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

Autumn 2024

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

Contents

1 Introduction . . . 3
2 Background . . . 7
2.1 Post hoc vs. intrinsically explainable models . . . 7
2.1.1 Temporal aspect . . . 7
2.1.2 Nature of black box models . . . 7
2.1.3 Lack of transparency . . . 7
2.1.4 Intrinsically explainable models . . . 8
2.2 Fidelity of XAI methods . . . 8
2.3 Future-based explanations . . . 8
2.3.1 Design . . . 8
2.3.2 Use cases . . . 9
2.3.3 Definition of intent . . . 9
2.4 Relevant methods . . . 9
2.4.1 What did you think would happen? Explaining Agent Behaviour through Intended Outcomes . . . 10
2.4.2 Explainable Reinforcement Learning via a Causal world model 11
2.4.3 CrystalBox: Future-Based Explanations for Input-Driven Deep RL Systems . . . 11
2.4.4 Are Large Language Models Post Hoc Explainers . . . 11
2.4.5 ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing RL Policies . . . 12
3 Methodology . . . 13
3.1 Action importance . . . 13
3.1.1 Counterfactual sequences . . . 13
3.2 Feature importance . . . 13
3.2.1 Shapley values for observations . . . 13
3.2.2 Integrated Gradients . . . 15
3.3 Extracting intent . . . 16
3.4 –IPPO vs MAPPO . . . 16
3.5 –Reward function changes?. . . . 18
3.6 –How to identify intent . . . 18
3.7 –Synergy/intent collision . . . 18
3.8 –Lazy agents/sparse rewards . . . 19
3.9 –Same intent for same state? Different policies? . . . 19
3.10 –Change environment, related to permutation importance . . . 19
3.11 –Feature importance . . . 19
3.12 –Feature permutation changes, for other agents and zombies . . . 19
Bibliography . . . 21

Contents

Chapter 1

Introduction

Sequential decision-making 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 1.1. In a Multi-Agent Reinforcement Learning (MARL) system, we would have multiple agents.

A significant problem with many deep machine learning models, including MARL policies, is known as the black box problem[5]. 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[6]. 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.

Explainable Artificial Intelligence (XAI) is an umbrella term for all explainable

