Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Iver Osborg Myklebust

Explainable AI methods for Cyber-Physical systems

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios Lekkas
Co-supervisor: Sindre B. Remman
June 2021

NTNU
Norwegian University of Science and Technology

Iver Osborg Myklebust

Explainable AI methods for Cyber-Physical systems

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios Lekkas
Co-supervisor: Sindre B. Remman
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This thesis serves as the final work on my master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The work was conducted under the supervision of Anastasios Lekkas, during the spring of 2021.

The goal of this project is to develop and implement Explainable Artificial Intelligence (XAI) methods capable of extracting information from deep neural networks trained via reinforcement learning. Two of the most frequently used model-agnostic XAI-methods today, Local Interpretable Model-Agnostic Explanation (LIME) and SHapley Additive exPlanations (SHAP), have been used in experiments across two different simulated robotic environments, Cartpole from OpenAI Gym and a Robotic Manipulator from Robotis [1][2]. These have been used to provide local and global explanations on both environments. In addition, the two gradient methods Integrated Gradients (IG) and Saliency, from the XAI-subfield of Neural Network interpretations, have been implemented to compare the global explanations.

The implementation is done within an Anaconda environment in Jupyter Notebook, based on the machine learning library Pytorch [3]. The Github libraries of SHAP, LIME and Captum are used to provide the interpretations [4][5][6]. The research has been implemented on a workstation provided by NTNU.

The master project is a continuation of the author's pre-project done in the fall of 2020 [7]. Some of the theory and experiments overlap throughout this report and have been either reused or rewritten. This is clarified at the start of the sections where it applies. The report is written in a way that does not require any background knowledge about XAI. Therefore, the methods and theory are thoroughly explained. However, it is assumed that the reader has a basic understanding of machine learning and mathematics. All figures have been created by the author with *draw.io* unless otherwise stated.

This project is a part of the EXAIGON project at NTNU, which aims to meet society's and industry's standards for deployment of trustworthy AI systems [8].

Acknowledgement

A huge thanks go to Anastasios Lekkas for his encouraging support and supervision throughout the semester. Also, a big thank you to PhD-student Sindre B. Remman for his insights and help with understanding the XAI libraries. His master's thesis, *Robotic manipulation using Deep Reinforcement Learning*, has served as a basis for some parts of this project [9]. The robotic manipulator is used as one of the experiments, and the SHAP implementation is extended into more methods. Lastly, a special thank you goes to my friends and family in a tough semester characterized by Covid-19 restrictions and a demanding workload.

Iver Osborg Myklebust
June 6, 2021

Abstract

The advancements in Artificial Intelligence (AI) in the last decade have paved the way for an innovative and more digitalized society. To this date, AI solutions are present in many situations we encounter in our everyday life. Based on this evolution, more research is now being done into Reinforcement Learning (RL). For robotics, this could be a game-changer since RL makes systems able to learn from experience, which can take future robots into an even higher degree of autonomy. The biggest break-through is present in the field of Deep Reinforcement Learning (DRL), where RL is combined with Artificial Neural Networks (ANN).

However, the introduction of neural networks into reinforcement learning comes at a cost. The systems behave like black boxes that do not provide any explanations or justifications for their predictions. In robotics, where mistakes could lead to catastrophic consequences, these systems must be made more transparent and trustworthy before they can be deployed. The aim for Explainable Artificial Intelligence (XAI) is to interpret an agent's decision-making to obtain insight into the black-box systems.

In this project, four XAI-methods have been used to interpret the decisions made by DRL-agents across two robotic environments of different complexity. Four procedures have been implemented to investigate how these models predict local situations, globally across an entire episode, within the training phase, and with data adaptions, including feature space reduction and forced initializations. A "real-time" example is also included to demonstrate how such an explainer model can interact with an audience.

The research shows that much information can be collected from the XAI-experiments, both to confirm pre-existing human intuition about the models and discover new trends. However, unexpected interpretations, together with assumptions and weaknesses present in the methods, can put the explanations under a critical view. Although XAI could improve final decision-making, it is vital to keep the limitations within the explainers in mind. This especially yields for complex environments, where highly correlated features can cause problems. Feature removals were a great way to reduce some of these challenges, but much research remains before this could be seen in real-life robotic DRL-development.

Sammendrag

Fremskrittene i kunstig intelligens (AI) det siste tiåret har banet vei for et innovativt og mer digitalisert samfunn. Til dags dato er AI-løsninger til stede i mange situasjoner vi møter hver dag. Basert på denne evolusjonen blir det nå gjort mer forskning på forsterket læring (RL). For robotikk kan dette være en game changer, siden RL gjør systemer i stand til å lære basert på erfaring, noe som kan øke graden av autonomi i fremtidens roboter. Det største gjennombruddet er til stede i feltet dyp forsterkende læring (DRL), der RL kombineres med kunstige nevrale nettverk (ANN).

Imidlertid kommer innføringen av nevrale nettverk i RL med en ulempe. Systemene oppfører seg som en svart boks, og gir sjelden noen forklaringer eller begrunnelser for deres prediksjoner. I robotikk, der feil kan føre til katastrofale konsekvenser, må disse systemene gjøres mer transparente og pålitelige. Målet for forklarende kunstig intelligens (XAI) er å tolke beslutningene til en agent for å få mer innsikt og forståelse om systemene.

I dette prosjektet har fire XAI-metoder blitt brukt til å tolke beslutningene tatt av DRL-agenter i to robotmiljøer av ulik kompleksitet. Fire prosedyrer har blitt implementert for å undersøke hvordan disse modellene oppfører seg i lokale situasjoner, globalt over en hel episode, innen trenings-fasen, og med data-tilpasninger, inkludert reduksjon av tilstander og tvangsinitialisering. Et "sanntids"-eksempel er også inkludert for å vise hvordan en slik forklaringsmodell kan samhandle med et publikum.

Forskningen viser at mye informasjon kan samles fra XAI-eksperimenter, både for å bekrefte eksisterende menneskelig intuisjon om modellene og oppdage nye trender. Imidlertid kan uventede tolkninger, sammen med antakelser og svakheter i metodene, sette forklaringene under et kritisk syn. Selv om XAI kan forbedre den endelige beslutningstaking, er det viktig å ha begrensningene i bakhodet. Dette gjelder spesielt for komplekse miljøer, der sterkt korrelerte funksjoner kan forårsake problemer. Å redusere tilstandsrommet var en effektiv måte å løse noen av utfordringene, men mye forskning gjenstår før dette kan sees i avansert robotutvikling.

Contents

List of Tables

List of Figures

Acronyms

AI Artificial Intelligence. iii, iv, ix, 1, 2, 10, 12, 18

ANN Artificial Neural Networks. iii, iv, 13–16

DDPG Deep Deterministic Policy Gradient. 12, 16, 34

DL Deep Learning. 2, 12

DRL Deep Reinforcement Learning. iii, iv, 4–7, 9, 13, 15, 16, 18, 37, 43, 54, 68, 71, 80, 84–86

GDPR General Data Protection Regulation. 1, 85

GUI Graphical User Interface. 76

HER Hindsight Experience Replay. 17

IG Integrated Gradients. i, 19, 27–29, 40, 44, 47, 53, 54, 59–64, 66, 68–71

LIME Local Interpretable Model-Agnostic Explanation. i, 19, 20, 22, 31, 37, 39, 40, 44, 46, 48, 54, 55, 58, 72–74, 76, 81

MDP Markov Decision Process. 10–12

RL Reinforcement Learning. iii, iv, 1, 2, 6, 10–13, 15, 17, 32

SHAP SHapley Additive exPlanations. i, ii, 19, 24, 25, 31, 37–39, 44, 46–49, 51, 53–56, 58, 60–64, 66–74, 76, 79, 81, 83

XAI Explainable Artificial Intelligence. i–iv, 2–7, 9, 15, 18, 19, 31, 37, 39–41, 43, 44, 54–56, 59, 61, 62, 64, 67, 68, 70, 71, 80, 83–86

Chapter 1

Introduction

1.1 Background and motivation

With the progress in intelligent systems research in the last decade, Reinforcement Learning (RL) has become a widely-used technique for training agents to solve some of the challenges within autonomy. Human-machine interaction is now a part of our everyday life, with successful implementations in, for example, robotics, autonomous driving, and other safety-critical applications [10]. Unfortunately, when working in such domains, the black box nature (Fig 1.1) of RL-models can lead to legal and ethical concerns [11]. In many situations, like life-changing decisions in hospitals or on the roads, the need for trusting these systems is crucial. The EU's General Data Protection Regulation (GDPR) came into effect in 2018. They aimed to ensure *"a right to explanation"*, meaning these implementations must increase their consideration towards more transparent decision-making and human-machine interaction [12][13].

According to DARPA, the evolution of AI has seen three waves (Fig 1.2) [14]. In the first wave, sets of logic rules were created to represent knowledge in limited domains. This enabled reasoning over narrowly defined problems, but the algorithms had no learning capability and did not handle uncertainty well. With the progress in machine learning, statistical models are created for specific problem domains and trained on big data. Learning is introduced in the second wave, where supervised and unsupervised

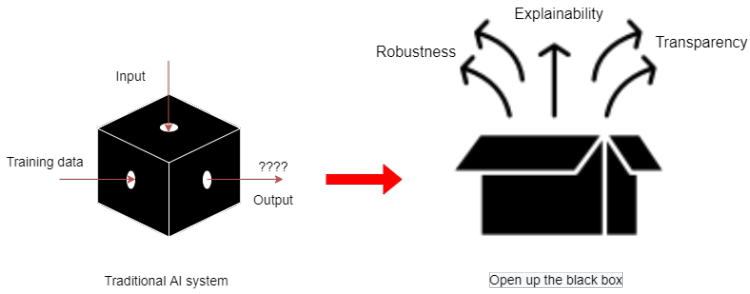


Figure 1.1: The black box problem

learning are driven by tasks and data. These are typically used to solve classification tasks in defined environments. Reinforcement Learning differs by aiming to solve problems through experience, and error handling [10].

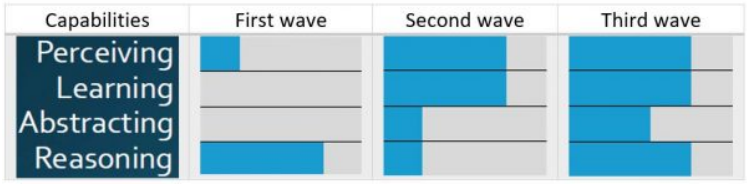


Figure 1.2: DARPA's three waves of AI, from [14]

As shown in Fig 1.2, introducing learning abilities into these systems leads to increased prediction probabilities. However, because of the enormous amounts of parameters in the statistical models that do not correspond to intuitive variables, it also meant less transparency. Thus, the incentives behind the development of Explainable Artificial Intelligence (XAI) are the transition into a third wave, where statistical learning meets transparency and reasoning abilities.

How to open up the black box?

The problem of model transparency has been well known in the AI/ML community with the introduction of Deep Learning (DL). Artificial Intelligence has made tremendous progress recently by combining statistical learning and neural network classifiers. The

effectiveness of the neural networks is achieved by passing the input through many hidden layers with millions of parameters and different activation functions [15]. In many ways, neural networks are spreadsheets on steroids. By stretching and squashing each layer many times, a complex space can be represented in lower-dimensional structures. This has opened up for a considerable number of practical applications to emerge, such as systems that can do everything from facial recognition to beat humans in video games [10][14].

However, because of the complex nature of deep neural networks, it lacks transparency, which makes it difficult to get insight into the mechanisms that produce the output [10]. Recently, the research into XAI, where the goal is to understand the inner workings of black box models, has increased [16]. XAI-methods have mainly been used in financial and medical applications with labeled data, which means the explanations more efficiently can be verified. An example in supervised XAI could be to explain a financial loan rejection, where the user wants to get a reason from the automated banking system. Often the reason behind such decisions can be trivial, for example the customer got rejected because of a too low income.

The goal of this project is to use these XAI-methods to explore robotic environments. If a robotic failure occurs, many factors play a role since uncertainty and intuition must also be considered. Therefore, explanations in these environments will often be compared directly to the intuition of a human controller or by designing answerable human-machine interaction agents.

Even though XAI could be used as a tool to getting information from the black box, there exist several goals and types of transparency. Before presenting the objectives for this project, these will be defined.

The ideal human-machine interaction

Even though the number of papers published about XAI has increased heavily in the last decade, the terms transparency, interpretability, and explainability are often used equivalently. Lipton [17] defines transparency as the opposite of a black-box-ness; it gives an understanding of how the model works. Interpretability refers to which extent

a learned model makes sense to a user, while explanations are a way to clarify how the learned model works. The rest of the report follows this terminology.

In the concluding remarks from the author's project thesis, the main factor to consider when choosing between XAI-methods was the given use case. For a developer, it is essential to understand how the system works to be able to debug or improve it. Typical things to consider are time limitations, expertise level, and the need for local or global explanations. From the user's perspective, it is more important to get a sense of trust in the technology, to get comfortable with the predictions [18]. The common factor is to define a target audience and ask why, for who, and how the model could get more interpretable. Depending on the use case, this could be data scientists, managers, or the end-user. Still, the need for increased model understanding and future regulatory compliance are often goals that exist through all the target audiences.

When it comes to cyber-physical systems, in addition to inspiring trust, transparency and interpretability can help a lot when something has gone wrong. They make it possible to go to the logged data, investigate what went wrong, and engineer a better solution to avoid a similar phenomenon in the future. Explainable methods can be embedded directly into the neural network model or applied as an external post hoc algorithm. In this project, different types of post-hoc methods will be compared with each other and explored throughout environments of different complexity. The motivation of this thesis is, therefore, to investigate how methods from XAI can improve the understanding and make robotic systems trained with Deep Reinforcement Learning more trustworthy.

1.2 Objectives and research questions

The main goal of this thesis is to answer the following research questions:

- **How do state-of-the-art methods from Explainable Artificial Intelligence (XAI) perform on simulated robotic systems? What are the crucial factors to consider when choosing between these XAI-frameworks?**
- **Can these XAI-explanations be used to engage with end-users, and how does this affect the trust of the DRL-models used to control robotic systems?**

To answer these questions, the semester was divided into a series of objectives to track the project progress:

1. Continuation from the pre-project with a literature review recap. Start exploring which XAI-methods that can fit into the Cartpole environment from OpenAIGym [1].
2. Implement Integrated Gradients, described in Section 2.8.2, on the Cartpole environment. Compare globally with SHAP-explanations from the pre-project [7]. From the Captum package of Pytorch [6], Saliency, and weight analysis were also included.
3. Transfer all the XAI-methods onto a more complex environment, the Robotic Manipulator lever model from [9].
4. Implement training procedures, feature removals, initialization- and normalization techniques throughout both environments to learn more about the agents.
5. Implement episode plots for local explanations (SHAP/LIME), and find ways to visualize it in a real-time manner for both environments. Transfer it into a possible simulation procedure. Compare the methods, search for good situations and visualizations throughout the procedures to highlight the trends observed.

1.3 Contributions

- Two pre-trained reinforcement learning environments, described in Chapter 3, are used to show how some of the most promising XAI-methods can interpret the decisions of a DRL agent. This project combines two state-of-the-art fields of study that still are primarily in the development phase. DRL for robotic problems is getting increased attention, and XAI methods have started to be explored in supervised machine learning problems. The thesis shows that XAI methods can give some interpretations in robotic environments that are in line with human intuition. Such insight can increase or decrease the trust of using these DRL-models in real-life robotic problems. However, with increased dimensionality, some challenges are discovered regarding correlation and keeping control over all the different features.
- The project contributes with four XAI-implementation procedures, each containing two robotic environments and in total four XAI-methods. In the first part, these XAI-methods get compared across local and global explanations. The explainers usually agree on the most influential features in both environments, but challenges with both the perturbation and gradient methods are observed. The assumptions being made in these methods and how this can be deployed when trying to interpret RL-agents with no clear answers is discussed in connection with this part.
- This forms as a motivation for the next parts, where some of the methods are put under "pressure" with forced initializations and by collecting explanations within the training procedure. After seeing how few of the manipulator features contribute significantly according to the XAI-methods, the state-space is reduced to explain the same agent, but with fewer active states. In the last part, episode plots and a "real-time" simulation are shown as examples of using these explanations to engage with an end user. Although probably still being early in the developments of transparent, explainable robotic models, this project will hopefully show some of the possibilities and challenges from today's available methods. In the end, the conclusion is to keep the human side of the equation in mind. XAI methods can be used as a tool to interpret results in DRL environments, but the limitations of the methods on robotic systems should put the explanations under a critical view.

1.4 Outline of the report

The rest of the report is divided into five chapters:

- Chapter 2: **Theory**
  - This chapter introduces terminology and theory that is important for the rest of the thesis. It starts with an overview of the DRL methods used to train the agents before explaining the theoretical foundation behind the XAI methods. In the end, the characteristics of each method are compared in table 2.1.
- Chapter 3: **Methodology and experiments**
  - An overview of the main software used in this thesis is provided. The two environments are presented along with the implementation of the XAI-methods.
- Chapter 4: **Results and Discussion**
  - This chapter is divided into four parts where results from the two environments are presented.
    - Method Comparison - Local and global explanations across four different methods for both environments.
    - Data adaptions - Forced initialization in the Cartpole environment and feature removal within the Robotic Manipulator environment.
    - Training analysis - Global explanations collected throughout the training phase in the Cartpole environment.
    - Real-time simulations - Episode plots for both environments with an attached example of using these explanations to engage with a human operator.

  After each part, the results are discussed in regards to the research questions of the project.
- Chapter 5: **Conclusion**
  - A conclusion to the thesis is given. The research questions are answered, and in the end, the possible extensions to future work are described.

Chapter 2

Theory

As this thesis is a continuation of the specialization project done in Fall 2020, the required theory is mostly similar. This means that the theory chapter is an updated version of the author's earlier work [7], with additional XAI-methods presented in Section 2.8.

The theory chapter will introduce the background theory used in this thesis. Since the project implements XAI methods on models trained with deep reinforcement learning, the theory behind DRL-algorithms will first be described. The algorithms used in this project are based on the respective policy gradient methods:

- **Cartpole** - Monte Carlo policy gradient (REINFORCE), described in Section 2.4.1.
- **Robotic Manipulator** - Deep determinstic policy gradient (DDPG) with Hindsight experience replay (HER), described in Section 2.4.2.

In contrast to traditionally more common control methods, reinforcement learning makes the robots learn from experience. The learning procedure helps tackle uncertainty, but today it comes at the cost of less knowledge about the robots' decisions. The theory chapter is therefore divided into two parts, where the XAI methods that are used to derive interpretations from the neural networks are presented from Section 2.5.

2.1 Machine Learning

Oxford dictionary defines Artificial Intelligence as *"the theory and development of computer systems able to perform tasks normally requiring human intelligence"* [19]. Machine learning is a branch of AI which allows models to improve performance based on processed data. In other words, algorithms learn from experience. There are three main types of machine learning [20][21]:

- Supervised learning - The model learns by using labeled output data as guidance.
- Unsupervised learning - The model learns by finding patterns in unlabeled data without any guidance.
- Reinforcement learning - The model learns by interacting with the environment. In many ways more similar to how humans learn. An agent learns how to behave in an environment, and the goal is to maximize the feedback reward signal in the long run. In this work RL will be the focus since it can enable robots to improve their performance gradually.

2.2 Reinforcement Learning

A lot of literature exists on reinforcement learning, and depending on the problem, different approaches can be relevant. This thesis will mostly focus on the literature behind the algorithms used to solve the environments in this project. Similarly to how the human brain learns, RL employs positive and negative feedback to learn how to perform various tasks. This can be compared with closed-loop problems where the goal is to maximize the reward. The environment defines the task that is going to be solved. It is modeled as a Markov Decision Process (MDP), and RL is a way to solve problems described by MDPs. An agent observes and acts on the environment based on the goal to maximize reward. To control the agent, a decision process needs to be defined, inspired by [22] and [23].

- A Markov Decision Process (MDP) is defined as a tuple $\langle S, A, T, R \rangle$, where S is a set of states that forms the state space, and the set of actions A forms the action space. T is the transition function when applying an action a going from a state s

to a new state s', while the reward function R is used to give direction for which way the system (MDP) should be controlled. MDPs have three different optimality criteria: finite horizon, discounted infinite horizon, and average reward.

