Explainable Hierarchical Reinforcement Learning For A Cooperative Sorting Task

MSc Final Project - Written Report

Asger Printz Madsen
Márk Adamik

Aalborg University
Faculty of Engineering and Science
Board of Studies for Electronics and IT
7 Fredrik Bajers Vej
DK-9220 Aalborg

AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Title:
Explainable Hierarchical Reinforcement Learning For A Cooperative Sorting Task

Theme:
MSc Final Project

Project Period:
Spring Semester 2020

Project Group:
ROB 1065

Participant(s):
Asger Printz Madsen
Márk Adamik

Supervisor(s):
Matthias Rehm

Page Numbers: 98

Date of Completion:
June 3, 2021

Abstract:

This report presents the design and implementation of a Hierarchical Reinforcement Learning (HRL) system with a focus on explainability aimed at the non-expert users of the system, using a UR-3 collaborative robot. Taking a cooperative waste-recycling sorting task as a use-case, an experiment is conducted, where participants were assessed on the sorting task in a simulated environment while being presented different explanation types by the robot. The results of the experiment indicated that graphical explanations could lead to significantly better performance compared to no explanations.
These results are considered while designing the HRL-based system, which consists of three layers. The low-level policies of the system are presented and tested, while the mid-level and high-level layers could not be tested and are left for future work.

Contents

Preface

We would like to express our sincere gratitude to our supervisor Matthias Rehm, whose guidance and assistance greatly aided in the evolution of this report. Additionally we would also like to acknowledge the technical support provided by Carlos Gomez Cubero. This is the documentation of a study of human-robot interaction with an investigation of explainable reinforcement learning at its core, written by two 4^{th} semester MSc Robotics students at Aalborg University, Denmark.

Aalborg University, June 3, 2021

Asger Printz Madsen
<aspa12@student.aau.dk>

Márk Adamik
<madami15@student.aau.dk>

Chapter 1

Introduction

This chapter introduces the project premise, provides context for the topic of explainable artificial intelligence and lays out the motivation for pursuing deep reinforcement learning with robotic manipulators.

1.1 Motivation

As science and technology advance over time, the computers that already permeate our everyday lives are becoming increasingly powerful, equipped with algorithms that are able to make decisions autonomously. This increase in computing power and algorithmic efficiency gave the field of Artificial Intelligence (AI) an unprecedented pace of development.

Interest in a special sub-field of AI called Machine Learning (ML) spiked after successes such as AlexNet [1], which won the ImageNet object detection competition in 2012, beating handcrafted image classification algorithms, or Deepmind's AlphaGo [2], that defeated the world champion Lee Sedol at the ancient game of Go.

In recent years, practical applications of different machine learning approaches range from medical image analysis [3] through speech recognition and machine translation to marketing and financial market analysis [4].

As a result of the huge potential, some of the world's biggest technology companies, such as Facebook, Google and Amazon, have spent billions of dollars developing their own custom AI algorithms [5]. A surge in this area of research resulted in a 20-fold increase of publications on the topic of AI between 2010 and 2019, reaching around 20,000 yearly [5].

However, as the variance and power of these applications increase, so does the danger of misuse, and the impact of potential errors. Although researchers and

policy creators from all around the globe are trying to find ways to mitigate the risks posed by AI and many different approaches are proposed, no definitive solutions or regulations have been agreed upon.

1.2 Why Explainability?

Most of these systems that are built on machine learning, are based on a specific method called Deep Learning (DL). This method allows systems to make precise predictions fast, often exceeding the capabilities of humans, thereby allowing them to make more decisions and take more actions than ever before [6]. However, one of the biggest problems with the applicability of DL methods is the so-called "black box" nature of these algorithms. As the name indicates, these algorithms are hard to understand complex nonlinear systems, often with thousands or sometimes even millions of parameters. This makes the AI systems decision-making process hard, if not impossible, to understand.

Explainability could also help the user understand the system's strengths and weaknesses, and understand how will it work in the future, when will it give good results and when will it fail to do so. It could allow the users or developers to focus on specific problems and fix them quicker.

Lipton [7] points out that in the literature, the terms transparency, explainability and interpretability are ill-defined, contributing to a quasi-scientific perspective on the issue. Although no consensus exists on the precise definition of these terms, we interpret these terms relying on the definitions provided by [8], [6] and [9], where transparency means that the model is understandable by itself, without any explanations, explainability means the ability to provide an explanation with regards to the inner processes of the algorithm and interpretability is a qualitative measure of the explanations (i.e. how easy it is for the human to interpret the explanation in a meaningful way).

1.3 The Use Case

Keeping the definitions described above in mind, in this thesis the main focus is on explainability: providing an explanation to the user that reveals useful and relevant information about the inner processes of the system. This system integrates a specific type of machine learning method called Reinforcement Learning (RL). This method consists of the agent observing the effect its actions have on the environment and deriving a reward from its actions. This way the robot learns to map different environments to different actions and aims to maximize the rewards (see

Chapter 2 for more details). Reinforcement Learning has been successfully applied in robotics in several different contexts, such as autonomous navigation of cars [10] or UAV-s [11], robotic manipulators like surgical robots [12], or bipedal robots such as Aldebaran's NAO [13].