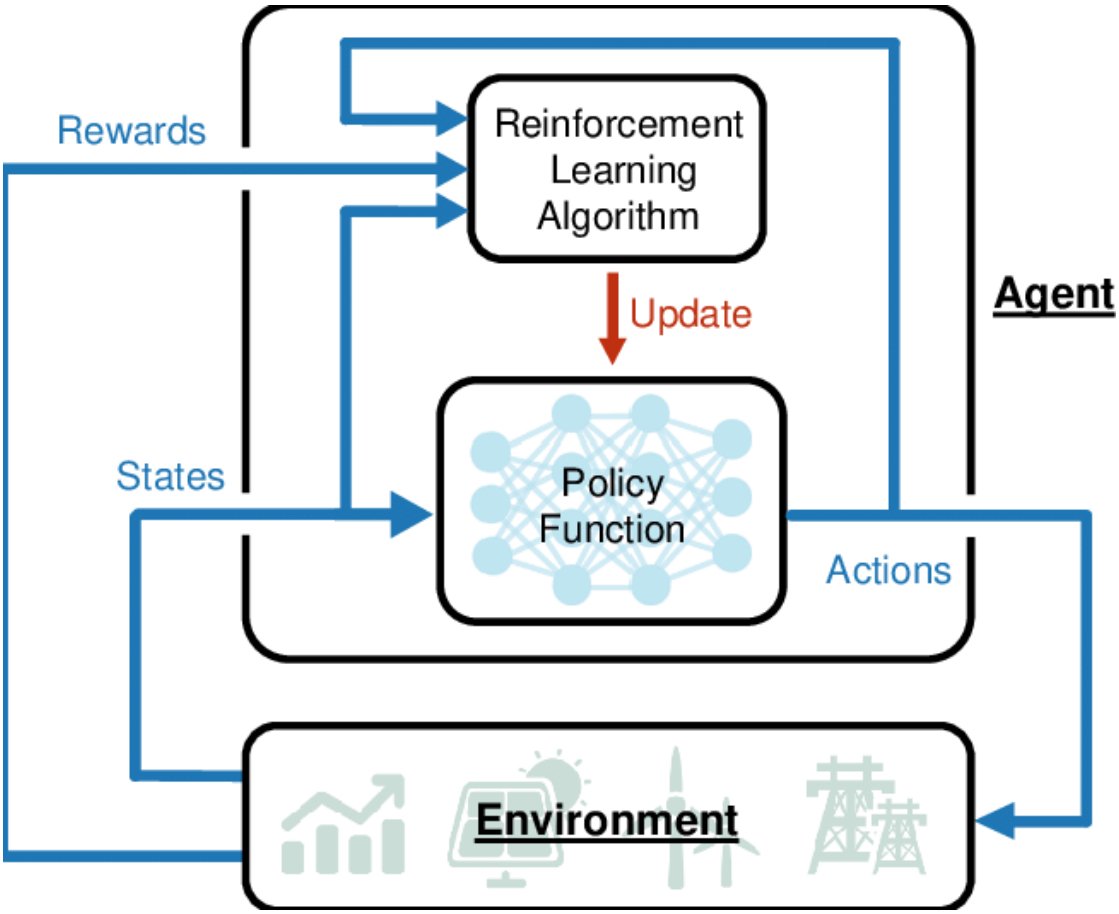

Figure 1.1: Structure of a simple RL system as found in Review of Learning-Assisted Power System Optimization [4]

artificial intelligence. The focus of this paper will be to rework and apply methods for general XAI to Reinforcement Learning (XRL). 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 [7] 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[8], 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.

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. First we will go a bit more in depth on what type of explainablity we want to use as well as

describe relevant methods already used, then we will attempt to integrate previous and own methods into a MARL policy we construct and train, and at the end we will discuss our findings.

Chapter 2

Background

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.

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.1.4 Intrinsically explainable models

Intrinsically explainable models has the aforementioned drawback of relying on model construction to be done well, as we will be unable to use a model that is shown to be accurate, but since it doesn't get as affected by point II.I to II.III it will be the preferred method in this paper.

2.2 Fidelity of XAI 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[9].

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 [10]. 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?"

2.3 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.

2.3.1 Design

"Towards Future-Based Explanations for Deep RL Network Controllers"[11] 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.

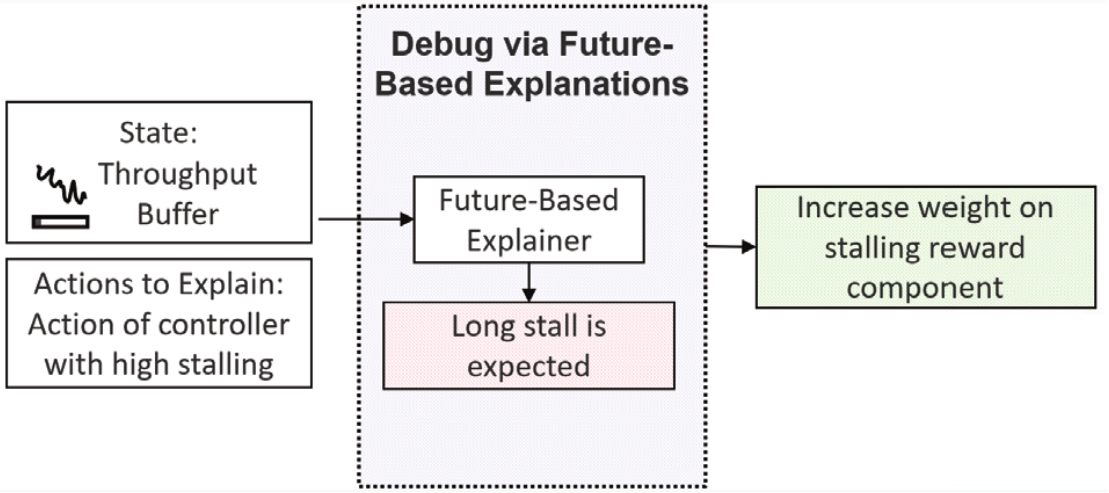

Figure 2.1: Using a future-based explainer as debugger to avoid stalling for adaptive bitrate streaming as found in "Towards Future-Based Explanations for Deep RL Network Controllers" [11]

2.3.2 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. See Figure 2.1.