- Policies - Determines which action an agent should take based on the environment state. Can be deterministic (direct mapping) or stochastic (maps over a probability distribution). Defined by $\pi : S \rightarrow A$.
- RL Process - In a basic RL operation, the agent receives a state from the environment, performs an action, and receives a reward continuously. This is illustrated in Fig 2.1

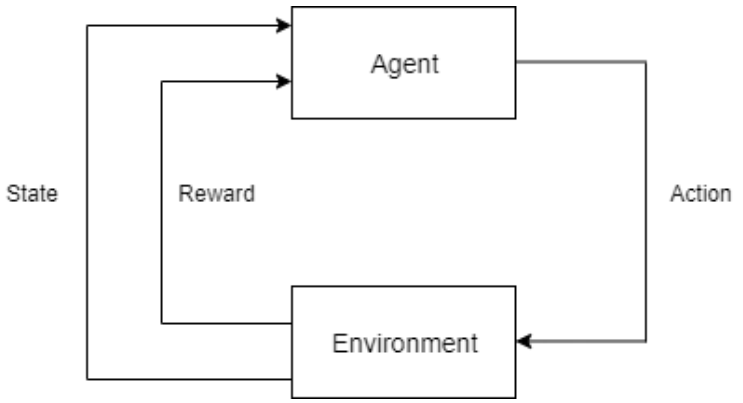


Figure 2.1: Reinforcement Learning process

Exploration and exploitation

When dealing with reinforcement learning algorithms, it is crucial to understand the trade-off between exploration and exploitation. This is what decides the agent's ability to discover new strategies, balancing between *exploiting the best actions* and *exploring the environment* by trying new strategies [20]. The most common exploration strategies are

- ϵ-greedy exploration - The agent chooses the greedy (exploiting the best action) with probability ϵ, or a random action with probability $1 - \epsilon$.

- Exploration noise - To get the agent to discover new strategies along with using the learned knowledge from the environment, noise can be added to the greedy action. This demands a continuous action space, which is usually present in robotic problems.

Solving an MDP-problem

Two of the most common methods when solving a Markov Decision Process are *Value iteration* and *Policy iteration*. Policy iteration focuses on evaluating and improving the policy at every step. In contrast, value iteration focuses purely on estimating the value function, and after it has converged towards V^*, the policy is computed. However, these techniques, called model-based RL, demands that the probability or/and transition matrix are known.

In Reinforcement Learning, a perfect model is often not defined since the probability, and transition matrices are unknown. When this is the case, statistical knowledge about the model needs to be gathered through MDP-sampling. This is called model-free RL, and there are two ways to sample the MDP [9][22]:

- Actor - Learn the Policy directly with a function approximation. The Value function is ignored.
- Critic - Learn Value function with a function approximation, and the policy gets derived implicitly by, for example, an exploration strategy.

The REINFORCE algorithm used in this thesis is an Actor-Only method, meaning it only samples the policy. DDPG is an Actor-Critic method; it learns both a policy and the value function. These algorithms use neural networks in their MDP-sampling, so before presenting the methods, some terms from Deep Learning (DL) need to be defined.

2.3 Deep Learning

Machine learning is a branch of Artificial Intelligence that provides systems the ability to learn from experience. Deep learning is a branch of machine learning that uses neural networks to solve complex problems. By using higher-level learned features defined

in terms of lower-level features, deep learning seeks to exploit unknown structures in the input distribution to discover good representations [24]. One of the Google Brain Project leaders, Andrew Ng, has compared deep learning with a rocket engine. With enormous amounts of fuel (data), the rocket needs powerful engines (deep learning models) to lift off the ground [25].

Before introducing the neural network that is used in deep reinforcement learning, some terms used in this section will be defined [26].

- **Perceptron** - A neuron in the human brain is a cell that transmits and processes information. Perceptrons are in many ways simplified versions of human brain cells that take several inputs and weigh them up to produce a single output. This was the first type of artificial neuron, but it is not so commonly used today.
- **Activation function** - To calculate the weighted neuron, an activation function is used. It calculates a weighted sum of inputs, adds bias, and from this information, it decides what should be fired to the next neuron. The most popular functions used today are Tanh, Softmax, and ReLU.
- **Gradient descent** - An algorithm to find the local minimum of a function. By guiding the solution in the direction of the steepest descent, it can be used to update the model's parameters.
- **Backpropagation** - Algorithm used to calculate the gradient descent. The goal is to minimize the error between the input and output, and backpropagation is used to train the neural network to an acceptable error margin.

Artificial Neural Networks

In the last decade, multiple types of neural networks have emerged, with different application features. For example, Convolutional Neural Networks (CNN) are often used in image processing, while Recurrent Neural Networks (RNN) have great ability in speech recognition. They are all a part of the broader family of Artificial Neural Networks (ANN). In this section ANNs will be defined, before the next section will introduce Deep Reinforcement Learning (DRL) by combining RL with the modelling

power of ANNs.

ANNs consist of artificial neurons, and have five main components: inputs x, outputs y, weights w, biases b, and an activation function $f(...)$. The relationship between these are given by

$$y = f(w^T x + b) \tag{2.1}$$

If this relationship gives 0 or 1 depending on the sign of the neural network, it is called a perceptron. However, since this means a slight change in the function's weights and biases can radically change the output, it is not commonly used. Other activation functions where small changes lead to a small output change are preferred.

When putting the artificial neurons into a network, this network is able to compute complex functions. Every ANN has at least one input and output layer and can also have multiple hidden layers. Hyperparameters are parameters where the value is set before the learning starts, and the number of neurons in each layer (width) and the number of layers (depth) are such hyperparameters. The main advantage with ANNs is the ability to reproduce and model nonlinear processes [27]. An example is shown in Fig 2.2.

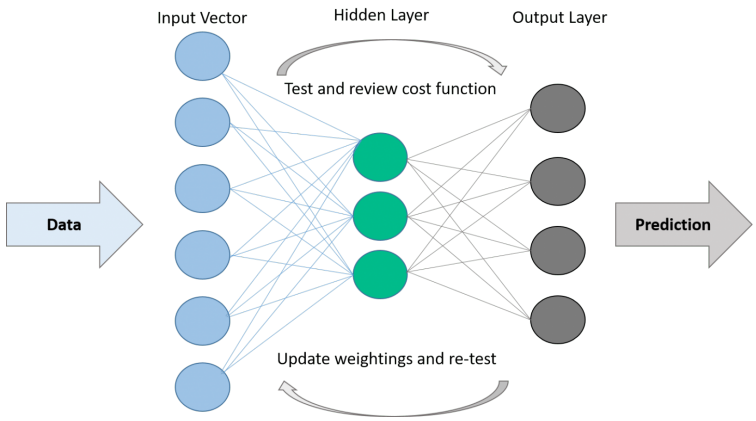


Figure 2.2: Artificial Neural Network example taken from [28]

Deep Reinforcement Learning

Robotic systems often have multiple joints, which means a high-dimensional state space. This is a challenge in reinforcement learning because the volume of the space increases so fast, hence the available data becomes sparse. The problem is called the curse of dimensionality, and it would take an enormous amount of computations, memory, and time to explore.

However, by combining RL with ANNs, a successful approach to this problem was discovered. The idea is to use a nonlinear function approximator to map state and action to a value. A Deep-Q-Network was first created by DeepMind in 2013 [29]. Deep-Q-Network is based on the Q-learning algorithm and learned control policies directly from a high-dimensional sensory input by using neural networks. To stabilize the training, a replay buffer and a target network were used.

- Replay buffer - By storing every transition between the explored samples in a replay buffer and sample a minibatch randomly in every update, the samples are independent of each other. This improves the generalization of the neural network.
- Target network - Instead of using a target that changes with every timestep, a network is used to minimize the target correlation. This makes the training easier.

2.4 Algorithms

Two DRL algorithms are used to train the different environments in this thesis. The trained model of each environment is used when implementing XAI-methods. Both algorithms presented below are temporal difference algorithms, meaning they are used to predict a measure of the total amount of reward expected over the future. Q-learning is often referred to as the most known model-free temporal difference DRL-algorithm [22]. Q-learning estimates the quality of an action that is taken to move to a state. It is a tabular method, which means that the learned Q-values are inserted into a state- and action space table, denoted by $|S|x|A|$. The update rule for the Q-learning algorithm is given by

$$Q(s,a) := Q(s,a) + \alpha\left(r + \gamma \max_{a' \in A(s')} Q(s',a') - Q(s,a)\right) \tag{2.2}$$

where α is the learning rate, r the reward and γ the discount factor. The algorithm is exploration insensitive under the assumption that a state-action can be visited infinite times so α can be decreased. This means it will converge to the optimal policy while following some exploration policy π [22].

2.4.1 Monte Carlo Policy Gradient (REINFORCE)

The REINFORCE algorithm, also called Monte Carlo Policy gradient, select actions based on a learned parameterized policy. It maximizes performance by updating a stochastic gradient ascent based on the policy gradient theorem. The actor parameter θ is updated using $G_t = \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}$ as an unbiased sample of $Q^{\pi}(S_t, A_t)$ [30].

$$\triangle\, \theta = \alpha * \gamma^t * \nabla_\theta \log \pi(S_t, A_t, \theta) * G_t \tag{2.3}$$

Because the algorithm relies on an estimated return by Monte Carlo methods, it plays out the whole episode to compute the total rewards. This means an entire episode is required before starting to train, which can be a challenge in many environments. For Cartpole, however, this is achievable and fast with a small state space. Another drawback is high gradient variance, which means lucky episodes can significantly affect the results, but this can be reduced by choosing an appropriate baseline. REINFORCE is an on-policy method, which means it updates the policy (Q-value) by using the next state and the current policy's action [31][32].

2.4.2 Deep Deterministic Policy Gradient (DDPG)

DDPG is an off-policy actor-critic DRL-algorithm. By being off-policy, it updates the Q-value using the next state along with a greedy action. Hence it can use samples generated from any time during training to optimize. Since DDPG is actor-critic, it trains two ANNs. An actor-network and a critic-network that approximates the policy and Q-value, respectively. The critic's role is to evaluate the performance of the actor, and DDPG is a suitable algorithm for robotics since it considers continuous states and actions. Two components are crucial for the success of this algorithm, a target network

and experience replay. The target network decreases the correlation with the target, which improves the training stability and slowly track the critic and actor networks by [9]

$$\theta^{Q'} \longleftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{2.4}$$

$$\theta^{\mu'} \longleftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{2.5}$$

where $\tau \in \mathfrak{R} : \tau \in (0,1), \tau \ll 1$

The replay buffer ensures that all the transitions are independent of each other and that previous transitions can be used multiple times. This is one of the benefits of off-policy algorithms. In [9], the novel technique Hindsight Experience Replay (HER) is used. It enables the RL-agent to learn from sparse rewards by the idea of substituting the actual goals with virtual goals. When using HER, the state-space is divided into two parts, one set of observation states and one for the goal states. In the Robotic Manipulator lever model, the "future" strategy of HER is used, meaning for every transition stored in the experience replay, k new versions of the transition are also stored. In these transitions, the goal states are substituted with randomly selected achieved goal states that were observed after the transition, and that came from from the same episode [32][9].

Pseudocodes of REINFORCE, DDPG and HER can be found here [30][33][34].

2.5 XAI Theory

The introduction of deep learning has opened up a new world in predictive modeling by making efficient decision-making following the works of the human brain. When these methods work well, it could be tempting just to trust the model and ignore why a particular decision was made. Until now, this has mostly been the case. Although major break troughs in complex machine learning, the models have been treated as black boxes.

However, knowing the 'why' can help learn more about the problem, the features and be helpful in possible error handling. In simple linear regression models, the number of parameters made it possible to explain the decisions made. With neural networks, this is much harder because of the enormous amount of connections. Even a narrow and shallow network can have tens of thousands of connections [9].

Explainable Artificial Intelligence (XAI) is a collective name for increasing transparency in AI-models. It can be divided into four different sub-fields [35]:

- **Interpretable models** - Linear/logistic regression and decision tree are commonly used interpretable models. From these decisions, rules can be extracted to say something about the feature importance.
- **Example-Based explanations** - Explain a model by selecting instances of the dataset and aim to represent them in a human-friendly way. They search for structures and more information within the data, which works well for images and text. Unfortunately, they are more challenging on tabular data where it is hard to represent it in a meaningful way.

  The two sub-fields above have many promising results within parts of the XAI-community [35]. Unfortunately, due to their limitations with tabular data from Deep Reinforcement Learning-models, they are not a part of this project.
- **Model-Agnostic methods** - Instead of selecting instances of a dataset, model-agnostic methods aim at creating summaries of features. This is done by manipulating perturbations of the neighborhood of data points. Moreover, by separating the explanations from the machine learning model, they offer great flexibility for

the developers. The theory behind model-agnostic methods will be presented below, and the methods LIME and SHAP are explained in the following sections.

- **Neural Network interpretation** - The amount of new deep neural network architectures in the last decade has exploded, and the trend is continuing towards even deeper networks with an increasing amount of weight parameters. Humans cannot follow the mapping from input to prediction when millions of mathematical operations must be considered. One way to do that is from the "outside," as with model-agnostic methods, but there are also benefits of using the neural network to increase transparency. First, uncovering features and weights directly from the hidden layers can theoretically improve the network. Secondly, by utilizing the gradient inside the neural network, it can be done in a more computationally efficient way than the model-agnostic approach [35]. The theory will be presented in Section 2.8, together with the methods Integrated Gradients and Saliency.

2.5.1 Surrogate models and local explainability

The idea behind Explainable Artificial Intelligence is to open up the black box by using some of the tools from simpler regression models. In such models, *Beta* coefficients are used to explain the prediction for all data points. This is called global fidelity; when a variable value increases by 1, prediction increases by *Beta* for every data point. Nevertheless, this does not explain the effect of individual data points; in other words, why the impact from one user's variable change could be different from another. This is called local fidelity, and local function explanations often have the property of linear and monotonic local regions. LIME and SHAP provide interpretability to black box models by exploring and use the property of local explainability. This is used to build surrogate models, which tweaks the input slightly and test how the prediction changes [36]. If the model prediction changes much by tweaking a variable value, that variable for that particular data point may be an essential predictor and vice versa.

Surrogate models still treat the system as a black box, which is called model agnostic methods since they are separating the explanations from the model (Fig 2.3) [37]. By exploring these local regions, the black box can be opened, and the difference between LIME and SHAP is how they build these surrogate models to make an interpretable data

representation. The following two sections will take a deeper look into both methods.

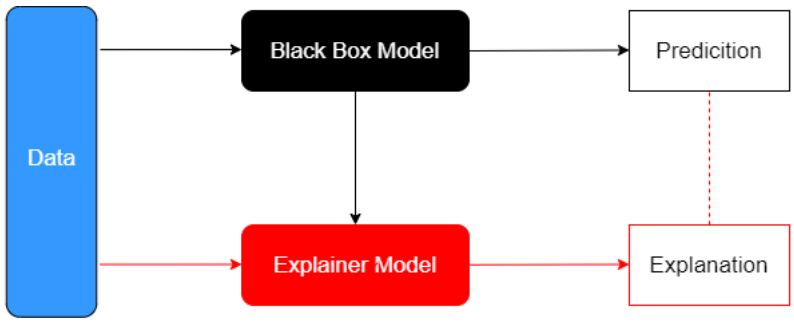


Figure 2.3: Model-Agnostic methods

2.6 LIME

The research paper *"Why Should I Trust You"* by Marco Ribeiro et al was one of the first to propose a technique to explain the black boxes of machine learning. It defines Local Interpretable Model-Agnostic Explanation (LIME) as *an algorithm that can explain the predictions of any classifier or regressor faithfully by approximating it locally with an interpretable model* [38].

Explaining a prediction means presenting artifacts, textual or visual, that improves the understanding between features (words, pixels, or robotic joints) and the model's prediction. Interpretable explanations need to present interpretations that are understandable by humans. While the classifier may represent more complex features, LIME uses a binary vector $x' \in \{0,1\}^{d'}$ to represent an instance, where $x \in \mathbb{R}^d$. The objective is to minimize the difference in prediction response between the instance x and its neighbor.

An explanation model $g \in G$ is defined as a class of potentially interpretable models, for example linear models (Fig 2.4) or decision trees, with the domain $g \in \{0,1\}^{d'}$. $\Omega(g)$ is the measure of complexity of the explanation $g \in G$ as not every model is simple alone to be interpretable. Examples of the complexity, $\Omega(g)$, can be the depth of the

three or the number of non-zero weights.

The model being explained is denoted $f : \mathbb{R}^d \rightarrow \mathbb{R}$, and $f(x)$ is the probability that x belongs to a certain class. To define locality around x, $\pi_x(z)$ is defined as a proximity measure between an instance z to x. The fidelity function $L(f, g, \pi_x)$ is a measurement of how unfaithful g is in approximating f in the locality defined by π_x. This means L must be minimized while $\Omega(g)$ must be low enough to be interpretable by humans. The LIME explanation is defined by

$$\xi(x) = arg \min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{2.6}$$

and this formulation can be used for multiple explanation models G, fidelity functions L, and complexities Ω [38].

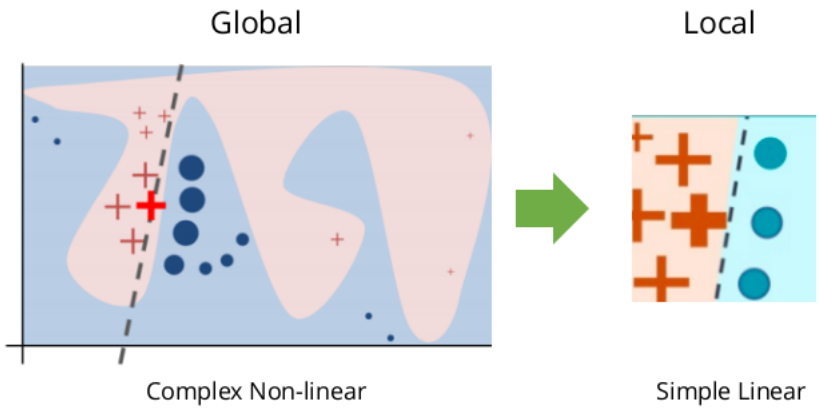


Figure 2.4: LIME localises a complex problem and defines a simpler linear model to explain a local prediction. Image taken from [39]

Lime Tabular

The LIME package is made for different datatypes. In this thesis, the environments generate matrix data, so the tabular method is used. The function explains predictions on numerical features from the training data. This is done by perturbing them, hence sampling from a normal $(0, 1)$, and doing the inverse operation of mean-centering and scaling. From this, neighborhood data is generated by randomly perturbing features, and a locally weighted linear model can be generated from a learned classifier. The models can be used to explain each of the classes in an interpretable way.

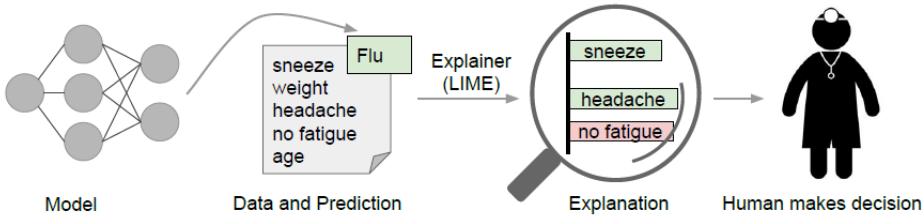


Figure 2.5: Explaining individual flu predictions with LIME [38]

2.7 SHAP

In the paper *A unified approach to interpreting model predictions* Scott M. Lundberg et al. proposed a new explanation method based on Shapley additive feature attribution methods [40]. This is also an approximation of the neural network when the model is too complex to be understood by humans. The algorithm used in this thesis is called DeepSHAP. This algorithm is based on two foundations, the DeepLift algorithm, and Shapley values, which will be presented first.

Shapley Values

As with LIME, the original prediction model is denoted by f and the explanation model by g. Simplified inputs, denoted by x', map to the original inputs using a mapping function $x = h_x(x')$. The goal of local methods is to make $g(z') \approx f(h_x(z'))$ given that $z' \approx x'$. This means an additive feature attribution can be defined

$$g(z') = \phi_0 + \sum_{t=1}^{M} \phi_t z'_t \tag{2.7}$$

where z' is a vector of binary variables with size M, M is the number of simplified input features, and $\phi_t \in \mathbb{R}$ is an effect that is assigned to each feature [40].

