new, with the first comprehensive surveys on the topic appearing in 2020 [Heuillet et al., 2021; Alharin et al., 2020; Puiutta and Veith, 2020]. With the continued development of DRL algorithms, DRL agents will increasingly be deployed in critical and complex domains, including healthcare, autonomous driving, and robotic navigation. This expansion will further raise the demand for XRL in the future. To meet this demand, this thesis presents several insights that inform the ongoing evolution of XRL. This section summarizes the key opportunities for future work identified in this dissertation.

## 15.1. Advancing Computational Metrics for XRL

Computationally evaluating the fidelity of saliency maps to the model’s internal reasoning is an active challenge in the field of XAI for image classifiers. The complex architecture of neural networks obscures the relevance of each pixel in the decision process, leaving us without definitive ground-truth saliency maps for comparison with the generated saliency maps. In Chapter 9, we explored this problem for the first time for XRL by using four variants of a computational XAI metric to assess the fidelity of saliency map methods in DRL scenarios. We found notable differences in the results between these computational metric variants. This observation reinforces the findings of Tomsett et al. [2020] that current fidelity metrics for saliency maps are vulnerable to nuances in their implementation. Moreover, our results extend these findings from general XAI to XRL. Specifically, in the context of value-based RL, our observations reveal substantial differences between metrics that focus on the action value and those assessing the state value.

Our results show that there are specific requirements for computational metrics in the context of XRL. Based on our results, future work should explore new computational metrics that more accurately reflect the unique aspects of DRL. Such DRL-focused metrics will enable a deeper and more precise evaluation of XRL methods.

## 15.2. Towards a Combined Local and Global Explanation Framework for Deep Reinforcement Learning

Part of this dissertation’s main contributions are the insights obtained from the user studies, which will inform the future development of XRL frameworks.

All three user studies described in Part V demonstrated the immense potential of integrating local and global explanations. Therefore, I recommend future researchers explore combined explanation frameworks that harness the strengths of both local and global explanation methods.

While global HIGHLIGHTS summaries enhanced the participants’ agent understanding and fostered appropriate trust, exploring alternative global explanation methods was not within the scope of this thesis. Given the substantial influence of global information in our studies, I think that investigating other global explanation methods is a promising future research direction.

For the local explanations, reward decomposition, which is intrinsically built into the agent, outperformed the post-hoc explanation methods, which are applied after the agent is trained. Therefore, future combinations of global and local explanations for DRL should aim to include and explore more intrinsic explanation methods.

In addition to these promising research directions, this thesis also identified three challenges for future XRL frameworks that combine global and local explanations.

### 15.2.1. Conversational Explanation Interfaces

One obstacle we identified in all three studies is the accessibility and usability of the combined explanation presentation.

The sequential decision-making nature of DRL requires an explanation presentation that reflects the temporal dynamics of the agents’ strategies. Initially, we tried to achieve this by presenting saliency maps as an overlay on HIGHLIGHTS videos in Chapter 10. The temporally connected action sequences within these videos were intended to establish a link between actions and their outcomes. However, the participants were overwhelmed by the amount of information in these combined videos.

The subsequent studies in Chapters 11 and 12 improved upon this by employing an interactive presentation with static images. While this aided participants with their objective tasks, it did not increase their subjective explanation satisfaction. Without subjective approval of the explanation framework, its adoption is unlikely, regardless of its objective merits.

Hence, I recommend that future research investigates more user-friendly explanation interfaces for the combined global and local explanation framework outlined in this thesis. Improving the presentation of explanations is essential to fully leverage the objective benefits demonstrated in our studies. Aligning the delivery of explanations more closely with user preferences and usability standards will ensure that the valuable insights provided by these methods are accessible and impactful.

As a starting point for an enhanced explanation interface, I suggest building on the interactive explanation presentation used in Chapters 11 and 12. Here, the participants could switch between the most important HIGHLIGHTS summary states, which were presented as static images with an additional local explanation. The main goal of these chapters was to investigate the benefits and limitations of explanation methods within a combined global and local explanation framework. Consequently, to maintain comparability in our evaluation of local explanations, we presented them alongside all images without allowing users to select their preferred types of explanation. While systematic, this approach diverges from the conversational nature of human-human explanations, where the explainer responds to the specific inquiries of the explainee [Madumal et al., 2019]. Leveraging the positive results for global strategy summaries, I suggest an explanation interface that initially presents summary states to users, providing a basic understanding of the agent’s policy. However, to preserve the dialog nature of human-human explanations and to reduce cognitive load, the local explanations should not be presented by default. Instead, users should have the flexibility to request specific types of explanations as needed, tailoring the information flow to their curiosity and comprehension needs.

### **15.2.2. Addressing Low Subjective Explanation Satisfaction**

In all three of our user studies, the local explanations did not increase the participants’ subjective explanation satisfaction despite aiding in objective tasks. In the second study, even the global explanations did not increase explanation satisfaction significantly. The lack of explanation satisfaction could partially stem from the usability challenges discussed in the previous subsection. However, we also identified additional factors that may have negatively impacted explanation satisfaction. These factors provide avenues for future work.

The first factor is that we conducted our experiments in a between-subject design where each participant experienced only a single condition. On the one hand, this setup meant that participants without explanations were unaware of the potential information they were missing. On the other hand, the participants who received explanations could focus on the explanations’ deficiencies. To gain a better understanding of the users’ subjective preferences in a combined global and local XRL framework, future work should more thoroughly explore the dis-

crepancy between objective utility and subjective explanation satisfaction. To this end, I suggest conducting a within-subject design study that allows participants to experience and directly compare different combinations of explanations or their absence.

Our second observation regarding explanation satisfaction concerns the different types of local explanations. We found that the local counterfactual explanations seemed to be less satisfying than the local saliency map and reward decomposition explanations. Specifically, in the third study, the two conditions with counterfactual explanations received significantly lower satisfaction ratings than the condition without counterfactuals. This result is in contrast to the first two studies, where the presence of reward decomposition or saliency maps had no significant effect on the explanation satisfaction. The comparative dissatisfaction with counterfactuals is particularly surprising since counterfactual explanations were more helpful than saliency maps in the objective agent understanding task. A possible reason for the dissatisfaction with counterfactual explanations might have been their visual quality. Some participants commented that the counterfactuals contained too many artifacts. Therefore, I believe that a promising direction for future research is to develop better counterfactual generation methods that create more realistic images with fewer artifacts. In our studies, counterfactual explanations were the best method for providing local post-hoc explanations, and they were almost as effective as the intrinsic reward decomposition method. Hence, more satisfying counterfactual explanations will help in situations where intrinsic explanation methods are not applicable.

### **15.2.3. Developing Local Explanations that Foster Appropriate Trust**

The last challenge we identified for future XRL frameworks is the development of local explanation methods that foster appropriate trust. In the first and third user studies, we used an agent comparison task to measure how well the participants' trust is calibrated. The results revealed that only the global HIGHLIGHTS summaries fostered appropriate trust. Adding local explanations did not improve the participants' performance in the agent comparison task beyond the level of HIGHLIGHTS.

However, the first study showed an encouraging trend, indicating that saliency maps fostered appropriate trust compared to the baseline condition without any explanations. This trend suggests that saliency maps have the potential to promote appropriate trust but have been hindered by the usability issues described in the previous two subsections. Therefore, I suggest addressing the aforementioned usability issues of saliency maps as a promising research direction for developing local explanations that foster appropriate trust.

### **15.3. Explanations for Multi-Agent Reinforcement Learning**

The ALE games used as test-beds within this dissertation were exclusively single-agent environments. However, real-world scenarios often involve multiple agents interacting within the same environment, necessitating the use of multi-agent reinforcement learning [Albrecht et al., 2024]. For example, several robots may need to collaborate within a warehouse.

The proposed methods and insights gained about explainable reinforcement learning for single-agent environments in this thesis provide a basis for developing XAI systems in multi-agent contexts. However, multi-agent teams introduce additional challenges. For instance, it is crucial to understand how agents distribute tasks among themselves and coordinate their actions efficiently. Future work should explore how the methods developed in this thesis can be extended to multi-agent settings.

# Bibliography

- Adadi Amina and Mohammed Berrada [2018]. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6, pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.
- Adebayo Julius, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim [2018]. “Sanity checks for saliency maps”. In: *Advances in Neural Information Processing Systems*, pp. 9505–9515.
- Alber Maximilian, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans [2019]. “iNNvestigate Neural Networks!” In: *Journal of Machine Learning Research* 20.93, pp. 1–8.
- Albrecht Stefano V., Filippos Christianos, and Lukas Schäfer [2024]. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press. URL: <https://www.marl-book.com>.
- Alharin Alnour, Thanh-Nam Doan, and Mina Sartipi [2020]. “Reinforcement Learning Interpretation Methods: A Survey”. In: *IEEE Access* 8, pp. 171058–171077. DOI: 10.1109/ACCESS.2020.3023394.
- Alqaraawi Ahmed, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze [2020]. “Evaluating saliency map explanations for convolutional neural networks: a user study”. In: *IUI ’20: 25th International Conference on Intelligent User Interfaces*, pp. 275–285. DOI: 10.1145/3377325.3377519.
- Amir Dan and Ofra Amir [2018]. “HIGHLIGHTS: Summarizing Agent Behavior to People”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’18, pp. 1168–1176.
- Amir Ofra, Finale Doshi-Velez, and David Sarne [2019]. “Summarizing agent strategies”. In: *Autonomous Agents and Multi-Agent Systems* 33.5, pp. 628–644. DOI: 10.1007/s10458-019-09418-w.
- Amitai Yotam and Ofra Amir [2023]. “A Survey of Global Explanations in Reinforcement Learning”. In: *Explainable Agency in Artificial Intelligence*. 1st ed., pp. 21–42. DOI: 10.1201/9781003355281-2.
- Ancona Marco, Enea Ceolini, Cengiz Öztireli, and Markus Gross [2018]. “Towards better understanding of gradient-based attribution methods for Deep Neural Networks”. In: *6th International Conference on Learning Representations, ICLR*. URL: <https://openreview.net/forum?id=Sy21R9JAW>.
- Anderson Andrew, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Bur-



- nett [2019]. “Explaining Reinforcement Learning to Mere Mortals: An Empirical Study”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 1328–1334. URL: <https://doi.org/10.24963/ijcai.2019/184>.
- Anjomshoae Sule, Amro Najjar, Davide Calvaresi, and Kary Främbling [2019]. “Explainable Agents and Robots: Results from a Systematic Literature Review”. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS ’19*, 1078–1088. ISBN: 9781450363099.
- Arrieta Alejandro Barredo, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bénénot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera [2020]. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58, pp. 82–115. DOI: <https://doi.org/10.1016/j.inffus.2019.12.012>.
- Atrey Akanksha, Kaleigh Clary, and David D. Jensen [2020]. “Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning”. In: *8th International Conference on Learning Representations, ICLR*. URL: <https://openreview.net/forum?id=rkl3m1BFDB>.
- Bach Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek [2015]. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* 10.7, pp. 1–46. DOI: [10.1371/journal.pone.0130140](https://doi.org/10.1371/journal.pone.0130140).
- Bansal Gagan, Besmira Nushi, Ece Kamar, Walter S. Lasecki, Daniel S. Weld, and Eric Horvitz [2019]. “Beyond Accuracy: The Role of Mental Models in Human-AI Team Performance”. In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 7, pp. 2–11. ISSN: 2769-1349. DOI: [10.1609/hcomp.v7i1.5285](https://doi.org/10.1609/hcomp.v7i1.5285).
- Bansal Gagan, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld [2021]. “Does the Whole Exceed Its Parts? The Effect of AI Explanations on Complementary Team Performance”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. CHI ’21*, pp. 1–16. ISBN: 978-1-4503-8096-6. DOI: [10.1145/3411764.3445717](https://doi.org/10.1145/3411764.3445717).
- Bastani Osbert, Yewen Pu, and Armando Solar-Lezama [2018]. “Verifiable Reinforcement Learning via Policy Extraction”. In: *Advances in Neural Information Processing Systems*. Vol. 31. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2018/file/e6d8545daa42d5ced125a4bf747b3688-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2018/file/e6d8545daa42d5ced125a4bf747b3688-Paper.pdf).
- Bellemare Marc G., Yavar Naddaf, Joel Veness, and Michael Bowling [2013]. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *J. Artif. Intell. Res.* 47, pp. 253–279. DOI: [10.1613/jair.3912](https://doi.org/10.1613/jair.3912).

- Bewley Tom and Jonathan Lawry [2021]. “TripleTree: A Versatile Interpretable Representation of Black Box Agents and their Environments”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.13, pp. 11415–11422. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i13.17360.
- Bewley Tom, Jonathan Lawry, and Arthur Richards [2022]. “Summarising and Comparing Agent Dynamics with Contrastive Spatiotemporal Abstraction”. In: *CoRR* abs/2201.07749. arXiv: 2201.07749.
- Brockman Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba [2016]. “OpenAI Gym”. In: *CoRR* abs/1606.01540. arXiv: 1606.01540.
- Buçinca Zana, Phoebe Lin, Krzysztof Z. Gajos, and Elena L. Glassman [2020]. “Proxy tasks and subjective measures can be misleading in evaluating explainable AI systems”. In: *IUI '20: 25th International Conference on Intelligent User Interfaces*, pp. 454–464. DOI: 10.1145/3377325.3377498.
- Byrne Ruth M. J. [2019]. “Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 6276–6282. DOI: 10.24963/ijcai.2019/876.
- Cauchy Augustin [1847]. “Méthode générale pour la résolution des systemes d'équations simultanées”. In: *Comp. Rend. Sci. Paris* 25.1847, pp. 536–538.
- Chandrasekaran Bruce, Michael C Tanner, and John R Josephson [1989]. “Explaining control strategies in problem solving”. In: *IEEE Intelligent Systems* 1, pp. 9–15.
- Chen Ziheng, Fabrizio Silvestri, Gabriele Tolomei, He Zhu, Jia Wang, and Hongshik Ahn [2021]. “ReLACE: Reinforcement Learning Agent for Counterfactual Explanations of Arbitrary Predictive Models”. In: *CoRR* abs/2110.11960. arXiv: 2110.11960.
- Choi Yunjey, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo [2018]. “Stargan: Unified generative adversarial networks for multi-domain image-to-image translation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8789–8797.
- Coppens Youri, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni [2019]. “Distilling deep reinforcement learning policies in soft decision trees”. In: *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*, pp. 1–6.
- Dai Jim G and Mark Gluzman [2022]. “Queueing Network Controls via Deep Reinforcement Learning”. In: *Stochastic Systems* 12.1, pp. 30–67. DOI: 10.1287/stsy.2021.0081.
- Danesh Mohamad H., Anurag Koul, Alan Fern, and Saeed Khorram [2021]. “Re-understanding Finite-State Representations of Recurrent Policy Networks”. In: *Proceedings of the 38th International Conference on Machine Learning*,

- ICML*, pp. 2388–2397. URL: <http://proceedings.mlr.press/v139/danesh21a.html>.
- Dhariwal Prafulla, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov [2017]. *OpenAI Baselines*. <https://github.com/openai/baselines>.
- Doshi-Velez Finale and Been Kim [2017]. “Towards A Rigorous Science of Interpretable Machine Learning”. In: *CoRR* abs/1702.08608. arXiv: 1702.08608.
- Erwig Martin, Alan Fern, Magesh Murali, and Anurag Koul [2018]. “Explaining deep adaptive programs via reward decomposition”. In: *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*.
- Fan Tingxiang, Pinxin Long, Wenxi Liu, and Jia Pan [2020]. “Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios”. In: *Int. J. Robotics Res.* 39.7. DOI: 10.1177/0278364920916531.
- Goel Vikash, Jameson Weng, and Pascal Poupart [2018]. “Unsupervised Video Object Segmentation for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 31*, pp. 5688–5699. URL: <http://papers.nips.cc/paper/7811-unsupervised-video-object-segmentation-for-deep-reinforcement-learning>.
- Goldberg Yoav [2017]. “Neural network methods for natural language processing”. In: *Synthesis Lectures on Human Language Technologies* 10.1, pp. 1–309.
- Goodfellow Ian, Yoshua Bengio, and Aaron Courville [2016]. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.
- Goudet Olivier, Diviyani Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag [2018]. “Learning Functional Causal Models with Generative Neural Networks”. In: *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pp. 39–80. ISBN: 978-3-319-98131-4. DOI: 10.1007/978-3-319-98131-4\_3.
- Greydanus Samuel, Anurag Koul, Jonathan Dodge, and Alan Fern [2018]. “Visualizing and Understanding Atari Agents”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML*, pp. 1787–1796.
- Gunning David and David W. Aha [2019]. “DARPA’s Explainable Artificial Intelligence (XAI) Program”. In: *AI Mag.* 40.2, pp. 44–58. DOI: 10.1609/AIMAG.V40I2.2850.
- Halasz Frank G and Thomas P Moran [1983]. “Mental models and problem solving in using a calculator”. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 212–216. DOI: 10.1145/800045.801613.
- Haykin Simon [2009]. *Neural networks and learning machines*. Vol. 3. Pearson Upper Saddle River, NJ, USA:

- Hessel Matteo, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver [2018]. “Rainbow: Combining Improvements in Deep Reinforcement Learning”. In: *Proceedings of the 32nd Conference on Artificial Intelligence, AAAI 2018*, pp. 3215–3222.
- Heuillet Alexandre, Fabien Couthouis, and Natalia Díaz Rodríguez [2021]. “Explainability in deep reinforcement learning”. In: *Knowl. Based Syst.* 214, p. 106685.
- Hoffman Robert R, Shane T Mueller, Gary Klein, and Jordan Litman [2018]. “Metrics for explainable AI: Challenges and prospects”. In: *CoRR* abs/1812.04608. arXiv: 1812.04608.
- Hsieh Hsiu-Fang and Sarah E Shannon [2005]. “Three approaches to qualitative content analysis”. In: *Qualitative health research* 15.9, pp. 1277–1288.
- Huang Sandy H, Kush Bhatia, Pieter Abbeel, and Anca D Dragan [2018]. “Establishing Appropriate Trust via Critical States”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3929–3936.
- Huang Sandy H., David Held, Pieter Abbeel, and Anca D. Dragan [2019]. “Enabling Robots to Communicate Their Objectives”. In: *Autonomous Robots* 43.2, pp. 309–326. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9771-0.
- Huber Tobias**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 10 pages.
- Huber Tobias**, Benedikt Limmer, and Elisabeth André [2022]. “Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents”. In: *Frontiers in Artificial Intelligence* 5. ISSN: 2624-8212. DOI: 10.3389/frai.2022.903875.
- Huber Tobias**, Silvan Mertes, Stanislava Rangelova, Simon Flutura, and Elisabeth André [2021a]. “Dynamic Difficulty Adjustment in Virtual Reality Exergames through Experience-driven Procedural Content Generation”. In: *IEEE Symposium Series on Computational Intelligence, SSCI*, pp. 1–8. DOI: 10.1109/SSCI50451.2021.9660086.
- Huber Tobias**, Dominik Schiller, and Elisabeth André [2019]. “Enhancing Explainability of Deep Reinforcement Learning Through Selective Layer-Wise Relevance Propagation”. In: *KI 2019: Advances in Artificial Intelligence*, pp. 188–202.
- Huber Tobias**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571.

- Iyer Rahul, Yuezhong Li, Huaoli Li, Michael Lewis, Ramitha Sundar, and Katia P. Sycara [2018]. “Transparency and Explanation in Deep Reinforcement Learning Neural Networks”. In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES*, pp. 144–150.
- Jaderberg Max, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel [2019]. “Human-Level Performance in 3D Multiplayer Games with Population-Based Reinforcement Learning”. In: *Science* 364.6443, pp. 859–865. DOI: 10.1126/science.aau6249.
- Jaunet Theo, Romain Vuillemot, and Christian Wolf [2020]. “DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning”. In: *Computer Graphics Forum* 39.3, pp. 49–61. ISSN: 1467-8659. DOI: 10.1111/cgf.13962.
- Juozapaitis Zoe, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez [2019]. “Explainable reinforcement learning via reward decomposition”. In: *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*.
- Keane Mark T., Eoin M. Kenny, Eoin Delaney, and Barry Smyth [2021]. “If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 4466–4474. DOI: 10.24963/ijcai.2021/609.
- Kindermans Pieter-Jan, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne [2018]. “Learning how to explain neural networks: PatternNet and PatternAttribution”. In: *ICLR*. URL: <https://openreview.net/forum?id=Hkn7CBaTW>.
- Kiran B Ravi, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez [2022]. “Deep Reinforcement Learning for Autonomous Driving: A Survey”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.6, pp. 4909–4926. DOI: 10.1109/TITS.2021.3054625.
- Lage Isaac, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir [2019]. “Exploring Computational User Models for Agent Policy Summarization”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 1401–1407. DOI: 10.24963/ijcai.2019/194.
- Landajuela Mikel, Brenden K. Petersen, Sookyoung Kim, Claudio P. Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F. Pettit, and Daniel Faissol [2021]. “Discovering Symbolic Policies with Deep Reinforcement Learning”. In: *Proceedings of the 38th International Conference on Machine Learning*, pp. 5979–5989.

- Langosco Lauro, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger [2022]. “Goal Misgeneralization in Deep Reinforcement Learning”. In: *International Conference on Machine Learning, ICML*, pp. 12004–12019.
- Lapuschkin Sebastian, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller [2019]. “Unmasking Clever Hans Predictors and Assessing What Machines Really Learn”. In: *Nature Communications* 10.1, p. 1096.
- Lee John D and Katrina A See [2004]. “Trust in automation: Designing for appropriate reliance”. In: *Human factors* 46.1, pp. 50–80.
- Lipton Zachary C. [2018]. “The mythos of model interpretability”. In: *Commun. ACM* 61.10, 36–43. ISSN: 0001-0782. DOI: 10.1145/3233231.
- Liu Guiliang, Oliver Schulte, Wang Zhu, and Qingcan Li [2019]. “Toward interpretable deep reinforcement learning with linear model u-trees”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD, Proceedings, Part II 18*. Springer, pp. 414–429.
- Looveren Arnaud Van and Janis Klaise [2021]. “Interpretable Counterfactual Explanations Guided by Prototypes”. In: *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD, Proceedings, Part II*, pp. 650–665. DOI: 10.1007/978-3-030-86520-7\_40.
- Lopez-Paz David, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Leon Bottou [2017]. “Discovering Causal Signals in Images”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Maaten Laurens van der and Geoffrey Hinton [2008]. “Visualizing Data Using T-SNE”. In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605. ISSN: 1533-7928.
- Madumal Prashan, Tim Miller, Liz Sonenberg, and Frank Vetere [2019]. “A Grounded Interaction Protocol for Explainable Artificial Intelligence”. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, pp. 1033–1041.
- Madumal Prashan, Tim Miller, Liz Sonenberg, and Frank Vetere [2020]. “Explainable Reinforcement Learning through a Causal Lens”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.03, pp. 2493–2500. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i03.5631.
- Matsui Teppei, Masato Taki, Trung Quang Pham, Junichi Chikazoe, and Koji Jimura [2022]. “Counterfactual explanation of brain activity classifiers using image-to-image transfer by generative adversarial network”. In: *Frontiers in Neuroinformatics* 15, p. 79.
- McKnight Patrick E and Julius Najab [2010]. “Mann-Whitney U Test”. In: *The Corsini encyclopedia of psychology*, pp. 1–1.
- Mertes Silvan, **Tobias Huber**, Christina Karle, Katharina Weitz, Ruben Schlagowski, Cristina Conati, and Elisabeth André [2024]. “Relevant Irrelevance: Gener-

- ating Alterfactual Explanations for Image Classifiers”. In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 467–475. DOI: 10.24963/ijcai.2024/52.
- Mertes Silvan, **Tobias Huber**, Katharina Weitz, Alexander Heimerl, and Elisabeth André [2022a]. “GANterfactual - Counterfactual Explanations for Medical Non-experts Using Generative Adversarial Learning”. In: *Frontiers Artif. Intell.* 5, p. 825565. DOI: 10.3389/frai.2022.825565.
- Mertes Silvan, Christina Karle, **Tobias Huber**, Katharina Weitz, Ruben Schlagowski, and Elisabeth André [2022b]. “Alterfactual Explanations - The Relevance of Irrelevance for Explaining AI Systems”. In: *CoRR* abs/2207.09374. arXiv: 2207.09374.
- Miller Tim [2019]. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artif. Intell.* 267, pp. 1–38. DOI: 10.1016/j.artint.2018.07.007.
- Miller Tim [2022]. “Are we measuring trust correctly in explainability, interpretability, and transparency research?” In: *CoRR* abs/2209.00651. DOI: 10.48550/arXiv.2209.00651.
- Miller Tim, Piers Howe, and Liz Sonenberg [2017]. “Explainable AI: Beware of inmates running the asylum”. In: *IJCAI-17 Workshop on Explainable AI (XAI)*. Vol. 36.
- Mnih Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu [2016]. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research, pp. 1928–1937. URL: <https://proceedings.mlr.press/v48/mniha16.html>.
- Mnih Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. [2015]. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540, p. 529.
- Mohseni Sina, Jeremy E Block, and Eric Ragan [2021a]. “Quantitative Evaluation of Machine Learning Explanations: A Human-Grounded Benchmark”. In: *26th International Conference on Intelligent User Interfaces. IUI '21*, pp. 22–31. ISBN: 978-1-4503-8017-1. DOI: 10.1145/3397481.3450689.
- Mohseni Sina, Niloofar Zarei, and Eric D. Ragan [2021b]. “A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems”. In: *ACM Transactions on Interactive Intelligent Systems* 11.3-4, pp. 1–45. ISSN: 2160-6455, 2160-6463. DOI: 10.1145/3387166.
- Molnar Christoph [2022]. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. URL: <https://christophm.github.io/interpretable-ml-book>.

- Montavon Grégoire, Wojciech Samek, and Klaus-Robert Müller [2018]. “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* 73, pp. 1–15. DOI: 10.1016/j.dsp.2017.10.011.
- Mopuri Konda Reddy, Utsav Garg, and R Venkatesh Babu [2019]. “CNN fixations: an unraveling approach to visualize the discriminative image regions”. In: *IEEE Transactions on Image Processing* 28.5, pp. 2116–2125.
- Mothilal Ramaravind Kommiya, Amit Sharma, and Chenhao Tan [2020]. “Explaining machine learning classifiers through diverse counterfactual explanations”. In: *FAT\* ’20: Conference on Fairness, Accountability, and Transparency*, pp. 607–617. DOI: 10.1145/3351095.3372850.
- Nemirovsky Daniel, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta [2022]. “CounterGAN: Generating counterfactuals for real-time recourse and interpretability using residual GANs”. In: *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022*, pp. 1488–1497. URL: <https://proceedings.mlr.press/v180/nemirovsky22a.html>.
- Norman Donald A [2014]. “Some observations on mental models”. In: *Mental models*, pp. 15–22. DOI: 10.4324/978131580272.
- Olson Matthew L., Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong [2021]. “Counterfactual state explanations for reinforcement learning agents via generative deep learning”. In: *Artif. Intell.* 295, p. 103455. DOI: 10.1016/j.artint.2021.103455.
- Pawelczyk Martin, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci [2021]. “CARLA: A Python Library to Benchmark Algorithmic Recourse and Counterfactual Explanation Algorithms”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks*.
- Petsiuk Vitali, Abir Das, and Kate Saenko [2018]. “RISE: Randomized Input Sampling for Explanation of Black-box Models”. In: *British Machine Vision Conference 2018*, p. 151. URL: <http://bmvc2018.org/contents/papers/1064.pdf>.
- Pierson Britt Davis, Dustin Arendt, John Miller, and Matthew E. Taylor [2024]. “Comparing Explanations in RL”. In: *Neural Computing and Applications* 36.1, pp. 505–516. ISSN: 1433-3058. DOI: 10.1007/s00521-023-08696-6.
- Pinkus Allan [1999]. “Approximation theory of the MLP model in neural networks”. In: *Acta numerica* 8, pp. 143–195.
- Prajod Pooja, **Tobias Huber**, and Elisabeth André [2022]. “Using Explainable AI to Identify Differences Between Clinical and Experimental Pain Detection Models Based on Facial Expressions”. In: *MultiMedia Modeling - 28th International Conference, MMM, Proceedings, Part I*, pp. 311–322. DOI: 10.1007/978-3-030-98358-1\_25.



- Prajod Pooja, Dominik Schiller, **Tobias Huber**, and Elisabeth André [2021]. “Do Deep Neural Networks Forget Facial Action Units? - Exploring the Effects of Transfer Learning in Health Related Facial Expression Recognition”. In: *International Workshop on Health Intelligence*. Springer, pp. 217–233.
- Puiutta Erika and Eric M. S. P. Veith [2020]. “Explainable Reinforcement Learning: A Survey”. In: *Machine Learning and Knowledge Extraction*, pp. 77–95.
- Puri Nikaash, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh [2020]. “Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution”. In: *8th International Conference on Learning Representations, ICLR*. URL: <https://openreview.net/forum?id=SJgzLkBKPB>.
- Rangelova Stanislava, Simon Flutura, **Tobias Huber**, Daniel Motus, and Elisabeth André [2019]. “Exploration of Physiological Signals Using Different Locomotion Techniques in a VR Adventure Game”. In: *Universal Access in Human-Computer Interaction. Theory, Methods and Tools - 13th International Conference, UAHCI 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Proceedings, Part I*. Vol. 11572. Lecture Notes in Computer Science, pp. 601–616. DOI: 10.1007/978-3-030-23560-4\_44.
- Rao Anand S. and Michael P. Georgeff [1995]. “BDI Agents: From Theory to Practice”. In: *Proceedings of the First International Conference on Multiagent Systems*, pp. 312–319.
- Ribeiro Marco Tulio, Sameer Singh, and Carlos Guestrin [2016]. ““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*, pp. 1135–1144. DOI: 10.1145/2939672.2939778.
- Rudin Cynthia [2019]. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nat. Mach. Intell.* 1.5, pp. 206–215. DOI: 10.1038/s42256-019-0048-x.
- Russell Stuart and Peter Norvig [2016]. *Artificial Intelligence: A Modern Approach Global Edition*. Pearson.
- Rutjes Heleen, Martijn Willemsen, and Wijnand IJsselsteijn [2019]. “Considerations on explainable AI and users’ mental models”. In: *Where is the Human? Bridging the Gap Between AI and HCI*.
- Sadler Matthew, Natasha Regan, Demis Hassabis, and Garry Kasparov [2019]. *Game Changer: AlphaZero’s Groundbreaking Chess Strategies and the Promise of AI*. New In Chess. ISBN: 978-90-5691-818-7.
- Samadi Amir, Konstantinos Koufos, Kurt Debattista, and Mehrdad Dianati [2024]. “SAFE-RL: Saliency-Aware Counterfactual Explainer for Deep Reinforcement Learning Policies”. In: *CoRR* abs/2404.18326. arXiv: 2404.18326.