The greatest advantage of Reinforcement Learning is its ability to equip the agent with sufficient flexibility to handle unknown or dynamically changing environments [14].

One of the areas where robotic manipulators and DL approaches start to be efficient is waste recycling. Before the advent of DL methods, it has been challenging for classification systems to recognize different waste products, as every piece of trash comes in different sizes, shapes and colors. To get around this problem, robotics companies such as ZenRobotics [15], Waste Robotics [16] or AMP Robotics [17] are using Deep Learning-based vision systems to classify waste and employ robotic manipulators to sort them. Due to the advancements in image recognition techniques that implement DL methods, these systems could be trained on thousands of images to recognize different objects based on their qualities such as shapes, texture or even brand logo.

There are several reasons why applying robotics solutions to waste recycling is both desirable and necessary. Besides the obvious environmental impact of increased recycling capability, human and political factors also play a huge role.

Before 2017, roughly 1500 shipping containers worth of recycled waste was shipped every day from the US to China [18], where the manual labor utilized for the sorting of these materials was less expensive and therefore could result in profitable recycling. Similarly, the EU also heavily relied on exportation. In 2016 for example, 1.4 million tonnes of plastic waste was shipped to China. This changed in 2017 however, when under a program called "Operation Green Fence" China introduced a restriction on several different types of solid waste imports, which resulted in a drop to 50.000 tonnes of plastic waste imported from the EU in 2018 and a further drop to 14.000 tonnes in 2019. Although other countries such as Malaysia or India increased their import capacity to fill out the gap left by China, the restriction still heavily affected the global waste management systems worldwide [19]. Starting from 2021 however, China plans to completely ban all solid waste imports [20], further increasing a need for an efficient recycling solution.

Furthermore, another aspect of the emerging need for automated recycling solutions is due to the global COVID-19 pandemic, as some of the recycling plants cannot afford to operate, partly because of worker's increased risk of contamination from the waste [21], and partly because of the design of the work-spaces not being fit to keep the social distancing restriction [22]. This situation highlights the potential

opportunities in augmenting manual labor with robotic solutions.

The rest of the thesis, therefore, aims to tie these above-mentioned issues and methods together and propose a solution of using explainable deep reinforcement learning techniques to teach robotic manipulators how to perform a sorting task.

1.4 Outline Of The Thesis

In order to establish a basic understanding of Deep Learning, Reinforcement Learning and the current theories of the psychology of explanations related to Artificial Intelligence, Chapter 2 provides a brief discussion on the background knowledge and examines the related concepts and methods that are referred to later in the project.

Chapter 3 introduces some of the most recent Deep Reinforcement Learning (DRL) techniques and Explainable Reinforcement Learning (XRL) methods and examines their applicability in a cooperative sorting task.

Based on the information conveyed in the previous chapters, Chapter 4 presents a problem formulation that is broken down into separate research objectives the following chapters aim at reaching. A delimitation is given to specify and narrow the scope of the solution. Furthermore, requirement specifications are set for the overall system.

Chapter 5 details the design, process and results of an experiment conducted to measure the effect of different explanation types.

Incorporating the results obtained in the previous chapter, in Chapter 6 the methods and materials used as well as the design and implementation of the algorithm is detailed in greater detail.

Chapter 7 presents the results of the tests and assesses whether the components of the system fulfill the specification requirements set in the chapter 4.

A discussion of the results is provided in Chapter 8, where the different research objectives are evaluated in light of the results, and future work and possible improvements are also presented.

Finally, Chapter 9 concludes the project by summarizing the previous chapters.

Chapter 2

Background

This chapter starts out by introducing some of the basic concepts of Deep Learning and Reinforcement Learning, paving the way to the fundamental ideas of their combination, Deep Reinforcement Learning. The chapter closes with an outline of some of the theories of explanations that are used in robotics and artificial intelligence.

2.1 Deep Learning

Basic Principles Of Artificial Neural Networks

The fundamental building blocks of Deep Learning techniques are Artificial Neural Networks (ANNs), which try to imitate the principle behind brain functioning. Within a human brain, neurons form networks, firing pulses of electricity that propagate through the connections, activating other neurons. As humans go through experiences, these neurons strengthen or weaken their connections to other neurons. ANNs try to imitate this by the use of artificial neurons that are organized in layers. Given an input x and activation threshold, if the input to the neuron exceeds the threshold, the signal is forwarded to all the other neurons it's connected to in the next layer. The strength of the connections of the artificial neurons is represented by weights w, while a bias b is used to affect the activation threshold of the neuron. Each neuron therefore can be modeled as a function $f(b + \sum_{i=0}^{n} x_i w_i)$ where n is the total number of connected neurons. To increase computational efficiency, the weights and biases of each neuron of each layer are vectorized and stored in special types of matrices called tensors.
The structure of the neurons allows ANNs to mimic all the basic logic functions (e.g. NOT, OR, AND, XOR). However, as the network is combined of linear functions,

the result is also linear. In order to allow it to approximate nonlinear function, non-linearity is introduced by the use of activation functions.