The method of finding the effect values ϕ is based on game theory. The Shapley value is a solution concept in traditional cooperative game theory to make a game "fair" according to the founder Lloyd Shapley [41]. To generate a total surplus of all players, four conditions must be met:

- The total reward should equal the sum of what everyone receives.
- The same amount of reward should be received from two people contributing the same value.
- No value contribution means nothing received.
- When playing two games, the individual's reward from both games should equal the reward sum from both the first and second games.

This can be transferred into three properties in Shapley value estimation:

- **Property 1 (Local accuracy)**

  $$f(x) = g(x') = \phi_0 + \sum_{t=1}^{M} \phi_t x'_t \tag{2.8}$$

  This means the explanation model $g(x')$ matches the original model $f(x)$ when $x = h_x(x')$

- **Property 2 (Missingness)**

  $$x'_t = 0 \rightarrow \phi_t = 0 \tag{2.9}$$

  Features missing in the original input have no impact.

- **Property 3 (Consistency)**
  If $f'(h_x(z')) - f'(h_x(z'_t = 0)) \geq f(h_x(z')) - f(h_x(z'_t = 0)), \forall z' \{0,1\}^M$, then $\phi_t(f', x) \geq \phi_t(f, x)$.
  This means that if a model changes so a simplified input's contribution increases, that input's attribution should not decrease.

Only one explanation model g follows definition 2.7 and the three properties above:

$$\phi_t(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f(h_x(z')) - f(h_x(z'_t = 0))] \tag{2.10}$$

where $|z'|$ is the number of non-zero entries in z' and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'. Equation 2.10 gives the solution of the SHAP values, where each value indicates how much a given state contributes to the magnitude of a given output (Fig 2.6) [40].

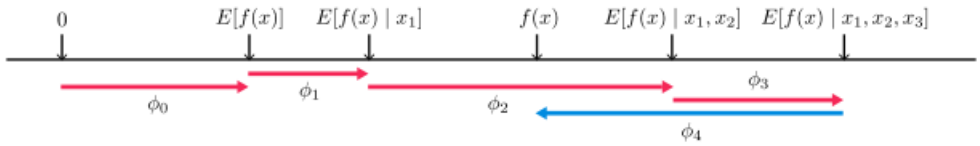


Figure 2.6: Shapley Values explain the output of a function f as a sum of the effects ϕ of each feature. Image taken from [42]

DeepLIFT

DeepLIFT is a recursive prediction explanation method for deep learning. With some modifications, this is an additive feature attribution method that converts binary values into the original inputs, $x = h_x(x')$. This binary value is decided by the effect of setting the input to the original value or a reference value that is decided by the user. This is denoted by $C_{\Delta x_t \Delta y}$, when $y = f(x)$ is the model output. DeepLIFT uses this in a "summation-to-delta" property

$$\sum_{t=1}^{n} C_{\Delta x_t \Delta y} = \Delta y \tag{2.11}$$

where $\Delta y = f(x) - f(r)$, $\Delta x_t = x_t - r_t$ and r the reference input. This matches equation 2.7 if we let $\phi_t = C_{\Delta x_t \Delta y}$ and $\phi_0 = f(r)$.

DeepExplain

It is challenging to compute exact SHAP values, but they can be approximated by combining insights from additive feature attribution methods. By assuming model linearity, the mapping can be approximated:

$$f(h_x(z')) \approx f([z_S, E[z_{\hat{z}}]] \tag{2.12}$$

where S is the set of non-zero indexes in z' and E the base value.

DeepExplain uses the connection between the DeepLIFT algorithm and the linear model approximation of Shapley values. By combining the two equations, and let the reference value from equation 2.11 represent $E[x]$ in equation 2.12, DeepLIFT approximates SHAP values.

This is equivalent to linearize the non-linear components of a neural network through backpropagation rules for each component. Furthermore, since DeepLift can be modified as an additive feature attribution method, it satisfies the properties of Shapley values. Thus, it motivates adapting a technique to approximate SHAP values for whole networks.

This method is called DeepExplain and combines SHAP values computed for smaller network components by recursively passing DeepLIFT's multipliers backward through the network (the composition rule). Fig 2.7 shows a simple component of a neural network, where the DeepLIFT approximation is given by

$$m_{x_j} f_3 = \frac{\phi_i(f_3, x)}{x_j - E[x_j]} \tag{2.13}$$

$$m_{y_i} f_j = \frac{\phi_i(f_j), y}{y_i - E[y_i]} \qquad \forall_j \in \{1, 2\} \tag{2.14}$$

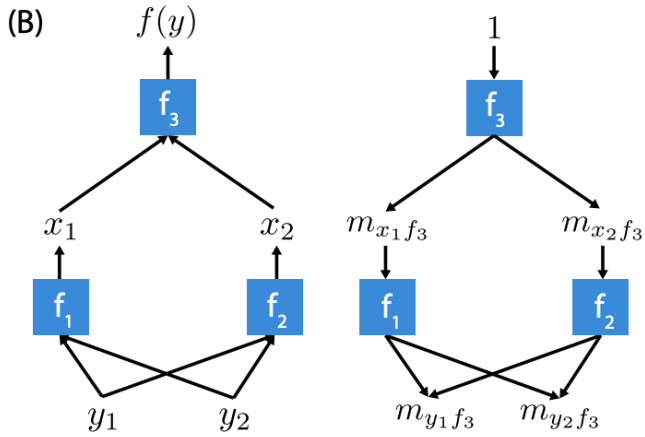


Figure 2.7: Neural Networks consist of many simple components, from [40]

$$m_{y_i f_3} = \sum_{j=1}^{2} m_{y_i f_j} m_{x_j f_3} \qquad \text{chain rule} \tag{2.15}$$

$$\phi_i(f_3, y) \approx m_{y_i f_3}(y_i - E[y_i]) \qquad \text{linear approximation} \tag{2.16}$$

Since such simple network components can be solved efficiently if they are linear and a deep neural network consists of many simple components, the composition rule enables a fast approximation for the full model. This makes it possible to explain deep neural networks efficiently [40].

2.8 Neural Network Interpretation

An attribution method assigns scores for each input feature, so it attributes input data based on predictions from the neural network. This means such a method can generate a score for each part of the input and say something about what part the scores played within a prediction [43].

2.8.1 Saliency Maps

Saliency Maps, also called Vanilla Gradients, was introduced in the paper *"Image-Specific Class Saliency"* in 2013 [44]. In simple terms, it calculates the gradient of the loss function for the score of interest with respect to the input. The size of the input features is then represented as a map that tells how much the prediction score would change with a slight increase in a highlighted prediction area [6].

The approach can be divided into three steps:

1. Perform a forward pass of the input
2. Compute the gradient of the class score of interest with respect to the input pixels

$$E_{grad}(I_0) = \frac{\partial S_c}{\partial I}|_{I=I_0} \tag{2.17}$$

3. The gradients can be visualized as absolute values or positive/negative contributions.

In many ways, Saliency maps are a simplified approach compared to Integrated Gradients (IG). However, one of its weaknesses is a saturation problem when the input gets capped at zero, and Integrated Gradients (IG) approach of capturing gradient information more globally makes them better at reflecting the importance of edges.

2.8.2 Integrated Gradients

The authors behind Integrated Gradients (IG) identified a shortcoming when attributing the prediction of a deep network to its input features in these attribution methods [45]. It was hard to separate errors from model misbehavior versus mistakes that stem

from the attribution method. Therefore, Integrated Gradients is based on an axiomatic approach, meaning it is based on desirable characteristics which increase the trust when attributing the correct scores to the right features. This also means Integrated Gradients can be computed independently of the network, using a few calls directly to the gradient operation.

In a simplified manner, Integrated Gradients is equal to multiplying the feature with the gradient. As explained in 2.4, during backpropagation in a neural network, a gradient tells the neural network how much a certain weight in the network should be changed. Therefore, each gradient associated with the input and output features can extract information about feature importance and hence more transparency into the network.

However, to avoid noisy gradients when the slope is zero, a baseline is often needed. This is especially important in object classification to avoid black images, but it will also make a big difference when working with tabular data. How to choose the baseline has been discussed and tested in multiple papers [43][45], where the findings indicated that a random baseline could be beneficial. When setting such a baseline, the focus should be on the interesting gradients shown in Fig 2.8:

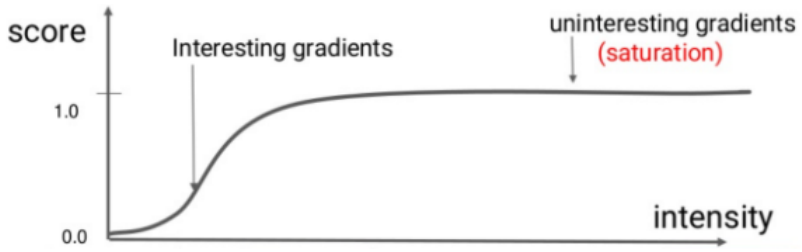


Figure 2.8: Integrated Gradients region of interest, from [46]

To understand Integrated Gradients, two axioms need to be described firstly.

- **Sensitivity** - A non-zero attribution should be given to every input or baseline that differs in one feature but have different predictions. This connects with the

idea of defining a baseline. When a function has a range between 0 and 1, and the input is bigger than 1, Integrated Gradients returns a non-zero attribution (1) instead of giving all differing features zero.

- **Implementation Invariance** - Two networks are functionally equivalent if all outputs are equal for all inputs. Attribution methods should satisfy implementation invariance; attributions are always identical for two functional equivalent networks, despite being implemented differently.

Integrated Gradients combines the axioms of Implementation Invariance and Sensitivity to produce explanations. A deep network can be represented by a function $F : R^n \longrightarrow [0, 1]$ The input is defined as $x \in R^n$ while $x' \in R^n$ is the baseline input.

By computing the gradients at all points along the straight-line path in R^n from the baseline to the input, IG can be obtained by finding the path integral of these gradients. Along the i^{th} dimension, it is defined as

$$IG(x) := (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{2.18}$$

where $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i^{th} dimension.

XAI methods' properties

All of the explainable methods used in this project are summed up below, together with their most essential characteristics.

Method	XAI field	Type	Scope	Package
LIME	Model-Agnostic	Perturbation	Local	LIME [5]
SHAP	Model-Agnostic	Perturbation	Local/Global	SHAP [4]
Integrated Gradients	NN Interpretation	Gradient	Global	Captum [6]
Saliency	NN Interpretation	Gradient	Global	Captum [6]

Table 2.1: XAI methods characteristics summarized

Chapter 3

Methodology and experiments

A total of four methods have been used to explore Explainable Artificial Intelligence on robotic systems. Both LIME and SHAP are relatively recent methods that have mainly been used to explore datasets and images. These were also compared in the author's pre-project, where three environments differed in state space, degree of explaining difficulty, and the amount of intuitive interpretation. They were provided to give local and global explanations for pre-trained solved agents to find possible indications of the approximations and assumptions done by these methods. Two of these environments are also included in this thesis, the least complex one (Cartpole) and a new version of the most complex one (Robotic Manipulator) with a lever task. In addition, another type of gradient method, Integrated Gradients, has been added to explore the global interpretations of the environments. Pytorch's explainability package, Captum AI, was used to implement the method. This package enabled further implementations of another gradient method, Saliency, and also some additional weight analysis.

In this chapter, the two environments will be presented together with the software used in the implementation. Towards the end, an overview of how the XAI methods were implemented will also be presented. The functions and different plot types will be reviewed to give a more straightforward interpretation of the results in the next chapter.

3.1 Software

PyTorch

PyTorch is a Python library for deep learning. It supports multiple features such as GPU for parallel computing, intuitive setups for standard neural network techniques, and effective debugging procedures. In addition, PyTorch's ability to support dynamic computation graphs makes it convenient and flexible to use compared to other deep learning frameworks such as Tensorflow, which uses static computation graphs [3].

Anaconda

Anaconda is an open-source distribution of Python for scientific computing that simplifies package management, and distribution [47]. Anaconda is used together with Jupyter Notebook, an open-source web application that contains live code and visualizations. This is useful when dealing with multiple environments and libraries where dependencies can arise, and it also makes it easier to explore explainability in pre-trained agents.

OpenAI Gym

OpenAI Gym is an open-source toolkit for developing and comparing reinforcement learning algorithms. It aims to provide an easy to set up and standardized environment so that published research becomes more easily reproducible [1][48]. This also means that RL algorithms can easily be adapted between environments. The environment used in this project is *CartPole-v1* from the classic control package, which consists of classic RL literature control theory problems. The main functions that need to be defined for Gym environments are [1]:

- make(environment_name) - Sets up a new instance of the environment and returns an object of the class.
- step(action) - Applies a step to the environment on the action used as the argument. Returns an observation of the environment, the transition reward, possible terminal state, and a dictionary of diagnostic information specific to the environment.

- reset() - Resets and then returns the first observation of the reset environment.
- render() - Visualizes the environment by rendering at each step. Used in the simulator implementation in Section 4.4 together with a Bar Chart Race package [49].

PyBullet

Robotic models trained with reinforcement learning is one of the main use cases for the PyBullet simulator [50]. PyBullet has several features that make it well suited for the Robotic Manipulator, as explained in Sindre Remman's master's thesis [9]. It has a built-in step function, which moves one time-step forward in the simulation when called. This functionality is well suited for reinforcement learning since the assumption is that the environment does not change without the agent performing an action. It is also a high-speed simulator compared to many of the alternatives, for example, Gazebo [51].

3.2 Environments

Cartpole-v1

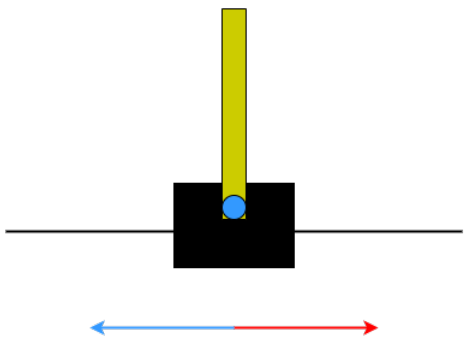


Figure 3.1: Cartpole schematic drawing

The *Cartpole-v1* environment is similar to an inverted pendulum with a gravity center above its pivot point. An un-actuated joint attaches the pole in Fig 3.1 to a cart, which moves along a frictionless track. The goal is to prevent the pendulum from falling over, defined as more than 15 degrees from the starting upright position. It is also not allowed to move the cart more than 2.4 units from the center [1]. A +1 reward is awarded for every time-step the pole remains upward (non-terminal step), and the pole is controlled by applying a discrete action, a force pushing the cart to the left or right (-1 or +1). The state-space for the Cartpole is given by

Nr	State	Min	Max
0	Cart position	-4.8	4.8
1	Cart velocity	-Inf	Inf
2	Pole angle	$-24°$	$24°$
3	Pole velocity at tip	-Inf	Inf

Table 3.1: State space Cartpole-v1

Cartpole is defined as solved when getting an average reward of 195.0 over 100 consecutive trials. In this project, the environment is solved using the REINFORCE-algorithm from [52]. This is a policy gradient method that solves the problem at around 500 episodes. The solution is stored in a checkpoint file that is used when implementing the XAI methods.

Robotic Manipulator

The second environment used in this project is a continuation from Sindre Remman's master's thesis where a robotic manipulator (OpenMANIPULATOR-X by Robotis [2]) is trained using deep reinforcement learning [9]. The manipulator has four revolute joints and a lever, which means the total number of degrees of freedom is five. In one of the tasks, *lever manipulation using DDPG*, the goal is to move the lever to a randomly selected goal angle.

If $|\theta_{lever} - \theta_{goal} < 0.025|$, it is classified as a success and a sparse reward is given. To make training faster and because the angle is trivial to find, the agent is restricted from not moving the first joint. This means the action space consists of the desired relative

angles of the three remaining joints and the choice of opening the gripper, a total action space of dimension four. The lever was created by Sindre in [9], using the open-source 3D graphics software Blender. This was transferred into the Pybullet simulator together with the manipulator by creating a mesh for the lever as a Unified Robot Description Format (URDF) model. The manipulator with and without the lever is shown in Fig 3.2 and Fig 3.3.

The manipulator consists of 19 observation states from the environment. These are angles and velocities from joints, Cartesian positions of the lever's base relative to the manipulator's base, the relative distance between the end-effector and the lever's base, and the current angle of the lever. In addition to this, the desired lever angle is defined as the goal state and is divided accordingly to the HER-procedure described in 2.4.2. This means the total state-space is of dimension 20. In the later stages of the project, the state-space was cut in two based on explaining the most influential features and trained in a new model with less complexity. These are described in the result part *Data Adaptions* in Section 4.2.

The explainable methods are run from a dataset that consists of 25 test episodes after the training procedure is completed.

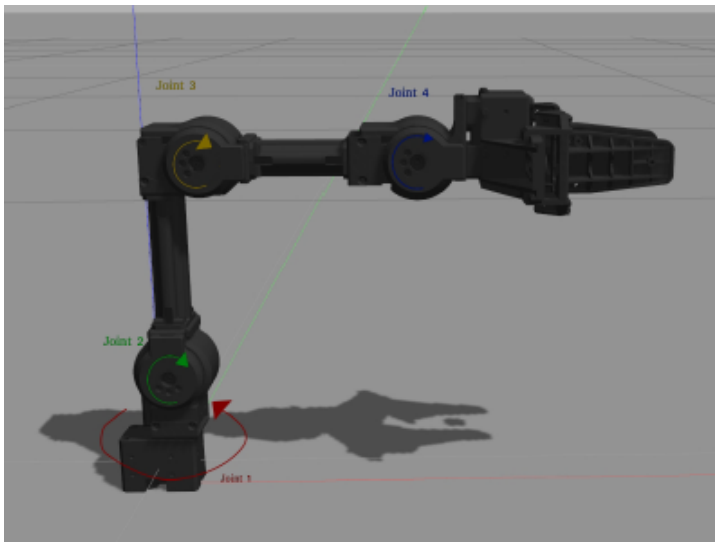


Figure 3.2: Robotic Manipulator scheme, from [9]

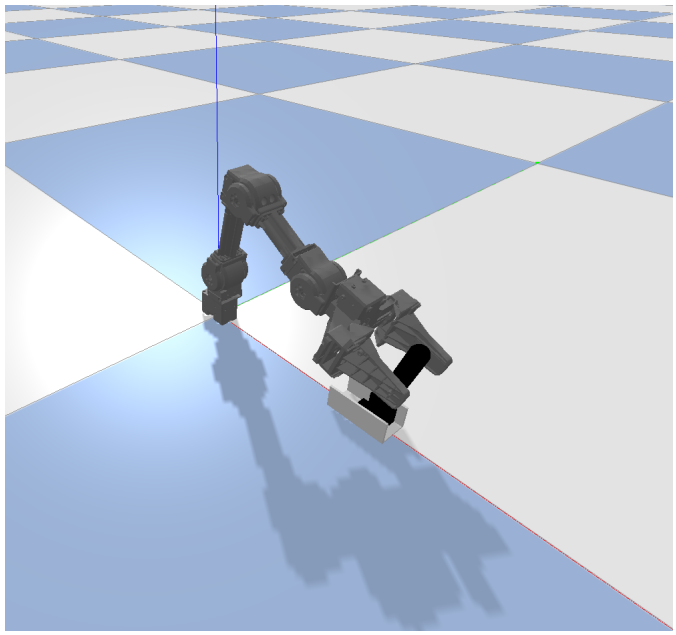


Figure 3.3: Robotic Manipulator Lever model, as seen in the Pybullet simulator

3.3 Methodology and XAI implementations

This project's scope is to implement XAI methods across the two environments presented above to investigate if these methods can produce plausible interpretations in robotic environments.

The research structure is built upon an Anaconda environment with all necessary libraries activated, and the implementation is done in Jupyter Notebook with Python 3.8 and Pytorch. After training the environments with DRL-algorithms, checkpoint-files, or datasets are saved, which has been beneficial so they can be used directly in further developments. A user-defined number of solved episodes are used in the XAI-methods. The explanations are also saved with checkpoints to be interpreted across multiple states locally or during the training phase and visualized in different procedures.