- Samek Wojciech, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller [2017]. “Evaluating the Visualization of What a Deep Neural Network Has Learned”. In: *IEEE Trans. Neural Networks Learn. Syst.* 28.11. URL: <https://doi.org/10.1109/TNNLS.2016.2599820>.
- Schiller Dominik, **Tobias Huber**, Michael Dietz, and Elisabeth André [2020]. “Relevance-Based Data Masking: A Model-Agnostic Transfer Learning Approach for Facial Expression Recognition”. In: *Frontiers in Computer Science* 2. DOI: 10.3389/fcomp.2020.00006.
- Schiller Dominik, **Tobias Huber**, Florian Lingenfelder, Michael Dietz, Andreas Seiderer, and Elisabeth André [2019]. “Relevance-Based Feature Masking: Improving Neural Network Based Whale Classification Through Explainable Artificial Intelligence”. In: *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*, pp. 2423–2427. DOI: 10.21437/Interspeech.2019-2707.
- Schlagowski Ruben, Frederick Herget, Niklas Heimerl, Maximilian Hammerl, **Tobias Huber**, Pamina Zwolsky, Jan Gruca, and Elisabeth André [2024]. “From a Social POV: The Impact of Point of View on Player Behavior, Engagement, and Experience in a Serious Social Simulation Game”. In: *Proceedings of the 19th International Conference on the Foundations of Digital Games*. FDG ’24. ISBN: 9798400709555. DOI: 10.1145/3649921.3649936.
- Schulz Karl, Leon Sixt, Federico Tombari, and Tim Landgraf [2020]. “Restricting the Flow: Information Bottlenecks for Attribution”. In: *8th International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=S1xWh1rYwB>.
- Schütt Anan, **Tobias Huber**, Ilhan Aslan, and Elisabeth André [2023]. “Fast Dynamic Difficulty Adjustment for Intelligent Tutoring Systems with Small Datasets”. In: *Proceedings of the 16th International Conference on Educational Data Mining*, pp. 482–489. ISBN: 978-1-73367-364-8. DOI: 10.5281/zenodo.8115740.
- Schütt Anan, **Tobias Huber**, Jauwairia Nasir, Cristina Conati, and Elisabeth André [2024]. “Does Difficulty Even Matter? Investigating Difficulty Adjustment and Practice Behavior in an Open-Ended Learning Task”. In: *Proceedings of the 14th Learning Analytics and Knowledge Conference*. LAK ’24, pp. 253–262. ISBN: 9798400716188. DOI: 10.1145/3636555.3636876.
- Selvaraju Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra [2020]. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *Int. J. Comput. Vis.* 128.2, pp. 336–359. DOI: 10.1007/s11263-019-01228-7.
- Septon Yael, **Tobias Huber**, Elisabeth André, and Ofra Amir [2023]. “Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection* -

- 21st International Conference*. Vol. 13955. Lecture Notes in Computer Science, pp. 320–332. DOI: 10.1007/978-3-031-37616-0\_27.
- Sequeira Pedro and Melinda Gervasio [2020]. “Interestingness Elements for Explainable Reinforcement Learning: Understanding Agents’ Capabilities and Limitations”. In: *Artif. Intell.* 288, p. 103367. ISSN: 0004-3702. DOI: 10.1016/j.artint.2020.103367.
- Silva Andrew, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son [2020]. “Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. ISSN: 2640-3498, pp. 1855–1865. URL: <https://proceedings.mlr.press/v108/silva20a.html>.
- Silva Andrew, Mariah Schrum, Erin Hedlund-Botti, Nakul Gopalan, and Matthew Gombolay [2022]. “Explainable Artificial Intelligence: Evaluating the Objective and Subjective Impacts of xAI on Human-Agent Interaction”. In: *International Journal of Human-Computer Interaction* 39.7, pp. 1–15.
- Silver David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis [2018]. “A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play”. In: *Science* 362.6419, pp. 1140–1144. DOI: 10.1126/science.aar6404.
- Simonyan Karen, Andrea Vedaldi, and Andrew Zisserman [2013]. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *CoRR* abs/1312.6034. arXiv: 1312.6034.
- Singla Sumedha, Motahhare Eslami, Brian Pollack, Stephen Wallace, and Kayhan Batmanghelich [2023]. “Explaining the black-box smoothly - A counterfactual approach”. In: *Medical Image Anal.* 84, p. 102721.
- Sixt Leon, Maximilian Granz, and Tim Landgraf [2020]. “When Explanations Lie: Why Many Modified BP Attributions Fail”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, pp. 9046–9057.
- Springenberg Jost Tobias, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller [2014]. “Striving for Simplicity: The All Convolutional Net”. In: *CoRR* abs/1412.6806. arXiv: 1412.6806.
- Such Felipe Petroski, Vashisht Madhavan, Rosanne Liu, Rui Wang, Pablo Samuel Castro, Yulun Li, Jiale Zhi, Ludwig Schubert, Marc G. Bellemare, Jeff Clune, and Joel Lehman [2019]. “An Atari Model Zoo for Analyzing, Visualizing, and Comparing Deep Reinforcement Learning Agents”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3260–3267. ISBN: 978-0-9992411-4-1.

- Sundararajan Mukund, Ankur Taly, and Qiqi Yan [2017]. “Axiomatic attribution for deep networks”. In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 3319–3328.
- Sutton Richard S and Andrew G Barto [2018]. *Reinforcement Learning: An Introduction*. MIT press.
- Swartout William R [1983]. “XPLAIN: A system for creating and explaining expert consulting programs”. In: *Artificial intelligence* 21.3, pp. 285–325.
- Tabrez Aaqib, Matthew B. Luebbbers, and Bradley Hayes [2022]. “Descriptive and Prescriptive Visual Guidance to Improve Shared Situational Awareness in Human-Robot Teaming”. In: *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pp. 1256–1264.
- Todorov Emanuel, Tom Erez, and Yuval Tassa [2012]. “MuJoCo: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. DOI: 10.1109/IRoS.2012.6386109.
- Tomczak Maciej and Ewa Tomczak [2014]. “The need to report effect size estimates revisited. An overview of some recommended measures of effect size.” In: *Trends in Sport Sciences* 21.1.
- Tomsett Richard, Dave Braines, Dan Harborne, Alun D. Preece, and Supriyo Chakraborty [2018]. “Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems”. In: *CoRR* abs/1806.07552. arXiv: 1806.07552.
- Tomsett Richard, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun D. Preece [2020]. “Sanity Checks for Saliency Metrics”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pp. 6021–6029. URL: <https://aaai.org/ojs/index.php/AAAI/article/view/6064>.
- Topin Nicholay, Stephanie Milani, Fei Fang, and Manuela Veloso [2021]. “Iterative Bounding MDPs: Learning Interpretable Policies via Non-Interpretable Methods”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.11, pp. 9923–9931. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i11.17192.
- Topin Nicholay and Manuela Veloso [2019]. “Generation of Policy-Level Explanations for Reinforcement Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 2514–2521. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33012514.
- Torrey Lisa and Matthew Taylor [2013]. “Teaching on a budget: Agents advising agents in reinforcement learning”. In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, AAMAS*, pp. 1053–1060.
- Van Hasselt Hado, Arthur Guez, and David Silver [2016]. “Deep reinforcement learning with double q-learning”. In: *Proceedings of the AAAI conference on artificial intelligence*, pp. 2094–2100.

- Van Seijen Harm, Mehdi Fatemi, Joshua Romoff, Romain Laroche, Tavian Barnes, and Jeffrey Tsang [2017]. “Hybrid Reward Architecture for Reinforcement Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/1264a061d82a2edae1574b07249800d6-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/1264a061d82a2edae1574b07249800d6-Paper.pdf).
- Vogt W.P. [2005]. *Dictionary of Statistics & Methodology: A Nontechnical Guide for the Social Sciences*. Sage Publications. ISBN: 9780761988557.
- Wachter Sandra, Brent Mittelstadt, and Chris Russell [2017]. “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. In: *Harv. JL & Tech.* 31, p. 841.
- Wang Ziyu, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas [2016a]. “Sample efficient actor-critic with experience replay”. In: *CoRR* abs/1611.01224. arXiv: 1611.01224.
- Wang Ziyu, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas [2016b]. “Dueling Network Architectures for Deep Reinforcement Learning”. In: *Proceedings of the 33rd International Conference on Machine Learning, ICML*, pp. 1995–2003.
- Weber Klaus, Lukas Tinnes, **Tobias Huber**, Alexander Heimerl, Marc-Leon Reinecker, Eva Pohlen, and Elisabeth André [2020]. “Towards Demystifying Subliminal Persuasiveness: Using XAI-Techniques to Highlight Persuasive Markers of Public Speeches”. In: *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*. Vol. 12175, pp. 113–128. ISBN: 978-3-030-51923-0 978-3-030-51924-7. DOI: 10.1007/978-3-030-51924-7\_7.
- Weitkamp Laurens, Elise van der Pol, and Zeynep Akata [2019]. “Visual Rationalizations in Deep Reinforcement Learning for Atari Games”. In: *Artificial Intelligence, 30th Benelux Conference, BNAIC*, pp. 151–165. ISBN: 978-3-030-31978-6. DOI: 10.1007/978-3-030-31978-6\_12.
- Weitz Katharina, Teena Hassan, Ute Schmid, and Jens-Uwe Garbas [2019a]. “Deep-learned faces of pain and emotions: Elucidating the differences of facial expressions with the help of explainable AI methods”. In: *tm-Technisches Messen* 86.7-8, pp. 404–412. DOI: 10.1515/teme-2019-0024.
- Weitz Katharina, Dominik Schiller, Ruben Schlagowski, **Tobias Huber**, and Elisabeth André [2019b]. ““Do you trust me?”: Increasing User-Trust by Integrating Virtual Agents in Explainable AI Interaction Design”. In: *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents, IVA*, pp. 7–9. DOI: 10.1145/3308532.3329441.
- Weitz Katharina, Dominik Schiller, Ruben Schlagowski, **Tobias Huber**, and Elisabeth André [2021]. ““Let me explain!”: exploring the potential of virtual agents in explainable AI interaction design”. In: *J. Multimodal User Interfaces* 15.2, pp. 87–98. DOI: 10.1007/S12193-020-00332-0.

- Wells Lindsay and Tomasz Bednarz [2021]. “Explainable AI and Reinforcement Learning - A Systematic Review of Current Approaches and Trends”. In: *Frontiers Artif. Intell.* 4, p. 550030. DOI: 10.3389/frai.2021.550030.
- Yang Mengjiao and Been Kim [2019]. “Benchmarking attribution methods with relative feature importance”. In: *CoRR* abs/1907.09701. arXiv: 1907.09701.
- Yu Chao, Jiming Liu, Shamim Nemati, and Guosheng Yin [2021]. “Reinforcement Learning in Healthcare: A Survey”. In: *ACM Computing Surveys* 55.1. ISSN: 0360-0300. DOI: 10.1145/3477600.
- Zahavy Tom, Nir Ben-Zrihem, and Shie Mannor [2016]. “Graying the black box: Understanding DQNs”. In: *Proceedings of the 33rd International Conference on Machine Learning, ICML*, pp. 1899–1908. URL: <http://proceedings.mlr.press/v48/zahavy16.html>.
- Zeiler Matthew D. and Rob Fergus [2014]. “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision - ECCV 2014 - 13th European Conference, Proceedings, Part I*, pp. 818–833.
- Zhang Baobao and Allan Dafoe [2019]. “Artificial intelligence: American attitudes and trends”. In: *Available at SSRN 3312874*.
- Zhao Xiangyu, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Jiliang Tang, and Hui Liu [2021]. “DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 1, pp. 750–758. DOI: 10.1609/aaai.v35i1.16156.
- Zhao Yunxia [2020]. “Fast Real-time Counterfactual Explanations”. In: *CoRR* abs/2007.05684. arXiv: 2007.05684.
- Zhou Yilun, Serena Booth, Marco Tulio Ribeiro, and Julie Shah [2022]. “Do Feature Attribution Methods Correctly Attribute Features?” In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. Vol. 36. 9, pp. 9623–9633.

**Part VII.**  
**Appendix**

# A. My Publications

## A.1. Main Publications

This dissertation builds upon the following five peer-reviewed publications. Some of the content included in this thesis – such as text, figures, and results – has previously appeared in these publications.

- **Tobias Huber**, Dominik Schiller, and Elisabeth André [2019]. “Enhancing Explainability of Deep Reinforcement Learning Through Selective Layer-Wise Relevance Propagation”. In: *KI 2019: Advances in Artificial Intelligence*. Springer International Publishing, pp. 188–202

**My Contribution:** I conceived and implemented the proposed algorithm, conducted the experiments, and wrote major parts of the paper.

- **Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

**My Contribution:** Together with Ofra Amir, I conceived the proposed combined explanation method. I designed major parts of the user study, conducted the study, and wrote major parts of the code and paper.

- **Tobias Huber**, Benedikt Limmer, and Elisabeth André [2022]. “Benchmarking Perturbation-Based Saliency Maps for Explaining Atari Agents”. In: *Frontiers in Artificial Intelligence* 5. ISSN: 2624-8212. DOI: 10.3389/frai.2022.903875

**My Contribution:** I supervised Benedikt Limmer’s bachelor thesis, which was the starting point for this paper. Building on this thesis, I wrote major parts of the paper and code.

- Yael Septon, **Tobias Huber**, Elisabeth André, and Ofra Amir [2023]. “Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection - 21st International Conference*. Vol. 13955. Lecture Notes in



Computer Science. Springer, pp. 320–332. DOI: 10.1007/978-3-031-37616-0\_27

**My Contribution:** I supervised the student projects by Julian Stockmann and Simone Pompe that implemented reward decomposition for the Atari game Pacman. Based on this, I trained the final Pacman agents for the user study. I helped design the study and wrote major parts of the paper.

- **Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS. IFAAMAS*, 10 pages

**My Contribution:** Together with Silvan Mertes, I conceived the proposed counterfactual explanation approach. I supervised Maximilian Demmler’s master’s thesis, which was the starting point for this paper. Building on this thesis, I wrote parts of the final code and trained additional agents and models. I designed and conducted major parts of the user study and wrote major parts of the paper.

## A.2. Other Publications

This section lists other publications that are not part of this dissertation but were conducted or written during the doctorate:

- **Tobias Huber**, Silvan Mertes, Stanislava Rangelova, Simon Flutura, and Elisabeth André [2021a]. “Dynamic Difficulty Adjustment in Virtual Reality Exergames through Experience-driven Procedural Content Generation”. In: *IEEE Symposium Series on Computational Intelligence, SSCI*, pp. 1–8. DOI: 10.1109/SSCI50451.2021.9660086
- Silvan Mertes, **Tobias Huber**, Katharina Weitz, Alexander Heimerl, and Elisabeth André [2022a]. “GANterfactual - Counterfactual Explanations for Medical Non-experts Using Generative Adversarial Learning”. In: *Frontiers Artif. Intell.* 5, p. 825565. DOI: 10.3389/frai.2022.825565
- Silvan Mertes, Christina Karle, **Tobias Huber**, Katharina Weitz, Ruben Schlagowski, and Elisabeth André [2022b]. “Alterfactual Explanations - The Relevance of Irrelevance for Explaining AI Systems”. In: *CoRR* abs/2207.09374. arXiv: 2207.09374

- Silvan Mertes, **Tobias Huber**, Christina Karle, Katharina Weitz, Ruben Schlagowski, Cristina Conati, and Elisabeth André [2024]. “Relevant Irrelevance: Generating Alterfactual Explanations for Image Classifiers”. In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*. International Joint Conferences on Artificial Intelligence Organization, pp. 467–475. DOI: 10.24963/ijcai.2024/52
- Pooja Prajod, Dominik Schiller, **Tobias Huber**, and Elisabeth André [2021]. “Do Deep Neural Networks Forget Facial Action Units? - Exploring the Effects of Transfer Learning in Health Related Facial Expression Recognition”. In: *International Workshop on Health Intelligence*. Springer, pp. 217–233
- Pooja Prajod, **Tobias Huber**, and Elisabeth André [2022]. “Using Explainable AI to Identify Differences Between Clinical and Experimental Pain Detection Models Based on Facial Expressions”. In: *MultiMedia Modeling - 28th International Conference, MMM, Proceedings, Part I*, pp. 311–322. DOI: 10.1007/978-3-030-98358-1\_25
- Stanislava Rangelova, Simon Flutura, **Tobias Huber**, Daniel Motus, and Elisabeth André [2019]. “Exploration of Physiological Signals Using Different Locomotion Techniques in a VR Adventure Game”. In: *Universal Access in Human-Computer Interaction. Theory, Methods and Tools - 13th International Conference, UAHCI 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Proceedings, Part I*. ed. by Margherita Antona and Constantine Stephanidis. Vol. 11572. Lecture Notes in Computer Science. Springer, pp. 601–616. DOI: 10.1007/978-3-030-23560-4\_44
- Dominik Schiller, **Tobias Huber**, Florian Lingensfelder, Michael Dietz, Andreas Seiderer, and Elisabeth André [2019]. “Relevance-Based Feature Masking: Improving Neural Network Based Whale Classification Through Explainable Artificial Intelligence”. In: *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association*, pp. 2423–2427. DOI: 10.21437/Interspeech.2019-2707
- Dominik Schiller, **Tobias Huber**, Michael Dietz, and Elisabeth André [2020]. “Relevance-Based Data Masking: A Model-Agnostic Transfer Learning Approach for Facial Expression Recognition”. In: *Frontiers in Computer Science* 2. DOI: 10.3389/fcomp.2020.00006
- Ruben Schlagowski, Frederick Herget, Niklas Heimerl, Maximilian Hammerl, **Tobias Huber**, Pamina Zwolsky, Jan Gruca, and Elisabeth André [2024]. “From a Social POV: The Impact of Point of View on Player