There are several different types of activation functions in use, some of the most widely used of which are the ReLU (Rectified Linear Unit) and the Sigmoid. ReLU is a simple function that introduces a cut-off value at 0, meaning that $f(x) = max(0, x)$. As the negative values are all assigned to 0 in this case, and therefore potentially valuable information is lost, a variation called Leaky ReLU is introduced, where instead of assigning 0 to all negative outputs, they are multiplied by a small weight, therefore preserving some information while still introducing non-linearity. The Sigmoid function (also called logistic function), besides introducing non-linearity, has the special property that it maps the output into a range between 0 and 1. For this reason, a generalization of this function called the Softmax function is often used as the last activation function of the network to output a normalized probability distribution over the predictions.

ANNs can have different structures based on how the layers of nodes are connected to each other. The simplest structure is called *fully connected* layers, where each node in layer i is connected to every node in layers $i + 1$ and $i - 1$. Other structures however also exist, such as *convolutional* or pooling layers.

In this thesis, only fully connected layers are used with either ReLU or Softmax activation functions.

Deep Learning Networks

An ANN is called "Deep" if it has at least one hidden layer, meaning a layer between the input and the output layer.

In the case of a single hidden layer, each neuron of the layer acts as a binary decision boundary. These boundaries can be combined to produce an output that is able to classify more complicated data distributions.

The "learning" part comes from the concept that the weights and biases of the network can be adjusted based on the error between the actual output of the network, and the desired output. The error is usually determined by a loss function (sometimes called cost function) $J(\vec{\theta})$. The parameters of the network are often adjusted using a method called *gradient descent*, which using the partial derivatives of the loss function updates the weights and biases of the network. As calculating the gradients of the loss function can be computationally expensive, usually, a computationally inexpensive method called *back-propagation* is used to calculate the partial derivatives with respect to the parameters using the chain rule[23].

Today, deep learning networks are commonly combined with three main types of learning problems:

- **Supervised Learning:** Given input x, the system is trying to predict the output y. In supervised learning, labeled training data (correct output y) is provided, and the label acts as ground truth or supervision, as it can tell the system how far off its predictions are. The two main types of supervised learning models are classification models, where the predictions are discrete variables and regression models, where the prediction can be a continuous variable. Some examples of these types of deep learning applications are image classification or market price prediction respectively.

- **Unsupervised Learning:** Unlike supervised learning, unsupervised learning does not use labels in the training data, but instead it tries to detect structures and patterns within the unlabeled (also called unstructured) training data. An example real-life use case is anomaly detection.

- **Reinforcement Learning**: Falling between supervised and unsupervised learning methods, reinforcement learning uses the occasional rewards as labels. In the simplest case, the reinforcement learning agent associates the rewards with the actions and observations provided by the surrounding environment. When deep neural networks are combined with reinforcement learning, the method is called Deep Reinforcement Learning (DRL).

The following section focuses on the basic concepts of reinforcement learning and describes the working principles behind the method in greater detail, followed by some of the fundamental algorithms.

2.2 Reinforcement Learning

As described in section 2.1, reinforcement learning (RL), as a relatively new member of the deep learning family. Especially within the field of robotics, RL has seen a huge surge in popularity within recent years. In the following section, general concepts involving RL will be laid out.

RL allows an agent to make decisions and learns from these decisions in an environment. It does this by maximizing reward in particular beneficial situations. However, to fully understand RL the concepts will now be described and each of these will be given in a robotics context. The **Agent** is the entity performing the action and learning the behavior needed to complete the objective of the training. Within robotics, this agent can control the actions of a mobile robot or manipulator. The **Environment** is the scenario the agent has to perform within. This can be everything from simple 2D games to complex real-world conditions. Within robotics, advanced simulation software is often used as the environment. **Reward** is the

feedback the agents get when performing an action. These rewards can either be sparse or dense. Sparse rewards are only given when the agent reaches some goal within the environment. In robotics this could happen when a mobile robot manages to go from point A to point B. Dense rewards are given each time the agent is in a preferable state e.g. when the robot moves closer to a goal. **State** is the condition of the environment at any given time. In robotics, this can be the joint angles of a manipulator or the position of a goal. **Policy** is the learned behavior of the agent, which means its strategy is utilized to take an action in a given state. [24]

These concepts can be seen in the following graph 2.1 representing all these concepts work together.

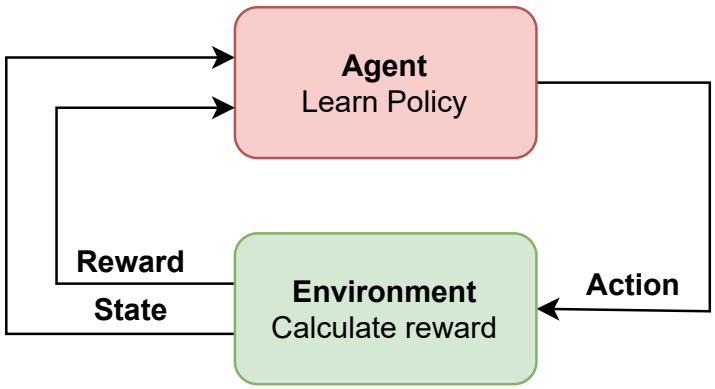


Figure 2.1: The relationship between the agent and the environment in reinforcement learning.