The framework for this research was set up based on being as flexible as possible. One of the reasons for this was the uncertainty from the Covid-19 pandemic, meaning it was beneficial to be able to work from home. Also, with the perturbation and gradient approaches in the XAI-methods used, the explanation methods do not have a direct influence on the DRL-algorithms. This means the setup is extendable to other problems, and one change can be transferred between the environments.

An overview of the XAI-implementation is presented below, including functions and helpful plot descriptions used from the libraries.

SHAP implementation

The SHAP DeepExplain function is used to get an explainer object. The function takes a neural net model from Pytorch together with a background dataset from the trained episodes. The explainer object is used in the ShapValues function together with the test state to approximate the SHAP values for the deep learning model. This can be done locally with one state or over multiple episodes.

Two plot types from SHAP are used in this project in addition to an episode plot made from the SHAP and LIME explanations.

- **Summary plot** - Shows a summary of the feature importance, how much impact

each feature has on the model output across the different actions. The parameters are the SHAP values, a NumPy array of the features, and a parameter for how many included features. Summary plots are shown throughout the result chapter, for example in Fig 4.4.

- **Local Force plot** (Fig 3.4) - The force plot visualizes the predictions in an additive way. The function takes in a reference (base) value that the feature contributions start from, and *"arrows"* are used to show if the predicted value is pushed higher or lower. The bold number on the number line shows the predicted value. The base value is the average model output over the training period. Features contributing to pushing the prediction higher are shown in red. Features pushing it lower appear in blue. The SHAP values and the test state are inputs here as well. The logit operator is used to convert the output values from log-odds to probabilities.

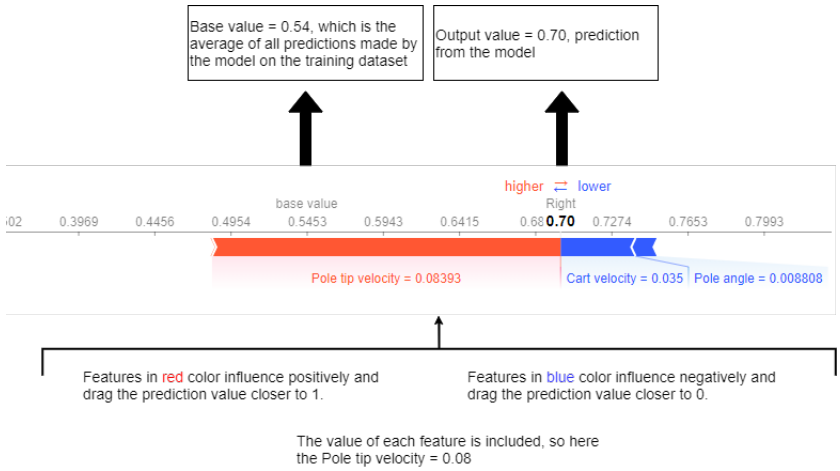


Figure 3.4: Example of a Force plot

- **Global Force plot** - The global force plot is a merging between the summary plot and the local force plot. It shows the most influential features across a full episode for one specific action, where the x-axis shows the state number and the y-axis the SHAP-magnitude. As with the local force plot, the blue color indicates not choosing that action, while a red feature color will take that action. The higher

the feature height, the more influence across that specific state and also across the episode globally. Two global force plots is included in the result section, Fig 4.10 and Fig 4.18.

- **Episode plots** (Fig 3.5) - This plot type is also included in this section, although it is used to produce both SHAP and LIME-explanations in the last result part section when implementing "real-time" analysis. The idea behind this episode plot is to visualize each feature throughout a full episode. The feature value is plotted as a light blue line while scatter points are plotted above each state. The color of the scatter point indicates a positive (red) or a negative (blue) influence from the XAI-explainer of that specific feature. The intensity of the points can be transferred to the magnitude of the SHAP or LIME values, meaning a sharp red scatter point has a high magnitude while a grey scatter point has no significant influence. An example done with SHAP-values for two of the features in the Cartpole environment is shown below in Fig 3.5

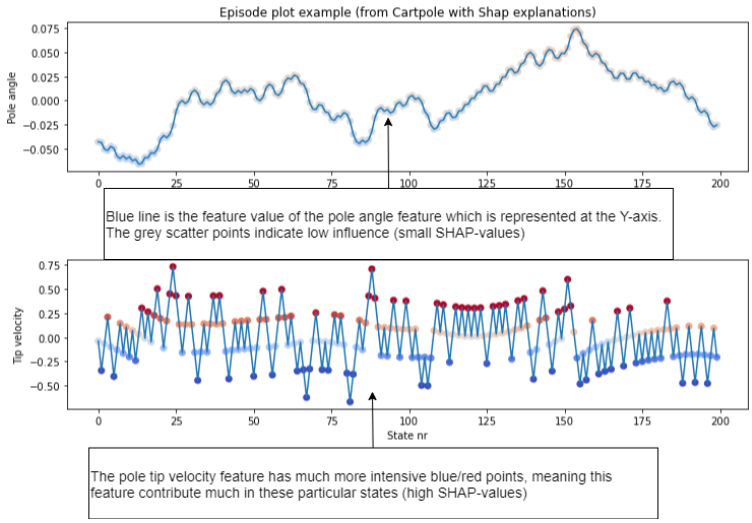


Figure 3.5: Example of an Episode plot

LIME implementation

The LimeTabular function has been used from the LIME package, and the function takes a background training dataset to produce a LIME explainer object. This function also has some hyperparameter adjustments available. The kernel width adjusts the exponential kernel of the neighborhood sampling. In this project, the author has used the standard value, $\sqrt{col_{nr}} * 0.75$. It is also possible to test different discretize options. It was decided to use quartiles, as it gave the most stable and plausible results. This means all non-categorical features will be discretized into quartiles [53].

The explain instance function is used to generate explanations for a local prediction by using the explainer object. This function generates neighborhood data by randomly perturbing features from the test state, one of the parameters. This is used to learn locally weighted linear models on this neighborhood data to explain each class in an interpretable way. The function uses a prediction function that outputs prediction probabilities for a given state. Defining the prediction function across two different environments was one of the challenges encountered in this project. An explanation object is returned with the corresponding explanations and can be visualized, showing prediction probabilities and feature importance for the local prediction.

Captum implementation

Captum is a new model interpretability toolbox for Pytorch [6]. It supports most types of Pytorch models, meaning the agents in this project can be modified to fit the methods in the library. As an open-source library for interpretability research, it includes several XAI-methods, mostly attribution methods that predict the importance of input features to Neural models. Integrated Gradients, Saliency, and also some weight analysis have been implemented from the Captum package in this project.

A wrapper function was made to fit the agents from the two robotic environments into the Captum methods. This wrapper function changes the shapes of the network correctly, so the gradient methods fit with the correct dimensions. It is mostly just a workaround to connect an advanced Pytorch agent with the Captum library, so multiple methods can be run directly by just calling the function. For Integrated Gradients (IG),

multiple baseline choices exist, and four different ones were tried. These were mean-, zero-, one- and random-base, where the last one was used. This was both because it was an advisable suggestion from the original authors in [45], and the observed results looked most promising when comparing the different baselines. However, there are also disadvantages of choosing this baseline, which will be discussed in the next chapter. Saliency has no baseline choices so it can be implemented directly using the respective Captum Saliency function together with the wrapper function. The learned model weights can be extracted from the Pytorch agent by using the *linear.weight* function. The Captum plots included in this thesis are standardized bar plots comparing the feature attribution across different XAI-methods.

Chapter 4

Results and Discussion

The result section is divided into four parts. The two environments, Cartpole and Robotic Manipulator are included throughout these sections. Each part starts by describing the problems to be addressed in connection with the research questions of this project. It also presents an outline of the section so that the reader can navigate the different situations and plots, followed by a discussion about the results in the end. The implementation procedures are presented in Chapter 3 together with an explanation on how to interpret the visualizations from the XAI-methods.

A big challenge with DRL-models is that no simple answers about the model's decision exist. In many ways, it is comparable to human reasoning, and interpretation is based on intuition. When implementing XAI-methods in such environments, it can be hard to separate between the performance of the agent versus the correctness of the explainer. The goal is to trust both of them, meaning more transparency instead of a black box architecture, to see if these results can increase trust instead of more uncertainty.

The goal of this section is to increase the trust of the DRL-agents by looking for trends throughout multiple robotic scenarios and XAI-methods. The advantage of this is that a great selection will hopefully lead to a better understanding of how the methods and models work. A possible disadvantage is the need for many different plot types in

this section, where each method has its way to present the data where a large amount of information can be extracted. The author would like to emphasize that all of this info will not be commented on in the result section. It will mainly be used to form a basis for answering the research questions.

4.1 XAI method comparison

In the author's project thesis, the model-specific XAI-methods LIME and SHAP were implemented to produce local and global explanations. The method comparison section is an extension of this project, adding two more methods (Integrated Gradients and Saliency) from another XAI-subfield called neural network interpretation. These are global methods so that they will be compared against the global SHAP-analysis. In addition to this, the weights of the model's neural network have also been included.

The section is divided into local and global explanations for both environments. Explanations from the Cartpole environment, with four states and two actions, will be presented firstly, followed by the more complex manipulator explanations.

4.1.1 Results

Local explanations Cartpole

One local state, from now on called *situation* is used to visualize local explanations, shown in Fig 4.1. The key for Cartpole is using inertia and acceleration to balance the pole. In this example, the Cartpole has a negative (left) *pole tip velocity*, meaning the intuitive action can be to push the cart to the left to stabilize the pole in the other direction. However, the *cart velocity* is also negative, meaning to slow the cart down by pushing right could also be considered. LIME explanations are shown in Fig 4.2. The LIME output consists of prediction probabilities, explanations comparing actions, and the feature values in the example state.

The local SHAP explanations are illustrated by a Force plot (Fig 4.3). In the Force plot, the red "arrows" illustrate a push to the right, and the blue "arrows" illustrate a

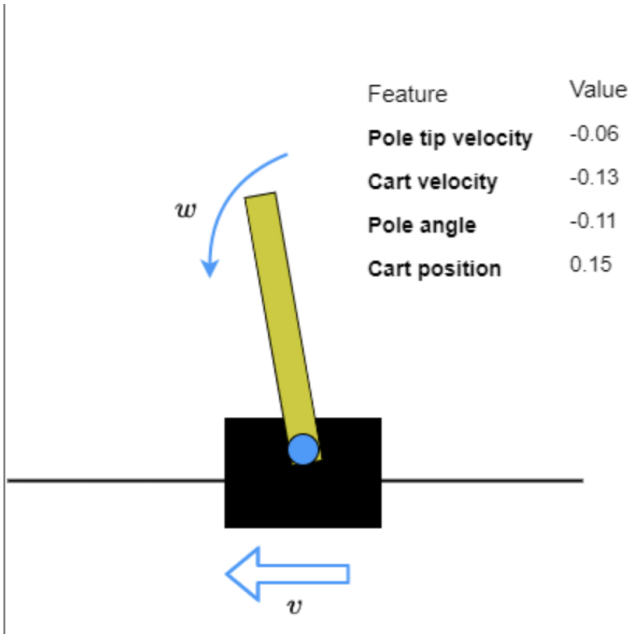


Figure 4.1: Cartpole: Schematic figure for Situation 1 with feature values

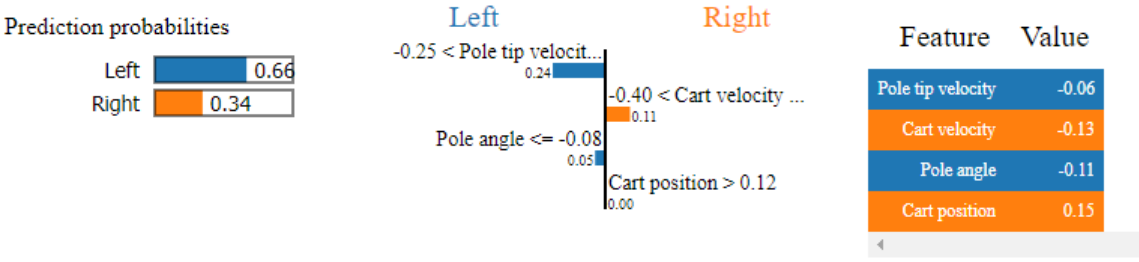


Figure 4.2: Cartpole: LIME Local explanations for Situation 1

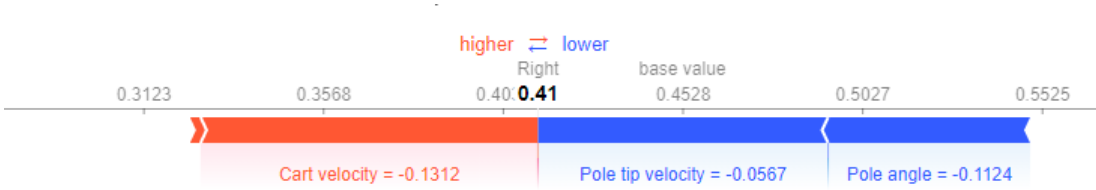


Figure 4.3: Cartpole: SHAP local Force plot situation 1. Baseline 0.45, Red "arrow": push in the right direction, blue: push in the left direction

push to the left. DeepSHAP uses the model instead of a prediction function, which means the explanations are presented in SHAP-value magnitude. However, here the logic operator is used to transfer these values from log-odds into probabilities. The base value in the SHAP plot is 0.45, which is the expected value of choosing to push the cart to the right overall, based on the training episodes. In this case, ten solved episodes have been used for training.

As seen in Fig 4.3, SHAP proposes the action of pushing the cart to the left, but the value of 0.59 ($1-0.41 = 0.59$) is lower than LIME's 0.66 in Fig 4.2. Both methods highlight the same features in the same directions, which also makes sense when looking at the feature values of the example state. The difference in the prediction probability can be because LIME emphasizes the *pole tip velocity* value bigger than SHAP. This could be because of the approximations in DeepSHAP or the neighborhood sampling of the LIME method and will be discussed more thoroughly below. The common pattern when testing these two methods in the Cartpole environment was mostly similar predictions, but SHAP tending to be a bit more conservative in the estimations.

Global explanations Cartpole

To produce global SHAP explanations, data was collected from ten solved episodes. This means 2000 states are used when training the explainers. Calculating global SHAP values took 2-3 minutes on the school computer while calculating the gradient methods was done within a couple of seconds.

The global explanations highlight the most important features over these solved episodes. The Summary plot of SHAP (Fig 4.4) shows how much the different features contribute to the magnitude of the different actions.

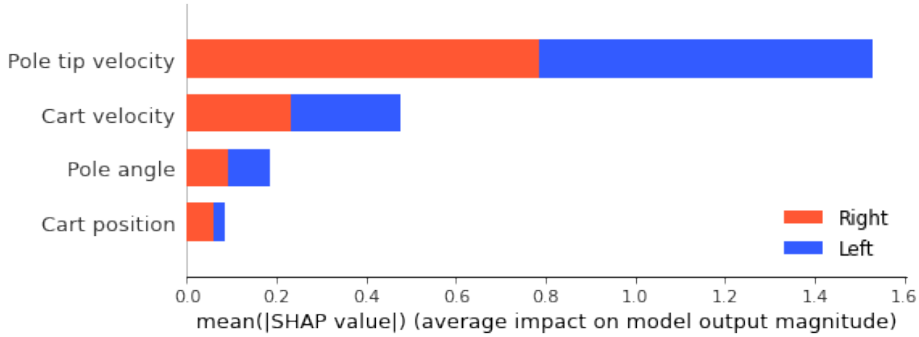


Figure 4.4: Cartpole: SHAP Global Summary plot over 10 episodes

Integrated Gradients is shown as the pink bar chart in Fig 4.5. It only shows the magnitude of each feature independently of which action to choose. Both SHAP and IG highlight *pole tip velocity* as the dominant feature and *cart position* as the one with the least impact. However, the methods disagree between *pole angle* and *cart velocity*. This pattern was discussed in the pre-project, where the *pole angle* had a much higher relative value in a Morris analysis [7]. Integrated Gradients reflects the indication that SHAP possibly underestimates the magnitude of the *pole angle* because of small feature changes in the training episodes. One theory is that the perturbations of the model-specific method would not pick this up while the gradient discovers this phenomenon. The pattern is interesting and will be further analyzed in the discussion of the Data Adaption part in Section 4.2.2.

The Captum plot in Fig 4.5 also includes the methods of Saliency and weight analysis. The attributions of IG and Saliency are primarily similar for all features, with Saliency having, in general, an overall higher magnitude. However, some interesting notes from the weight plotting are that some of the least influential features, according to the explainers, have a certain weight attribution.

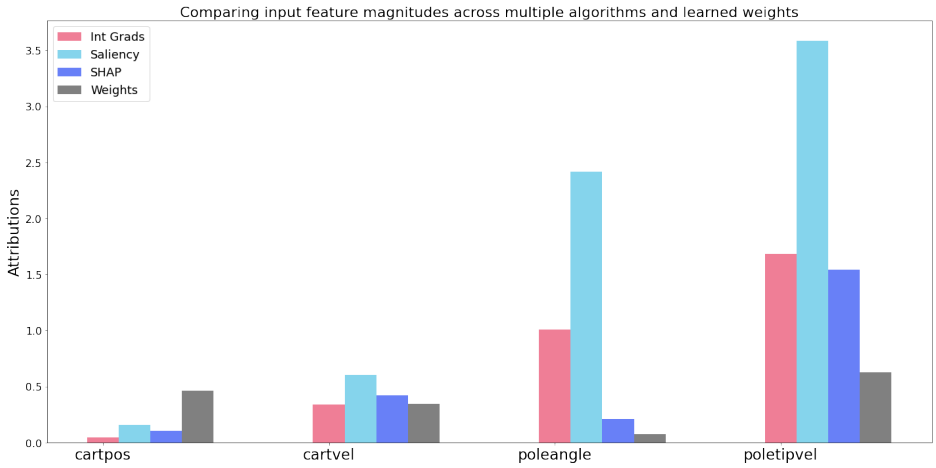


Figure 4.5: Cartpole: Captum Global methods attributions

Local explanations Robotic Manipulator

In the author's pre-project, local SHAP and LIME explanations were used to interpret results on a Robotic Manipulator reacher task with 15 states and four actions. These explanations have been extended to a lever manipulation task with 20 states and four actions in this project. In the situation used, the agent is just a single step away from the goal state (Illustrated in Fig 4.6) in the test episode, with four training episodes used to train the explainers. Local explanations for two of the joints (Joints 3 and 4) are included in this section, where the LIME explanations are shown in Fig 4.7. SHAP explanations are shown as Force plots in Fig 4.8.

The features of the lever angle have the most significant impact from both explainers, which can be explained in regards to the manipulator being only one step away from the goal. However, there are some disagreements between the methods about the other features, and none of them give a clear interpretation of this specific situation nor which action to choose. Only two of the four actions have been included here, but the same trend is present for the remaining actions and throughout other test episodes.

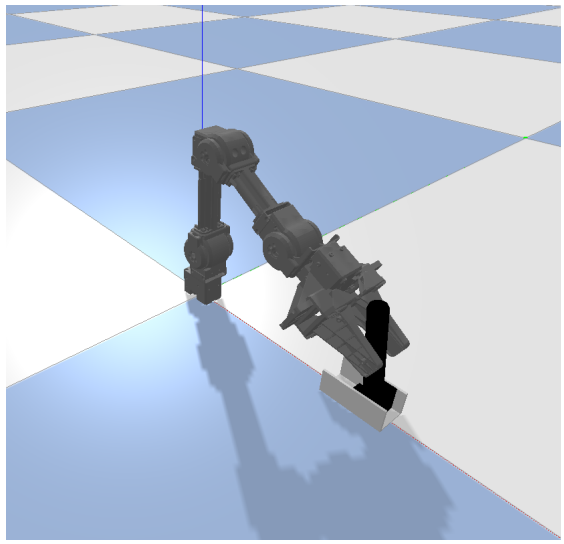


Figure 4.6: Robotic Manipulator: Illustration of Robotic Manipulator being close to finishing the episode. Picture taken from Pybullet simulator

Global explanations Robotic Manipulator

The global explanations of the lever model are obtained using 25 solved episodes. For the SHAP-explanations, 24 episodes were used to generate the explanation model, and the remaining episode explained. This means 1200 states are used for training. For the Captum methods, the explanation model is also generated using the same episodes. The data for these explanations are generated from a normalized dataset from the real-world manipulator.