- Behavior, Engagement, and Experience in a Serious Social Simulation Game”. In: *Proceedings of the 19th International Conference on the Foundations of Digital Games*. FDG '24. New York, NY, USA: Association for Computing Machinery. ISBN: 9798400709555. DOI: 10.1145/3649921.3649936
- Anan Schütt, **Tobias Huber**, Ilhan Aslan, and Elisabeth André [July 2023]. “Fast Dynamic Difficulty Adjustment for Intelligent Tutoring Systems with Small Datasets”. In: *Proceedings of the 16th International Conference on Educational Data Mining*. Ed. by Mingyu Feng, Tanja Käser, and Partha Talukdar. Bengaluru, India: International Educational Data Mining Society, pp. 482–489. ISBN: 978-1-73367-364-8. DOI: 10.5281/zenodo.8115740
  - Anan Schütt, **Tobias Huber**, Jauwairia Nasir, Cristina Conati, and Elisabeth André [Mar. 2024]. “Does Difficulty Even Matter? Investigating Difficulty Adjustment and Practice Behavior in an Open-Ended Learning Task”. In: *Proceedings of the 14th Learning Analytics and Knowledge Conference*. LAK '24. New York, NY, USA: Association for Computing Machinery, pp. 253–262. ISBN: 9798400716188. DOI: 10.1145/3636555.3636876
  - Klaus Weber, Lukas Tinnes, **Tobias Huber**, Alexander Heimerl, Marc-Leon Reinecker, Eva Pohlen, and Elisabeth André [2020]. “Towards Demystifying Subliminal Persuasiveness: Using XAI-Techniques to Highlight Persuasive Markers of Public Speeches”. In: *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*. Ed. by Davide Calvaresi, Amro Najjar, Michael Winikoff, and Kary Främling. Vol. 12175. Cham: Springer International Publishing, pp. 113–128. ISBN: 978-3-030-51923-0 978-3-030-51924-7. DOI: 10.1007/978-3-030-51924-7\_7
  - Katharina Weitz, Dominik Schiller, Ruben Schlagowski, **Tobias Huber**, and Elisabeth André [2019b]. ““Do you trust me?”: Increasing User-Trust by Integrating Virtual Agents in Explainable AI Interaction Design”. In: *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents, IVA*, pp. 7–9. DOI: 10.1145/3308532.3329441
  - Katharina Weitz, Dominik Schiller, Ruben Schlagowski, **Tobias Huber**, and Elisabeth André [2021]. ““Let me explain!”: exploring the potential of virtual agents in explainable AI interaction design”. In: *J. Multimodal User Interfaces* 15.2, pp. 87–98. DOI: 10.1007/S12193-020-00332-0

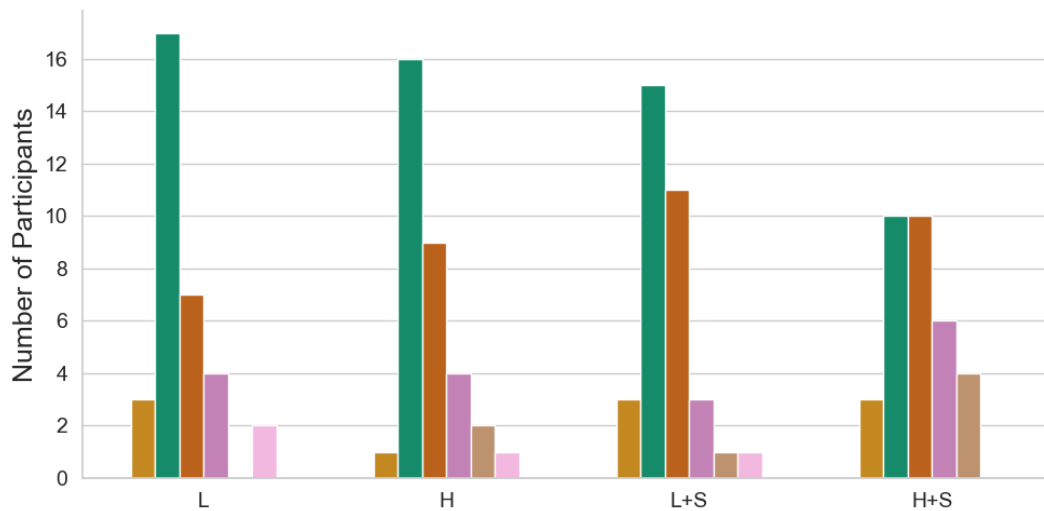
## B. Appendix to the First User Study

This chapter provides additional details for the study described in Chapter 10. It is based on the appendix in our paper:

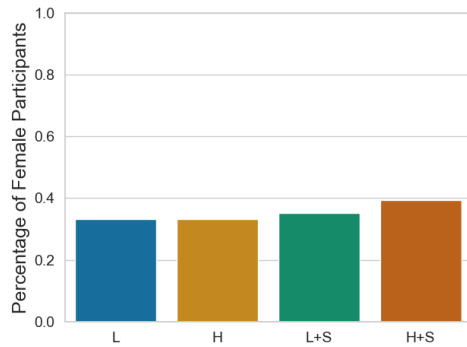
**Tobias Huber**, Katharina Weitz, Elisabeth André, and Ofra Amir [2021b]. “Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps”. In: *Artif. Intell.* 301, p. 103571. DOI: 10.1016/j.artint.2021.103571

### B.1. Participants Demographics

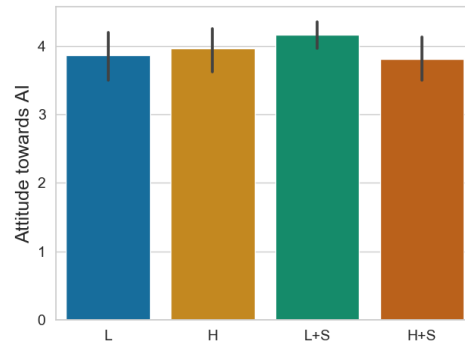
In this section, we provide more details regarding participants’ demographics. As Figure B.1 shows, most participants were between 18 and 34 years old.



**Figure B.1.:** The number of participants in each age group per condition. The bars show from left to right: “18-24”, “25-34”, “35-44”, “45-54”, “55-64” and “65 or older”. The categories “17 or younger” and “do not want to specify” were never selected.



**Figure B.2.:** Percentage of female participants per condition.



**Figure B.3.:** The average attitude towards AI, rated on a 5-point Likert scale.

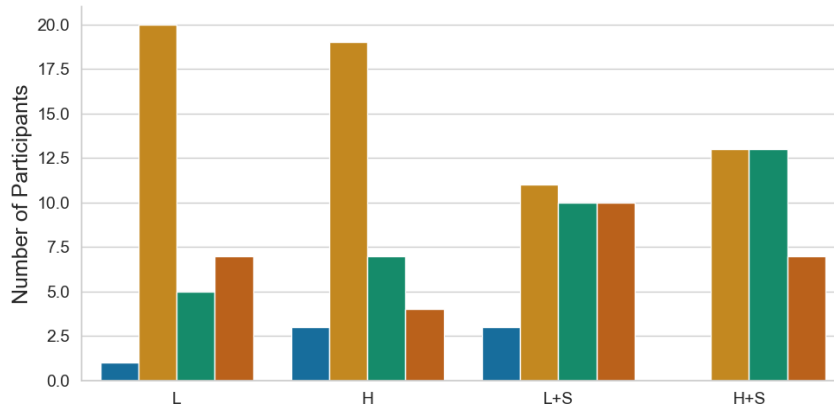
There were no major differences in gender distribution between the four conditions (Figure B.2).

We verified that participants in different conditions did not differ much in their AI experience and views and in their Pacman experience. To this end, we asked them when they played Pacman for the last time (1=“never”, 2=“more than 5 years ago”, 3=“less than 5 years ago”, 4=“less than 1 year ago”). Across all four conditions, the median group was 2: “I played Pacman more than 5 years ago”. A comparison is shown in Figure B.4.

For the AI experience, we adapted a description of AI from Zhang and Dafoe [2019] and Russell and Norvig [2016] to “The following questions ask about Artificial Intelligence (AI). Colloquially, the term ‘artificial intelligence’ is often used to describe machines (or computers) that mimic ‘cognitive’ functions that humans associate with the human mind, such as ‘learning’ and ‘problem solving’. AI agents are already able to perform some complex tasks better than the median human (today). Examples for such intelligent agents are search engines, chatbots, chess bots, and voice assistants.”

After that, every participant who stated to have AI experience (104 across all conditions) had to select one or more of the following items:

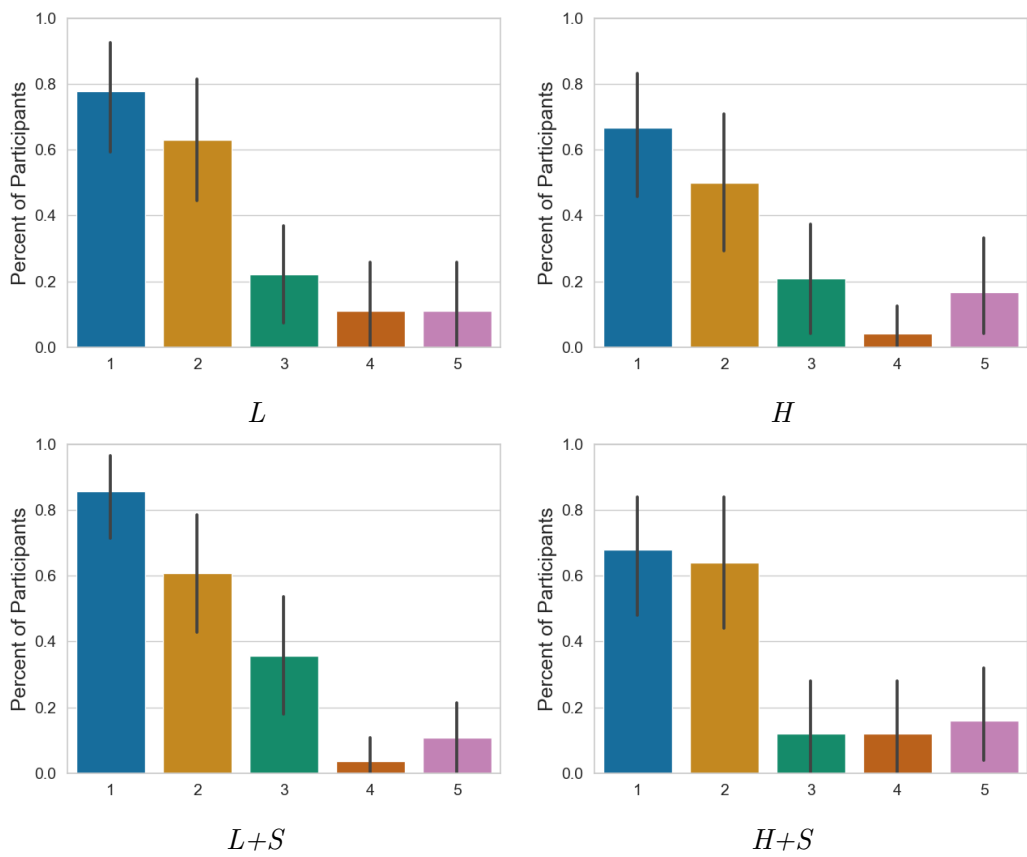
- ☛ 1: I know AI from the media.
- ☛ 2: I use AI technology in my private life.
- ☛ 3: I use AI technology in my work.
- ☛ 4: I took at least one AI related course.
- ☛ 5: I do research on AI related topics.
- ☛ Other:



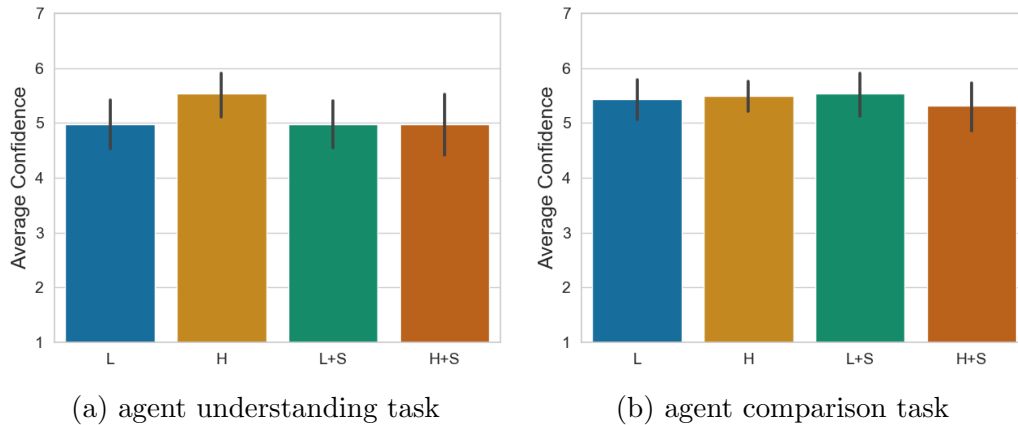
**Figure B.4.:** The Pacman experience across all conditions. The bars depict when the participants played Pacman the last time. From left to right, the bars represent: “never”, “more than 5 years ago”, “less than 5 years ago” and “less than 1 year ago”.

The last free form option was used exactly once, and it read “work on MTurk”. The distribution of the other items for each condition is shown in Figure B.5.

To measure the participants’ attitude towards AI, we adapted a question from Zhang and Dafoe [2019] and asked them to rate their answer to the question “Suppose that AI agents would achieve high-level performance in more areas one day. How positive or negative do you expect the overall impact of such AI agents to be on humanity in the long run?” on a 5-point Likert scale from “Extremely negative” to “Extremely positive”. The results are shown in Figure B.3.



**Figure B.5.:** Distribution of the chosen AI experience items for each condition. The x-axis depicts the items described above.



**Figure B.6.:** The average confidence that participants in each condition had in their answers during each task.

## B.2. Supplementary Results

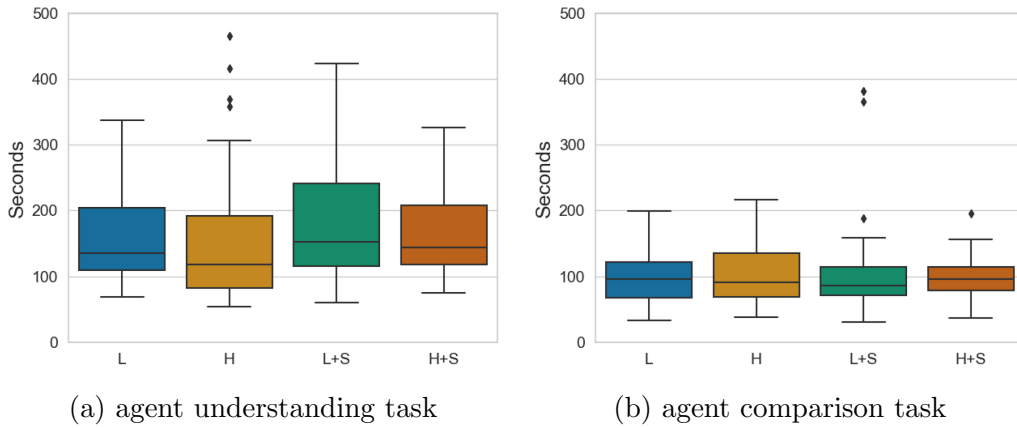
In this section, we present additional information about the results of the study that goes beyond the main hypotheses we explored and described in the paper.

**Confidence, Time and Pauses.** To investigate whether participants were confident in their decisions, they had to rate their confidence in each of their selections (item selection in the agent understanding task and agent selection in the agent comparison task) on a 7-point Likert scale. The results across each task are shown in Figure B.6.

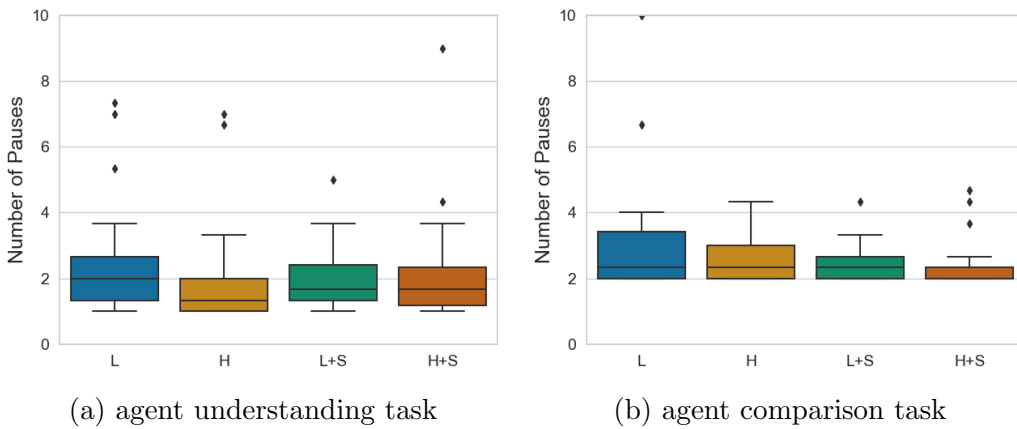
To evaluate whether participants were especially diligent or effective during the tasks, we measured the time that each participant stayed on each page of the survey and calculated the average time per task (each task consists of three pages). Furthermore, we kept track of each time a video was paused, as described in Section 10.1.3. The average completion times of participants and the average number of pauses are shown in Figure B.7 and B.8, respectively (shown in boxplots due to the presence of several outliers that strongly affect the mean values).