This method of an agent taking action and receiving a reward and state is called the Markov Decision Process. However, Markov Decision Processes assume a fully observable environment which is not always the case of RL applications. For these reasons, Partially Observable Markov Decision Process (POMDP) can be used. This method is formulated is a 7-tuple $(S, A, T, R, \omega, O, \gamma)$ where

- S is the state
- A is the action
- T is the conditional transition probabilities between states
- R is the reward
- ω is the observations
- O conditional observation probabilities

- $\gamma \in [0, 1]$ is the discount factor

At each discrete time step in the environment, the agent takes an action $a \in A$ based on a S, resulting in the environment transitioning to state s' with probability $T(s' \mid s, a)$. After this the agent obtains an observation $o \in \Omega$ of the new state of the environment, s', and on the just taken action, a, with probability $O(o \mid s', a)$. The agent gains a reward r equal to $R(s, a)$. After this, the process repeats. The goal is for the agent to choose actions at each time step that maximize its expected future discounted reward: $E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where r_t is the reward earned at time t. The discount factor γ determines how much immediate rewards are favored over more distant rewards. [24]

The immediate reward and future discounted rewards can be seen in the bellman equation 2.1 which is the function used to find the optimal policy.

$$V(s) = max_a(R(s, a) + \gamma V(s')) \tag{2.1}$$

where $V(s)$ is the value of a state, max_a is the maximization of the action given the immediate reward and future rewards.

2.3 Model-free Reinforcement Learning

Model-free is used in POMDP when the model if the environment is unknown or too complex to be used efficiently and instead data is sampled with sensors. This data can then be used as the state representation of the environment. This way of representing the state is often the only option when dealing with highly dynamic and complex real-world environments like the control of robots among humans. Model-free RL can be categorized into two policy evaluation methods, Monte Carlo and Temporal-Difference [24].

Monte Carlo

Monte Carlo (MC) learns from complete episodes of experience where the value of a state will be the mean of the return. This method attempts to find the optimal policy v_π from the sampled observation after each episode. Two types of MC learning policy evaluation methods exist [24].

- First-visit MC evaluation: The interaction with a state is only counted the first the agent visits a unique state. In order to evaluate states, the number of encounters of a state is set to zero, $N_{(s)} = 0$. The first time a state is visited the counter incremented by one, $N_{(s)} = N_{(s)} + 1$ and the total return $G_{(s)}$ is

calculated for that state. If the agent visits this state again the state is not counted. The value of the state is then estimated by $V_{(s)} = G_{(s)}/N_{(s)}$ and by the law of large numbers, the value will converge to v_π as $N_{(s)}$ approaches infinity [24].

Every-visit MC evaluation: The value of a state is calculated the same way as First-visit, however subsequent visits of the same state are counted within an episode.

Temporal-Difference

Instead of estimating the value of a state after each episode, Temporal-Difference (TD) learning attempts this after each time step t is $v(S_t)$. TD does this by substituting the remainder of an incomplete episode with an estimate. The estimation is called bootstrapping. The bootstrapped estimate is improved dynamically as the agent goes through experiences each time step within an episode. The estimate is based on the Temporal-Difference Error (TD-error) δ, which represents the difference between the expected value estimate from a state at time t, $v(S_t)$ and the actual reward gained by taking action A_t. $\delta_t = r(S_t, A_t) + \gamma v(S_{t+1}) - v(S_t)$. This is the error signal that allows the agent to better it's decisions. If the value is parameterized by w, the task becomes to choose the parameters so as to minimize the TD-error: $w_{t+1} = w_t + \alpha\delta_t \nabla_w v : w(S_t)$

As mentioned before, the difference between TD and MC is TD capability of learning before the end of an episode. In an environment related to robotics, this means that if a robot makes a mistake and collides with an object, MC will not learn from that experience before the end of an episode. TD on the other hand is capable of learning from this experience and using it throughout the same episode. This means that if each episode of the training consists of a lot of time steps, TD should be able to learn faster. [24]

2.4 Challenges Of Reinforcement Learning

In their review of reinforcement learning applications with robotic manipulators, Nguyen et al.[25] mentions three main issues even state-of-the-art reinforcement learning implementations face.

1. **Sample inefficiency:** It takes a long time to learn a movement, they learn from scratch and there is currently no straightforward way to incorporate new data.

2. **Exploration vs. Exploitation** questions address the issue of "Should the agent go for the uncertain, potentially great reward or settle for the reward that is known?". The problem in the case of manipulators is that the exploration can be dangerous for the physical safety of robots when performed in real-time.