The global explanations from SHAP are visualized in two ways. One Summary plot in Fig 4.9 shows all SHAP magnitudes of the features across the four actions. As with the local explanations, the *goal lever angle* and *current lever angle* are the most influential features according to the explainers. The agent aims to move the current lever angle towards the goal lever angle, so this is trustworthy. These angles are used as an explanation both for and against doing the specific action. For example, the position of the *current lever angle* is an argument for moving Joint 3 while the *goal lever angle*

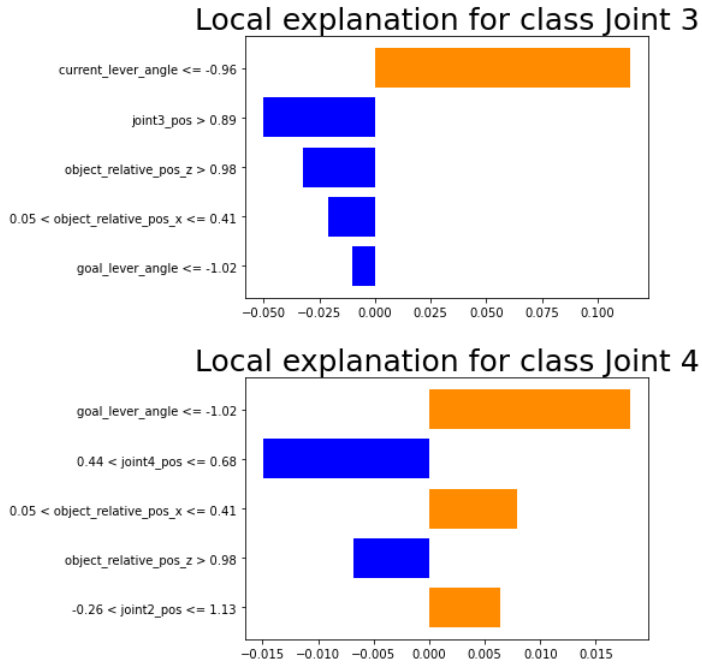


Figure 4.7: Robotic Manipulator: LIME Local explanations Joints 3 and 4

Figure 4.8: Robotic Manipulator: SHAP Local explanations Joints 3 and 4

pushes in the opposite direction. It can also be observed that moving Joint 2 has a more significant impact than the other actions. However, this varied depending on the choice of the test episode. If one action has an unexpectedly significant impact, it can indicate a place to start looking for a failed operation. On the other hand, considerable changes between different test episodes can also suggest that too few training episodes are used when training the explainer, which could be observed in the local explanations where only five training episodes were used.

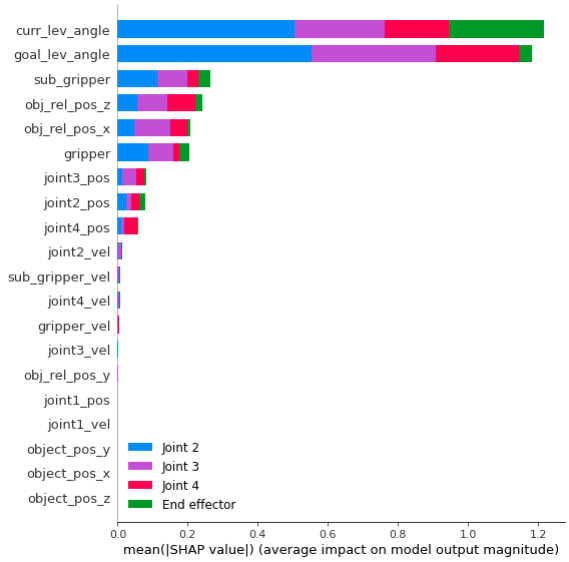


Figure 4.9: Robotic Manipulator: SHAP Global Summary plot

The Summary plot shows that under half of the features have a significant impact on the explainer. This was the motivation behind using these global plots to reduce the model to fewer trained features and see how this affects the interpretations. The results from these operations are presented below in Section 4.2.1.

The Force-plot of SHAP can also be used to show the features of the highest magnitude throughout the test episode. As seen in Fig 4.10, it confirms the lever angle importance. Without these two features, accomplishing the episode's goal for the agent

will be virtually impossible, so this is a positive sign. A reason behind the explainer not proposing the *current lever angle* action (blue color) early in the episode to proposing it (red color) towards the end is that the manipulator is getting a grip of the lever. Then these two states would try to equalize each other.

The magnitude of the relative z and x distances is also one of the more influential features across these two plots. This makes sense since, combined with the *current lever angle*, these two states can be used to calculate where the end-effector is relative to the lever. On the other hand, it would be expected that the angles of the manipulator's joints contribute more towards the actions.

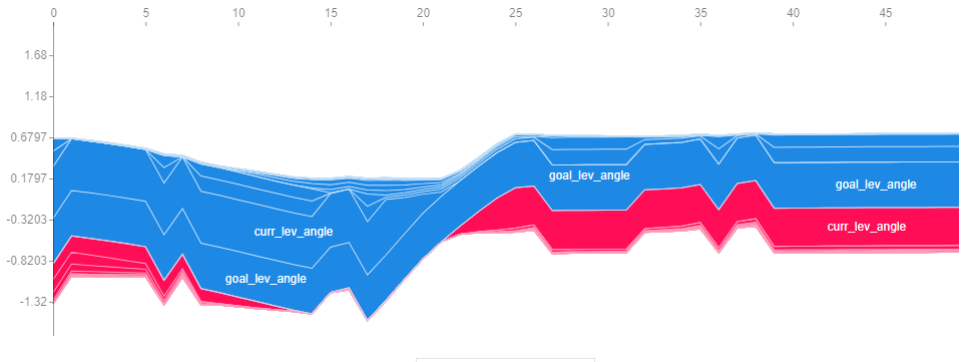


Figure 4.10: Robotic Manipulator: SHAP global Force plot action 1 (Moving joint 2). SHAP-values on the Y-axis vs State number on the X-axis.

Many of the same observations can be seen from Integrated Gradients in the Captum plot in Fig 4.11. The lever angles have feature importances much higher than the rest. Most of the features also have no significant impact, which further amplifies reducing the state space. It can be noted that IG has, on average, pretty similar attribution values compared to SHAP. Because of the big state-space, the Captum plots comparing the gradient methods are divided in two (Fig 4.11 and Fig 4.12). Saliency and Integrated Gradients follow the same trends here as well, but as opposed to the Cartpole environment, Saliency does, in general, have a lower overall magnitude contribution. The weights from the relative objective positions are influential in these two plots. However, the phenomena regarding the small influence from the joint angles are also present across the gradient methods.

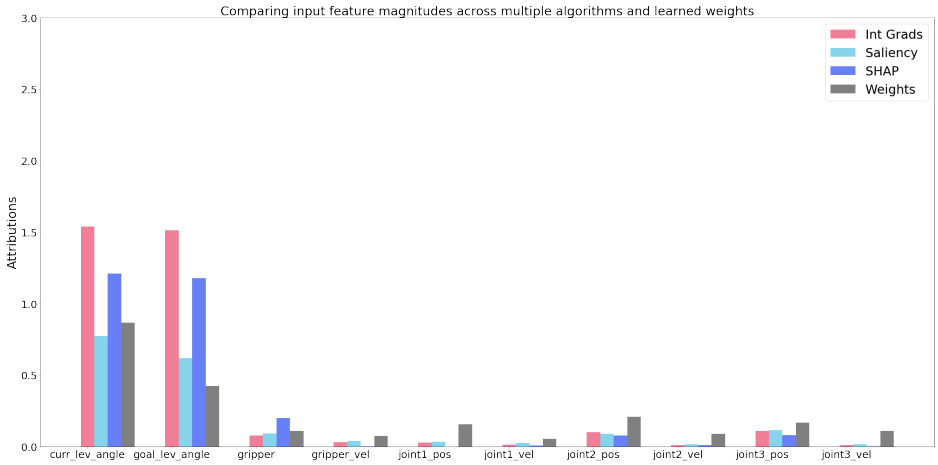


Figure 4.11: Robotic Manipulator: Global Captum attributions top 10 features

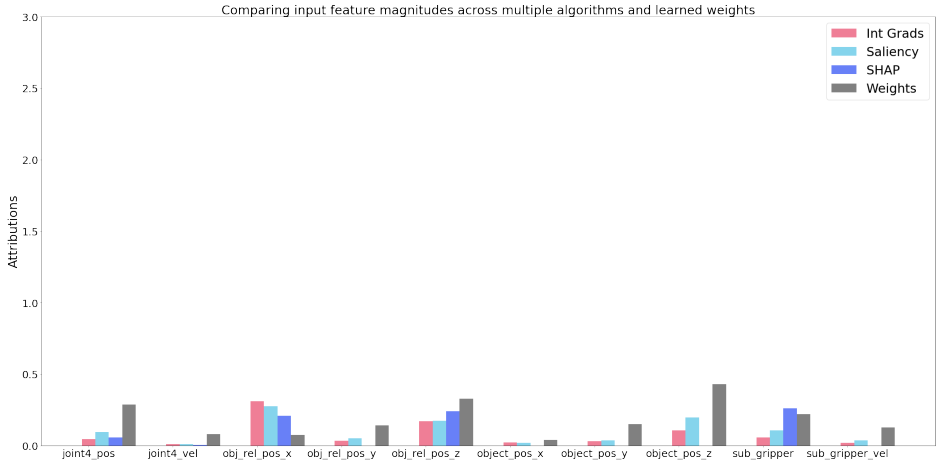


Figure 4.12: Robotic Manipulator: Global Captum attributions remaining features

4.1.2 Discussion

In the author's pre-project, the explanations from the model-specific methods SHAP and LIME was discussed. With the addition of the Captum package, this has been extended with global explanations from Integrated Gradients and Saliency. Two DRL-models have been used to explore the performance of these XAI-methods on robotic environments. The results in Section 4.1 above show that the methods produce quite similar explanations across both environments. From the author's perspective, the Cartpole environment results often coincide with the predictions a human would have made in the same situation.

When increasing the dimensionality, a complex state space and multiple actions make it difficult to get a clear sense of the specific situation. Working with the Cartpole environment, the simpler state-space gives room for better human-machine interaction with the interpretations. Some trends about the Robotic Manipulator can be discovered locally and possibly say something about a possible failure. It is still possible to detect some of the main characteristics of the agent, and the most influential features and their change across an episode are often in line with the expectations in advance, which means the trust of the model increases. On the other hand, it could also lead to an even

higher degree of confusion about the black box. This was one of the main concerns in the pre-project and one of the motivations for exploring feature minimization and real-time analysis in the following sections.

Multiple issues with the XAI methods used in this project could also amplify the agents' disconcerting factors when the problems just as well could arise in the explainers. This is the motivation behind comparing these XAI-methods, focusing on what their characteristics and theoretical foundation mean for the explanations observed above. This will be based on the models and literature available [35][54][55][56][57].

SHAP

- The SHAP framework is based on solid theory. The properties from game theory give the algorithm a good foundation, and the efficiency axiom makes the difference between the prediction and average prediction fairly distributed among the feature values. This could emphasize the fact that SHAP results were more stable throughout these experiments. However, SHAP-values can be misinterpreted since it explains the contribution of a feature value to the difference between the actual prediction and mean prediction. This could make it more challenging to compare directly with other methods.

- SHAP is able to produce both local and global explanations. This is a huge benefit when analyzing deep learning models, and both local and global insights could be beneficial. Producing global explanations takes some time, though, especially with many training episodes. The weighting when getting the SHAP values for local explanations could be affected by the approximation in DeepSHAP. Even though it fulfills the Shapley value properties, it could miss some of the actual behavior of the model in the local neighborhood. When using the model to make these predictions, the XAI-method becomes by definition model-specific instead of model-agnostic where the neural network is decoupled. This means the agent is used directly to produce the explanations in a post-hoc matter, but it also makes it prone to adversarial attacks when making these approximations as shown in [58]. This could explain some of the differences between SHAP and LIME's local

explanations in the previous section.

- The assumption of feature independence is the elephant in the room of this project. Shapley value methods suffer from the inclusion of unrealistic data instances when features are correlated. Marginalizing the features is fine as long as the features are independent. When they are dependent, feature values that do not make sense might be sampled. Fig 4.13 shows the correlation over ten episodes in the Cartpole environment, while Fig 4.14 does the same over 25 episodes for the Robotic Manipulator. The findings from the pre-project were that even though the correlation between some of the features is high in the Cartpole environment, the explanations are still plausible. In the complex manipulator environment, this is still a challenge. From the values in Fig 4.14, this could also explain why the lever angles have such a significant impact because it correlates highly across all the other influential features across the training episodes. With a complex feature space, perturbations or highly correlated feature pairs could influence the predictions from the SHAP-explainer.

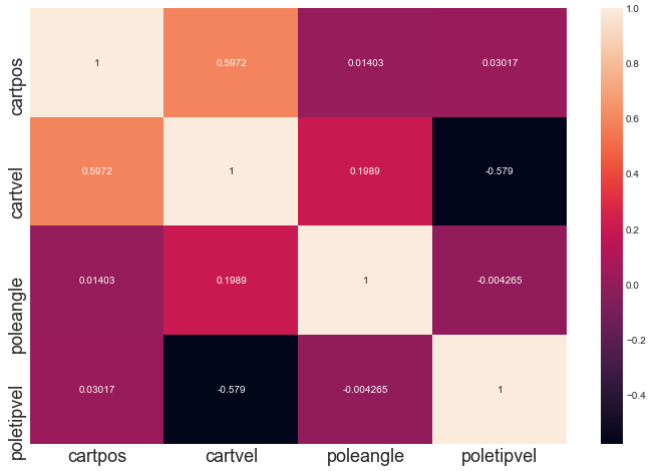


Figure 4.13: Cartpole: Correlation. Values closer to ±1 are more connected (high correlation), and no feature pairs have values above ±0.6

Overall, this is still a drawback when applying XAI-methods to robotic problems since the features are naturally correlated. Some solutions to this problem were

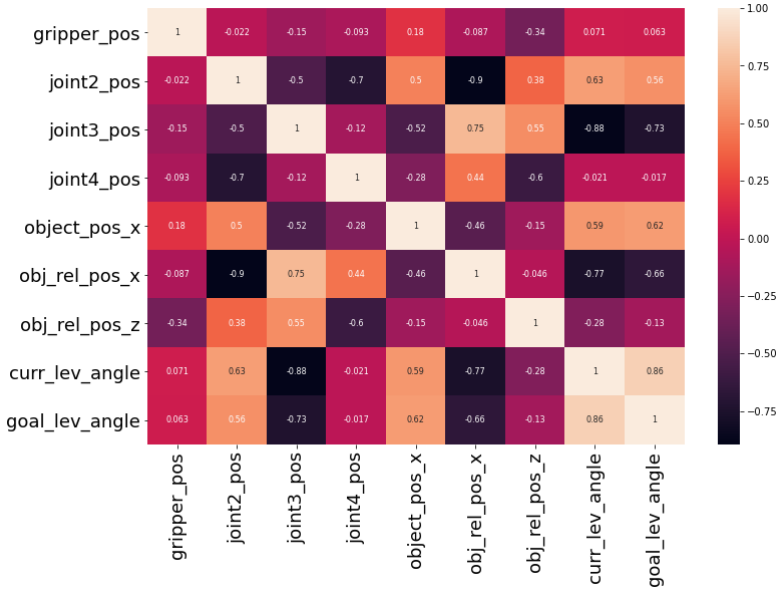


Figure 4.14: Robotic Manipulator: Correlation most influential features. Values closer to ±1 are more connected (high correlation), and many feature pairs have values near ±1

proposed after the pre-project, such as permuting correlated features together, but this could violate the symmetry axiom in the Shapley estimation. Therefore, the proposal was to reduce the state space for the manipulator to only include the most influential features and look more into the initialization and training phase for the simpler Cartpole agent. It would not remove the challenge of feature independence, but it would hopefully lead to a greater understanding of how this affects the explanations. In many ways, this becomes an amplified SHAP-procedure since the interpretation perspective for the method is to probe feature correlations by removing features according to the game-theoretic framework.

LIME

- LIME is easy to implement and use across different models and works for multiple data types. The interpretations given are easy to understand. It gives precise prediction probabilities together with feature importance. It does not say anything about magnitudes or full attributions but gives straightforward human-friendly explanations. LIME can only make local predictions, but it calculates these very fast. In this project, all of the explanations, even the more complex ones, were predicted within a few seconds. The visualization of the LIME-predictions was easy to follow in both environments.

- As with SHAP, the correlation between features is ignored since sampling is from a Gaussian distribution. Another problem being observed with LIME is the instability of the explanations, suddenly changing between similar situations. This could be because the correct definition of the neighborhood is a huge, unsolved problem when using LIME. This is especially important with the environments that use tabular data. Samples near the point of interest are weighted more heavily than samples far away, and when using discretization, it is impossible to differentiate within the discretized bins. The way sampling is done in LIME can lead to unrealistic data points, and the local interpretation can be biased towards those data points. This makes the LIME-method even more vulnerable for adversarial attacks than SHAP, where [58] found ways to fool these post-hoc explanation techniques with discriminatory behavior. The instability and vulnerability of LIME were observed throughout the testing period, where a small

situation change could lead to significant explanation differences. This is not a good sign when explaining robotic models and makes it more challenging to trust the explanations.

Gradient approaches

As these methods are very similar in their building blocks, the characteristics will be presented together. In many ways, Saliency maps are a simplified approach compared to Integrated Gradients (IG). Therefore, the main focus will be on presenting the characteristics of Integrated Gradients, with supplements of the Saliency approach towards the end. Attribution methods are hard to evaluate because it is challenging to distinguish explanation errors from the gradient method versus errors of the agent. The idea is to use the observed project explanations together with papers evaluating the methods.

Ïntegrated Gradients and Saliency

- By going directly into the gradient, this method is by definition also model-specific. However, the framework is simpler and much faster to compute than the perturbation methods. It is easy to only relate to the magnitude of each feature. However, on the other hand, it is limited how much information that can be extracted from only the feature magnitude without taking the different actions into account.
- With only the feature magnitudes from a global perspective, it is not easy to know whether an explanation is correct when it primarily gives a qualitative evaluation. Multiple papers have also shown that these gradient methods can be fragile for adversarial attacks and unreliable [59][60][61]. An investigation regarding insensitivity revealed that some types of gradient methods primarily focus on small changes instead of using the actual training data. This is an ongoing discussion within the XAI-field, and more research on how to evaluate these methods more thoroughly will hopefully be available soon. The overall observation from the results in this project showed that at least IG mainly was in line with the other perturbation predictions. The insensitivity theory is probably an important factor for the observed differences with the *pole angle* feature in the Cartpole environment.

- The idea behind Integrated Gradients is to compute the average gradient while the input varies along a line path depending on a baseline. The developer chooses this baseline, and multiple alternatives exist. In this project, a random baseline has been used because the authors of IG suggested that using a random distribution is advisable [45]. However, the approach has two main drawbacks; it could change at every run because of random sampling and also introduce patterns that cause biased attributions [62]. On the other hand, compared to other baselines (zero, one, and mean-base), the results with the random baseline were very close to the author's expectations. The similarities between the SHAP- and IG-results could be further amplified by the close connection between the gradient method and the DeepLift algorithm used in SHAP. DeepLift could actually be used as a good approximation of Integrated Gradients since it approximates an average partial derivative at each non-linearity where IG computes this gradient in regards to a baseline, both while varying the input. Overall, the similarity between the IG and SHAP explanations increase the trust in both methods.

- The main use case for the Saliency method has been primarily images trained with Convolutional Neural Networks. Saliency takes the absolute value directly of the gradient, indicating which feature that can be perturbed the least to change the output the most. However, choosing the absolute value directly instead of using a baseline can limit the detection quite significantly [44]. Overall, Saliency also performs well across both these environments, but it seems to either be over-estimating or underestimating each feature. When having the better alternative of using IG, both theoretically and from the observed explanations, this will be the chosen gradient method for the procedures in the following sections. In addition, the magnitudes of learned model weights tell us about the correlations between the dependent variable and each independent variable. Zero weight means no correlation, whereas positive weights indicate positive correlations and negatives the opposite. This could explain some of the discoveries regarding the *pole angle* disagreements between the perturbation and gradient approaches in the Cartpole environment and will be further investigated in the next section.