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.

2.3.3 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[12], see section 2.4.5. Details in section 3.1.1.

2.4 Relevant methods

There are several papers written on XAI and XRL problems. Milani et al.[13] 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 Recurrent Neural Network (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" [14], "Explainable Reinforcement Learning via a Causal world model" [15] and "CrystalBox: Future-based explanations for input-driven deep RL systems" [16] 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" [17] Could be relevant as using other explainers to compare explanations is useful. "ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing RL Policies" [12]

2.4.1 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 intended model 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. See Figure 2.2

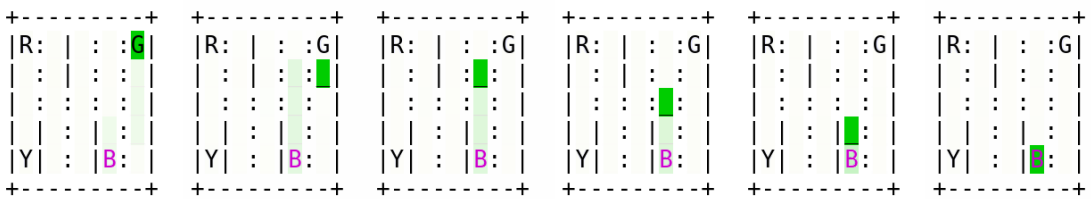

Figure 2.2: Belief map of a Taxi driving from G to B, where the opacity describes the likelihood of the taxi entering that square as found in "What did you think would happen? Explaining Agent Behaviour through Intended Outcomes" [14]

2.4.2 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.

2.4.3 CrystalBox: Future-Based Explanations for Input-Driven Deep RL 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.

2.4.4 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.

2.4.5 ACTER: Diverse and Actionable Counterfactual Sequences for Explaining and Diagnosing RL 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 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.

Chapter 3

Methodology

3.1 Action importance

3.1.1 Counterfactual sequences

To discover counterfactual sequences we use the EA known as NSGA-II (Non dominated Sorting Genetic Algorithm 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. [**?**] 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.

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

Algorithm 1 Fast Non-Dominated Sort

```
F_i ← ∅
for all p ∈ P do
    S_p ← ∅
    n_p ← 0
    for all q ∈ P do
        if p ≺ q then
            S_p ← S_p ∪ q
        else if q ≺ p then
            n_p ← n_p + 1
        end if
    end for
    if n_p == 0 then
        F_1 ← F_1 ∪ p
        p_rank ← 1
    end if
end for
i ← 1
while F_i ≠ ∅ do
    Q ← ∅
    for all p ∈ F_i do
        for all q ∈ S_p do
            n_q ← n_q − 1
            if n_q ← 0 then
                q_rank ← i + 1
                Q ← Q ∪ q
            end if
        end for
    end for
    i ← i + 1
    F_i ← Q
end while
```

