UNIVERSITY
OF OSLO

Master's thesis

Explainable Reinforcement Learning

Discovering intent based explanations for heterogeneous cooperative multi agent reinforcement learning agents

Ada Hatland

Informatics: Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
The Faculty of Mathematics and Natural Sciences

Spring 2025

Ada Hatland

Explainable Reinforcement Learning

Discovering intent based explanations for heterogeneous cooperative multi agent reinforcement learning agents

Supervisors:
Dr. Dennis Groß
Prof. Kyrre Glette
Dr. Helge Spieker

Declaration of AI use

In this thesis generative models have been used for topic suggestion, a small part of figure creation, equations and parts of the code used for creating plots, as well as debugging and documentation. All code suggested has been verified by a human. Generative models have **not** been used for text generation. I, the author, take full responsibility for the contents of this thesis.

Acknowledgments

I would like to thank my external supervisor Dennis Groß, as well as my internal supervisor Kyrre Glette for their guidance and support in writing this thesis. They have been of significant help when discussing how to proceed and what remains on the to-do list. I would also like to thank my cat Nokia for emotional support throughout the writing of this thesis.

List of Acronyms

RL	Reinforcement Learning
XRL	Explainable Reinforcement Learning
AI	Artificial Intelligence
XAI	Explainable Artificial Intelligence
MARL	Multi Agent Reinforcement Learning
MDP	Markov Decision Process
DQN	Deep Q-Network
PPO	Proximal Policy Optimization
DRL	Deep Reinforcement Learning
RNN	Recurrent Neural Network
MBRL	Model-Based Reinforcement Learning
LLM	Large Language Model
EA	Evolutionary Algorithm
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
NN	Neural Network
AEC	Agent Environment Cycle
CNN	Convolutional Neural Network
DRC	Deep Repeated ConvLSTM
POMDP	Partially Observable Markov Decision Process

Abstract

This is the abstract

Contents

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Problem statement 2
- 1.3 Scope and limitations 2
- 1.4 Research methods 3
- 1.5 Ethical considerations 4
- 1.6 Main contributions 4
- 1.7 Thesis outline 5

2 Background and Related Works 7
- 2.1 Reinforcement Learning Fundamentals 7
  - 2.1.1 Markov Decision Process in Reinforcement Learning 7
  - 2.1.2 Markoc Decision Process in Multi Agent Reinforcement Learning 7
  - 2.1.3 Deep Reinforcement Learning 8
  - 2.1.4 Model-Free vs. Model-Based Reinforcement Learning 8
  - 2.1.5 Recurrency in Deep Reinforcement Learning Policies 9
- 2.2 Approaches to Explainability in Reinforcement Learning . . 10
  - 2.2.1 Post Hoc vs. Intrinsically Explainable Models . . . 10
  - 2.2.2 Fidelity of Explainable Artificial Intelligence Methods 11
  - 2.2.3 Future-Based Explanations. 11
- 2.3 Relevant Methods 12
  - 2.3.1 What did you think would happen? Explaining Agent Behaviour through Intended Outcomes 13
  - 2.3.2 Explainable Reinforcement Learning via a Causal world model 13
  - 2.3.3 CrystalBox: Future-Based Explanations for Input-Driven Deep Reinforcement Learning Systems . . 14
  - 2.3.4 Are Large Language Models Post Hoc Explainers . 14
  - 2.3.5 ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing Reinforcement Learning Policies 15
- 2.4 Related Works. 15
  - 2.4.1 Predicting Future Actions Of Reinforcement Learning Agents 15
  - 2.4.2 An Investigation of Model Free Planning 16
  - 2.4.3 Stabilizing Transformers for Reinforcement Learning 16

3 Methodology 17
3.1 Action importance 17
3.1.1 Counterfactual sequences 17
3.2 Feature importance 19
3.2.1 Shapley values for observations 19
3.2.2 Integrated Gradients 19
3.3 Extracting intent 21
4 Experiments 23
4.1 General experimental setup 23
4.2 Action inclusion 24
4.2.1 Setup 24
4.2.2 Results 24
4.2.3 Analysis 24
4.3 Experiment 2 24
4.3.1 Setup 24
4.3.2 Results 24
4.3.3 Analysis 24
4.4 Experiment 3 24
4.4.1 Setup 24
4.4.2 Results 24
4.4.3 Analysis 24
5 Discussion 25
5.1 Conclusion 25
5.2 Limitations 25
5.3 Future work 25

Chapter 1

Introduction

Sequential decision-making problems are problems where one decision leads to a new state that requires a new decision to be made. An example of this is autonomously driving from point A to point B through city streets. For these types of problems *Reinforcement Learning (RL)* systems are used. RL, both multi- and single-agent models have seen a significant rise in successful use and applicability in recent years, with models like AlphaGo[1] and smacv2[2]. These models learn by agents performing actions in a given state decided by a policy that lead to a new state, and learning by receiving rewards depending on if the new state is preferable to the previous, the aim is to learn a near optimal policy for achieving a fixed goal[3]. The agent uses an observation of the state to choose an action. The environment describes how an action affects the state. See Figure 1.2. In a *Multi Agent Reinforcement Learning (MARL)* system, we would have multiple agents.

1.1 Motivation

A significant problem with many deep machine learning models, including MARL policies, is known as the black box problem[4]. This problem describes how the processing of information is hidden from a human user due to the opacity of the model, and we just have to trust that the model uses relevant data to come to the conclusion it does, which it often doesn't do. For many tasks and models where the impact of the output isn't highly significant this isn't a big issue, but for tasks like autonomous driving we simply cannot use these models without trust for the models processing of data and that the solution given wont hurt anyone. We can considering autonomous driving as a MARL system if we consider each vehicle as an agent. To solve this we would like to have models that along with an output can provide us with some kind of reasoning for what information is used and how.

All this essentially means, until the black box problem is solved, autopilot will always require a driver behind the wheel[5]. Despite having high accuracy, precision and recall, a reinforcement learning model might choose a less than preferable action in edge cases or states not well covered by training and test sets, for example driving on snowy roads when all images in the training data is from warmer climates, or combinations that aren't well covered, like a foggy, snowy road with a sharp right turn. With a way to ask the agent for intent, we could find that it has learned to always expect there to not be a car around the corner, which is not always obvious from looking at the dataset, but if it in a decision relies on the road to be empty to choose a safe route, this is obviously a huge issue. This paper aims to explain how an agent in a MARL setting decides on an action due to what it expects from other agents in the future. The field working on combating this issue is known is Explainable reinforcement learning.