4.2 Data adaptions

The comparison between the XAI-methods above often confirms the human intuition in regards to the influence from some of the features. Especially in the Cartpole environment, XAI can increase the understanding of the agent. Some interpretations from the Robotic Manipulator can also be collected through different methods. However, regarding the research question, the XAI-methods struggle more when the state space increases in complexity.

This forms a basis for the underlying motivation of the data adaption procedures. For the Cartpole environment, where it is simpler to follow the development from a human perspective, forced implementations were implemented. This would hopefully give an increased understanding of how the model performs when putting it under pressure from the start of the episode. It could possibly also explain some of the observed differences between SHAP and IG regarding the *pole angle* feature.

For the robotic manipulator, the big and complex state space creates challenges both for the XAI methods and the interpretations from the author's side. A reduced model with the nine most influential states active in the method comparison was therefore made and trained with the same goals. Suppose the XAI-methods can be used firstly to interpret a complex model, reduce the trained state space without a significant decrease in performance, and learn even more about the reduced agent. In that case, this could be a beneficial area to investigate further.

4.2.1 Results

Forced initializations Cartpole

To produce the initialization plots in the Cartpole environment, ten solved episodes have been used. A test episode is used to produce the explanations, and for every test, the model gets initialized with a feature change from the start. This gets done for all of the four features, ten data points each time. For example, the *cart velocity* feature gets varied from starting at zero velocity to a maximum of 2.5 through 10 steps, where each step gets increased by 0.25. The maximum boundary is decided by trial-and-error

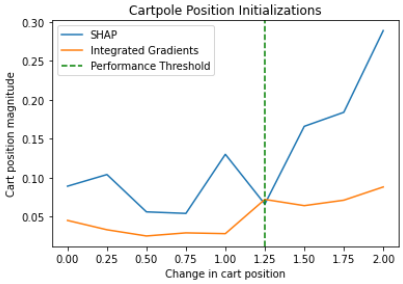
to see how much change is needed for the episode to fail. In addition, a performance threshold is included for each feature, which reflects where the model starts to cross the boundary of not solving the episode (score under 195) a majority of the time.

The total magnitude of the specific feature is compared against the initialization change for the same feature. It should be noticed that a zero change reflects the same magnitude value as the method comparison above. Hence for a zero change, both XAI-methods produce the same magnitudes as above, meaning each of the feature magnitudes starts at a level well above zero. All of the features and the corresponding magnitude changes are included to form a total of four subplots. The expectation from the author is that an initialization change for a specific feature will increase the importance of the explanation model. This is because the agent needs to compensate when the forced initialization gets more extensive with actions that stabilize the given feature in the other direction.

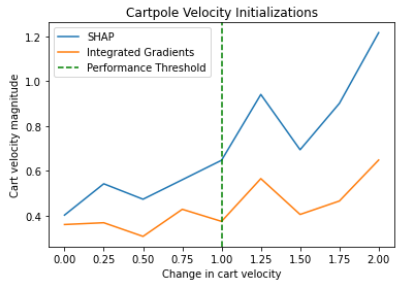
In Fig 4.15, initializations are included for the two first features, *Cart Position* and *Cart Velocity*. Both feature magnitudes increase dramatically when the change gets bigger. Some fluctuations are expected and also present in these two subplots. Still, both features follow the same upwards trend in SHAP-magnitudes, with a big increase after the performance threshold. IG also goes a bit upwards in magnitude, but not nearly as much as SHAP. More specific comments about the numbers are included under each subplot in Fig 4.15(a) and Fig 4.15(b).

In Fig 4.16, the remaining two features are presented using the same initialization procedures as above. The *pole tip velocity* in Fig 4.16(b) follows the same trend, with an almost exponential increase when passing the performance threshold. It should be noted that the magnitude is already pretty high from the start, as this is the most influential feature. IG follows SHAP more closely here after the threshold compared to the other features.

The most significant difference in these plots is the *pole angle* feature. The disagreements between the gradient and model-specific methods observed in the previous section appear even stronger here. It is hard to say if it is an underestimation from SHAP



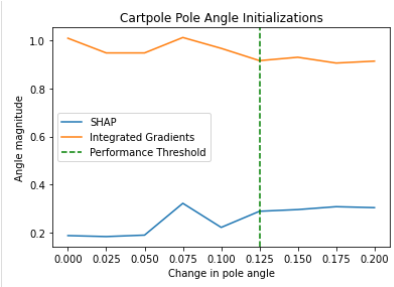
(a) Position
SHAP starts at 0.1, with small fluctuations towards the performance threshold. After passing, the magnitudes get much bigger with 0.3 in magnitude when the position gets initialized to the right. IG varies much less, only ranging from 0.05 to 0.1 in the end.



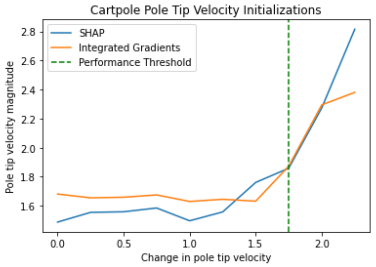
(b) Velocity
SHAP starts at 0.4, and the magnitudes gets much bigger also here after passing the threshold, going above 1. IG starts a bit under with less variation, but an increase can be observed towards the end going over 0.6

Figure 4.15: Forced initializations for Cart Position and Cart Velocity

or an overestimation from IG, probably somewhere in between. A strange phenomenon is also that both methods are very stable even after the threshold. Even though the change is minor here compared to the other features, an increase will be expected when the *pole angle* initializes away from zero to compensate. The question is if the SHAP-perturbations account for the other features to cover this start angle or if the perturbations from the training episodes only account for tiny changes in the *pole angle*.



(a) Pole Angle
SHAP starts at 0.2, and finishes only slightly above, even after passing the performance threshold. IG also has less variation compared to the other features, but starts and finishes at a much higher lever, around 1.



(b) Pole tip velocity
Both SHAP and IG starts in the interval $1.5 - 2$, moves steadily towards the performance threshold, followed by a big increase for both methods.

Figure 4.16: Forced initializations for Pole Angle and Pole Tip Velocity

Feature space reduction Robotic Manipulator

When interpreting the Robotic Manipulator explanations above, some frustration about keeping control over 20 different features in a complex model could probably be read through the lines. The motivation to reduce the model complexity to see how this affects both the agent performance and the XAI-methods was also discussed above. In this section, the state space of the agent has been reduced considerably to 9 states. The same plots from SHAP, IG and Saliency are presented in Fig 4.17, Fig 4.18 and Fig 4.19.

The explanations in these plots are pretty similar to the ones with all features included. The lever angles are still dominant, but especially from the SHAP-analysis, it can be observed that more features have a significant impact on the model. The same yields for the gradient methods, however the increase is not as significant. Other trends to notice are that all of the actions are more involved in each feature's magnitudes, but this also varies slightly depending on how the agent reaches the goal in that specific episode. It shows that very many of the same properties are transferred from the original to the reduced model, benefiting the user of explaining under half of the state space.

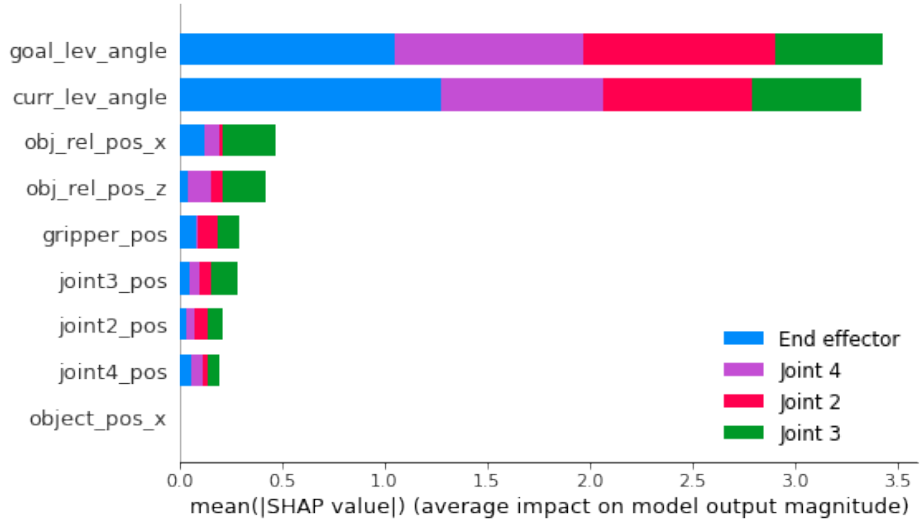


Figure 4.17: Robotic Manipulator: Shap Global Summary plot reduced model. Note that the colors for each action has changed from the Summary plot with all features

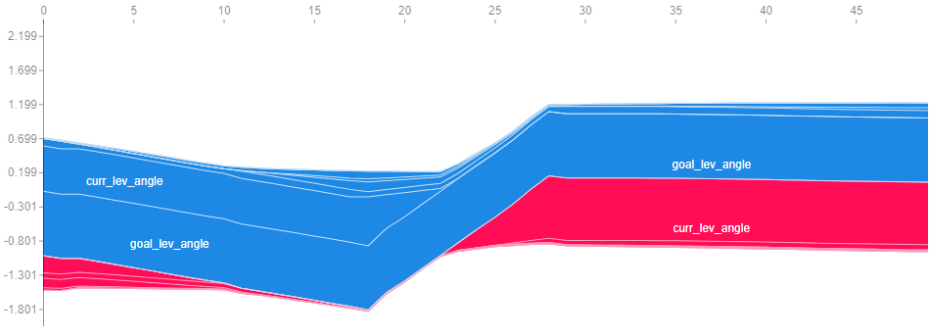


Figure 4.18: Robotic Manipulator reduced model: Shap Global Force plot action 1 (moving joint 2). SHAP-values on the Y-axis vs State number on the X-axis.

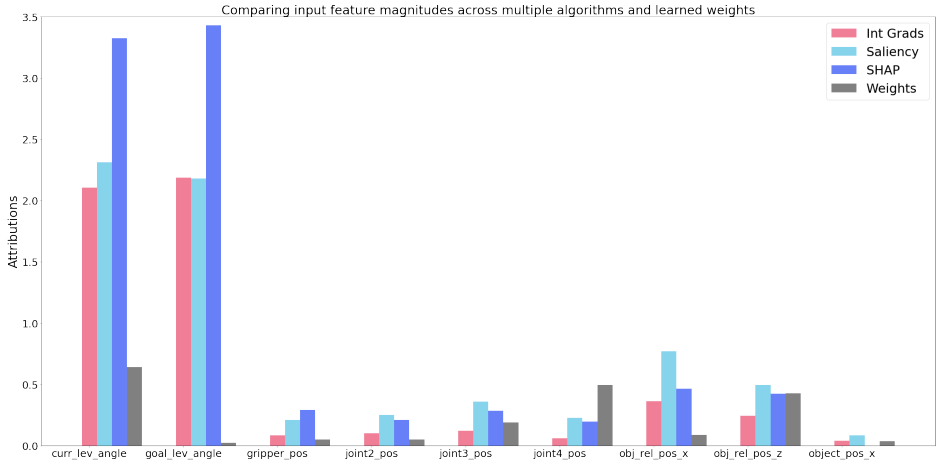


Figure 4.19: Robotic Manipulator: Captum attributions reduced model

4.2.2 Discussion

The trends from the data adaption procedures are primarily described above. Therefore, the goal of the discussion regarding this part is to connect the characteristics from the comparison above with the observations in this section.

For the Cartpole initializations, the primary trend to note is that the SHAP-explanations are more influenced by a more challenging start position for all the features. In regards to the biased "fooling" experiments in [58], this shows that small changes affect the perturbation approach much more than the gradient. On the other hand, from a human intuition, the influence of a feature should increase when it is put under pressure from the start. IG shows a slight increase across the features, but not nearly as evident as SHAP. If, for example, the position feature starts in a wrong position, this could maybe be easier to spot right away from the SHAP-plots, where the magnitude increases more dramatically.

The trend regarding the *pole angle* feature is somewhat disconcerting and has been a focus area across multiple result parts. The point on insensitivity could explain why IG rates this feature important constantly across the increased starting points. Even though it gets initialized with an increased angle, the gradient is still perturbed to produce

changes within a similar range. These changes are probably much more significant than the neighborhood perturbation sampling done in SHAP, which is based on the actual values across the training episodes. Therefore, it could be argued that the *pole angle* feature, which has a significant impact if it varies, amplifies the differences between these approaches.

The idea behind reducing the Robotic Manipulator state space is to make the explanations from a complex model easier to interpret. An increase in feature impact from the XAI-method output is expected since fewer features contribute towards the agent's actions. This is most present in the SHAP-explanations while the gradient methods do not increase significantly. This is probably an effect of the perturbation sampling, where each feature will be more affected with fewer features.

Overall, using the entire model to produce explanations leads to discovering the most influential states. Using these explanations to reduce the model, and keeping the same performance with under half of the features, increases the trust of both the agent and the XAI-methods. The same trends regarding the most influential features are still present in the reduced model, which is very favorable, and it shows an exciting area of use for complex robotic XAI-research.

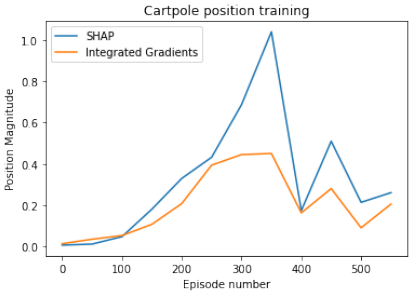
4.3 Training analysis

In the explanations above, the XAI-methods have been applied to environments where the agent already has been trained to solve the episode. This leads to information about how the agent emphasizes the different features to guide the Cartpole or Robotic Manipulator to a particular goal point. This is useful when learning more about the basis for the agents' decisions. Still, it is also interesting to learn more about how the DRL-algorithms train the agent to be able to solve the environment. When working with robotic environments, one of the main motivations with XAI is to be able to spot critical errors at an early stage. To do this, the black box needs more transparency, but how about opening up the box before it develops into a black box architecture?

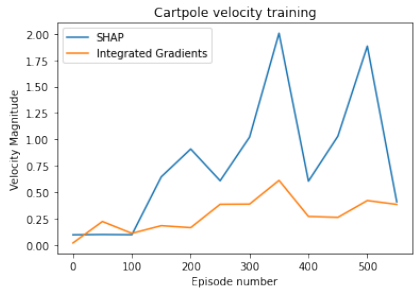
The methods used in this project are post-hoc, and they are not coupled directly to the DRL-algorithm, meaning they cannot influence the training procedure. More on the potential for doing this in the discussion, but since SHAP and IG cannot affect the agents' decisions, a creative solution had to be implemented. This was done in the Cartpole-environment, where the REINFORCE DRL-algorithm needs approximately 500 episodes to solve the OpenAI environment. The training procedure was divided into steps of 50 episodes, where the XAI-methods use 1000 states in each step to train the explainer. These states can be everything from 5 solved episodes in the end to 50 unsolved episodes in the start when the agent struggles. SHAP uses the last episode to produce global explanations, while Integrated Gradients sets a baseline for those 1000 states to find the feature magnitudes. Overall, this increases the computation time of the training procedure, especially the SHAP-explanations take some time to compute after each 50 episode step. The faster gradient approach could be beneficial if time is limited, since it does not significantly delay the training procedure.

4.3.1 Results

The expectations regarding this procedure are not necessarily straightforward in advance. From the start, when each episode only contains a few states until the Cartpole loses balance, the magnitudes of all features are probably at a low level. This would increase towards the solved episode explanations above when the agent gets better.



(a) Position:
IG starts at zero and increases towards a magnitude of 0.4 after 350 episodes, and has a magnitude around 0.2 at the 500 mark with some decrease in the end.
SHAP has a big magnitude (1) between episode 300 and 400, before decreasing towards almost the same level as IG in the end.



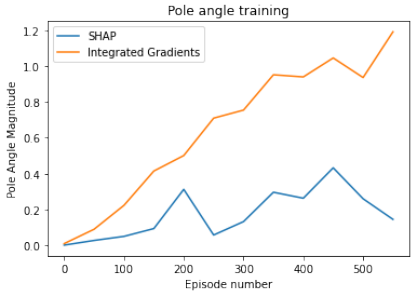
(b) Velocity
IG increases slowly from zero to a bit under 0.5. In this feature, SHAP varies very much up and down, from zero to almost 2 in magnitude, before ending at the same level around 0.5 in magnitude. Interestingly, the feature also starts a bit over zero, indicating that the velocity features contribute more to the failed episodes from the start.

Figure 4.20: Training plots: Cart Position and Cart Velocity

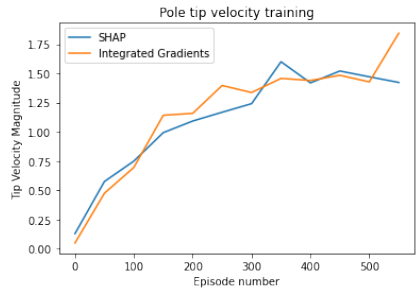
However, the agent could also encounter problems under- or over-estimating one or multiple features in the middle of the training process where the episodes vary in length. The expectation is that this will stabilize when reaching the threshold of solving the environment, around the 500 episode mark.

In Fig 4.20 the position and velocity features are plotted against their respective magnitude. Both features follow the same pattern with small magnitudes from the start that increases when the number of episodes gets larger. Some over-estimations compared to the solved episodes are also observed, especially around the halfway mark. As with the forced initializations, SHAP varies much more than Integrated Gradients, with big ups and downs. Especially the velocity feature reaches almost two in SHAP-magnitude closely before the end, which can be a sign regarding the instability of the perturbations. Integrated Gradients are much more stable throughout both features.

This is not the case for the two remaining features in Fig 4.21. Here, the magnitude of Integrate Gradients increases while the agent gets better. This is the case both for the *pole angle* and *pole tip velocity*. For the latter feature, SHAP follows the same trend, and



(a) Angle:
IG increases from zero magnitude in the start towards 1.2 when the agent solves the environment after 500 episodes.
SHAP has much lower magnitude through all episodes, also beginning from zero, but ending only at 0.1



(b) Pole tip velocity:
Both methods follows the same trend here, increasing from zero towards an magnitude between 1.5 and 1.75. Also with this velocity feature, SHAP, initialize a bit over zero (0.1) from the start.

Figure 4.21: Training plots: Pole Angle and Pole Tip Velocity

this is also the feature with the highest magnitude in the end, as seen in the sections above. The big difference between the two methods for the *pole angle* is also present here. However, in these two states, the SHAP-explanations do not fluctuate so much up and down as seen above. A theory can be that when the magnitude of the *pole tip velocity* gets big enough, together with stabilizing the magnitudes of the *position* and *velocity*, the agent also stabilizes enough to solve the environment. However, the differences between the methods with the *pole angle* feature could decrease this hypothesis, as many more factors probably play a role in keeping the Cartpole stable. The discussion below will cover more about this theory and evaluate how these plots can decrease or increase the trust of the agent's learning period.

4.3.2 Discussion

The expectations about how the XAI-methods develop throughout the training procedure have already been covered above. In summary, three of the four features follow the same trends, and the same observations regarding the initialization procedures are visible. SHAP varies much more, and there are still significant disagreements about the

pole angle feature. A sign of strength for IG is that this feature increases in magnitude while training, meaning the changes are directly dependent on the trained agent. This could be a sign that it is the SHAP-explainer that is most off about the interpretations of this specific feature, which is backed up by the importance with other analysis of the Cartpole environment [63]. The faster computation time of IG is also very useful in these implementations, especially when only observing the changes in feature magnitude over the training phase.

Even though these training plots show explanations across the agent's training phase, the limitation of the post-hoc approach becomes more visible. The reason model-agnostic and gradient methods have been used in this project is that they are decoupled from influencing the underlying machine learning model. This makes them easier to implement across multiple environments and gives explanation and representation flexibility. Because of these scaling abilities, they are expected to be a dominant factor in the XAI-field [64].