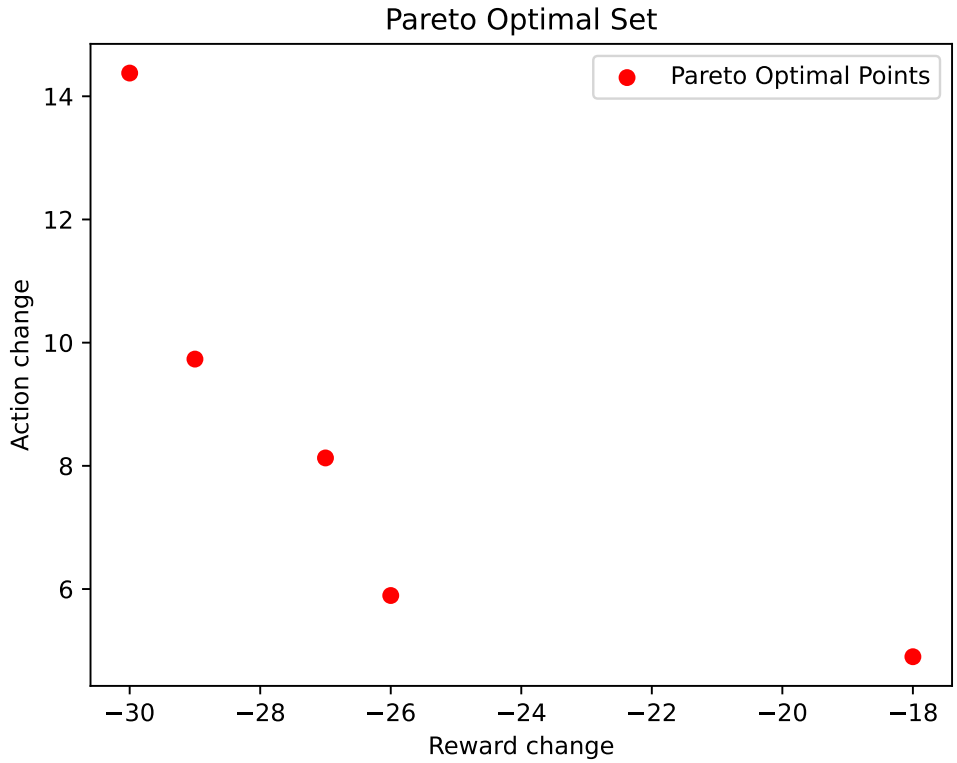

Figure 3.1: Pareto optimal points obtained with 2000 generations and population of 400 with NSGA-II

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

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 [18]. See algorithm 3. Comparing figure 3.4 with figure 3.3 we can see that the values

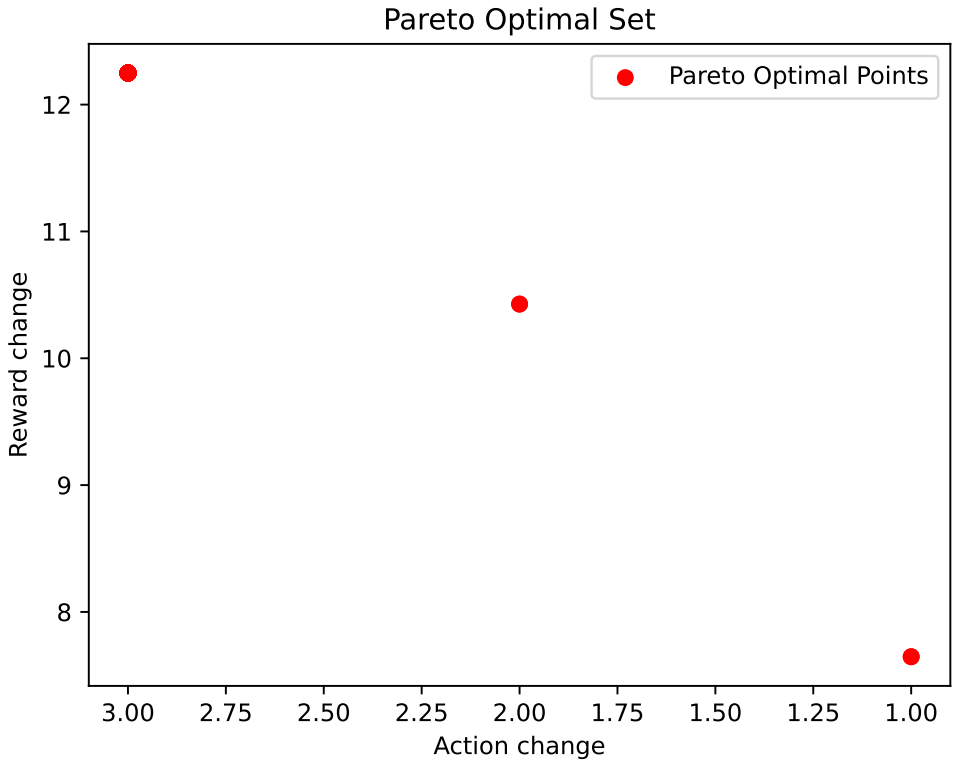

Figure 3.2: Pareto optimal points obtained with 2000 generations and population of 400 with NSGA-II

are similar. This means that in the case of this observation, the fidelity of the Shapley explanation is high.