1.2 Problem statement

The focus of this paper will be to rework and apply methods for general *Explainable Artificial Intelligence (XAI)* to Reinforcement Learning, and expand on *Explainable Reinforcement Learning (XRL)* methods already developed. In particular we will focus on an agent and how the expected future states of other agents will affect it in its decision making process. The environment we will focus on is a MARL environment known as Knights Archers Zombies [6] made for Python. We use this because it is a cooperative environment where each agent has to consider other agents, both of the same type as itself, and other types. If we used an environment where all the agents are identical, our findings will be less useful for other environments where the agents aren't identical. In many real life scenarios agents will not be identical. If we once again consider autonomous vehicles, most vehicles are different in some way. A very obvious example of this is considering cars and motorcycles. Where because of their size difference, their paths chosen will often be different.

In psychology it is well known that most human based decisions are made with intent[7], and by focusing on the expectation of what future states will look like we can consider this the intent of the agent. If we are accurately able to extract the intent of an RL policy we are better able to explain the choice made and we are more comfortable with trusting that the choice made is a good decision. Concretely we hypothesise that by analyzing expected future states and events we could find the intent of the agent, which we could use to ensure the agent acts in a safe or desirable manner. The main question we aim to answer is what information, found in weights or design of the policy, or information extracted by knowing the weights and design, is the most important for separate prediction models to be able to predict future states and actions, in cases where we do not have access to a decent world model, or such a model does not exist.

1.3 Scope and limitations

We aim to construct a framework to describe the intent of the agents in the KAZ environment, and we aim to make it applicable to other environments as well. However, due to time limitations we restrict the framework to only accept PettingZoo example environments or environments created with the PettingZoo environment creation tool. MARL environments can be divided into two major categories, *Agent Environment Cycle (AEC)* and parallel. AEC environments are environments where the state transition happens after each agent chooses an action, and parallel environments are environments

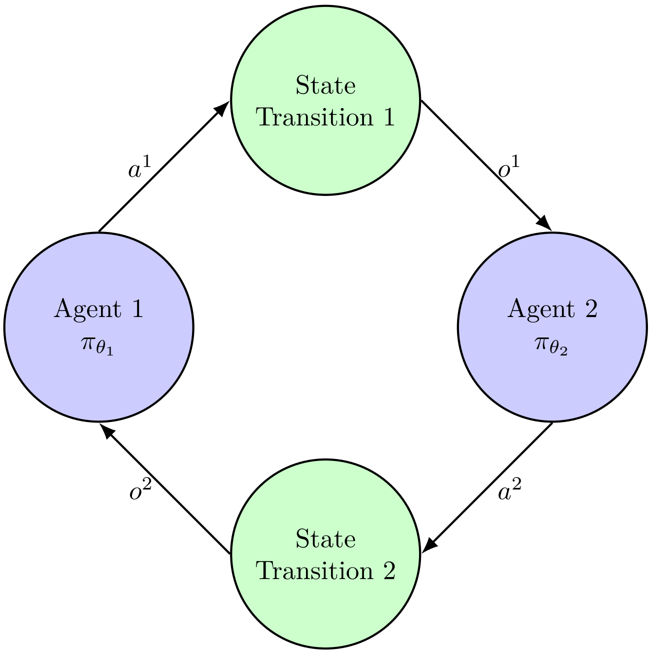

Figure 1.1: Example AEC in an environment with 2 agents.

where all the agents choose an action each before the transition happens. See figure 1.1.

Due to limited time and resources the size of policy or value networks, as well as any other *Neural Networks (NNs)* or otherwise computationally expensive algorithm used, will have to be significantly limited, this will impact results in a meaningful way, and therefore we will keep all NNs comparable sizes, and focus on doing experiments on how other factors than network size impact the results.

We do not have access to any real world datasets, so its hard to ensure the methods we have developed will be directly applicable, however the results we get will still be relevant for later research in the area.

1.4 Research methods

This section describes the methods we use to answer the questions we have posed. The methodology mainly focuses on produced datasets made with a digital RL environment, which does mean we have access to a world model, however as we want to answer the research questions without the use of a world model we will only be using the results of simulations to create our methods. The effect of this is that the methods will be compatible with RL systems where we do not have access to a world model.

Producing our own dataset has the benefit of no restrictions, other than hardware restrictions, on the datasets size. We can also produce variations of these datasets by altering the simulations used to create them which would in a real world setting often be impossible, e.g. access to gradients during inference.

The datasets produced will be sets of full or partial trajectories in given environments along with other relevant data like integrated gradient, or Shapley values for the observations in the trajectories. We use these datasets to train predictive NNs with two main types. Event prediction, for example whether an agent encounters a critical state within the next 10 timesteps, which could represent a dangerous situation for an autonomous vehicle, and state prediction, which for example could be the x and y coordinates of a given agent 10 timesteps into the future.

We will first be focusing on including explanations for the observations and observe and analyze the effects on event and state predictors, then we will be using NN policies with access to memory and include their representation of memory along with the observation, hidden values for lstm, and attention matrix for GTrXL, and once again observe and analyze effect on the predictors.

1.5 Ethical considerations

MARL is an important field in development for military purposes[8], and while this research is not supposed to effectivize warfare its possible that it ends up being relevant for unintended areas of research, including but not limited to military use-cases. This is especially the case because I aim to develop methods for general MARL use and not a specific area of research, like medicine or sports. It is more or less impossible to restrict access to these methods for specific areas of research.

If this research aids to make MARL methods applicable to real world settings there is the possibility it risks increasing job displacement issues. This is an issue that affects most *Artificial Intelligence (AI)* research and can be mitigated by programs to reeducate people whose jobs it might affect, and increasing employment in developing and monitoring AI tools.

1.6 Main contributions

As discussed in 1.1, 1.2 and 1.4, there are certain specific objectives we aim to reach. In general we aim to expand on current future prediction research.

- **Objective 1: Observation space**
  Most current methods assumes the observation space to be image based, and bases research on information found in convolutional layers, either regular *Convolutional Neural Networks (CNNs)*, or convLSTM layers.
- **Objective 2: Feature importance as input**
  Using feature importance from the policy to improve future predictions is a novel idea, and has potential to improve future predictions by a significant margin.
- **Objective 3: Memory as input**
  Earlier research has found that including hidden states from the *Deep Repeated ConvLSTM (DRC)* architecture had a significant effect on accuracy when predicting future states and events. We aim to expand on this and use hidden states from ***lstm!s (lstm!s)*** layers and observe the effect and analyze how it differs from using the ConvLSTM hidden values, as well as including attention from the GTrXL architecture.