Figure B.6 (a) shows that participants in condition *H* were slightly more confident on average in their analysis of the agents. This is also reflected by the lesser amount of time per analysis (Figure B.7 (a)) and pauses (Figure B.8 (a)). Apart from this, there are no obvious differences between the average confidence, time, and pause values for each task (Figure B.6 to B.8).



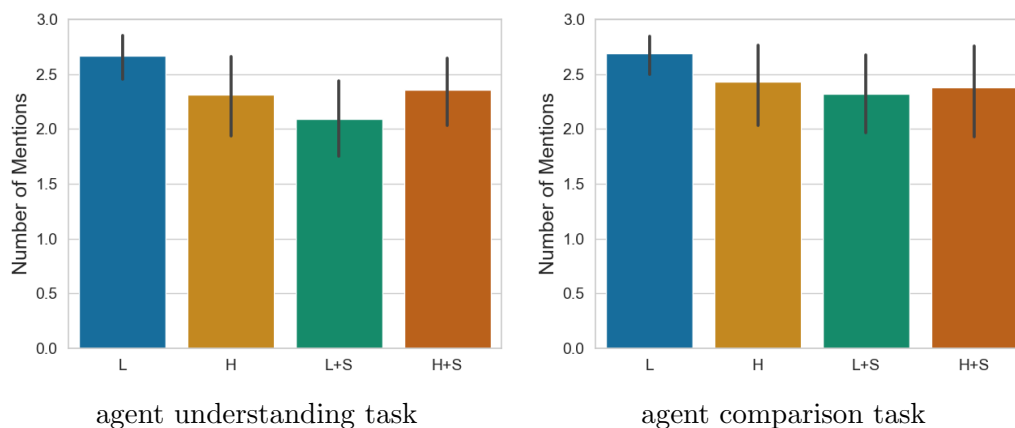


**Figure B.7.:** The average time taken by participants in each condition per agent analysis (a) and comparison of agent pairs (b).

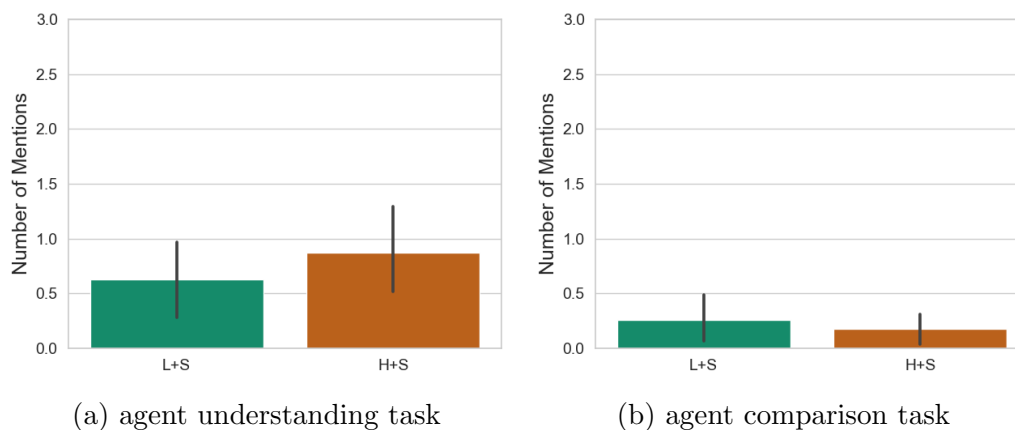


**Figure B.8.:** The average number of times that participants in each condition paused the videos during each agent analysis (a) and comparison of agent pairs (b).

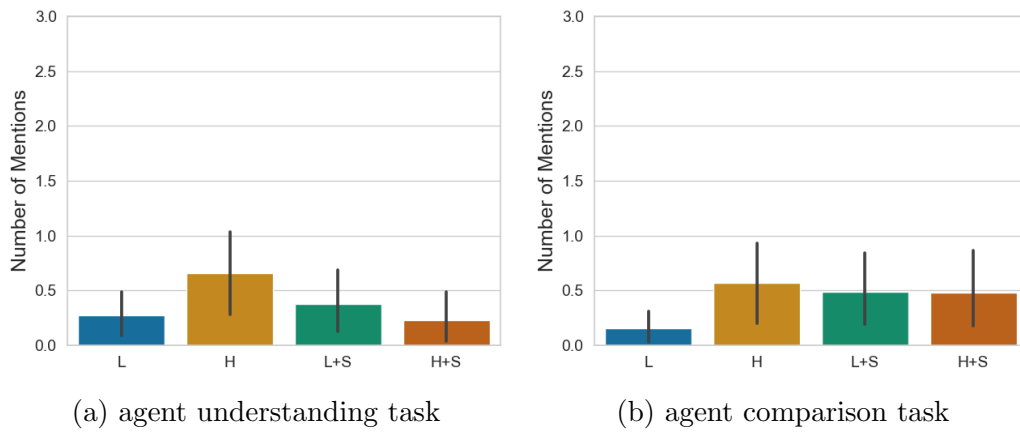
**Participants' Justifications.** As described in Section 10.1.3, an independent coder identified different concepts inside the participants' justifications. Figures B.9 and B.10 show the average number of mentions of *gameplay* and of *saliency maps* in the different tasks across the different conditions. As discussed in Section 10.2, most participants mainly based their justifications on the agents' gameplay (Figure B.9). In the saliency conditions, participants seldom mention the saliency maps in their justifications (see Figure B.10). Finally, Figure B.11 shows that participants in condition *H* gave more unjustified explanations in the agent understanding task. However, this observation did not repeat in the agent comparison task.



**Figure B.9.:** Comparison of how often the participants referenced the agents' gameplay in their justifications for their answers.



**Figure B.10.:** Comparison of how often the participants referenced the green highlighting of the LRP-argmax saliency maps in their justifications for their answers.



**Figure B.11.:** Comparison of how often the participants justifications contained **unjustified** arguments.

**Table B.1.:** The specific scores that participants received for selecting each object during the agent understanding task.

selected object	<i>Power pill agent</i>	<i>Regular agent</i>	<i>Fear-ghosts agent</i>
“Pacman”	1	1	1
“normal pill”	−1	−0.5	−0.5
“power pill”	1	−0.5	−0.5
“normal ghost”	−1	−0.5	1
“blue ghost”	−1	1	1
“cherry”	−1	−0.5	−0.5

### B.3. Evaluation of the Retrospection Task

As described in Section 10.1.3, we evaluated participants’ scores in the object selection part of the retrospection task with a simple scoring function based on predefined answers by two of the authors involved in the training of the agents. Hereby, we assign a score of 1 to each object that is connected to the agents’ specific goal and their source of information (Pacman’s position for all agents), −1 for each object that was not related to the agents’ reward function and −0.5 to objects that were related to the reward but on which the agent did not focus. The specific scores are shown in Table B.1.

For the free form answers to the question “Please describe the strategy of the AI agent”, an independent coder identified various not mutually exclusive concepts contained in the participants’ answers. We aggregated these concepts into the following 16 groups, where the coder used ‘G’ for ghosts, ‘PP’ for power pills, and ‘NP’ for normal pills:

1. *eating power pills*: “eating PP”, “eating as many PP as possible”, “eat PP when ghosts are near”, “eat PP when ghosts are near”, “prioritizing PP”, “prioritizing PP to eat ghosts”, “prioritizing PP , but not eat ghosts”, “eat PP to get points”
2. *ignore power pills*: “do not care about PP”
3. *eat normal pills*: “eat NP to get points”, “eating NP”, “eating as many NP as possible”, “prioritizing NP”, “clearing the stage”
4. *ignore normal pills*: “do not care about NP”, “focus on areas without [sic] NP”
5. *avoid ghosts*: “avoiding G”, “avoiding G strongly”, “wait for G to go away”, “outmanoeuvring G”, “hiding from G”, “mislead ghosts”, “avoids being eaten / caught”, “avoiding to lose / staying alive”, “stays away from danger”

6. *move towards ghosts*: “being close to G”, “trying to eat G NON blue”, “(easily) caught by G”, “easily caught by G”
7. *ignore ghosts*: “do not care about G”
8. *making ghosts blue*: “making G blue”
9. *eat blue ghosts*: “being close to blue G”, “eating as many G as possible”, “eat blue G to get points”, “chasing/going for G”, “eating the blue G”, “eating to jail many G”(jailing since the ghosts move back to jail after being eaten), “prioritizing PP to eat ghosts”
10. *avoid blue ghosts*: “avoiding blue G”
11. *ignore blue ghosts*: “do not care about blue G”, “prioritizing PP , but not eat ghosts”
12. *eat cherry*: “prioritizing cherry”, “eat cherry to get points”, “going for cherry”, “eating cherry”
13. *ignore cherry*: “do not care about cherry”
14. *random movement*: “moving randomly”, “move all over map”, “switching directions /back&forth”, “not moving / being stuck”, “sticking to walls / outside”, “confused”, “without strategy /random”, “not planning ahead”, “switching directions”
15. *focus on Pacman*: “focus on PM”, “focus on whats in front of/around PM”, “stuck to itself”
16. *staying in corners*: “staying in corners”

These groups are used to define a simple scoring function. Depending on the agent, each group could either be positive, neutral, or negative. Positive groups contain concepts that are in line with the predefined descriptions of the agents’ strategies by two of the authors involved in the training. Neutral groups consist of correct observations, which are byproducts of the agent’s strategy, and negative concepts go against the agent’s strategy. Each positive group contained in an answer increased the participant’s score by 1, and each negative group decreased the score by  $-1$ . Here, we define a group to be “contained in an answer” if at least one concept of this group was included in the answer. Neutral groups did not affect the score.

*Power pill agent:*

- *positive*: “eat power pill”, “ignore normal pill”, “ignore ghosts”, “ignore blue ghost”, “ignore cherry”, “focus on Pacman”, “staying in corners”

- *neutral*: “eat normal pill”, “making ghosts blue”

*Regular agent*:

- *positive*: “ignore cherry”, “focus on Pacman”, “making ghosts blue”, “eat blue ghost”
- *neutral*: “eat normal pill”, “eat power pill”, “ignore ghosts”

*Fear-ghosts agent*:

- *positive*: “avoid ghost”, “focus on Pacman”, “making ghosts blue”, “eat blue ghost”, “ignore cherry”
- *neutral*: “eat normal pill”, “eat power pill”

## B.4. Questionnaire

In this section, we provide the complete questionnaire used in the first study. On the first page, the participants were asked to provide personal information:

Personal information

What is your age?

Choose one of the following answers

17 or younger

18 - 24

25 - 34

35 - 44

45 - 54

55 - 64

65 or older

I prefer not to specify

To which gender identity do you most identify?

Choose one of the following answers

Male

Female

I prefer not to answer.

Other:

Do you have a colour vision impairment?

Yes  No

Did you play Pacman before?

Choose one of the following answers

No

The last time I played Pacman was less than 1 year ago.

The last time I played Pacman was less than 5 years ago.

The last time I played Pacman was more than 5 years ago.

The following questions ask about Artificial Intelligence (AI). Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving". AI agents are already able to perform some complex tasks better than the median human (today). Examples for such intelligent agents are search engines, chatbots, chessbots and voice assistants.

Do you have experience with AI (Artificial Intelligence)?

Yes  No

Do you have experience with AI (Artificial Intelligence)?

Yes  No

\*What kind of AI experience do you have?

Check all that apply

- I know AI from the media.
- I use AI technology in my private life.
- I use AI technology in my work.
- I took at least one AI related course.
- I do research on AI related topics.
- Other:

\*Suppose that AI agents would achieve high-level performance in more areas one day.

	1: Extremely negative	2	3	4	5: Extremely positive	I don't know
How positive or negative do you expect the overall impact of such AI agents to be on humanity in the long run?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



## Information about Pacman:

### Instructions

In this survey, you will be shown summaries of Pacman games. These videos will show you parts of games played by various AI agents trained to play Pacman. Later you will be asked to complete 6 tasks based on those videos. You will earn a 10 cent bonus for each task you solved correctly.

Please carefully read the following description of Pacman. There will be a short quiz to make sure you understood the information correctly.

A Pacman game takes place in a Labyrinth:



Pacman's goal is to eat as many "pills" (pink rectangles) as possible to earn points, while escaping the ghosts.

Pacman receives 10 points for each pill, and dies when it's eaten by a ghost.

In addition to the regular pellets, there are also four special "power pills" (large pink rectangles). Pacman receives 50 points for eating a power pill.

Eating a power pill also gives Pacman limited time during which the ghosts become blue and Pacman can eat the ghosts. Pacman receives 200, 400, 800, 1600 points for each ghost it eats successively.

After a ghost is eaten, its eyes move back to the ghost box in the middle of the screen and the ghost restarts from there in his normal form.

At random intervals cherries spawn and move through the labyrinth.

If PacMan eats the cherry he gets 100 points.

Pacman starts the game with three lives and loses one life each time it gets eaten by a ghost.

Please use this link to play the game and get familiar with it. (While the objects look different in that version the rules are the same as described above.):

[Pacman](#)

This quiz tests whether the participants understood the information about Pacman. Participants were sent back to the previous page if they got an answer wrong.


### Quiz Pacman

Please fill out this quiz to show that you understood the information.

\*Eating  gives Pacman:


Choose one of the following answers

- No points
- 1 point
- 10 points

\*What is shown in this image: 

Choose one of the following answers

- A ghost.
- A blue ghost.
- Pacman.

\*What is shown in this picture? 

Choose one of the following answers

- a piece of wall.
- a power pill.
- a regular pill.

\*When do ghosts become blue?

Choose one of the following answers

- Every 2 minutes
- When Pacman eats a power pill
- When Pacman gets close to a ghost

\*What happens when ghosts are blue?

Choose one of the following answers

- Pacman can eat them
- Ghosts move faster
- Ghosts move slower

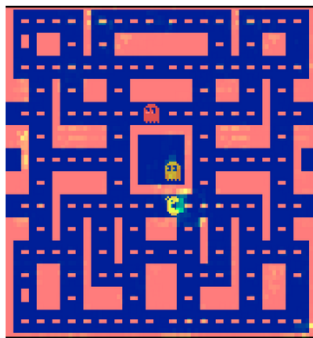
Additional information about the provided explainable AI methods. The information about saliency maps was only displayed if the participant was in one of the saliency conditions.


#### Additional Information

In the following questions you will be shown videos that **summarize the behavior** of an AI agent that was trained to play Pacman. These videos contain **several scenes** from a longer session of gameplay and aim to show you, in a limited amount of time (about 20 seconds), **how the agent plays the game**.

To aid you in your tasks, the screens will be augmented with green heatmaps that show **how relevant a pixel was for the decision of the AI agent** that controls Pacman ("what the agent is looking at"). On some screens the green color may appear yellowish.

The **brighter the green** highlighting of a pixel is, the **more relevance** it had for the decision of the AI agent.



In this example, Pacman is paying a lot of attention on the area in front of itself 

This quiz tests whether the participants understood the information about the provided explainable AI methods. Participants were sent back to the previous page if they got an answer wrong.

#### Second Quiz

Please fill out this quiz to show that you understood the information.

\*A green area is:

● Choose one of the following answers

- relevant for the Pacman agent.
- relevant for your decision.
- relevant for the ghosts.

\*If the green highlighting of an area is very bright then:

● Choose one of the following answers

- the area is worth more points.
- the area is more relevant.
- the area is less relevant.

\*The videos in the following questions show:

● Choose one of the following answers

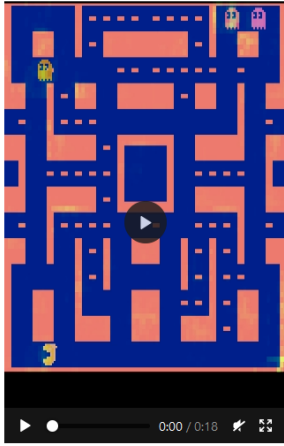
- scenes where the Pacman agent performed very good.
- a summary of how the Pacman agent behaves.
- scenes where the Pacman agent performed very bad.

This is the agent understanding task that was repeated for each of the three agents in a randomized order:

Analyze the agent

In these three tasks you will be shown **summaries of different AI agents** trained to play Pacman. For each AI agent, you will be asked to **analyze the strategy of that particular AI agent**. To this end, you will be asked to describe the AI agents' strategy and to select the objects that were the most important for the strategy of that particular AI agent. You will receive a bonus of 10 cents for each agent that you analyze correctly.







The video shows a summary of typical behavior of a Pacman agent. Please watch the whole video.