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 could train different models with different sets of inputs to identify what is important to identify intent. We could for example train one model that takes just the observation as input, one that takes observation and action, and one that takes observation and Shapley values.

3.4 –IPPO vs MAPPO

Using IPPO for now because [**?**]. Should be enough for explainability.

Algorithm 2 Shapley Values for Feature Importance in Reinforcement Learning

Input: RL policy π_θ, set of features F
Output: Shapley values ϕ_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 π_θ with feature set S
    $V_{S\cup\{f\}} \leftarrow \texttt{value}(\pi_\theta, S \cup \{f\})$ ▷ Evaluate RL model π_θ 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(π_θ, S)
  Input: RL model π_θ, feature set S
  Output: Model performance with feature set S
  `Train model` π_θ `using features in` S
  `Evaluate model` π_θ
  return `model performance`
end function

Algorithm 3 Integrated Gradients for Feature Importance in Reinforcement Learning

Input: RL policy π_θ, input x, baseline input $\bar{x}$, number of steps m
Output: Integrated gradients $\texttt{IG}(x, \bar{x})$
$\texttt{IG}(x, \bar{x}) \leftarrow 0 \; \forall f \in$ `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(π_θ, x)
  Input: RL policy π_θ, input x
  Output: Model prediction for input x
  `Feed input` x `to policy` π_θ
  return `model output`
end function