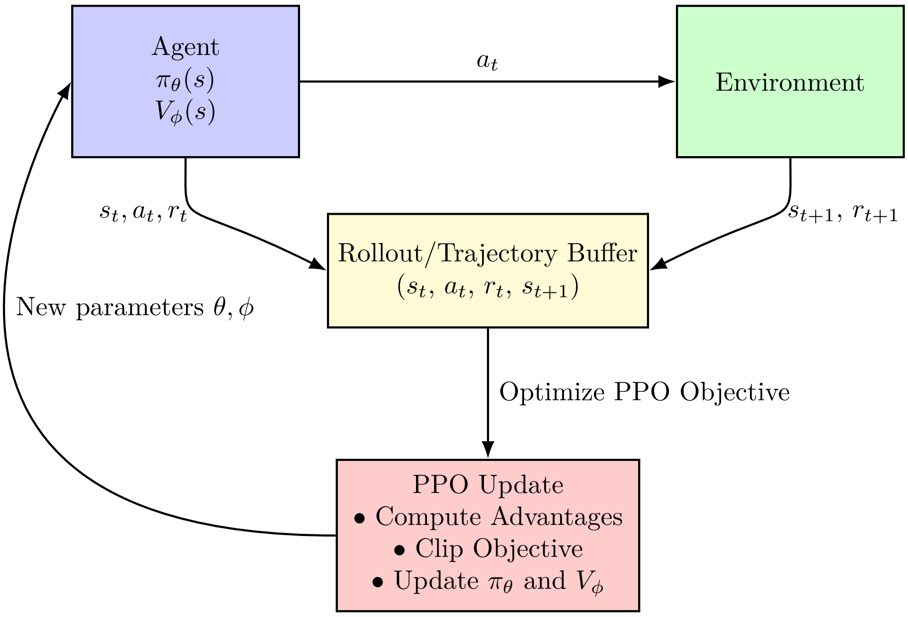

Figure 1.2: Diagram showing the *Proximal Policy Optimization (PPO)* update algorithm

1.7 Thesis outline

This thesis will be structured as follows:

- **Chapter 2: Background and related works**
  This chapter will introduce important concepts, both RL and explainability related, and briefly discuss the history of XAI methods, and how they are applicable to my use case. We will also consider XRL methods that are already developed and how to expand or integrate them into developed methods of our own.
- **Chapter 3: Methodology**
  This chapter discusses the specific principles, procedures and techniques used to conduct the XRL research done in this thesis, to ensure that the results we get are valid and reliable.
- **Chapter 4: Experiments**
  This chapter details the setup, results and analysis of the specific experiments we use to evaluate the hypothesis and answer research questions based on certain evaluation metrics, and baseline comparisons.
- **Chapter 5: Discussion**
  This chapter will analyze the results of the experiments and discuss their implication for XRL research. It will also contain the limitations of our work and briefly discuss how to expand on any research done.

Chapter 2

Background and Related Works

2.1 Reinforcement Learning Fundamentals

To make sure all methods are understood well its important to make sure we have a proper understanding of the fundamentals of RL, especially *Deep Reinforcement Learning (DRL)* and MARL. This section will go over the basics needed.

2.1.1 Markov Decision Process in Reinforcement Learning

A *Markov Decision Process (MDP)* is a framework used to model decision-making in stochastic environments, like we want to in an RL task. It is defined by a tuple with 5 elements:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$$

where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of possible actions, $P(s'|s,a)$ is the transition probability function, which defines the probability of moving to state s' when action a is taken in state s, $R(s,a,s')$ is the reward function that provides a reward for transitioning from state s to s' via action a, $\gamma \in [0,1]$ is the discount factor that determines the importance of future rewards.

The objective in an MDP is to find a policy $\pi(s)$ that maximizes the expected cumulative reward:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi\right]$$

where $V^{\pi}(s)$ is the value function that represents the expected return starting from state s and following policy π. See figure 1.2.

2.1.2 Markoc Decision Process in Multi Agent Reinforcement Learning

In a MARL setting agents could have shared or independent MDPs depending on architecture. There are three main branches of MARL, cooperative, competitive and mixed. Our focus will be on cooperative MARL. In a cooperative MARL setting the goal is some social welfare function that maximises rewards for the agents, either collective rewards or a mix of collective and individual rewards. In such a setting each agent has MDP. Assuming that state space $\mathcal{S}$ and discount factor γ is shared, which they will be in all my experiments, if two agents have the same reward structure R and action space $\mathcal{A}$ their objective would be the same, in which case they could share policy π and could therefore share $\mathcal{M}$.

2.1.3 Deep Reinforcement Learning

Traditional RL methods struggle with scalability due to the fact that they rely on discrete state representations. DRL uses deep NNs to approximate the policy function $\pi_\theta(s)$, the value function $V_\phi(s)$, or the Q-function $Q_\theta(s,a)$, making it possible to represent these functions in continuous and complex environments. See figure 2.1. Input layer is size of state for all three functions. Output layer is size of action space for $Q_\theta(s,a)$ and $\pi_\theta(s)$. $V_\phi(s)$ only has one output node. In a feed forward NN like we have here each node has the value of some activation function $\phi(x)$ where x is the sum of nodes of the previous layer multiplied by their respective weights, usually as well as adding some bias, shown as the connections between the nodes. Common activation functions are $ReLU(x) = max(0,v)$ and $tanh = \frac{e^v - e^v}{e^v + e^v}$ where v is the value before activation.