\*Please briefly describe the strategy of the AI agent shown in the video above:

\*Based on this video, select the objects that you think were most important for the strategy of this particular AI agent.  
 Select a **maximum of 3** (it does not necessarily have to be 3) Objects.

🟢 Check all that apply

- Pacman 
- Normal pills 
- Power pills 
- Ghosts 
- Blue Ghosts 
- Cherrys 

\*How confident are you in your selection?

	<b>Not at all confident</b>						<b>very confident</b>
How confident are you that you chose the right objects?	○	○	○	○	○	○	○

\*Please briefly explain how you came to your selection:

After all three agents, the participants were asked for their satisfaction:

Explanation Satisfaction

\*

	1 : I disagree strongly.	2	3	4	5 : I agree strongly.
From watching the videos of the AI agents, I got an idea of the agents' strategies.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos showing the AI agents play contain sufficient detail about the agents' behavior.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos showing the AI agents play contain irrelevant details.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The gameplay scenarios shown in the videos were useful for analyzing the agents' behavior	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The green highlighting in the videos was useful for analyzing the agents' behavior	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Do you have additional comments?

This is the agent comparison task that was repeated for each combination of the three agents in a randomized order:

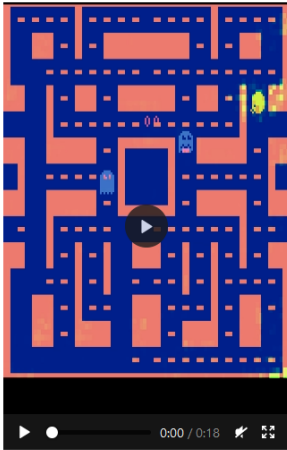
Compare the agents

Below are videos showing parts of games played by **two different Pacman agents:** Agent A (left) and Agent B (right).

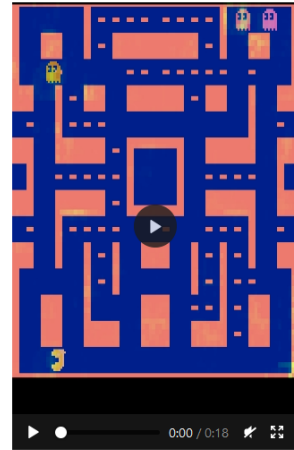
Based on these videos, choose which Pacman agent you'd like to play on your behalf. You will receive a bonus of 10 cents for each time you select the Pacman agent that achieves higher points on average.

**Note:** Press play to start a video. You can pause and rewatch the videos by pressing the button again. Please watch the whole videos.

Agent A



Agent B



Which of the Pacman agents do you choose to play on your behalf?

Choose one of the following answers

- Player A
- Player B

How confident are you that you chose the better agent?

	Not at all confident						very confident
How confident are you that you chose the better player?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Please briefly explain your selection:



After all three comparisons, the participants were asked for their satisfaction again:

#### Explanation Satisfaction

**\***

	1 : I disagree strongly.	2	3	4	5 : I agree strongly.
From watching the videos of the AI agents, I got an idea of the agents' strategies.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos showing the AI agents play contain sufficient detail about the agents' behavior.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos showing the AI agents play contain irrelevant details.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The gameplay scenarios shown in the videos were useful for choosing the agent that performs better.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The green highlighting in the videos was useful for choosing the agent that performs better.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Do you have additional comments?

## C. Appendix to the Second User Study

This chapter provides additional details for the study described in Chapter 11. It is based on the appendix in our paper:

Yael Septon, **Tobias Huber**, Elisabeth André, and Ofra Amir [2023]. “Integrating Policy Summaries with Reward Decomposition for Explaining Reinforcement Learning Agents”. In: *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection - 21st International Conference*. Vol. 13955. Lecture Notes in Computer Science. Springer, pp. 320–332. DOI: 10.1007/978-3-031-37616-0\_27

### C.1. Survey Information

The following figures are screenshots from the second user study. As an example, we show the survey only for the HIGHLIGHTS condition and only include one of the agents. The participants were shown multiple different agents, and depending on their condition, they saw explanations as shown in Figure 8.3.

Dear participant,  
Thank you for participating in this study.  
In this survey, you will be asked about the behavior of Pacman  
(as an artificial intelligence - AI agent).

Estimated study time: 10-15 minutes.  
You can receive a bonus of up to 0.9 dollars for answering  
questions correctly!  
Your responses are confidential and anonymous.

We do not ask for your name or other identifying information.  
Whatever information you convey in this study cannot be traced  
back to you.  
Please read all the instructions and information very carefully.  
This study is very important for us.

We appreciate your cooperation!

I agree to participate



Age:

18 30 43 55 68 80

years

To which gender do you most identify

Male

Female

Non-binary / third gender



Prefer not to say

In this survey, you will be shown videos that demonstrate different actions which Pacman takes in different game situations. The game is played by an AI agent trained to play Pacman. Later on, you will be shown 3 different Pacmans (different AI agents) and you will need to answer questions regarding the Pacmans' behavior. You will get a 30 cent bonus for each Pacman (AI agent) you answer correctly.

Please carefully read the following description of Pacman. There will be a short quiz to ensure you understand the information correctly.

A Pacman game takes place in a Labyrinth:



Pacman's goal is to eat as many "normal pills" (pink rectangles ) as possible to earn points while escaping the ghost (  ). Pacman receives 10 points for each pill and dies when it's eaten by a ghost.

In addition to the normal pills, there are also four special "power pills" (large pink rectangles  ). Pacman receives 50 points for eating a power pill.

Eating a power pill also gives Pacman limited time during which the ghosts become blue (  ) and Pacman can eat the ghosts. Pacman receives 200, 400, 800, 1600 points for each ghost it eats successively.

After a ghost is eaten, its eyes (  ) move back to the ghost box in the middle of the screen (  ) and the ghost restarts from there in its normal form.

At random intervals cherries (  ) spawn and move through the labyrinth.

If Pacman eats cherries he gets 100 points.


Pacman starts the game with three lives and loses one life each time it gets eaten by a ghost.

Eating a "normal pill"  gives Pacman:

1 point

No points

10 points

What is shown in the image: 

A piece of wall

A normal pill

A power pill

When do ghost become blue?

When Pacman gets close to a ghost

Every 2 minutes

When Pacman eats a power pill

What happens when ghosts are blue?

Ghosts move faster

Pacman can eat them

Ghosts move slower

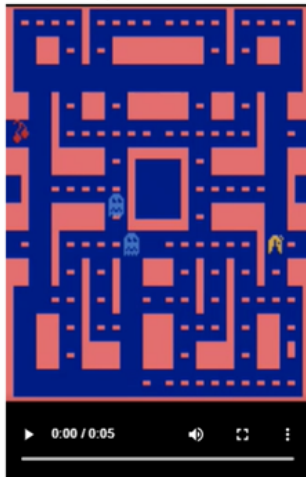
In this survey, you will be shown short scenarios that represent the Pacman's behavior.

When deciding which action to take, Pacman tries to predict its future total score. This score takes into account: (1) the immediate points it gets for choosing the current action, and (2) the points for all of its future actions. These future points are also affected by Pacman's current choice.

Importantly, the scores consist of 4 components: normal pill, power pill, eating a blue ghost and dying.

The Pacman agent chooses the action for which it predicts the highest score. Different Pacman agents might give different weights to the different components. For example, one agent might care more about eating normal pills, while another agent might care more about eating power pills.

For Example:



In this video clip, we can see that Pacman goes directly to the power pill and then returns. This suggests that he is interested in the power pill

How many different components is the score composed of?

- 3
- 4
- 5
- 6

### Explanation of the task

In the task, you will be given a link that leads to a webpage. On the page, you will see a header, a video and buttons (as shown in the picture).

Each video consists of 5 scenarios showing the same agent in different scenarios.

When clicking on a button, the video will jump to a certain time in the scenario.

For example - when clicking on the "Scenario 3" button the simulation jumped to time "0:15"



Note: There are 5 buttons - a button for each scenario. Each scenario shows:

- The same agent in different situations
- Different agents

Open link in a new tab:

[http://ec2-54-149-86-189.us-west-2.compute.amazonaws.com/Pacman/agent2\\_PP\\_R3.html](http://ec2-54-149-86-189.us-west-2.compute.amazonaws.com/Pacman/agent2_PP_R3.html)

If the link doesn't load copy and paste it in a new tab

Please briefly describe the strategy of the Pacman based on the video

Based on the video, which component do you think the Pacman cares about more, eating normal pills or eating power pills:

- eating normal pills
- eating power pills
- Cares about both the same

Based on the video, which component do you think the Pacman cares about more, eating power pills or eating blue ghosts:

- eating blue ghosts
- Cares about both the same
- eating power pill



Based on the video, which component do you think the Pacman cares about more, eating normal pills or eating blue ghosts:

- Cares about both the same
- eating blue ghosts
- eating normal pills

What helped you determine your answer to the previous question?

How confident are you in your answer?

- very confident
- confident
- neutral
- not confident
- not confident at all

Please answer the following questions regarding all the videos you've seen in the study:

	Strongly disagree			Neither agree nor disagree			Strongly agree
	1	2	3	4	5	6	7
The videos helped me to recognize agent strategies.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos contain sufficient detail for recognizing agent strategies.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos contain irrelevant details.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
This is an attention check question, please select option two.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The videos were useful for the tasks I had to do in this survey.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The specific scenarios shown in the links were useful for the tasks I had to do in this survey	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

# D. Appendix to the Third User Study

This chapter provides additional details for the implementation of the approach presented in Chapter 7 and its computational evaluation, as well as the study described in Chapter 12. It is based on the appendix in our paper:

**Tobias Huber**, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André [2023]. “GANterfactual-RL: Understanding Reinforcement Learning Agents’ Strategies through Visual Counterfactual Explanations”. In: *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. IFAAMAS, 10 pages

## D.1. Implementation Details

In this section, we provide implementation details regarding the training of our counterfactual generation methods. Our full implementation can be found online.<sup>1</sup>

### D.1.1. Training Data

For the size of our training data sets, we aimed for around 200000 states since the StarGAN architecture from Choi et al. [2018] that we use in our approach was fine-tuned for the CelebA dataset, which contains around 200000 images. To this end, we started by sampling 400000 states for each game and each RL agent. For the Pacman agents, after duplicate removal and under-sampling (see Section 3.2), this leaves us with 230450 states for the *blue-ghost agent*, 277045 states for the *power pill agent* and 40580 states for the *fear-ghosts agent*. For Space Invaders, the dataset size is only slightly reduced due to the removal of training samples that are duplicates of test samples. For the normal agent, this resulted in 382989 states, and for the flawed agent, it resulted in 376711 states.

As is custom for the Atari environment (Section 2.1.2), we use a random amount (in the range  $[0, 30]$ ) of initial *Do Nothing* actions for each episode to make the games less deterministic.

---

<sup>1</sup><https://github.com/hcmlab/GANterfactual-RL>

### D.1.2. Training GANterfactual-RL

For training the StarGAN within our GANterfactual-RL approach, we tried to stay as close to Choi et al. [2018] as possible. We built our implementation upon the published source code<sup>2</sup> of Choi et al. [2018] and used their original settings. The network architecture is the same as in Choi et al. [2018]. For the loss functions specified in the main paper, we use  $\lambda_{cls} = 1$ ,  $\lambda_{rec} = 10$ , and  $\lambda_{gp} = 10$ . For training, we use an ADAM optimizer with  $\alpha = 0.0001$ ,  $\beta_1 = 0.5$  and  $\beta_2 = 0.999$ . The model is trained for 200,000 batch iterations with a batch size of 16. The learning rate  $\alpha$  linearly decays after half of the batch iterations are finished. The discriminator is updated five times per generator update during training.

One thing we change compared to Choi et al. [2018] is that we do not flip images horizontally during training. This is an augmentation step that improves the generalization on datasets of face images. However, it is counterproductive for Atari frames since flipped frames would often leave the space of possible Atari states or change the action that the agent would select.

### D.1.3. Training the Counterfactual State Explanations Model

For training the counterfactual state explanation model proposed by Olson et al. [2021], we reuse their published source code<sup>3</sup> to ensure comparability and reproducibility. For this reason, we also use the same Training parameters and network architecture. The only change we had to make to the network architecture is that the size of the latent space of our DQN Pacman agents is 256 compared to the Space Invaders agents in Olson et al. [2021] that have a latent space size of 32.

---

<sup>2</sup><https://github.com/yunjey/stargan>

<sup>3</sup><https://github.com/mattolson93/counterfactual-state-explanations/>

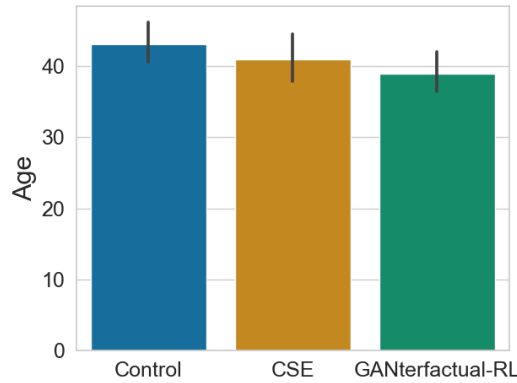


Figure D.1.: The participants’ age per condition.

## D.2. User Study Demographics

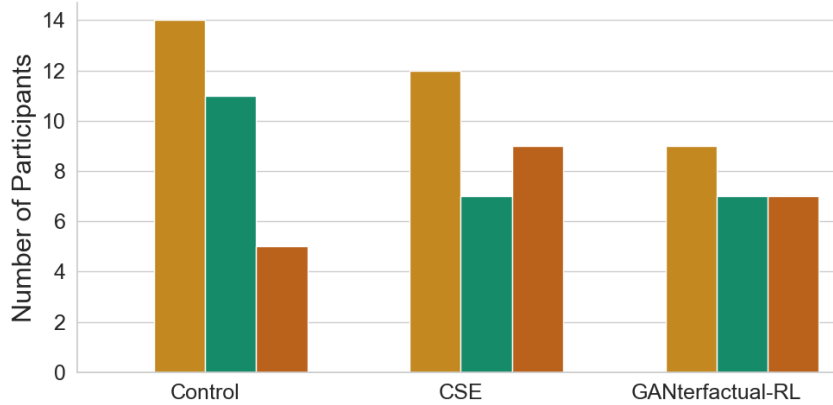
In this section, we provide more details regarding the demographic of the participants in our user study. As Figure D.1 shows, the mean age for each condition was around 40.

We verified that participants in different conditions did not differ much in their AI experience and views and their Pacman experience. To this end, we asked them when they played Pacman for the last time. The results are shown in Figure D.2.

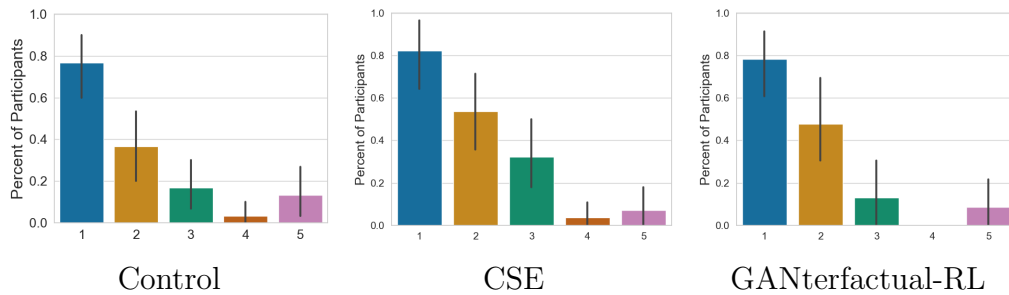
For the AI experience, we adapted a description of AI from Zhang and Dafoe [2019] and Russell and Norvig [2016] to “The following questions ask about Artificial Intelligence (AI). Colloquially, the term ‘artificial intelligence’ is often used to describe machines (or computers) that mimic ‘cognitive’ functions that humans associate with the human mind, such as ‘learning’ and ‘problem solving’.” After this description, participants had to select one or more of the following items:

- 1: I know AI from the media.
- 2: I use AI technology in my private life.
- 3: I use AI technology in my work.
- 4: I took at least one AI-related course.
- 5: I do research on AI-related topics.
- Other:

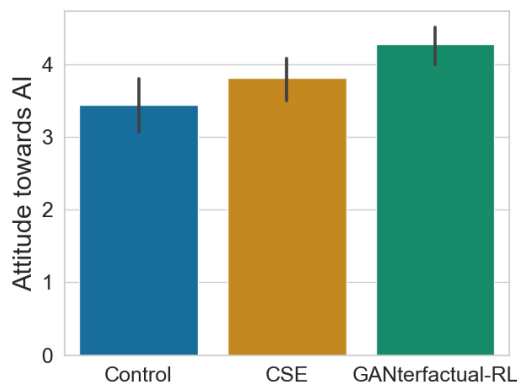
The distribution of the items for each condition is shown in Figure D.3. The option *Other* was never chosen.