When seeing the underlying basis on how these post-hoc methods can extract information from the training phase, an even more exciting direction would be to see the performance of intrinsic XAI-methods that are directly coupled to the DRL-model in these environments. Unfortunately, the author did not manage to find any promising XAI-methods that are intrinsic, and at the same time, able to explain robotic environments. In the future, this could make it possible to use the interpretations to influence the agent while learning. Hopefully, this could improve the final results without developing the system into a black box, eventually leading to the third wave described by DARPA [14].

4.4 Real time XAI

The third research question in this project investigates how an audience can engage with robotic models. In the sections above, the plots interpret the agent with explanations produced after the episode has finished. Can this be visualized in a way that's more related to a real-time analysis, where a human operator can engage with the explanations?

The idea behind this last result section is to use local explanations to interpret all of the features in one episode and visualize it in a way that could be beneficial for a human operator. The goal is to transfer the attempts into a simulation using an OpenAI rendering video of the Cartpole. The model-agnostic methods of SHAP and LIME have been used in this task because they produce local explanations and also say something specific about prediction values and feature magnitude in each state.

As discussed in the method comparison section, one of the benefits of LIME is that local explanations are produced very fast. However, when producing 200 local explanations in a row, SHAP has the benefit that it can use a global training procedure and then use these explanations locally. This means that the SHAP-explainer turned out to be the choice when applying these interpretations in a simulated environment. There are still some challenges with the methods being post-hoc in this case, so it is not really real-time in theory. However, the computation time for SHAP is hopefully fast enough to produce explanations rapidly after the episode.

However, before testing in the simulator, episode plots were produced for both SHAP and LIME in the Anaconda environment.

4.4.1 Results

The episode plots visualize how each feature changes locally across a complete episode, and an example of how to interpret the plot type is shown in Section 3.3. The state values are shown as a blue line with scatter points in red/blue, where the intensity indicates the magnitude of a push in one of the directions. This can, for example, reveal some of the critical states where the Cartpole or Robotic Manipulator struggles. Cartpole's two actions are easy to control, while the Robotic Manipulator proposes some challenges since the action space increases. The plots have been made by looping over an entire episode and calling the explainer at each state. In the end, the explanations are linearly normalized between data points to better reflect each feature in context with each other.

Episode plots Cartpole

The episode plots for the Cartpole environment are shown in Fig 4.22 (LIME) and in Fig 4.23 (SHAP). As seen in the situation from the local explanation in Section 4.1.1, the *pole tip velocity* feature is dominant in both explainers. This can be recognized by more intensive red/blue data points. When the *pole tip velocity* is positive, both explainers indicate a push in the right direction to stabilize the pole. The opposite happens when the feature is negative, which makes sense according to balancing the Cartpole.

There are not too many indications from the other features, but some intensive dots can be spotted when the feature is well above or below zero. The most concerning feature is maybe *cart velocity*, showing some red dots when being positive, which does not make much sense. The intuitive action with a positive (right) velocity will be a push to the left (blue dot), but the SHAP-plot does not always reflect this action. Some comments about all the features are presented below.

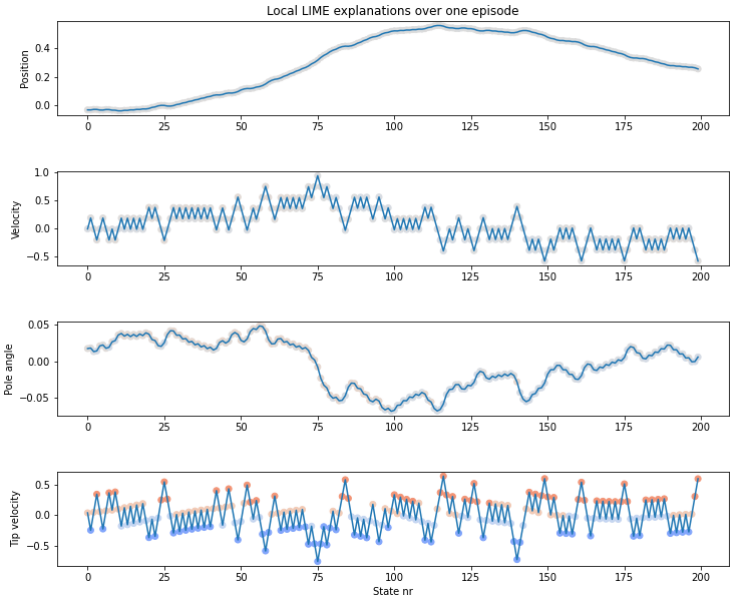


Figure 4.22: Cartpole: Episode plots Lime

- **Cart Position**: The position feature varies from 0 to 0.4, so the Cartpole moves a bit to the right and back during the episode, well within the environment boundaries. Both SHAP and LIME explanations have a grey tone in almost all states, which indicates that this feature does not have a big influence on any of the local explanations.

- **Cart Velocity**: This feature varies much more up and down, but the scatter points still have little intensity. Some red or blue tones can maybe be spotted in the outliers for the SHAP-explanations, commented above as a disconcerting factor. However, overall in this episode, the velocity feature nor does have any special impact.

- **Pole Angle**: The angle starts with a positive value (tipping right) before switching in a negative direction after 75 states. Throughout the episode, this feature does not contribute significantly to the explanations either.

- **Pole Tip Velocity**: The dominant feature where the explainer proposes a right push when the feature is positive and a left push when the feature is negative. As it is a velocity feature, this feature also varies a lot.

Episode plots Robotic Manipulator

The reduced manipulator model is used in this section, but it still has four actions, meaning an episode plot above does not contribute as much as in the Cartpole environment. One way to visualize this is to show each action, where blue/red indicates the intensity of choosing this action or not for that specific feature. Another way is to choose the features with the greatest influence, in this case, the lever angles, and plot them across all four actions. Both procedures have been done with SHAP and LIME. These are presented across two figures in Fig 4.24 and Fig 4.25. Note that these plots are of good quality, so it is possible to zoom in to visualize better.

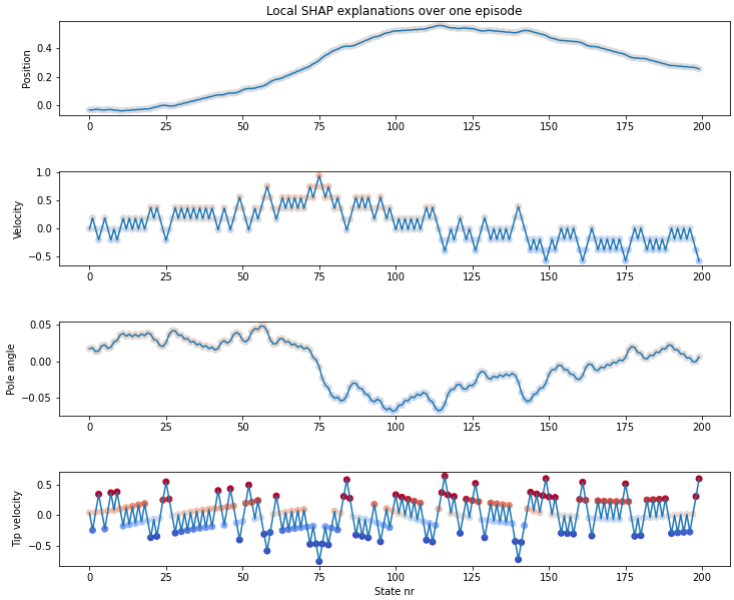


Figure 4.23: Cartpole: Episode plots SHAP

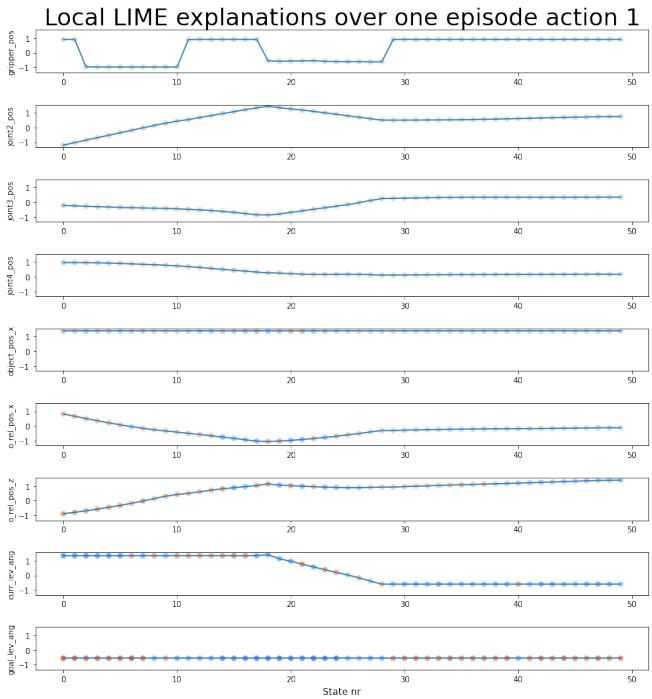
In Fig 4.24 all features are shown across the first action, which is to move joint 2. As with the global plots, this is done by training the explainers with 24 episodes (1200 states) and then running it across a test episode with 50 transitions. The primary trend to spot across these two methods is that the lever angles take a huge portion of the total magnitude. As with the Cartpole, SHAP has in general higher magnitudes, which can be seen by more intensive color points. LIME has a bit more even distribution across the nine features, while SHAP gives the dominant impact to the lever angles.

From the lever angle plots in Fig 4.25 it can be implied that the SHAP-explanations give a more stable interpretation across the episode. Before the *current lever angle* changes, both angles push in the same direction for every action. Afterward, they push in the opposite direction, which is expected since the angle is in the correct place. LIME however, has some outlier points that suddenly push in another direction, and the same trends are not so obvious to spot. These observations can be seen in direction with the method comparison discussion in Section 4.1.2, where one of LIME's disadvantages was unstable values because of adversarial perturbations.

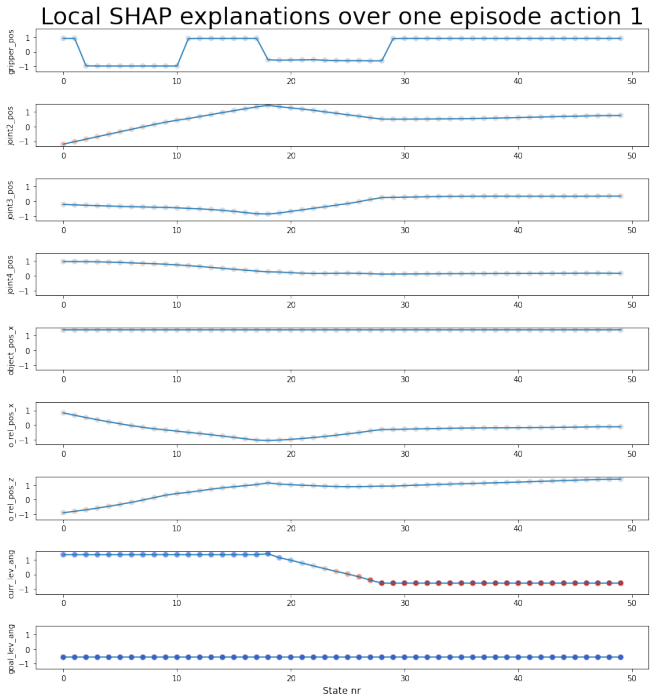
Even though some trends can be spotted across the two episode plots above, having nine features and four actions make it difficult and unnecessary to focus on all the information. Using the original model with 20 features would have been even worse. Arguably, these explanations do not add a significant amount of information compared to the global force plots above, where SHAP provides a good visualization technique to discover the essential features.

Simulation example

In the Cartpole environment, the episode plots were much more helpful since all four features contribute to keeping the pole upright. It is also much simpler to keep track of only four features and two actions. At the end of this project, the episode plot from SHAP was used to make a simulation example from the OpenAI render environment. This was mostly done because of an underlying motivation to show how a human operator could interpret these results regarding the second research question, which will be discussed below. There are multiple ways to implement such a solution, and one way could be to make an Graphical User Interface (GUI), another is to generate a

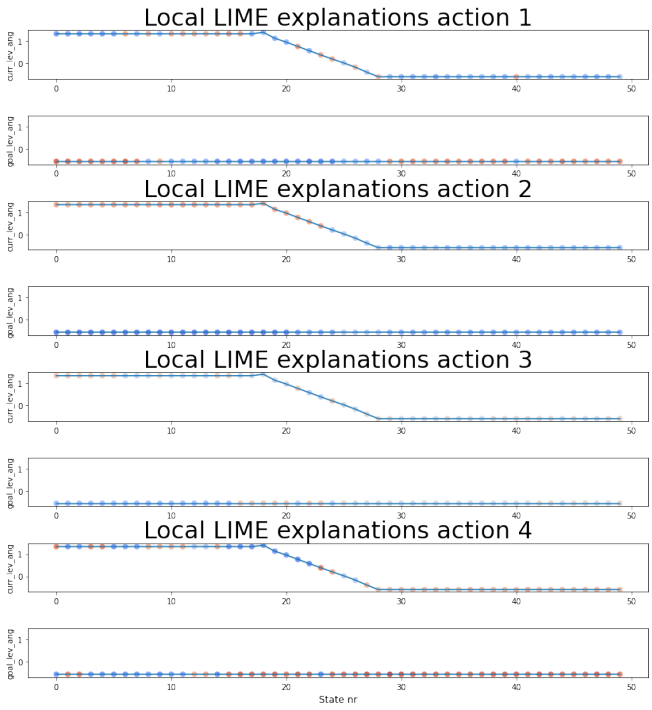


(a) LIME Explanations

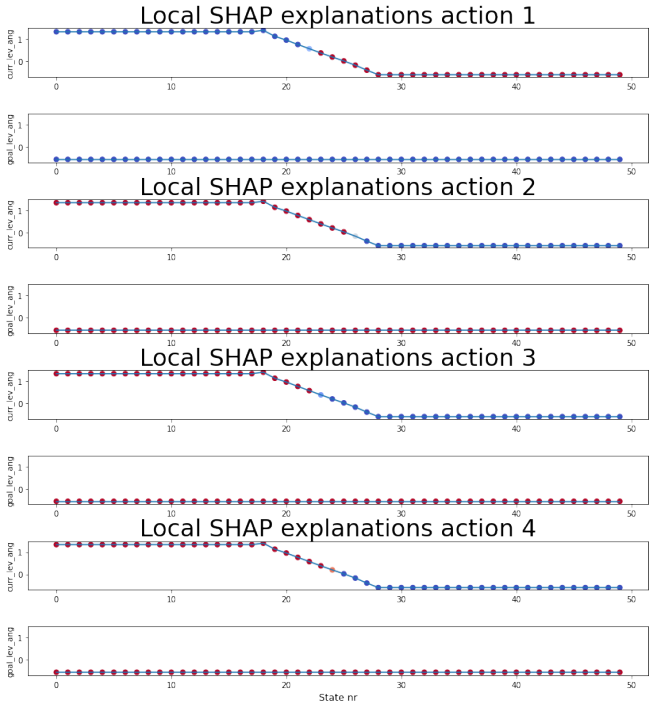


(b) SHAP Explanations

Figure 4.24: Episode plots Robotic Manipulator for action 1 (Moving joint 2). Plot is of good quality, the author recommends to zoom to see details



(a) LIME Explanations



(b) SHAP Explanations

Figure 4.25: Episode plots RM for all 4 actions across the lever angles. Plot is of good quality, the author recommends to zoom to see details

video file in real-time. A simplified version of the last one was implemented. Since the SHAP-method used is post-hoc, the simulation is not "real-time" by definition, but it is still a way to follow the process, either in a testing environment or during training.

The implementation was done by making a GIF from the OpenAI render feature of one solved episode, playing in 2 frames per second with the given feature values and state number. The SHAP-explanations from the episode plots were inserted into a package called *barchartrace*, which varies each feature represented for every frame according to its value [49]. This was done for both actions. A screenshot from the simulation, a thought scenario of a human operator's eyes, is shown in 4.26. Here, *state 67* is shown, and as with the local SHAP-explanations, a negative *pole tip velocity* pushes for a left action, while a positive *pole angle* does the opposite. Keep in mind that this interface was mainly done at the end of the project to show a possible use case for these episode plots and that a proper video with better labels and smoother graphics is a task for future work.

One interesting note when implementing this simulation was the author's observation on how the agent chooses its actions to keep the pole upright during the episode. This was very easy to spot when playing it as a bar chart race but was not discovered before with "traditional" plots. During this specific episode, and also the other ones that were tested, it could be argued that the feature *pole tip velocity* "controls" the agent. Every time it becomes positive, the SHAP-value increases from 0 to around 1. The agent suggests an action of pushing in the right direction as long as there are no significant values from the other features, which makes the *pole tip velocity* decrease back to 0. During the episode's 200 states, this happens back and forth. If the other features have a significant impact, leading the agent to take action in contrast to the *pole tip velocity*-proposal, the SHAP-value of this feature will increase to 2, and the agent must consider it in the next step.

In conjunction with the summary plot in Fig 4.4, these values make sense. Overall, such interactive explanations are likely a better way to spot trends and characteristics of the agent from an operator's eyes. The discussion below will move into the possible use-cases, pitfalls to be aware of, and how to get audience engagements across these two environments of different complexity.

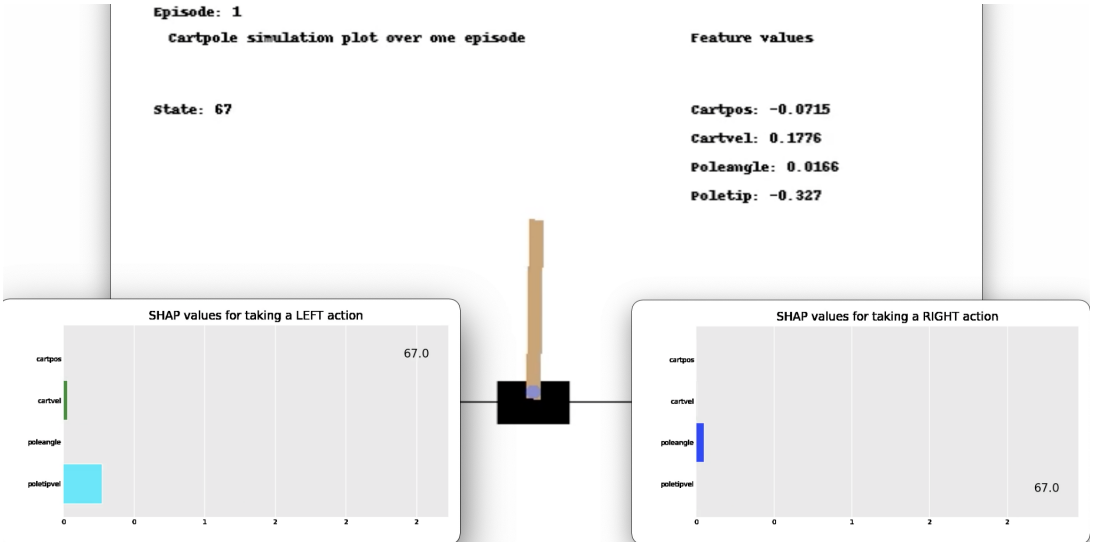


Figure 4.26: Cartpole: Simulation example

4.4.2 Discussion

The last part of this project is closely connected with the second research question. To address the possibilities of audience engagement with XAI-explanations, real-time interaction will probably play a big part in the future. The post-hoc methods used in this project make the direct engagement a bit limited for such attempts, but the part above shows that it is still possible to create visualizations and simulations that could increase interaction between the DRL-model and a human operator.

As with the previous result sections, a simpler environment makes it easier to follow the explanations across such an episode plot. For the Cartpole environment, it was a useful way to visualize the local explanations over a full period. It also made it possible to learn more about the agent, for example, by discovering the trend on how the *pole tip velocity* in practice *"controls"* the agent's decisions. Such an analysis makes room for multiple extra attributes that could be useful from an operator's perspective. Examples can be an alert system for outlier states, mysterious explanation values, or within critical phases.

The same patterns could also, to a certain extent, be found in the episode plots for the Robotic Manipulator. Even though this is the reduced model with a smaller feature space, four actions make it more challenging to visualize in such a manner. In the Cartpole environment, the intensity specifies the action one way or the other, while the intensity in the Manipulator plots indicates if this specific action should be chosen or not. When having four actions in total, it is difficult to separate each feature. The use case for the lever angle plots could be more beneficial for a human operator to see if it changes when the manipulator reaches the lever. However, the global Force plot shown in Fig 4.18 covers a lot of this information already. Therefore, other "real-time" visualization techniques should probably be investigated further to see if it is possible to plot one action against the remaining three over an entire episode.