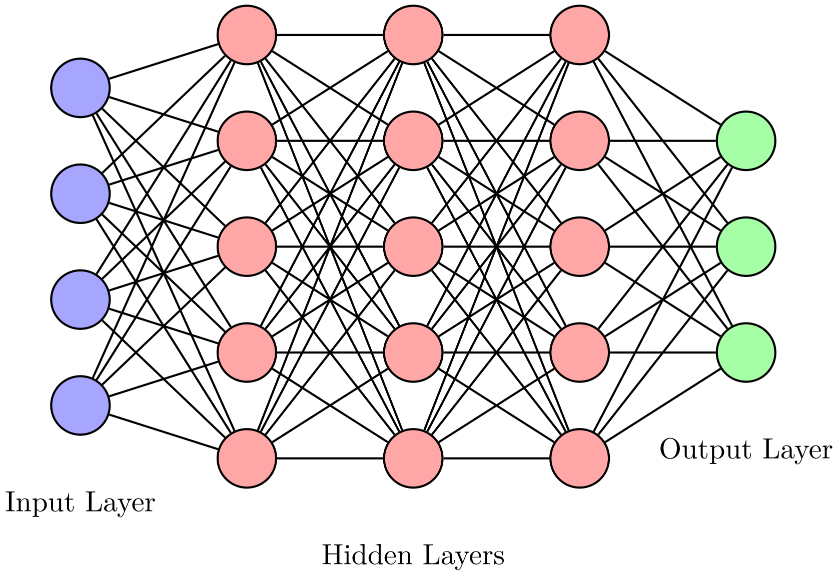

Figure 2.1: Illustration of an example of a NN with three hidden layers

2.1.4 Model-Free vs. Model-Based Reinforcement Learning

Model-based and model-free reinforcement learning differ in how they represent environment transitions. In model-based reinforcement learning, the agent learns or has access to a model of the environment, which includes a transition function $P(s'|s,a)$ and a reward function $R(s,a)$ that determines the expected reward for taking action a in state s [9]. This allows the agent to simulate future outcomes without taking actions in the real environment. As a result, model-based methods are able to update policies based on imagined rollouts, which increases learning speed compared to executing

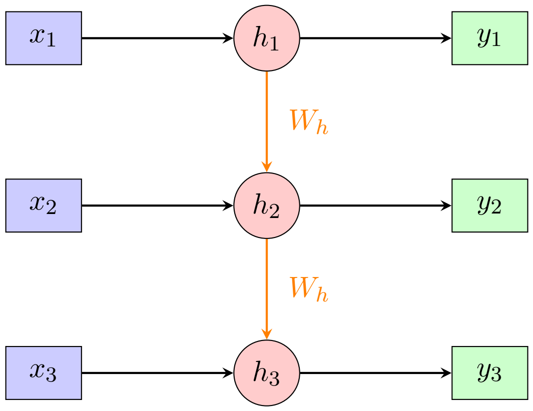

Figure 2.2: Recurrent neural network

the real environment transitions. Examples of model-based algorithms are Dyna[10] and DreamerV3[11].

In contrast, model-free reinforcement learning does not learn or utilize an explicit model of the environment. Instead, the agent learns by interacting directly with the environment and updating its network weights based on observed rewards. This approach is generally less sample-efficient but is applicable to environments where modeling the transition dynamics isn't feasible. Examples of model-free algorithms include Q-learning, *Deep Q-Network (DQN)* [12], and PPO [13].

Model-based methods enable explicit planning by using the learned environment transitions in methods such as Monte Carlo Tree Search [14], while model-free methods rely on direct experience to optimize behavior. We will be using PPO to optimize our algorithms, and will not be using simulations or any other form of explicit planning, ensuring that the methods developed will be applicable in the most amount of environments.

2.1.5 Recurrency in Deep Reinforcement Learning Policies

Recurrency in NN is a way to carry over information from previous timesteps to current ones, if we believe that information could be useful. *Recurrent Neural Networks (RNNs)* are sometimes used when constructing the architecture of DRL policies [15], mainly for two reasons. The first one is for when your environment can be modeled as a *Partially Observable Markov Decision Process (POMDP)*, in which case memory is important as the current observation is not always enough to capture all the information an agent could have. The second reason is for planning, like in the DRC architecture[16]. See figure 2.2.

2.2 Approaches to Explainability in Reinforcement Learning

2.2.1 Post Hoc vs. Intrinsically Explainable Models

This section will discuss benefits and drawbacks of explainable models divided into two main categories, Post hoc models and intrinsically explainable models.

Post hoc explainability refers to finding ways to understand already trained models. Instances where this doesn't pose more of a challenge can often be more efficient as you do not need to construct and train a model, which could often pose an issue both considering time spent and accuracy of the model. Making a model explainable only makes sense if we know it usually makes sensible decisions. There are however issues with making post hoc models.

Temporal Aspect

RL policies all have a temporal aspect, which means several different actions over a certain period of time might contribute to a singular outcome, this could make it very challenging to pinpoint which actions were made for which outcome, especially if the outcome happens several states after the initial decision was made.

Nature of Black Box Models

Many RL policies, especially deep RL policies, which are the policies we will be working with, have complex inner connections due to the hierarchical structure of learning features in the input, sequences of non linear connections that are very hard to understand without spending a lot of time studying the specific connections learned by a model. It's often very difficult to accurately pinpoint the purpose of a node, especially because we do not know if it even has a purpose at all or it could have a lot of different purposes. Especially in environments with complex observation spaces.

Lack of Transparency

Tracing how an observation leads to a state, through the action chosen and the environment, is often challenging in models constructed without this in mind. If we do not know the processing of information of a model it's very hard to explain the rationalisation of an agent.

Intrinsically Explainable Models

Intrinsically explainable models have the already stated drawback of relying on careful model construction, which can be challenging and may impact model performance. While these models offer transparency from the get go, they are often less flexible and might not achieve the same level of accuracy as more complex RL models. Given that we aim to analyze and compare already trained RL policies, we will mostly focus on post hoc explainability methods in this thesis.

2.2.2 Fidelity of Explainable Artificial Intelligence Methods

It is important to understand why the current XAI methods not meant for RL applications aren't always useful to us.

XAI methods not intended for reinforcement learning often provide explanations that don't necessarily represent the inner workings of an RL policy, because an RL policy has a temporal aspect as well. Broadly these methods can be categorised as feature-based explainers, and they often struggle to fully explain an agents behaviour or specific actions because they cannot capture the future view of an RL policy.

Saliency maps which have been successfully used for classification of images provide explanations about what part of the input was important for the outcome, which is highly relevant for classification tasks, but using the same method for an RL policy doesn't sufficiently explain the intent of the agent[17].

Another commonly used XAI method is model distillation, which works by transferring knowledge from a large model to a smaller, usually interpretable, one, for instance a deep learning network to a decision tree [18]. This has use cases in verifyability, but struggles to fully explain the temporal aspect of RL policies, and are therefore not sufficient as an explainer.

However, these methods might still prove insightful in conjunction with other intent-based methods, the state in which a decision is made is obviously very relevant to why that particular decision was made. We could perhaps use these methods to answer questions such as "What part of agent As Observation this state, lead it to believe agent B would end up at these coordinates at a later state?"

2.2.3 Future-Based Explanations

Next, I will describe in slightly more detail, what is meant by an intent based explainer, like I want to develop, and how to use it.