3. **Generalization and reproducibility:** Most algorithms are fine-tuned for a specific problem or set of problems, making their implementation approach narrow. It is often hard to reproduce results from state-of-the-art papers, due to a lack of details on implementation. Even random seeds can cause vastly different performances. As the authors mention, no efficient bench-marking exists as of now.

The problem of sample inefficiency is overcome by implementing the DRL system in a simulation environment, where a powerful GPU can greatly speed up the training process. The problem of exploration vs. exploitation is partially solved. Although there is no physical threat posed to the robot due to the nature of the simulation, the lack of exploration can still cause the robot to get stuck at a local minimum.

The greatest challenge, however, that even the simulation environment cannot solve is the problem of generalization and reproducibility. There are many hyperparameters that could be tuned to increase performance, but tuning is a process that takes up a lot of time, without a guarantee of improvement. Some measures however can be taken, such as fixing the seed of the simulation and of the models to reduce the variation in the results.

2.5 Explanations In Artificial Intelligence

The aim of producing explainable artificial intelligence (XAI) systems is not a new trend. Researchers working on expert (rule-based) systems over 40 years ago were already considering what type of explanations an artificial agent should give to the user, and what are the requirements of these explanations in order to increase credibility and trust [26]. This is a reasonable approach since, in order to provide sufficient explanation, one has to understand what constitutes an effective explanation. As the explanation is aimed at and processed by humans, it is necessary to have some understanding of the psychological processes that lie beneath.

The matter of explanations however is an incredibly complex topic. It involves cognitive psychology, as it has to do with the way memories and knowledge are produced and stored in the brain, and it rests on the foundations of neuropsychology and neuroscience. Furthermore, as knowledge and memories are gathered in a social environment, the topic is also linked to social psychology [27].

The exploration of the topic of explanations in this project is therefore heavily limited and deals mainly with the theories and methods of other research within the field of human-robot interaction, with a special focus on mental models.

Mental Models

A systematic review of the literature around XAI was conducted by Anjomshoae et al.[28] and revealed that most studies that relied on a theory of explanation used a mental model-based approach (either Theory of Mind (ToM) model or folk psychology)

Other social scientists such as Miller [29] also argue that basing the system on

People create internal conceptualizations called mental models of both the environment and the agents acting in it. These mental models serve as a starting point for the predictions of events and actions that might happen around the people, and drive much of the social interaction with people [30], as they contain a representation of other agent's beliefs and intentions. Explanations can provide an understanding of an agent's behavior, as the uncertainty about the intentions and beliefs lying behind the actions of the agents decreases, therefore serving as a way to update these mental models. If people have insufficient or incorrect mental models of the robots that are surrounding them, the risk of the robot causing injury or harm is higher, while explanations can aid in creating more accurate mental models and adjust the trust in the systems [31].

Explanations can be evaluated based on the quality of mental models, asking questions like ("How does it work?" Why did it act like this?). Tullio et al. [32] for example queried the mental models people created of their system by asking them to describe the system components and their relevant importance to the system's functioning.

Categorization Of Explanations

There can be several different ways to categorize explanations, depending on which aspect is focused on. In ancient Greece, Aristotle for example put explanations of events into four categories based on their different causes [33]. Or, in a more recent study, Graaf et al. in their study measuring mental models of robots[34] for example grouped the explanations given by participants to describe the robot behaviour into *cause*, *causal-history* and *reason* explanations. One categorization that is considered relevant is one mentioned by Stange and Kopp [35], where they describe three different functions of explanations:

- **What-explanations**: These explanations are used to clarify a situation. In our specific use-case, a what-explanation could clarify what the robot is going to do, or which piece of waste it's planning to pick up.
- **Why-explanations** can reveal underlying reasons and effects of causality. Regarding the sorting task, these explanations can produce explanations why a certain can has or has not been picked or considered for sorting.
- **How-explanations** enable users to perform a specific intentional action, by providing relevant information about the method or means of achieving a goal. This function of explanation however is not considered relevant for our use case.

The survey by Anjomshoae previously mentioned determines six types of explanations discovered in the literature based on the presentation modality of the explanations. They have found that the most frequently used explanations in the studies they reviewed are textual explanations (47%), followed by graphical explanations (21%), logs (11%), expressive motions (11%), indication lights (7%) and lastly speech (3%).

In order to determine which modalities would potentially yield the most benefit to the end-users, the basic principles of explanations should be considered, as not all modalities could be applicable to every use case.

Principles Of Explanation

Johnson [36] distinguishes two different principles of explanations. An explanation should either transfer knowledge that allows users to carry out their tasks, or it should provide a reason for an action carried out or situation created by the explainer. Given the use case provided in Chapter 1, both principles could be fulfilled. The first principle of explanation could be fulfilled, as the robot's explanations might allow the user to pick different types of waste (what-explanation), and the second principle could also be fulfilled by providing an explanation regarding why a certain can has or has not been picked (why-explanation). In either case, the explanation could allow users to increase their productivity by adjusting their actions or planning ahead.