Both SHAP and LIME have been used to produce these episode plots. In the comparison above, the characteristics of each method have been presented, and one of the advantages with LIME was the fast computational time. However, when looping all local explanations, the total time for LIME suddenly becomes pretty extensive, while SHAP benefits over collecting global SHAP-values directly. Therefore, the simpler LIME approach is not so valuable for these settings. With the extra factor that SHAP explanations produce more consistent explanations, it was preferred for the simulation part.

The simulation model presented in Fig 4.26 is mainly included to show how the operator from a control station can follow this. It is easier to understand the model when seeing both the features and explanation values changing throughout a rendered episode of the environment. A natural next step for this project is to improve this example to an actual video or graphical user interface and run test episodes for a selection of users. For instance, multiple attributes can be added, such as warning lights, coloring the pole features, and improving the bar chart variations. Both the author's supervisors and fellow students agreed that a form for a bar chart race could be a good visualization idea to follow the interpretations, so it would be interesting to test this further.

Chapter 5

Conclusion

5.1 Answering the research questions

How do state-of-the-art methods from Explainable Artificial Intelligence (XAI) perform on simulated robotic systems? What are the crucial factors to consider when choosing between these XAI-frameworks?

Much information can be collected about the environments from running these XAI-methods across four different procedures. It has been interesting to confirm the pre-existing human intuition about the models and use these explanations to discover new trends. The impression throughout these attempts is that a lot of the explanations make sense, but on the other hand, it is almost always possible to find unexpected predictions. The assumptions and weaknesses of the methods presented in the comparison section also put the explainers under an extra critical view. It is sometimes challenging to know whether to trust the agent or the XAI-explanations, and this means that in the worst case, the black box gets even more confusing.

When going through the results in this project, it is interesting to compare it to other user studies done in the XAI-field. For example, one study indicated that SHAP-explanations are consistent with human explanations [56]. On the other side, recent studies argued that SHAP, albeit good in generating explanations, does not improve final

decision-making [65][66]. The author will argue that it could improve final decision-making, but it is vital to keep the limitations in mind with a critical interpretation viewpoint. A lot of new XAI-methods are being developed at the moment. However, in addition, it would be great to increase the focus on existing methods for several use cases and possible improvements. For example, a review of how to evaluate these methods against each other in different settings would be beneficial.

When choosing a framework, the main points to consider are the given use case, time limitations, and how knowledgeable the audience is. Gradient methods have the benefit of being faster and possibly less influenced by the correlation problem. The model-agnostic methods give a broader explanation model with more information, but it could come at the cost of perturbations errors. The author's proposal is to use the benefits of both types of methods as insurance to increase the trust of the explainers as well as the agents.

The overall results in this project show that promising XAI techniques could help interpret the black-boxes in DRL and give explanations that make sense from a human perspective. In simple environments (Cartpole) where the features are not too correlated, this can be used as support to understand the model, both in a local and global scope. This shows that XAI could enlighten insight about the deployment of everyday problems in the RL community. However, when the environments get more complex with highly correlated features, some of the challenges by using XAI methods in DRL are more prominent. Feature removals were an effective way to reduce some of the challenges in complex environments. However, much research remains before this could be seen in real-life robotic DRL-development.

Can these XAI-explanations be used to engage with end-users, and how does this affect the trust of the DRL-models used to control robotic systems?

The simulation examples and "real-time" analysis done in the latter parts of this project could possibly increase audience engagement with the DRL-agents. It makes it easier to discover trends from the models and more straightforward to spot outlier explanations. However, more user studies across different end-users could investigate

how these explanations can engage with diverse target audiences. These studies could also include better visualization and simulation techniques, which would make the development phase easier and could convey the results in a better way. Assessing how XAI techniques can understand models beyond classification tasks is starting to get more attention, and a new subfield in Explainable Reinforcement learning (XRL) is emerging [67]. The most crucial thing in XRL is to keep the human side of the equation in mind [64]. No clear answers make interpretations more complicated, and explainability should possibly be used as a tool instead of a solution.

The goal with XAI methods is to open up the black box problem to interpret decisions made by the agents. One of the drawbacks of robotic reinforcement learning is that mistakes in such systems could lead to dramatic consequences. Therefore the amount of transparency must increase before it gets more adapted in the industry. The methods used in this thesis show that XAI could help increase the trust about simulated robotic problems trained with DRL. It can give information about the agent's predictions and important features, both locally and globally. The problem is that these predictions do not have any clear basis for comparison, in contrast to machine learning problems where it is possible to classify a correct prediction. This makes the XAI methods more challenging to visualize and convey, and it depends a lot more on human intuition. However, human intuition is almost a black box itself. When GDPR starts to considerate *"a right to explanation"*, a part of the project development could be to set some transparency boundaries, which both makes the system easier to explain and the end-user more prepared on how to interpret different explanations.

5.2 Further work

The natural next step in this project would be to improve the simulated Cartpole-environment explanations by making a graphical user interface or video. This could include multiple extra features and a smoother explanation interface, as discussed in the last section of the result chapter. It could be customized to being flexible to include all the XAI-methods tested in this project. This interface could also be transferred to the more complex Robotic Manipulator by, for example, using the Pybullet simulator.

After working one year with post-hoc XAI-methods, the desire for seeing the explanations from an intrinsic method that are directly coupled to the DRL-agent has increased more and more. It would be interesting to see how such a method interprets these agents compared to the methods used in this project. According to the answer to the research questions above, this is probably an essential factor to consider when developing methods for the new subfield XRL. Unfortunately, this is a field where it has been hard to find methods, so more research is needed.

This project has been a part of the EXAIGON project at NTNU, which aims to meet society's and industry's standards for the deployment of trustworthy AI systems in social environments and business-critical applications. This will be done by developing methods to understand how black-box models make their predictions and what their limitations are. Hopefully, the foundation from testing XAI in simulated environments assist in producing knowledge and understanding of how best to make use of these methods in different applications. Gradually, several use cases and data from industry players in the EXAIGON project will be available. This can be used to test the methods more thoroughly in relevant environments. Several platforms can be used for testing, including robotic arms, drones, and marine vessels, so the possible applications are huge.

References

[1] OpenAI, *Gym*, https://gym.openai.com/, [Online; accessed 11-May-2021], 2021.

[2] emanual.robotis.com, *Openmanipulator*, http://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/, [Online; accessed 25-March-2021], 2019.

[3] Pytorch, *From research to production*, https://pytorch.org/, [Online; accessed 10-February-2021], 2021.

[4] S. Lundberg, *Shap*, https://github.com/slundberg/shap, [Online; accessed 1-April-2021], 2021.

[5] M. Ribeiro, *Lime*, https://github.com/marcotcr/lime, [Online; accessed 20-April-2021], 2021.

[6] Captum, *Algorithm descriptions*, https://captum.ai/docs/algorithms, [Online; accessed 02-April-2021], 2021.

[7] I. O. Myklebust, "Explainable ai (xai) methods for deep reinforcement learning," *NTNU Project Thesis, unpublished*, 2020.

[8] A. Lekkas et al, *Exaigon*, https://www.ntnu.edu/exaigon, [Online; accessed 07-May-2021], 2021.

[9] S. B. Remman, "Robotic manipulation using deep reinforcement learning," *NTNU Master Thesis*, 2020.

[10] A. Adadi and M. Berada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, no. 1, pp. 52 138–52 160, 2018.

[11] A. Mishra, U. Soni, J. Huang, and C. Bryan, *Why? why not? when? visual explanations of agent behavior in reinforcement learning*, 2021. arXiv: 2104.02818 [cs.HC].

[12] W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," *Lecture Notes in Computer Science*, pp. 5–22, 2019.

[13] E. Puiutta and E. M. Veith, *Explainable reinforcement learning: A survey*, 2020. arXiv: 2005.06247 [cs.LG].

[14] D. Gunning, *Explainable artificial intelligence*, Program Update Darpa, 2017.

[15] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne, *Learning how to explain neural networks: Patternnet and patternattribution*, 2017. arXiv: 1705.05598 [stat.ML].

[16] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[17] Z. C. Lipton, *The mythos of model interpretability*, 2016. arXiv: 1606.03490 [cs.LG].

[18] A. Weller, *Transparency: Motivations and challenges*, 2017. arXiv: 1708.01870 [cs.CY].

[19] O. Dictionary, *Artificial intelligence*, https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095426960, [Online; accessed 01-February-2021], 2020.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, The MIT Press, 2018.

[21] D. Fumo, *Types of machine learning algorithms you should know*, https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861, [Online; accessed 03-March-2021], 2017.

[22] A. Lekkas, *Lecture notes in ttk23 introduction to autonomous robotics systems for industry 4.0*, https://www.itk.ntnu.no/emner/fordypning/TTK23, October-November 2021.

[23] M. Wiering and M. v. Otterlo, *Reinforcement Learning: State of the art*. Springer-Verlag Berlin Heidelberg, 2012.

[24] Y. Bengio, G. Guyon, V. Dror, G. Lemaire, D. Taylor, and D. Silver, "Deep learning of representations for unsupervised and transfer learning," vol. 7, Jan. 2011.

[25] A. Ng, *Deep learning*, https://on-demand.gputechconf.com/gtc/2015/presentation/S5818-Keynote-Andrew-Ng.pdf, [Online; accessed 10-March-2021], 2015.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[27] M. A. Nielsen, *Neural networks and deep learning*, http://neuralnetworksanddeeplearning.com/, [Online; accessed 10-March-2021], 2018.

[28] Claire, *4 simple steps to powerful artificial neural networks in python*, https://www.artificiallyintelligentclaire.com/artificial-neural-networks-python/, [Online; accessed 20-April-2021], 2020.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, 2013. arXiv: 1312.5602 [cs.LG].

[30] L. Weng, *A (long) peek into reinforcement learning*, https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning, [Online; accessed 12-May-2021], 2018.

[31] J. Hui, *Rl - policy gradient explained*, https://jonathan-hui.medium.com/rl-policy-gradients-explained-9b13b688b146, [Online; accessed 07-May-2021], 2018.

[32] A. Rao, *Policy gradient algorithms*, https://web.stanford.edu/class/cme241/lecture_slides/PolicyGradient.pdf, [Online; accessed 07-May-2021], 2021.

[33] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, 2019. arXiv: 1509.02971 [cs.LG].

[34] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, *Hindsight experience replay*, 2018. arXiv: 1707.01495 [cs.LG].

[35] C. Molnar, *Interpretable machine learning - a guide for making black box models explainable*, https://christophm.github.io/interpretable-ml-book/index.html, [Online; accessed 02-May-2021], 2021.

[36] A. Nayak, *Idea behind lime and shap*, https://towardsdatascience.com/idea-behind-lime-and-shap-b603d35d34eb, [Online; accessed 20-April-2021], 2019.

[37] H. D. Harder, *Model-agnostic methods for interpreting any machine learning model*, https://towardsdatascience.com/model-agnostic-methods-for-interpreting-any-machine-learning-model-4f10787ef504, [Online; accessed 20-April-2021], 2020.

[38] M. T. Ribeiro, S. Singh, and C. Guestrin, *"why should i trust you?": Explaining the predictions of any classifier*, 2016. arXiv: 1602.04938 [cs.LG].

[39] J. Manu, *Interpretability part 3: Opening the black box with lime and shap*, https://www.kdnuggets.com/2019/12/interpretability-part-3-lime-shap.html, [Online; accessed 15-May-2021], 2019.

[40] S. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions*, 2017. arXiv: 1705.07874 [cs.AI].

[41] R. Cubitt, "The shapley value: Essays in honor of lloyd s. shapley," *The Economic Journal*, vol. 101, no. 406, pp. 644–646, 1991.

[42] S. M. Lundberg and S.-I. Lee, *Consistent feature attribution for tree ensembles*, 2018. arXiv: 1706.06060 [cs.AI].

[43] R. Khandelwal, *Understanding deep learning models with integrated gradients*, https://towardsdatascience.com/understanding-deep-learning-models-with-integrated-gradients-24ddce643dbf, [Online; accessed 02-May-2021], 2020.

[44] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, 2014. arXiv: 1312.6034 [cs.CV].

[45] M. Sundararajan, A. Taly, and Q. Yan, *Axiomatic attribution for deep networks*, 2017. arXiv: 1703.01365 [cs.LG].

[46] K. Bhardwaj, *Integrated gradients for deep neural networks*, https://medium.com/@kartikeyabhardwaj98/integrated-gradients-for-deep-neural-networks-c114e3968eae, [Online; accessed 29-April-2021], 2019.

[47] *Anaconda software distribution*, https://docs.anaconda.com/, version Vers. 2-2.4.0, 2020.

[48] Unknown, *Openai*, https://en.wikipedia.org/wiki/OpenAI, [Online; accessed 11-April-2021], 2021.

[49] Dexplo, *Explainability in deep reinforcement learning*, https://github.com/dexplo/bar_chart_race, [Online; accessed 10-May-2021], 2020.

[50] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, http://pybullet.org, [Online; accessed 01-May-2021], 2016-2021.

[51] A. Howard and N. Koenig, *Gazebo - robot simulation made easy*, http://gazebosim.org/, [Online; accessed 01-June-2021], 2019.

[52] prabodhhere, *Solving cartpole-v0 using reinforce*, https://www.kaggle.com/prabodhhere/solving-cartpole-v0-using-reinfor, [Online; accessed 07-April-2021], 2018.

[53] M. Ribeiro, *Lime documentation*, https://lime-ml.readthedocs.io/en/latest/index.html, [Online; accessed 20-April-2021], 2021.

[54] J. Mak, *Introduction to model interpretability*, http://web.stanford.edu/class/cs224u/materials/cs224u-2020-model-explainability.pdf, [Online; accessed 14-May-2021], 2020.

[55] T. Sigma, *Interpretability methods in machine learning: A brief survey*, https://www.twosigma.com/articles/interpretability-methods-in-machine-learning-a-brief-survey/, [Online; accessed 02-May-2021], 2020.

[56] A. Das and P. Rad, *Opportunities and challenges in explainable artificial intelligence (xai): A survey*, 2020. arXiv: 2006.11371 [cs.CV].

[57] A. Alqaraawi, M. Schuessler, P. Weiß, E. Costanza, and N. Berthouze, *Evaluating saliency map explanations for convolutional neural networks: A user study*, 2020. arXiv: 2002.00772 [cs.HC].

[58] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, *Fooling lime and shap: Adversarial attacks on post hoc explanation methods*, 2020. arXiv: 1911.02508 [cs.LG].

[59] A. Ghorbani, A. Abid, and J. Zou, *Interpretation of neural networks is fragile*, 2018. arXiv: 1710.10547 [stat.ML].

[60] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, *The (un)reliability of saliency methods*, 2017. arXiv: 1711.00867 [stat.ML].

[61] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, *Sanity checks for saliency maps*, 2020. arXiv: 1810.03292 [cs.CV].

[62] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, *Towards better understanding of gradient-based attribution methods for deep neural networks*, 2018. arXiv: 1711.06104 [cs.LG].

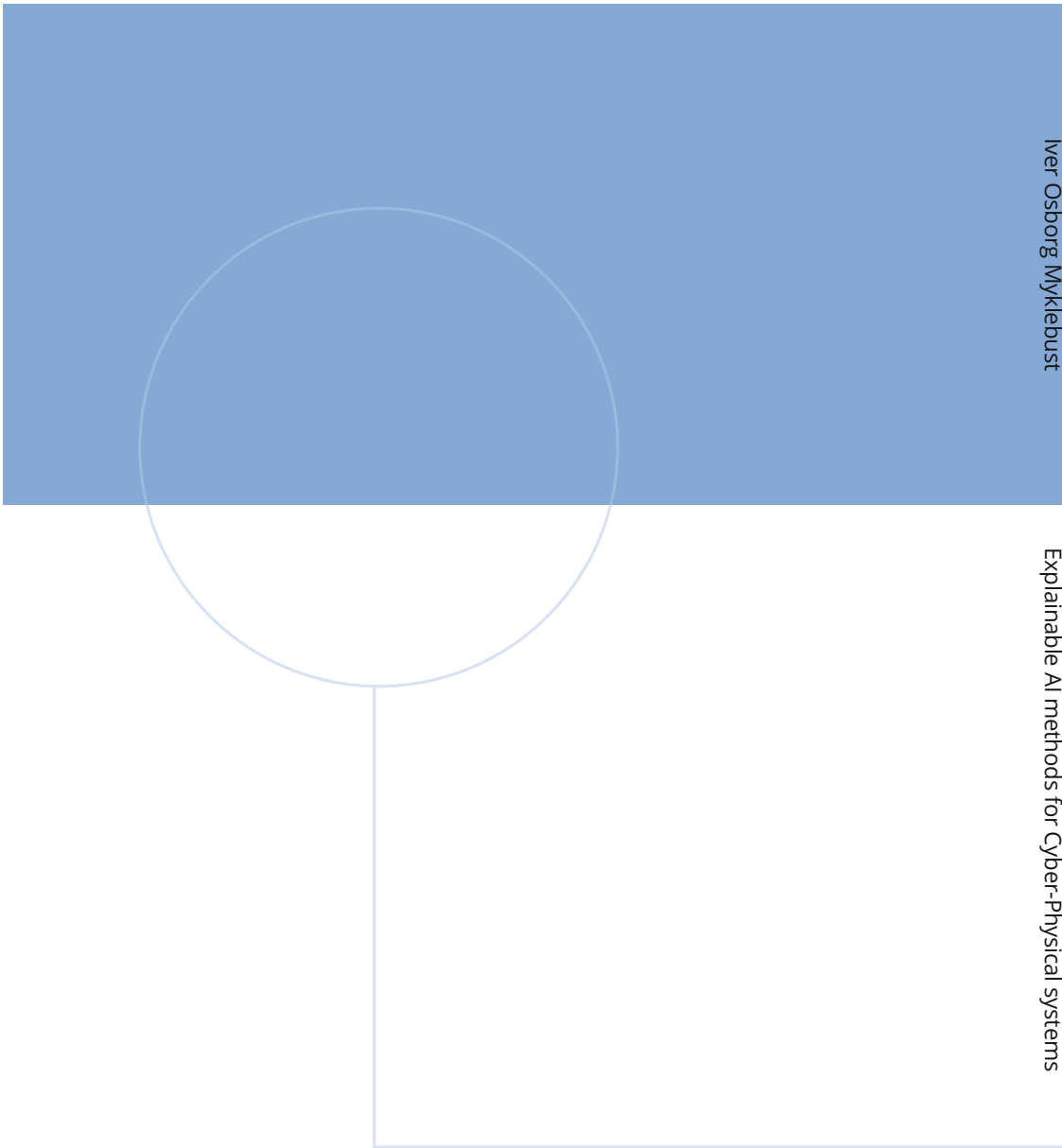
[63] G. Liu, O. Schulte, W. Zhu, and Q. Li, *Toward interpretable deep reinforcement learning with linear model u-trees*, https://arxiv.org/pdf/1807.05887.pdf, [Online; accessed 15-February-2021], 2018.

[64] F. C. Alexandre Heuillet and N. Díaz-Rodríguez, *Explainability in deep reinforcement learning*, http://web.stanford.edu/class/cs224u/materials/cs224u-2020-model-explainability.pdf, [Online; accessed 29-April-2021], 2020.

[65] S. Mohseni, J. E. Block, and E. D. Ragan, *A human-grounded evaluation benchmark for local explanations of machine learning*, 2020. arXiv: 1801.05075 [cs.HC].

[66] H. J. P. Weerts, W. van Ipenburg, and M. Pechenizkiy, *A human-grounded evaluation of shap for alert processing*, 2019. arXiv: 1907.03324 [cs.LG].

[67] E. Puiutta and E. M. Veith, *Explainable reinforcement learning: A survey*, https://arxiv.org/pdf/2005.06247.pdf, [Online; accessed 20-April-2021], 2020.



Iver Osborg Myklebust

Explainable AI methods for Cyber-Physical systems

NTNU
Norwegian University of
Science and Technology