Design

"Towards Future-Based Explanations for Deep RL Network Controllers"[19] broadly describes future-based intrinsically explainable methods. Future-based intrinsically explainable methods for DRL policies often take three inputs, the trajectories experienced by an agent, the environment and the agent. Then, they collect the rewards and interactions and use this information to train an explainer.

During inference, we can then apply the explainer to a state and action to get the expected future consequences of that action. Depending on the architecture we could either get the expected future consequence of any action, or just the one the agent decides on.

Use Cases

Designing a DRL solution requires choosing features in the observation, hyperparameter tuning, policy design and reward function among other things. This is a time, and resource, consuming process. These are usually picked by trial-and-error but could be made easier with assistance from an explainer.

Another, and perhaps more important, use case is safety. If when online, i.e. the chosen action will actually affect the state, we are expecting high likelihood of an unsafe state, we could instead of opting for the chosen action fall back to a known safe action, that could have a lower expected return, i.e. breaking instead of turning a corner if we have limited vision around the corner.

Definition of Intent

We define the state transition function $T(s, \mathbf{a}, s')$ where $\mathbf{a}$ is the set of simultaneous actions made by the set of active agents, in our case, one action per agent. Given T the agent should when prompted output a set of trajectories $\tau = \{(s_0, \mathbf{a}_0), (s, \mathbf{a}), (s_1, \mathbf{a}_1)...(s_n, \mathbf{a}_n)\}$ where s_0 is the current state, $\mathbf{a}_i$ is the set of actions taken in s_i, and s_{i+1} is the state reached by $T(s_i, \mathbf{a}_i, s_{i+1})$. We will apply a series of methods on τ to extract intent. One of these methods is discovering counterfactual trajectories, $'\tau$, where the actions made by the agents in $'\tau$ are as similar as possible to the actions made by the agents in τ, but the total reward gained is as low as possible. $'s_0 = s_0$ and given an identical transition function T the goal is to discover which actions are most important to receive the reward. This method is similar to and inspired by ACTER[20], see section 2.3.5. Details in section 3.1.1.

2.3 Relevant Methods

There are several papers written on XAI and XRL problems. Milani et al.[21] categorise XRL into three main categories, feature importance (FI), learning process and MDP (LPM) and Policy-level (PL). FI considers the immediate context for actions, i.e. what part of the input was important for a single action, LPM considers techniques that highlight which experiences in training were influential for the final policy and PL focuses on long term effects and behaviour. Since we are interested in future states and actions we will look at influential trajectories and transitions within these trajectories. It is important to view these transitions in the context of the trajectory to understand the long term effects and not just immediate, which are of less interest in this paper, if we find the state with a high state importance $I(s)$, $I(s) = max_a Q(s, a) - min_a Q(s, a)$ most similar by some similarity measure to an arbitrary current state we could find the resulting trajectory and expect the agent to intend a similar outcome. There are also ways to convert RNN policies to an interpretable models post-hoc, which might be relevant if we use an RNN. This paper will explore PL explainability further.

In particular "What did you think would happen? Explaining Agent Behaviour through Intended Outcomes" [22], "Explainable Reinforcement Learning via a Causal world model" [23] and "CrystalBox: Future-based explanations for input-driven deep RL systems" [24] are highly relevant due to the fact that they all describe temporal connections between current actions and future states or actions.

"Are large language models post-hoc explainers" [25] Could be relevant as using other explainers to compare explanations is useful. "ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing RL Policies" [20]

2.3.1 What did you think would happen? Explaining Agent Behaviour through Intended Outcomes

What did you think would happen describes what an agent expects to happen in future states, and why the current action is chosen based on the future expectations. As stated in the paper, a limitation of their method means it doesn't work well with high dimensionality. The two main difference between this paper and the problem i aim to solve is that they're focusing on an environment with a single agent and that our observation space will be high dimensionality. It uses Q-learning and Markov-chains that train simultaneously with a "belief map" that shows what the agent expects the environment to look like in future states. In the simple examples used in the paper it shows where it believes the taxi should drive and therefore chooses an action to follow this path. This is not directly applicable to my thesis as it's unlikely that Q-learning or Markov chains will be viable for the policy and explainer based on the environment. However the paper is successful in explaining an agents underlying motivations and beliefs about the causal nature of the environment, and using similar methods might be an effective means for making MARL agents with higher dimensionality understandable from a human perspective.

2.3.2 Explainable Reinforcement Learning via a Causal world model

Explainable Reinforcement Learning via a Causal world model constructs a sparse model to connect causal relationships, without prior knowledge of the causal relationship, rather than a fully connected one, but they still achieve high accuracy and results comparable to other fully connected *Model-Based Reinforcement Learning (MBRL)* policies. Which is important, as there is often a trade off between explainability and performance. Using the same model for both explanations and choosing actions also make the explanations faithful to the intentions of the agent. The paper also describes a novel approach to derive causal chains from the causal influence of the actions, which lets us find the intent of the agent. The paper is successful being applied to MBRL, and is also applicable to models with infinite actions spaces, which is a limitation of some other models, see previous sub section.

A limitation of the paper is that it requires a known factorisation of the environment, denoted by $\langle S, A, O, R, P, T, \gamma \rangle$, where S is state space, A is action space, O is observation space, R is reward function, P is probability function for the probability of transitioning from state s to state s' given action a, T is the termination condition given the transition tuple (s, a, o, s'), and γ is the discount factor. Considering we will be working with hand crafted simulations we will have access to all of these, however its not certain that if we depend on this method that our contributions will be applicable to certain other environments where the factorisation is not known.

2.3.3 CrystalBox: Future-Based Explanations for Input-Driven Deep Reinforcement Learning Systems

CrystalBox introduces a model-agnostic, post-hoc, future based explanation for DRL. It doesn't require altering the controller, and works by decomposing the future rewards into its individual components. While this isn't exactly the kind of explanation we are looking for, it could be a great tool in developing an explainer that considers other agents actions in a multi agent cooperative environment, which is the goal of our paper, because it is post-hoc, and easily deployable. Especially because it was constructed for input-driven environments. The original paper claims it offers high fidelity solutions for both discrete and continuous action spaces. KAZ has a discrete action space but we might do some work with other environments as well.

It's not certain to be useful because it works by decomposing the reward function, and it's not safe to assume the reward function will even be useful to decompose.