Gunning [37], [38] describes two main principles of XAI that should be respected by the designer of the systems. *Explainability* aims at giving sound, complete explanations, with the proper amount and type of information. *Correctability* refers to the extent to which the system is able to compensate for the occasional errors, and takes in user feedback. Within a reinforcement learning setting the correctability could be achieved by leaving the network open for learning, and updating the network parameters, however, the implementation of such a system exceeds the scope

of the project. Explainability, especially regarding the amount of information could be a question worth discovering, as some researchers such as the authors of [39] also expressed concern that too much explanation could have negative effects on both the user evaluation of the explanations and on the task performance.

Evaluating Explainability In AI

Hoffman et al. [40] proposed a trust measurement metrics for explainable artificial intelligence systems.

Schraagen et al.[41] put the metrics mentioned above to the test while examining the effect of different explanation techniques on the user's trust in the context of self-driving cars. The authors concluded that the test appeared reliable, although the validity was only shown for the third measurement. According to a plausible explanation, no trust was developed between the users and the system in the first two measurements. They compared three different types of explanations, causal explanation presenting causes (also called unintentional), intentional, where reasons are presented and mixed. The results showed a preference for intentional and mixed explanations. A serious limitation of the study was however the use of PowerPoint presentation as simulation, and only including textual explanations.

Although it could be interesting to test these metrics in a more realistic setting, they, however, seem to focus more on the argumentative and reasoning functions of the explanations, which might not be completely applicable to our use-case.

Holzinger et al.[42] proposed a system Causability Scale, which is a tool for the measurement of the quality of explanations. They define explainability as determining the parts of the machine's system that are having an influence on the model prediction, while they refer to the term *causability* as the human's understanding or interpretation of the causal elements of the explanation. It can be viewed as a mapping from the explanation between the machine learning model and the mental model of the user. Based on a System Usability Scale [43], a 10 item questionnaire with a 5-point Likert scale was developed for rating the causality of the explanations provided by the machine.

As with the previous metrics, this scale also puts a greater emphasis on cause-and-effect relationships, which might not be relevant to our use case.

A potentially relevant measure of explanation is proposed by Gunning [37], who describes five main measures for explanation:

- **User Satisfaction** refers to the user's subjective evaluation of the clarity and usefulness of the explanations.

- **Mental Model** refers to the user's understanding regarding the system's actions and choices.
- **Task Performance** can be used to evaluate the impact of the explanation on both the user's and the machine's individual or collective performance.
- **Trust Assessment** determines the future use of the system and the trust it induces.
- **Correctability** refers to the ability to identify and correct errors as well as the integration of the corrections to improve future performance.

The experiment design presented in another work of the author[38] consists of four conditions, where the explained system is tested with and without providing explanations, with providing partial information and finally a condition where a state-of-the-art "black box" system is used as a baseline performance.

Challenges And Opportunities

As the area of explainable artificial intelligence is a rapidly growing field, several challenges have been identified that could serve as opportunities for contribution to the field. As noted in [28], many studies of explainability remain in the conceptual domain, merely proposing a method that is not evaluated. Further directions for exploration and research mentioned in a review by Xie et al. [44] that could be interesting and relevant are:

1. Most explanations are aimed at experts and not at the layman. Explainable systems that produce easily understandable explanations accessible for laymen are rarely considered in the current literature.
2. The time-critical aspect of situations where explanations are presented is also not considered, meaning that it is often neglected to take into account how much time the user takes to reach a decision. This point could also be related to the amount of information or explanations that should be displayed to the user that is discussed above.

The first point could potentially be addressed in a cooperative waste-sorting scenario if the explanations are aimed at the other people cooperatively sorting the waste (who we would consider potentially unfamiliar with the programming and system design of the robot and therefore label as a layman). The second point could also be considered, as the sorting task usually involves a conveyor belt that introduces a sensitivity to the timing of the operations and a temporal limitation to the explanation evaluation and decision-making process.

An additional challenge is the tailoring of the explanations to the audience. Workers for example, who need to know what the robot they are cooperating or collaborating with is going to do next doesn't need the same level of explanation as a developer who aims to maintain or develop the controller of the robot. Another problem is that the comprehensibility of a model also depends on a users' cognitive ability, and on their familiarity with the topic. Therefore while trying to adjust the explanations to fit the principles discussed earlier it is also important to keep the audience and specific use-case in mind.

Lastly, an additional problem is the question of the technology's capabilities to produce an explanation. As discussed earlier in the chapter, deep learning methods have a "black-box" nature that might not allow every type of explanation to be extracted for the user. Therefore, in order to examine how recent works tackled the issue, the following chapter introduces some of the state-of-the-art solutions on explainable artificial intelligence and reinforcement learning.

Chapter 3

State of the Art

This chapter reviews some of the most recent literature on Deep Reinforcement Learning, Explainable Artificial Intelligence and Explainable Reinforcement Learning, with the application of these techniques to robotics solutions.

3.1 Deep Reinforcement Learning techniques

When combining RL with deep learning, it is called Deep Reinforcement Learning (DRL). In the following sections, two influential DRL techniques will be explained.