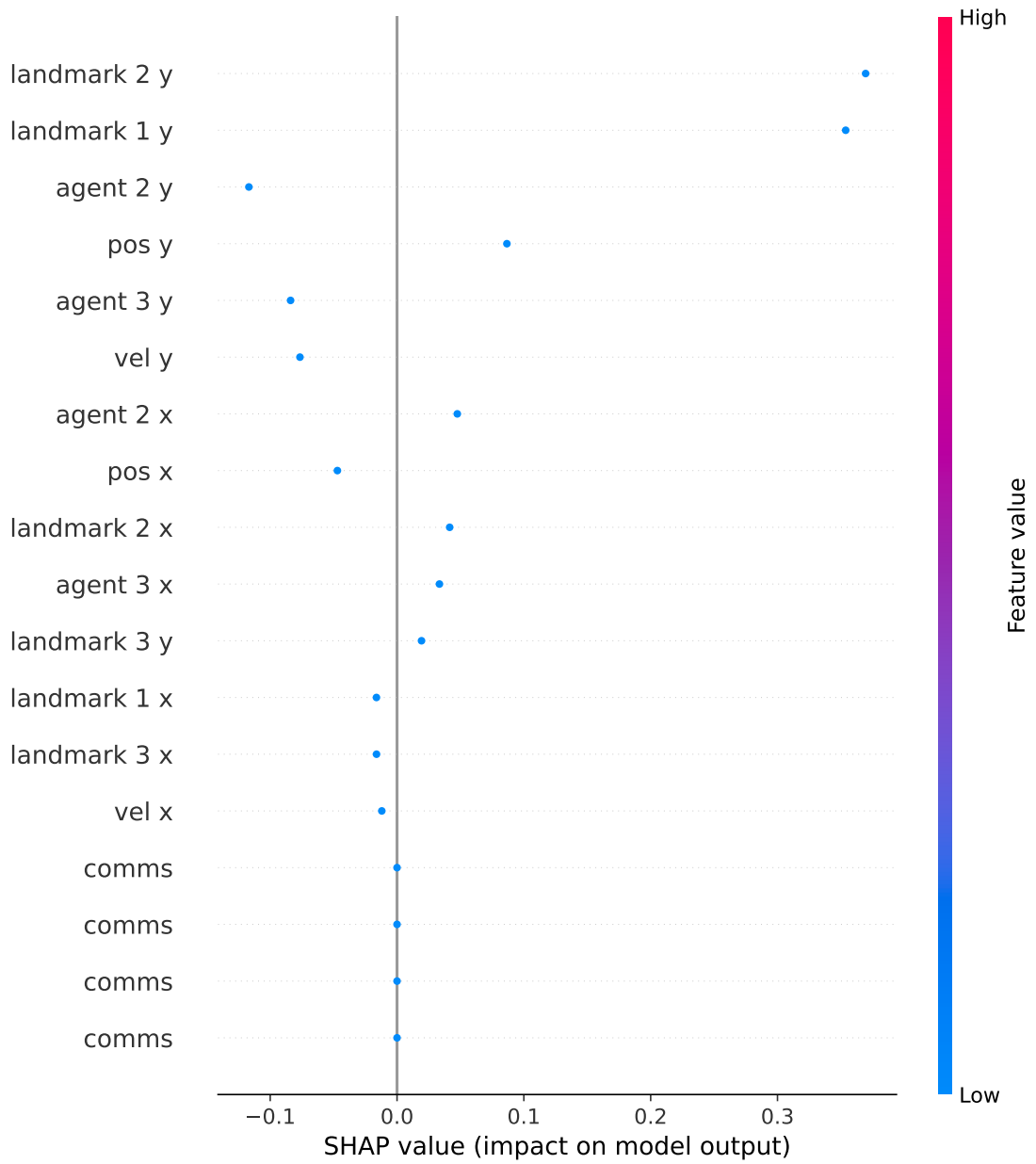

Figure 3.3: Shapley values for moving up in a critical state in which the agent decided to move up with a confidence of 0.810

3.5 –Reward function changes?

3.6 –How to identify intent

3.7 –Synergy/intent collision

Cooperation index

3.8 –Lazy agents/sparse rewards

archers train fine, knights only start learning anything useful after a very long time, 12M timesteps, then not stable

-alternate reward structure?

-memories to focus more on knights?

-policy architecture?

-start with only training knights at first then add archers as well? (eg. 4 knights -> 2 knights, 2 archers) for consistent action and observation space. Similar to using a reward structure where only knights gain rewards for n timesteps before both knights and archers gain rewards.

-Possible to not have the same reward for every agent, but then no centralised critic

3.9 –Same intent for same state? Different policies?