2.3.4 Are Large Language Models Post Hoc Explainers

A *Large Language Model (LLM)* is a predictive model that generates text based on a prompt you give it, be it continuing the prompt or responding, and can often give the impression of comprehension of human language and a deeper understanding of the topic at hand. The paper aims to investigate the question "Can LLMs explain the behaviour of other complex predictive models?" by exploiting the in-context learning (ICL) capabilities of LLMs. ICL allows LLMs to perform well on new tasks by using a few task samples in the prompt. A benefit of using an LLM as a post-hoc explainer, is that the output given by the model will already be written in natural language and should be understandable by a layman. The paper presumes that the local behaviour of a model is a linear decision boundary, and by using a sample x and perturbing it to x', and presenting both the samples and the perturbation as natural language we could get an explanation from the LLM for the outcome. With a sufficient number of perturbations in a neighbourhood around x the LLM is expected to explain the behaviour in this neighbourhood, and rank the features in order of importance.

While the faithfulness of the LLM as an explainer is on par with other XAI methods used for classification, meaning that the reasons provided are enough to explain the output, I am sceptical of the fidelity of the LLM for two reasons. One is the same as for why other XAI models often struggle with fidelity, the temporal aspect. If applied in the same way as in the paper it would not consider the intent or the past and only what part of the current observation made it make a certain decision. The other is that I am sceptical of claims presented by an LLM in general as these are all just guesses. Good guesses a lot of the time, but still just guesses.

We could however potentially change the implementation so it considers the temporal aspect, and this might be a viable post hoc explainer after some more research into prompt engineering.

2.3.5 ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing Reinforcement Learning Policies

The paper presents ACTER, an algorithm that uses counterfactual sequences with actionable advice on how to avoid failure for an RL policy. It does this by using an *Evolutionary Algorithm (EA)* known as *Non-Dominated Sorting Genetic Algorithm II (NSGA-II)* to generate counterfactual sequences that don't lead to failure as close as possible to factual sequences that lead to failure. This paper presents counterfactual sequences and not just actions, which means it also presents how to avoid the state that lead to failure to begin with, which should, if ACTER is implemented correctly, allow us to significantly reduce the amount of times our policy fails. It also offers multiple counterfactuals to allow the end user to decide which counterfactual is preferable to their use case.

There are 4 hypothesises tested by the paper. The last two considers laymen users and are therefore not as interesting to us. The first two however "ACTER can produce counterfactual sequences that prevent failure with lower effort and higher certainty in stochastic environments compared to the baselines." and "ACTER can produce a set of counterfactual sequences that offer more diverse ways of preventing failure compared to the baselines." are partially and fully confirmed respectively. Which means ACTER will likely be a useful tool to explain and diagnose our RL policy.

2.4 Related Works

This section describes recent development in the field of planning and future predictions. All the papers described here are directly relevant to the methods i developed.

2.4.1 Predicting Future Actions Of Reinforcement Learning Agents

This paper describes a method developed by the author that predicts future actions and events [26]. It does this for non planning policies, like Impala, implicit planning policies like DRC, see section 2.4.2, and explicit planning policies like MuZero and Thinker.

Non planning policies are policies designed without planning in mind. Pure NN policies without recurrency are considered to be non planning, as they have not been found to exhibit planning like behaviour. This paper found that when predicting future states and actions both inner states from implicit planners, and explicit planners improved accuracy over non planning agents.

Explicit planners are agents who simulate future states with a world model before making a decision on what action to take. They rely on the world model being accurate, and so does using them to predict future states and actions. In some environments learning an accurate world model is not feasible, an example of this is autonomous driving.

The paper proposes two methods on predicting future states and actions, simulation based and inner state based. The inner state approach is most effective when working with explicit planners, but also shows significant improvement for implicit planner agents. The simulation based approach performs very well on implicit planner agents, but requires the opportunity to train a decently performing world model. The paper also shows with an ablated world model, the inner state approach performs better.

2.4.2 An Investigation of Model Free Planning

2.4.3 Stabilizing Transformers for Reinforcement Learning

Chapter 3

Methodology

3.1 Action importance

3.1.1 Counterfactual sequences

To discover counterfactual sequences we use the EA known as NSGA-II. This algorithm is a significant improvement over earlier multi-objective EAs that use non dominated sorting. It has a complexity of $O(MN^2)$ instead of $O(MN^3)$, elitist approach and a specified sharing parameter, all three of which, many earlier algorithms lacked. [**Deb2001AFA**] Pseudocode for sorting in algorithm 1. We use multi objective search because we want to minimize action change while maximizing total reward change throughout a simulation. This defines our two objectives. This way we can discover what actions or combinations there of have the highest impact on the environment. When working with a discrete action space, that is when we have a finite, usually relatively small, amount of actions to choose from, we define the action objective that we want to minimize as how many actions in a single timestep are different from a predefined sequence found with a trained model. After the timestep with the found actions we again use the model to roll out the rest of the sequence. We compare the altered sequence to the original sequence and measure the total reward both of these sequences receive. We want to maximize this difference. See figure 3.2. In the case of continuous actions we instead alter actions in multiple timesteps and use a distance measure that we want to minimize. The reward objective is similar to the case of discrete actions. See figure 3.1.

Figure 3.1: Pareto optimal points obtained with 2000 generations and population of 400 with NSGA-II

Figure 3.2: Pareto optimal points obtained with 2000 generations and population of 400 with NSGA-II

Algorithm 1 Fast Non-Dominated Sort

```
$F_i \leftarrow \emptyset$
for all $p \in P$ do
    $S_p \leftarrow \emptyset$
    $n_p \leftarrow 0$
    for all $q \in P$ do
        if $p \prec q$ then
            $S_p \leftarrow S_p \cup q$
        else if $q \prec p$ then
            $n_p \leftarrow n_p + 1$
        end if
    end for
    if $n_p == 0$ then
        $F_1 \leftarrow F_1 \cup p$
        $p_{rank} \leftarrow 1$
    end if
end for
$i \leftarrow 1$
while $F_i \neq \emptyset$ do
    $Q \leftarrow \emptyset$
    for all $p \in F_i$ do
        for all $q \in S_p$ do
            $n_q \leftarrow n_q - 1$
            if $n_q \leftarrow 0$ then
                $q_{rank} \leftarrow i + 1$
                $Q \leftarrow Q \cup q$
            end if
        end for
    end for
    $i \leftarrow i + 1$
    $F_i \leftarrow Q$
end while
```