**Figure D.2.:** The Pacman experience across all conditions where the bars depict when the participants played Pacman the last time. From left to right, the bars represent: “more than 5 years ago”, “less than 5 years ago” and “less than 1 year ago”.



**Figure D.3.:** Distribution of the chosen AI experience items for each condition. The x-axis depicts the items described above.

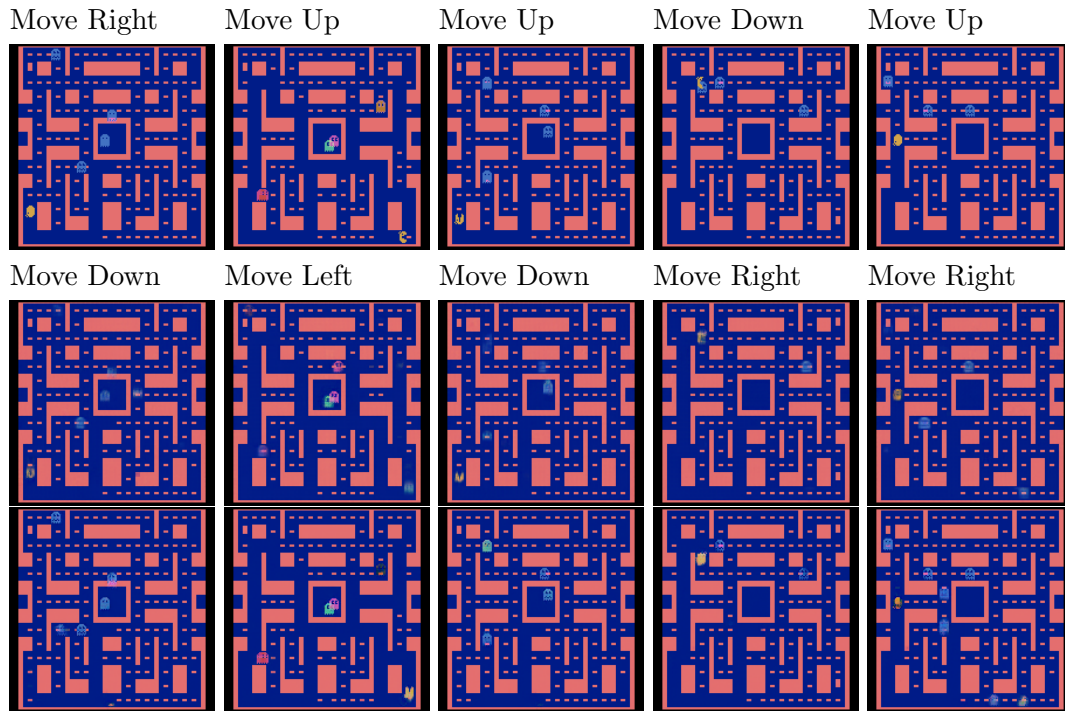


**Figure D.4.:** The average attitude towards AI, rated on a 5-point Likert scale.

To measure the participants' attitude towards AI, we adapted a question from Zhang and Dafoe [2019] and asked them to rate their answer to the question "Suppose that AI agents would achieve high-level performance in more areas one day. How positive or negative do you expect the overall impact of such AI agents to be on humanity in the long run?" on a 5-point Likert scale from "Extremely negative" to "Extremely positive". The results are shown in Figure D.4.

### D.3. Agent Performance

In this section, we want to report the average in-game score and survival time of our Pacman agents since we used this as ground truth for the agent comparison task. The *blue-ghost agent* got a mean score of 2035.6 and survived for 708.36 steps on average. The *power pill agent* got a mean score of 1488 and survived for 696.4 steps on average. The *fear-ghosts agent* got a mean score of 944.4 and survived for 6490.16 steps on average.

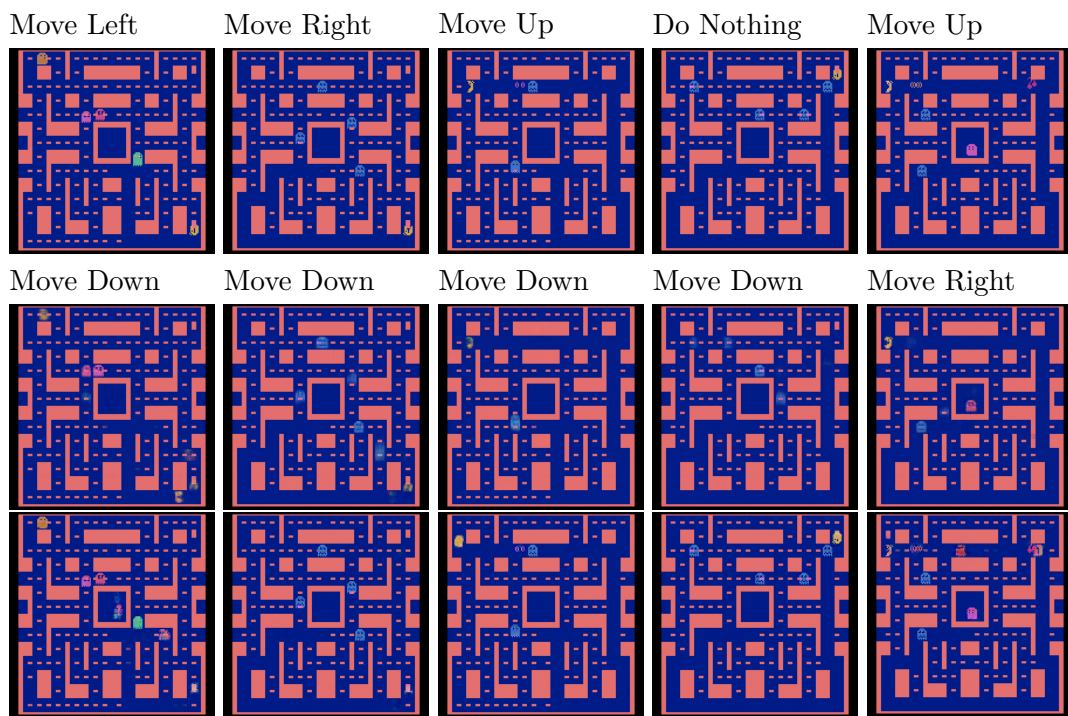


**Figure D.5.:** Example counterfactual states for the *blue-ghost agent*. The first row shows the original states, and the second and third rows show counterfactual states by Olson et al. [2021] and our GANterfactual-RL approach, respectively. The states and actions are the same states that were used during our user study and were chosen by the HIGHLIGHTS algorithm [Amir and Amir, 2018].

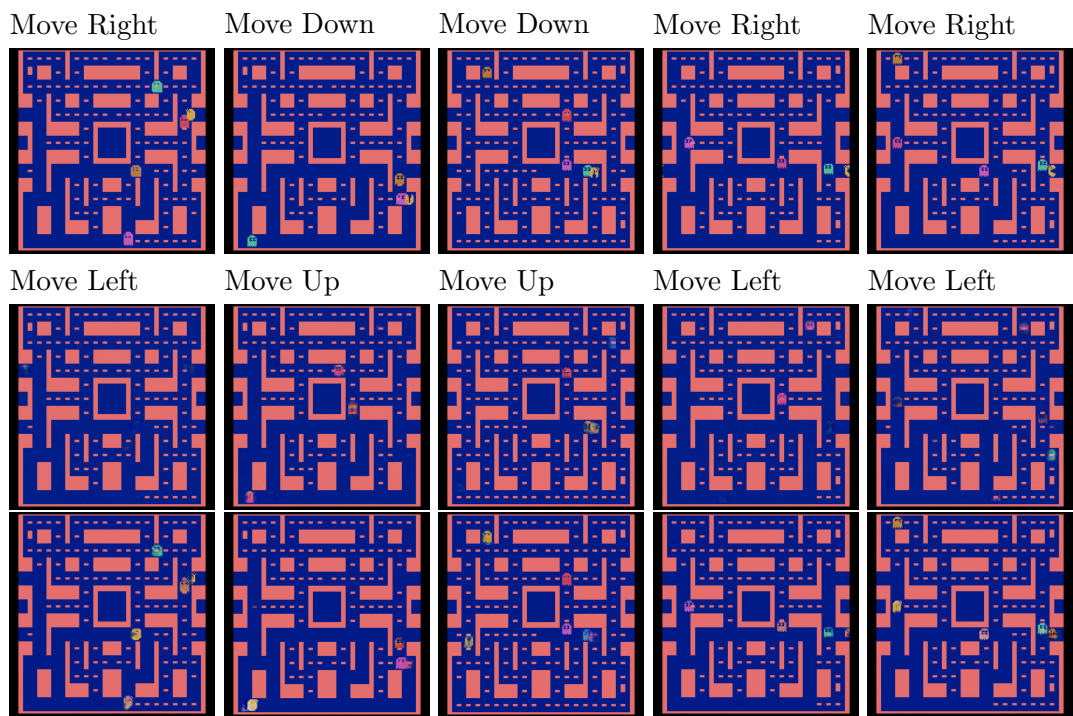
## D.4. Example Counterfactuals

Figures D.5, D.6, D.7, D.8, and D.9 show example counterfactuals for both approaches tested in the Chapters 7 and 12.

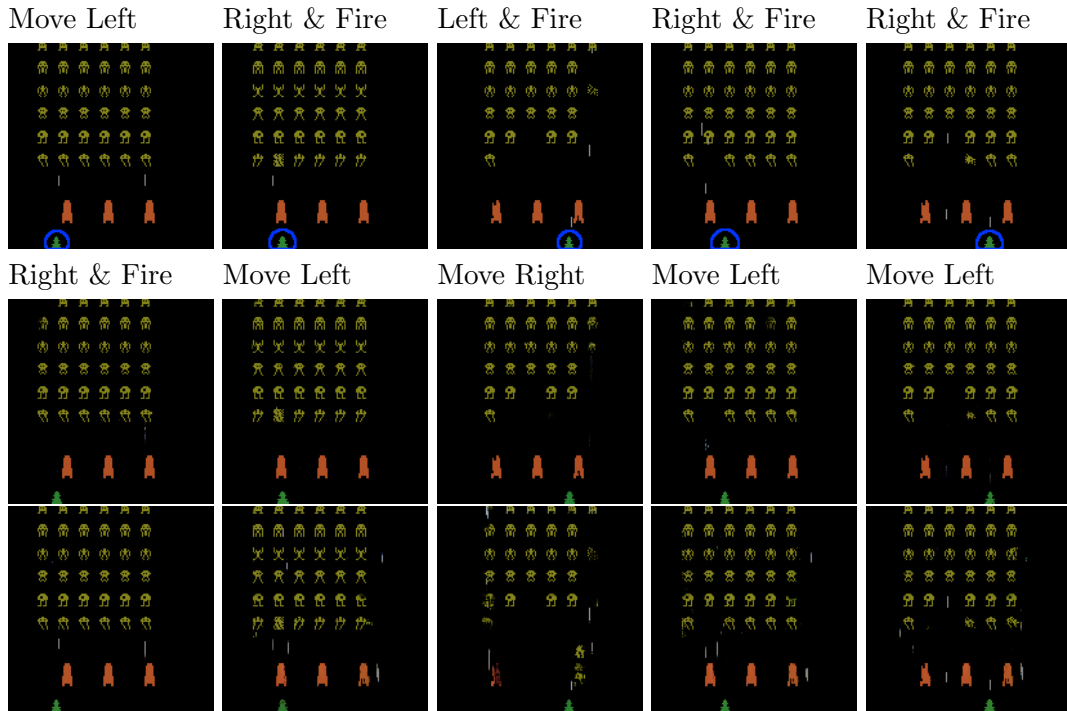




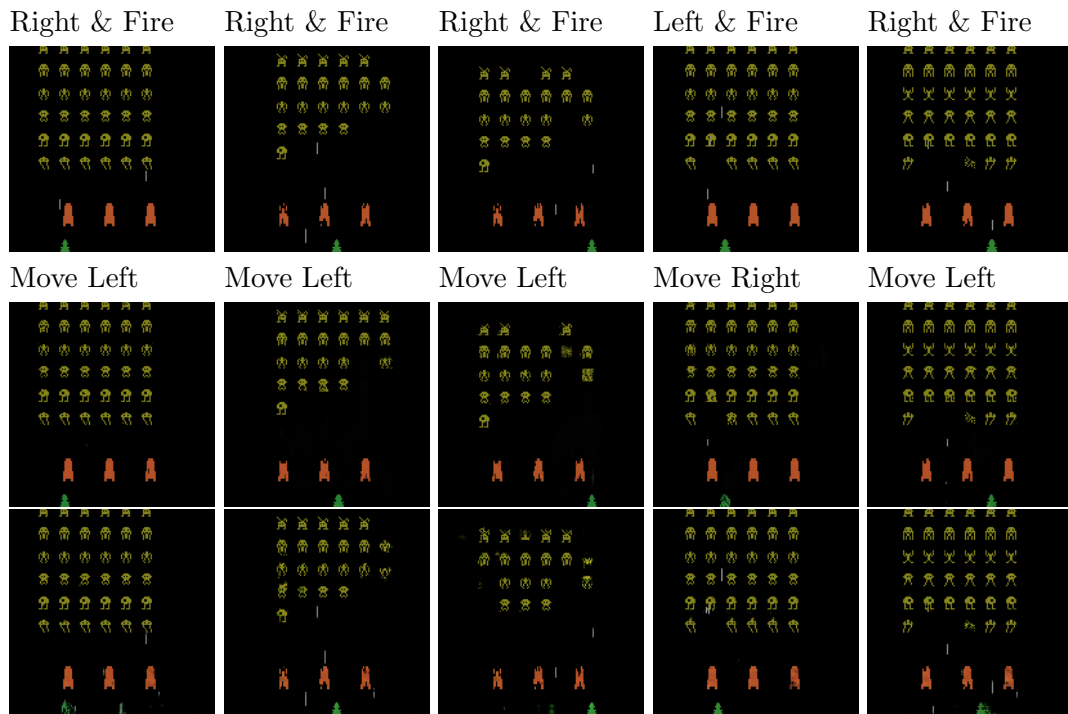
**Figure D.6.:** Example counterfactual states for the *power pill agent*. The first row shows the original states, and the second and third rows show counterfactual states by Olson et al. [2021] and our GANterfactual-RL approach, respectively. The states and actions are the same states that were used during our user study and were chosen by the HIGHLIGHTS algorithm [Amir and Amir, 2018].



**Figure D.7.:** Example counterfactual states for the *fear-ghosts agent*. The first row shows the original states, and the second and third rows show counterfactual states by Olson et al. [2021] and our GANterfactual-RL approach, respectively. The states and actions are the same states that were used during our user study and were chosen by the HIGHLIGHTS algorithm [Amir and Amir, 2018].



**Figure D.8.:** Example counterfactual states for the flawed Space Invader agent. The first row shows the original states, and the second and third rows show counterfactual states by Olson et al. [2021] and our GANterfactual approach, respectively. The states were chosen by the HIGHLIGHTS algorithm [Amir and Amir, 2018]. The counterfactual actions were chosen to be complete opposites of the original action. Despite this big difference in the action, neither approach moves the laser cannon (highlighted with a blue circle in the original frames) that the flawed agent does not see.



**Figure D.9.:** Example counterfactual states for the normal Space Invader agent. The first row shows the original states, and the second and third rows show counterfactual states by Olson et al. [2021] and our GANterfactual approach, respectively. The states are chosen by the HIGHLIGHTS algorithm [Amir and Amir, 2018]. The counterfactual actions are chosen to be complete opposites of the original action. In contrast to the counterfactuals for the flawed Space Invaders agent, both approaches sometimes modify the laser cannon.

## **D.5. Full User Study**

Figures D.10 to D.21 present screenshots of our user study. Exemplarily, we show the CSE condition.

## Personal information

**\*What is your age?**  
● Only numbers may be entered in this field.

**\*To which gender identity do you most identify?**  
● Choose one of the following answers

Male

Female

I prefer not to answer.

Other:

**\*Do you have a colour vision impairment?**

Yes  No

**\*Did you play Pacman before?**  
● Choose one of the following answers

No

The last time I played Pacman was less than 1 year ago.

The last time I played Pacman was less than 5 years ago.

The last time I played Pacman was more than 5 years ago.