Dueling Deep Q-Network

Dueling Deep Q-Network is a part of the value-based methods within DRL. Value-based methods attempt to estimate the expected return, the Q value, given a state and an action and by this create a policy of how to act in an environment. This method was used in one of the greatest breakthroughs in RL called Q-learning. In its most simple form Q-learning can be seen in equation 3.1,

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a_t) - (s_t, a_t)) \tag{3.1}$$

where Q is the value of being in a state with a given action, s and a are the state and action, respectively. α is the learning rate, γ is the discount factor and t denotes the time.

When a neural network is used as the function estimator for the Q-value this method is called Deep Q Learning (DQN) which has been highly influential in the field of RL. This algorithm is known for being able to handle simple environments with discrete actions with superhuman performance.

In 2015 Wang et al. [45] improved the original version of DQN with a new architecture called Dueling DQN. Instead of just estimating the Q value of a state

and action, this new method splits up this term into a value of a state and an action advantage, which can be seen in the following equation,

$$Q_t(s_t, a_t) = V_t(s_t) + A_t(s_t, a_t) \tag{3.2}$$

where V_t is the value of a state and A_t is the action advantage. This change is useful since it is sometimes more important to get a value of the state that the agent is in and sometimes it is more important to get the value of each state-dependent action. An example of this could be sitting on a bench in sunny weather. Here the value of the state is high since the weather is good and no matter what action you take, it will continue to be high. However, then it starts to rain and the state value becomes low and each possible action is not to be the contributing factor. Like this, the agent will learn better when a state is generally good and when actions are preferred. After each term is estimated they are aggregated together however simply adding them which each other leads to the problem of identifiability. It is impossible to know which term had the largest influence on the outcome. To solve this the maximum of the action advantage can be subtracted from the Q value leaving out only the state value. This can be seen in the following equation,

$$Q_t(s_t, a_t) = V_t(s_t) + (A_t(s_t, a_t) - max_{a' \in A} A(s_t, a_t)) \tag{3.3}$$

where the subtracted term is the maximum of all actions belonging to the action advantage. This can be further improved by taking the mean of the action advantages, which loses the original semantics of the state and action advantage terms since these are now both off by a constant value, however doing this is shown to improve stability of the learning. This improvement is seen in the following equation,

$$Q_t(s_t, a_t) = V_t(s_t) + (A_t(s_t, a_t) - \frac{1}{|A|} \sum_{a'} A(s_t, a_t)) \tag{3.4}$$

where the average is now subtracted. The full architecture of the Dueling DQN with the aggregation layer can be seen in the following figure. To conclude, the DQN networks described will be used whenever discrete actions are preferable by the agent. [46]

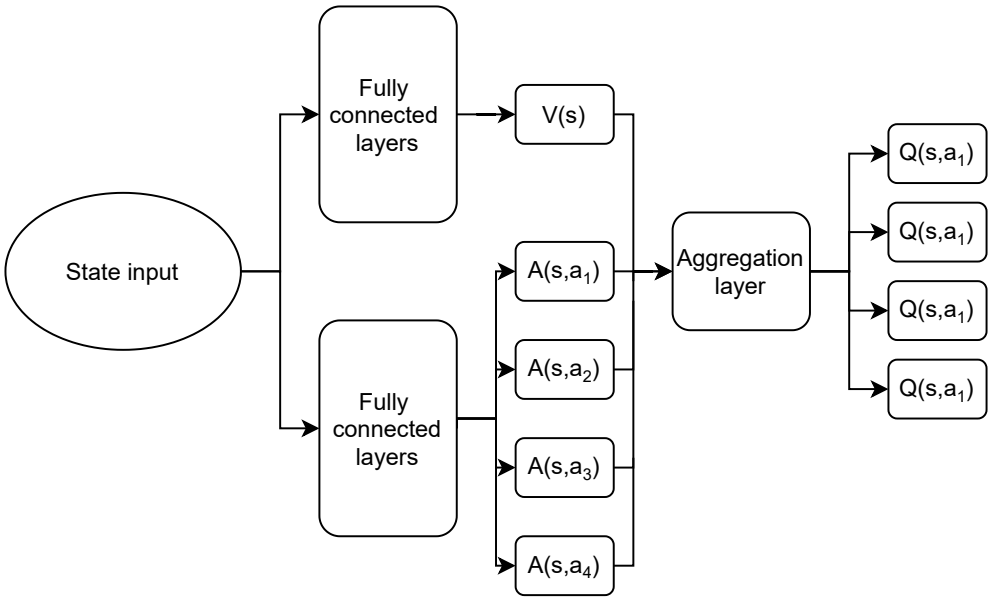


Figure 3.1: Architecture of the dueling DQN network

Soft Actor-Critic (SAC)

Actor-critic is a method of separating the memory structure to represent the policy independent of the value function. The policy part of the structure is the actor since it selects actions, and the estimation of the value function is the critic since it criticizes the actions made by the actor. The critic learns about and critiques whatever policy the actor is following. One of the most known actor-critic algorithms is called Soft Actor-Critic (SAC). SAC uses the idea of maximum entropy learning, which means that instead of trying to maximise a discounted future reward, it will attempt to find the optimal policy through entropy regularized rewards. This will encourage exploration dynamically as the agent learns the environment. This can be seen in the objective function,

$$\max_{\pi_\theta} E[\sum_t \gamma^t (r(S_t, A_t) + \alpha H(\pi_\theta(\cdot|S_t)))] \tag{3.5}$$

which consists of the reward term $r(S_t, A_t)$ and the entropy term $\alpha H(\pi_\theta(\cdot|S_t)))$ where H is the entropy and α is the weight this entropy should have.