3.2 Feature importance

3.2.1 Shapley values for observations

The Shapley value is an idea from cooperative game theory where it was used to measure the contribution of each player to the total outcome. It has been adapted to deep reinforcement learning where each feature is considered a player and the policy is considered a game. In our case we initially want to use it to explain actions so we consider that to be the outcome of the game. The Shapley value is calculated by taking each feature and finding the marginal contribution, i.e. seeing how the prediction changes when you include the feature and compare the outcome to when it wasn't included, you do this over all possible coalitions of features. See algorithm 2. The Shapley value has a computational complexity of $O(2^n)$, and because of this we often use estimates instead of exact Shapley values. There are two main ways of reducing the complexity. The first one is using a surrogate model instead of the policy to reduce the number of features, and the second is reducing the number of coalitions. In our case we reduce the number of coalitions to some number 2^K and use K-means clustering to increase representability and reduce variance. Since we have forward passes of the policy, and each can be considered one example of a game we sample the games and use the average marginal contribution over all the games instead of just one game. The Shapley value can help us understand how the policy acts in general. We can also use it to get explanations for single observations, however these explanations won't always have high fidelity. We will explore more of how the fidelity of Shapley values for single explanations compare to high fidelity explainability methods like gradient methods. See section 3.2.2

Figure 3.3: Shapley values for moving up in a critical state in which the agent decided to move up with a confidence of 0.810

3.2.2 Integrated Gradients

By integrating the gradient of the model prediction when going from a specified baseline value as input to our specific observation x we get a high fidelity explanation for what features were important in that specific observation. We integrate from the baseline instead of just getting the gradient in our observation because of the saturation problem. If a feature is important to the action the gradient could be small for the value x, but the integral of the gradient from the baseline to x would still be large. Conversely a gradient for the observation x could be significant in a small area around a feature in x even if that feature isn't as significant as the gradient in x would lead us to believe [27]. See algorithm 3. Comparing figure 3.4 with figure 3.3 we can see that the values are similar. This means that in the case of this observation, the fidelity of the Shapley explanation is high.

Figure 3.4: Feature importance found with integrated gradients method in a state where the agent decided to move up with a confidence of 0.810

Algorithm 2 Shapley Values for Feature Importance in Reinforcement Learning

```
**Input:** RL policy $\pi_\theta$, set of features $F$
**Output:** Shapley values $\phi_f$ for each feature $f \in F$
$\phi_f \leftarrow 0 \; \forall f \in F$    ▷ Initialize Shapley values to zero
$n \leftarrow |F|$    ▷ Number of features
for all $f \in F$ do
    for all $S \subseteq F \setminus \{f\}$ do
        $V_S \leftarrow \texttt{value}(\pi_\theta, S)$    ▷ Evaluate RL model $\pi_\theta$ with feature set $S$
        $V_{S\cup\{f\}} \leftarrow \texttt{value}(\pi_\theta, S \cup \{f\})$    ▷ Evaluate RL model $\pi_\theta$ with feature set $S \cup \{f\}$
        $\phi_f \leftarrow \phi_f + \frac{|S|!(n-|S|-1)!}{n!}\left(V_{S\cup\{f\}} - V_S\right)$    ▷ Update Shapley value for feature $f$
    end for
end for
function VALUE($\pi_\theta, S$)
    Input: RL model $\pi_\theta$, feature set $S$
    Output: Model performance with feature set $S$
    Train model $\pi_\theta$ using features in $S$
    Evaluate model $\pi_\theta$
    return model performance
end function
```

Algorithm 3 Integrated Gradients for Feature Importance in Reinforcement Learning

```
Input: RL policy $\pi_\theta$, input $x$, baseline input $\bar{x}$, number of steps $m$
Output: Integrated gradients $\texttt{IG}(x, \bar{x})$
$\texttt{IG}(x, \bar{x}) \leftarrow 0 \; \forall f \in \texttt{features of } x$    ▷ Initialize integrated gradients to zero
for $i = 1$ to $m$ do
    $\alpha_i \leftarrow \frac{i}{m}$
    $x_i \leftarrow \bar{x} + \alpha_i \times (x - \bar{x})$
    $grad_i \leftarrow \nabla_x \texttt{value}(\pi_\theta, x_i)$    ▷ Compute the gradient of the value function
    $\texttt{IG}(x, \bar{x}) \leftarrow \texttt{IG}(x, \bar{x}) + grad_i \times \frac{(x-\bar{x})}{m}$    ▷ Accumulate gradients scaled by the step size
end for
function VALUE($\pi_\theta, x$)
    Input: RL policy $\pi_\theta$, input $x$
    Output: Model prediction for input $x$
    Feed input $x$ to policy $\pi_\theta$
    return model output
end function
```

3.3 Extracting intent

In an environment where the agents successfully reaches a desirable state we can assume that at some point in the trajectory an agents intent is to reach this state. We can train a prediction model on such an environment that uses a certain state or observation as input and outputs features of interest some time in the future. In the example of autonomous vehicles the output could be predicted position of the vehicle at a future point in time. If our prediction model is successful, then it could be considered part of an explanation for the intent of an agent. Additionally, we train different models with different sets of inputs to identify what is important to identify intent. We train one model that takes just the observation as input, one that takes observation and action, and one that takes observation and action one hot encoded. While action as an integer and action one hot encoded contains the same amount of information it could be easier for a network to learn correlations with the additional inputs. Looking at the loss curve during training, as well as final loss, this suspicion is confirmed to be true when dealing with a network architecture and environment like we are currently using, 2 fully connected hidden layers with the hyperbolic tangent as the activation, in the environment simple_spread_v3. While the input layer for the three prediction models are slightly different the difference in training time is negligible. After 200 epochs the base prediction model reached a evaluation loss of 0.0407, the model with integer action as part of the input reached a evaluation loss of 0.0359 and the model with one-hot encoded action as part of the input reached an evaluation loss of 0.0322. See figure 3.5, 3.6 and 3.7. Both the base model and the one-hot action model stagnate long before 200 epochs is reached. If we let the model with integer action as input train for another 50 epochs it does not improve. The test loss for these three models respectively is ... . These graphs and values indicate that while action and an explanation are useful for extracting intent, in the context of our environment and model one-hot values and integrated gradient explanation are the most useful. The difference between using Shapley values and

Figure 3.5: Loss on evaluation set during training with only observation as input

Figure 3.6: Loss on evaluation set during training with observation and action as integer as input

Figure 3.7: Loss on evaluation set during training with observation and action one-hot encoded as input

Chapter 4

Experiments

4.1 General experimental setup

We have several NNs for each experiment. The goal is to train them to predict states and events. For the state prediction I initially trained a NN to take the observations made by a type of agent from step n and output the x and y coordinates of the same agent at step $n+m$ in a trajectory. The dataset D_pos was constructed such that each data point was from a different trajectory τ, to make sure all the data points were independent of each other, in order to make sure the NN did not learn any such connection. The variable n was chosen randomly for each trajectory while m stayed as a constant, in my case $m = 10$. I would for example have 50000 trajectories, each trajectory contains 2 archers, so we would have a dataset of size 100000. 100000 observations and their corresponding positions 10 steps into the future. We assume that both archers use the same policy as they are homogeneous.