Swap homogeneous agents to see intent difference

3.10 –Change environment, related to permutation importance

3.11 –Feature importance

Detect which zombies agents care about saliency maps

3.12 –Feature permutation changes, for other agents and zombies

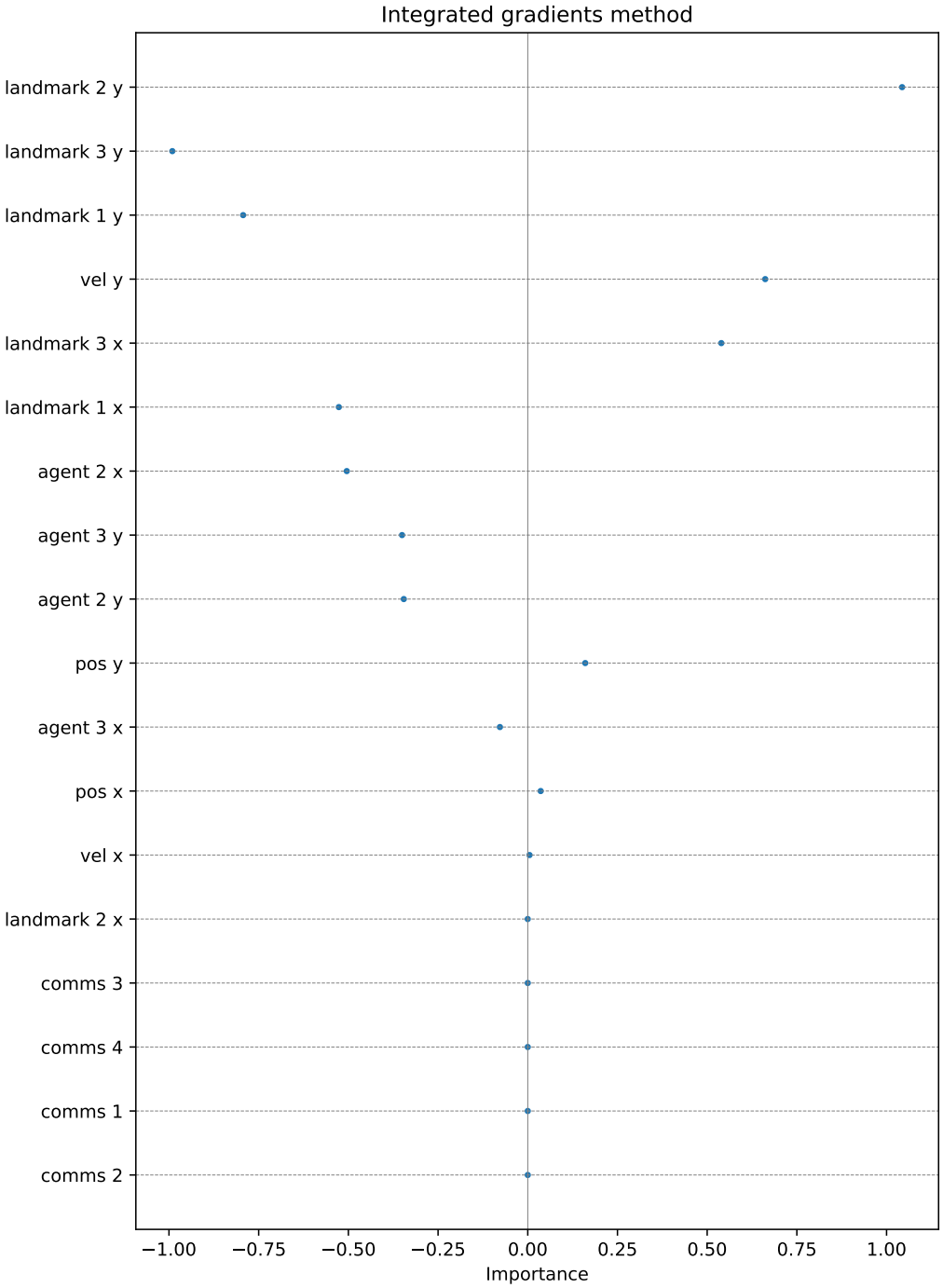

Figure 3.4: Feature importance found with integrated gradients method in a state where the agent decided to move up with a confidence of 0.810

Bibliography

[1] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[2] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning, 2023.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[4] Guangchun Ruan, Haiwang Zhong, Guanglun Zhang, Yiliu He, Xuan Wang, and Tianjiao Pu. Review of learning-assisted power system optimization, 09 2020.

[5] Carlos Zednik. Solving the black box problem: A normative framework for explainable artificial intelligence, 2019.

[6] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars, 2018.

[7] Farama. Knights archers zombies, 2024.

[8] Bertram Malle. *Folk Explanations of Intentional Action*, pages 265–286. 04 2001.

[9] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning, 2020.

[10] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction, 2019.

[11] Sagar Patel, Sangeetha Abdu Jyothi, and Nina Narodytska. Towards future-based explanations for deep rl network controllers. *SIGMETRICS Perform. Eval. Rev.*, 51(2):100–102, oct 2023.

[12] Jasmina Gajcin and Ivana Dusparic. Acter: Diverse and actionable counterfactual sequences for explaining and diagnosing rl policies, 2024.

[13] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. A survey of explainable reinforcement learning, 2022.

[14] Herman Yau, Chris Russell, and Simon Hadfield. What did you think would happen? explaining agent behaviour through intended outcomes, 2020.

[15] Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. Explainable reinforcement learning via a causal world model, 2024.

[16] Sagar Patel, Sangeetha Abdu Jyothi, and Nina Narodytska. Crystalbox: Future-based explanations for input-driven deep rl systems, 2024.

[17] Nicholas Kroeger, Dan Ley, Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju. Are large language models post hoc explainers?, 2024.

[18] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017.