**\*The following questions ask about Artificial Intelligence (AI). Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".**

Do you have experience with AI (Artificial Intelligence)? Check all that apply.

● Check all that apply

I do not have any experience in AI related topics.

I know AI from the media.

I use AI technology in my private life.

I use AI technology in my work.

I took at least one AI related course.

I do research on AI related topics.

Other:

**\*Suppose that AI agents would achieve high-level performance in more areas one day.**

	1: Extremely negative	2	3	4	5: Extremely positive	I don't know
How positive or negative do you expect the overall impact of such AI agents to be on humanity in the long run?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Next

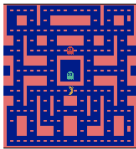
Figure D.10.: Demographic information.

## Instructions

In this survey, you will be shown images from Pacman games. These images will show you scenes of games played by various AI agents trained to play Pacman. Later you will be asked to complete 6 tasks based on those images. You will earn a 10 cent bonus for each task you solved correctly.

Please carefully read the following description of Pacman. There will be a short quiz to make sure you understood the information correctly.

A Pacman game takes place in a Labyrinth:



Pacman's goal is to eat as many "pills" (pink rectangles) as possible to earn points, while escaping four ghosts (blue, orange, pink, and red).

Pacman receives 10 points for each pill, and dies when it's eaten by a ghost.

In addition to the regular pills, there are also four special "power pills" (large pink rectangles). Pacman receives 50 points for eating a power pill.

Eating a power pill also gives Pacman limited time during which all ghosts become blue and Pacman can eat the ghosts. Pacman receives 200, 400, 800, 1600 points for each blue ghost it eats successively.

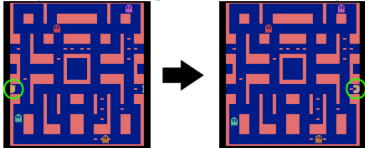
After a ghost is eaten, its eyes move back to the ghost box in the middle of the screen and the ghost restarts from there in his normal form.

At random intervals, cherries spawn and move through the labyrinth.

If Pacman eats the cherry it gets 100 points.

Pacman starts the game with three lives and loses one life each time it gets eaten by a ghost.

When Pacman moves through a tunnel on the side of the labyrinth, it reappears at the opposite side of the labyrinth:



Please use this link to play the game and get familiar with it (While the objects look different in that version the rules are the same as described above.):

[Pacman](#)

Next

Figure D.11.: The Pacman tutorial.

## Quiz Pacman

Please fill out this quiz to show that you understood the information.

\* Eating  gives Pacman:


● Choose one of the following answers

- No points
- 1 point
- 10 points

\* What is shown in this image: 

● Choose one of the following answers

- A ghost.
- A blue ghost.
- Pacman.

\* What is shown in this picture? 

● Choose one of the following answers

- a piece of wall.
- a power pill.
- a regular pill.

\* What happens when ghosts are blue?

● Choose one of the following answers

- Pacman can eat them
- Ghosts move faster
- Ghosts move slower

\* When do ghosts become blue?

● Choose one of the following answers

- Every 2 minutes
- When Pacman eats a power pill
- When Pacman gets close to a ghost

\* What happens when Pacman moves through a tunnel at the side of the labyrinth?

● Choose one of the following answers

- Pacman gets stuck.
- Pacman appears at the opposite side of the labyrinth.
- Pacman appears in the middle of the labyrinth.

Next

Figure D.12.: The Pacman quiz.



## Additional Information

In the following questions you will be shown images that summarize the behavior of an AI agent that was trained to play Pacman. These images show **several situations** from a longer session of gameplay and aim to show you, in a limited amount of time, **how the AI agent plays the game**. The situations are **not arranged chronologically**.

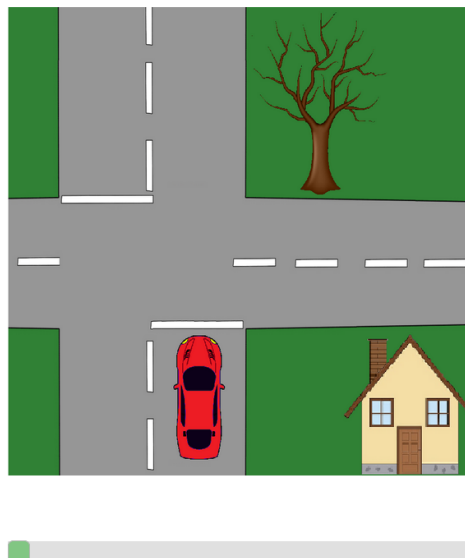
To aid you in your analysis of the AI, we will provide you with an additional explanation of the AI's decision. Below the images, you will always find a **slider**. By moving the slider to the **right**, the image will change to a so-called **counterfactual image**. This counterfactual image shows you how the game image could be modified, such that the AI makes a different decision than it actually did. The images answer the question: "What would the current screen need to look like for the AI to perform a different action?". To answer this question, the AI will only evaluate the current moment in the game, not the past or the future. Which action the AI originally took and which action it takes in the counterfactual image are displayed above the image.

For a more concrete example, consider the following. Imagine there is a red self-driving car that is taking you home. It approaches an intersection and it wants to turn right to take you to your destination. Now imagine a situation where the red car would choose to go straight instead of turning right. There are various reasons why this could happen. One example is if the brown tree fell over and blocked the road. In this example, an answer to a question of "what would need to change" right now for the car to choose go straight at this intersection, would be "the brown tree fell over which blocks the right turn". Now in the examples you will look at, the AI will answer the question of "what needs to change" by responding with a counterfactual image that can be seen by moving the slider to the right. This response shows the smallest amount of change in the game to take a different action. Back to the car example, the original image below shows the original image of the intersection. Moving the slider to the right shows a counterfactual image with the fallen brown tree.

Try it out!

Original Action: **Move Right**

Counterfactual Action: **Move Up**

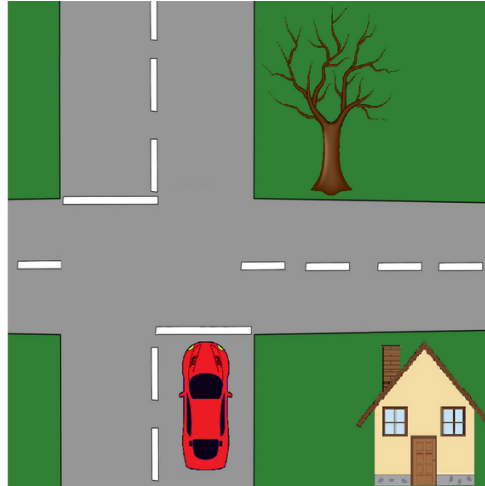


**Figure D.13.:** The first part of the counterfactual tutorial, which is built upon the tutorial by Olson et al. [2021].

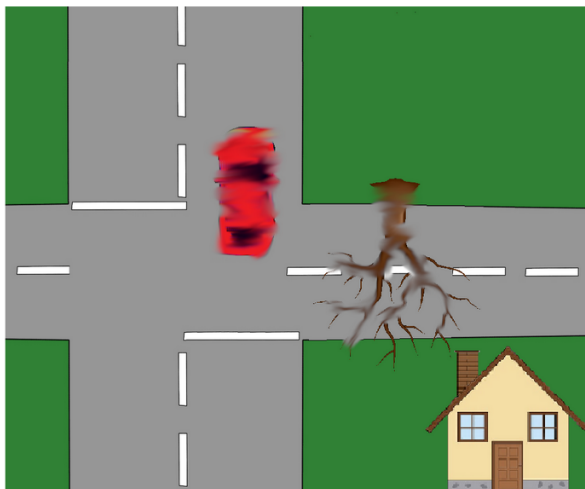
It is also possible for multiple objects to influence an AI's decision. In this example, two things influence the red self-driving car's decision to take the move straight action. The first is: if the brown tree has fallen over, but also if the red car's position changed such that it is past the intersection.

Original Action: **Move Right**

Counterfactual Action: **Move Up**



Finally, please note that **the counterfactual images in this study are generated automatically**. Therefore, **they may contain artifacts** that do not appear during normal pacman gameplay. **For example, some objects may be blurry**, as shown by the tree and the car in the following example:



Next

**Figure D.14.:** The second part of the counterfactual tutorial, which is built upon the tutorial by Olson et al. [2021].

## Second Quiz

Please fill out this quiz to show that you understood the information.

\*What do images on the **right** side of a slider show you?

Choose one of the following answers

- They show the original image that the AI agent saw.
- They show how the original image could be modified to change the decision of the AI.
- They always show an image where the AI would move up.
- They always show an image where the AI would move right.

\*What do images on the **left** side of a slider show you?

Choose one of the following answers

- They show the original image that the AI agent saw.
- They show how the original image could be modified to change the decision of the AI.
- They always show an image where the AI would move up.
- They always show an image where the AI would move right.

Next







**Figure D.15.:** The quiz about counterfactual explanations.

## Analyze the agent

On this page, you will be shown **example scenarios of an AI agent** trained to play Pacman (the scenarios are not ordered chronologically). Your task is to **analyze the strategy of that particular AI agent**. To this end, you will be asked to describe the AI agents' strategy and to select the objects that were the most important for the strategy of that particular AI agent. You will receive a bonus of up to 10 cents if you analyze the agent correctly. In total, you will see three different agents during this survey.

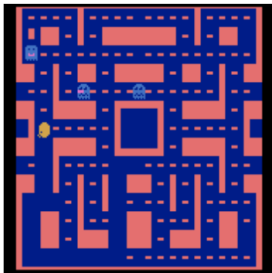
\*Based on the images below, select the objects that you think were most important for the strategy of this particular AI agent. Select a **maximum of 2** (it does not necessarily have to be 2) Objects.

• Check all that apply

- Pacman 
- Normal pills 
- Power pills 
- Ghosts 
- Blue Ghosts 
- Cherrys 

Original action: **Move Up**

Counterfactual action: **Move Right**



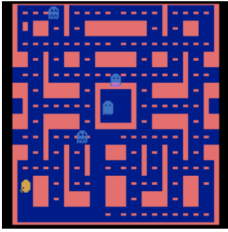
Original action: **Move Down**

Counterfactual action: **Move Right**

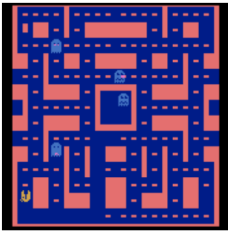


**Figure D.16.:** The first part of the agent understanding task. This task was repeated for all three agents. The order of the agents was randomized.

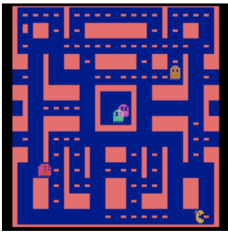
Original action: **Move Up**  
Counterfactual action: **Move Down**



Original action: **Move Up**  
Counterfactual action: **Move Down**



Original action: **Move Up**  
Counterfactual action: **Move Left**



\*Please briefly describe the strategy of the AI agent shown in the images above:

\*How confident are you in your selection?

	Not at all confident						very confident
How confident are you that you chose the right objects?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

\*Did you use the counterfactual images to come to your conclusion?

Yes  No

Next

**Figure D.17.:** The second part of the agent understanding task. This task was repeated for all three agents. The order of the agents was randomized.

### Explanation Satisfaction

\*

	1 : I disagree strongly.	2	3	4	5 : I agree strongly.
From watching the counterfactual images, I got an idea of the agents' strategies.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The counterfactual images contain sufficient detail about the agents' behavior.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The counterfactual images contain irrelevant details.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The counterfactual images were useful for analyzing the agents' behavior.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Do you have additional comments?

[Next](#)

**Figure D.18.:** Explanation Satisfaction for the agent understanding task. In the *Control* condition, *counterfactual images* was replaced by *images*.

### Compare the agents

Below are images showing parts of games played by **two different Pacman agents**: Agent A and Agent B.

You have already seen the individual agents in the previous tasks. Now you have to choose which of the AI agent you'd like to play on your behalf. You will receive a bonus of 5 cents for each time you select the Pacman agent that achieves higher points on average and another bonus of 5 cents for each time you select the Pacman agent that survives longer on average.

#### Agent A

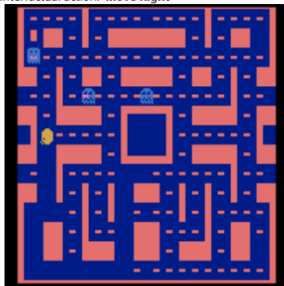
For this agent your previous description was:

██████

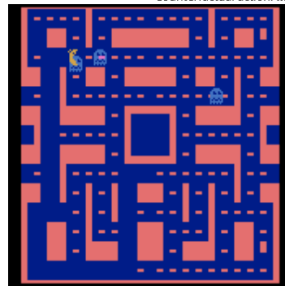
You can also explore the scenarios that you have seen before below. All of those scenarios show agent A:

Original action: **Move Up**

Counterfactual action: **Move Right**

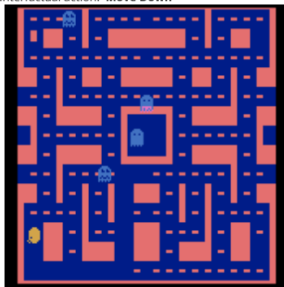


Original action: **Move Down**  
Counterfactual action: **Move Right**

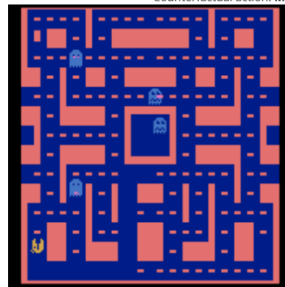


Original action: **Move Up**

Counterfactual action: **Move Down**

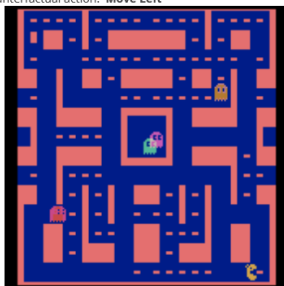


Original action: **Move Up**  
Counterfactual action: **Move Down**



Original action: **Move Up**

Counterfactual action: **Move Left**

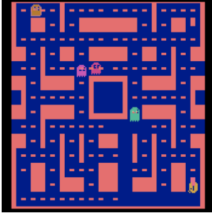


**Figure D.19.:** The first part of the agent comparison task. This task was repeated for all three agent pairs. The order of the pairs was randomized.

**Agent B**  
 For this agent your previous description was:  
 ■■■

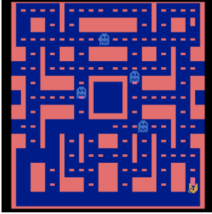
You can also explore the scenarios that you have seen before below. All of those scenarios show agent B:

Original action: **Move Up**  
 Counterfactual action: **Move Down**



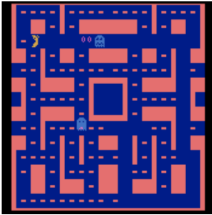
■■■

Original action: **Move Up**  
 Counterfactual action: **Move Down**



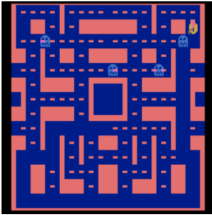
■■■

Original action: **Move Left**  
 Counterfactual action: **Move Down**



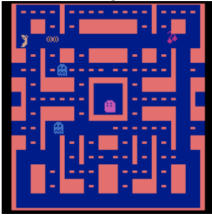
■■■

Original action: **Move Up**  
 Counterfactual action: **Move Down**



■■■

Original action: **Move Up**  
 Counterfactual action: **Move Right**



■■■

\*Which of the Pacman agents do you choose to play on your behalf to achieve more points?  
 ● Choose one of the following answers

Player A  
 Player B

\*Which of the Pacman agents do you choose to play on your behalf to survive longer?  
 ● Choose one of the following answers

Player A  
 Player B

\*How confident are you that you chose the better agent?

	Not at all confident						very confident
How confident are you that you chose the better player?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

\*Did you use the counterfactual images to come to your conclusion? (Please also choose "Yes" if you based your decision on insights that you gained through counterfactual images in the previous tasks).

Yes  No

Next

**Figure D.20.:** The second part of the agent comparison task. This task was repeated for all three agent pairs. The order of the pairs was randomized.

280



### Explanation Satisfaction

\*

	1 : I disagree strongly.	2	3	4	5 : I agree strongly.
From watching the counterfactual images, I got an idea of the agents' strategies.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The counterfactual images contain sufficient detail about the agents' behavior.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The counterfactual images contain irrelevant details.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The counterfactual images were useful for choosing the agent that performs better.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Do you have additional comments?

Next

**Figure D.21.:** Explanation Satisfaction for the agent comparison task. In the *Control* condition, *counterfactual images* was replaced by *images*.