We also trained a NN for event prediction. The event we trained a network to predict was whether an agent would encounter a critical state or not. See section placeholder. The dataset D_{crit} was similar to D_{pos}, however instead of positions, it would contain whether the agent would encounter a critical state within the next 5 steps.

The NN was a fully connected network, had two hidden layers of size 64 each and used the activation function $tanh$. This neural network got an accuracy of placeholder on the test set and the average distance from target was placeholder on the test set. This NN is purely a post hoc explainer and used no information about the policy what so ever to explain the agents intention. As aforementioned we want to include information about the policy to discover how we can improve this prediction, and better discover the intent of the agent.

All hyper parameters stayed the same. Only The dataset, and therefore also the input layer, was different between the runs. I trained the networks for a total of 200 epochs, after which improvement was negligible, if at all existent, with a learning rate scheduler that decreased learning rate when loss reached a plateau, which was measured on a validation set.

4.2 Action inclusion

4.2.1 Setup

4.2.2 Results

4.2.3 Analysis

4.3 Experiment 2

4.3.1 Setup

4.3.2 Results

4.3.3 Analysis

4.4 Experiment 3

4.4.1 Setup

4.4.2 Results

4.4.3 Analysis

Chapter 5

Discussion

5.1 Conclusion

5.2 Limitations

5.3 Future work

Bibliography

[1] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529 (Jan. 2016), pp. 484–489. DOI: 10.1038/nature16961.

[2] Benjamin Ellis et al. *SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning.* 2023. arXiv: 2212.07489 [cs.LG].

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[4] Carlos Zednik. *Solving the Black Box Problem: A Normative Framework for Explainable Artificial Intelligence.* 2019. arXiv: 1903.04361 [cs.GL].

[5] Yuchi Tian et al. *DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars.* 2018. arXiv: 1708.08559 [cs.SE].

[6] Farama. *Knights Archers Zombies.* 2024.

[7] Bertram Malle. "Folk Explanations of Intentional Action". In: Apr. 2001, pp. 265–286. ISBN: 9780262278942. DOI: 10.7551/mitpress/3838.003.0019.

[8] Josh Sammu. "Multi-Agent Collaboration in Autonomous Military Operations". In: (May 2024).

[9] Thomas M. Moerland et al. *Model-based Reinforcement Learning: A Survey.* 2022. arXiv: 2006.16712 [cs.LG]. URL: https://arxiv.org/abs/2006.16712.

[10] Richard S. Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *SIGART Bull.* 2.4 (July 1991), pp. 160–163. ISSN: 0163-5719. DOI: 10.1145/122344.122377. URL: https://doi.org/10.1145/122344.122377.

[11] Danijar Hafner et al. *Mastering Diverse Domains through World Models.* 2024. arXiv: 2301.04104 [cs.AI]. URL: https://arxiv.org/abs/2301.04104.

[12] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning.* 2013. arXiv: 1312.5602 [cs.LG]. URL: https://arxiv.org/abs/1312.5602.

[13] John Schulman et al. *Proximal Policy Optimization Algorithms.* 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[14] Maciej Świechowski et al. "Monte Carlo Tree Search: a review of recent modifications and applications". In: *Artificial Intelligence Review* 56.3 (July 2022), pp. 2497–2562. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10228-y. URL: http://dx.doi.org/10.1007/s10462-022-10228-y.

[15] Matthew Hausknecht and Peter Stone. *Deep Recurrent Q-Learning for Partially Observable MDPs.* 2017. arXiv: 1507.06527 [cs.LG]. URL: https://arxiv.org/abs/1507.06527.

[16] Arthur Guez et al. *An investigation of model-free planning.* 2019. arXiv: 1901.03559 [cs.LG]. URL: https://arxiv.org/abs/1901.03559.

[17] Akanksha Atrey, Kaleigh Clary, and David Jensen. *Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning.* 2020. arXiv: 1912.05743 [cs.LG].

[18] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. *Verifiable Reinforcement Learning via Policy Extraction.* 2019. arXiv: 1805.08328 [cs.LG].

[19] Sagar Patel, Sangeetha Abdu Jyothi, and Nina Narodytska. "Towards Future-Based Explanations for Deep RL Network Controllers". In: *SIGMETRICS Perform. Eval. Rev.* 51.2 (Oct. 2023), pp. 100–102. ISSN: 0163-5999. DOI: 10.1145/3626570.3626607. URL: https://doi.org/10.1145/3626570.3626607.

[20] Jasmina Gajcin and Ivana Dusparic. *ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing RL Policies.* 2024. arXiv: 2402.06503 [cs.AI].

[21] Stephanie Milani et al. *A Survey of Explainable Reinforcement Learning.* 2022. arXiv: 2202.08434 [cs.LG].

[22] Herman Yau, Chris Russell, and Simon Hadfield. *What Did You Think Would Happen? Explaining Agent Behaviour Through Intended Outcomes.* 2020. arXiv: 2011.05064 [cs.AI].

[23] Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. *Explainable Reinforcement Learning via a Causal World Model.* 2024. arXiv: 2305.02749 [cs.LG].

[24] Sagar Patel, Sangeetha Abdu Jyothi, and Nina Narodytska. *CrystalBox: Future-Based Explanations for Input-Driven Deep RL Systems.* 2024. arXiv: 2302.13483 [cs.LG].

[25] Nicholas Kroeger et al. *Are Large Language Models Post Hoc Explainers?* 2024. arXiv: 2310.05797 [cs.CL].

[26] Stephen Chung, Scott Niekum, and David Krueger. *Predicting Future Actions of Reinforcement Learning Agents.* 2024. arXiv: 2410.22459 [cs.AI]. URL: https://arxiv.org/abs/2410.22459.

[27] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks.* 2017. arXiv: 1703.01365 [cs.LG]. URL: https://arxiv.org/abs/1703.01365.