Because of this entropy term, SAC does not add noise to the training in order to smooth policy like other actor-critic methods. Through entropy, SAC trains a stochastic policy which inherently adds noise to the training to a similar effect. An added advantage of SAC is therefore not having to discover a suitable amount of

noise to explore the environment. For this reason, SAC is one of the chosen RL methods to be used in this project. [46]

3.2 Deep Reinforcement Learning with robotic manipulators

This section reviews some of the state-of-the-art DRL solutions focusing on using robotic manipulators. Noting the different state spaces, action spaces and reward structures used by these studies could give a deeper understanding of the nature of the problem and inspiration to apply it to our specific use-case. The two main tasks that are of particular interest are trajectory planning and grasping tasks.

Trajectory planning

Zhan et al. [47] introduced a Framework for Efficient Robotic Manipulation (FERM), that allows agents to learn tasks like opening a drawer or picking an object in between 15-50 minutes of training time using the SAC algorithm. The observation the agent receives is based on the raw pictures of a RealSense camera, and the reward structure consisted of a discrete reward of -1 if the task was not completed, and 0 otherwise. The nature of the action-space (joint space or Cartesian space) is not specified. The main contribution of the framework is to pre-train the agent using expert demonstrations, which significantly improved the performance of the algorithms.

Grasping tasks

Joshi et al.[48] developed a DQN-based network architecture called Grasp Q-Network for robotic grasping. The reward function used by the authors consists of a success reward of 10, a partial success (contact) reward of 1, a reward of -1 for inactivity, when no action is taken, and a constant negative reward of -0.025 to incentivize quick solutions. The algorithm achieved a success rate of approx. 80% when grasping a cylinder-shaped object. The observation space used consisted of RGB-D inputs of cameras and joint angles, while the action space consisted of a Cartesian position of the tool center point.

Zeng et al. [49] used a vanilla DQN network extended with convolutional neural networks to learn a grasping task from a processed camera image as the state representation. Interestingly, instead of motor commands, the authors used primitive actions for robotic grasping, where the grasping action at location q consisted of the

TCP moving down $3cm$, grasping, then moving up $3cm$. A simple reward structure of giving a reward of 1 for every successful grasp is used. The algorithm managed to successfully grasp the objects 77.2% of the time in a simulation setting, and 83.3% of the time in a real world environment.

3.3 Explainable Artificial Intelligence techniques

This section starts by looking at the different types of XAI algorithms. Gunning [37], [38] distinguishes three main classes of XAI algorithms depending on the model structure and explanation extraction method:

- **Deep Explanation** refers to DL models that modify or extend the model to be explained. Researchers for example have attempted to either use deconvolutional networks to visualize and explain the convolutional layers, or developed techniques that associated semantic concepts with individual nodes or node structures [38]. An example of this method could be the Gradient-weighted Class Activation Mapping (Grad-CAM).
- **Interpretable Models** are structured causal models that try to abstract causal relationships from the underlying models. An example of this structure is a Bayesian rule list, which presents an explanation as a collection of if-else statements and logical operators, assigning a prediction probability to each clause [50].
- **Model induction** is a technique, where an underlying "black box" model that is to be explained, is probed and experimented with until an explanation could be inferred by inspecting the input-output relationships. Popular explainability techniques such as *Shapley Additive Explanations* [51] could be categorized as a model induction method.

The rest of the section introduces an example of some of these methods, and examines their applicability to the use-case of the thesis.

Grad-CAM

While convolutional neural networks have allowed for unprecedented advancements in feature detection, only a few techniques have been developed for explaining these networks. One of these is the Gradient-weighted Class Activation Mapping (Grad-CAM) method, which uses the gradients of the last convolutional layer to produce a heatmap highlighting important regions and thereby explaining the underlying post-hoc attention of the classifier.

Selvaraju et al [52] proposing this technique, also described as being applicable for reinforcement learning, without architectural changes or re-training. As mentioned previously, Grad-CAM uses the last convolutional layer to generate the heatmap, which the authors of the original paper argue has the best combination of high-level semantics and spatial information. Grad-CAM generates this heatmap in three steps. Firstly, the raw output of the last layer is used to calculate the back-propagation score for a class c y^c with respect to the feature maps A. The following step then takes a global average pool of these gradients to highlight the important weights. Steps one and two can be seen in Figure 3.2 taken from the original paper, where α is the importance weight of each feature and y^c and A are the score of the class and the feature map, respectively. These gradients are taken in relation to the height and width of the feature map, i and j, and following that, the global average pooling over the same dimensions.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

Figure 3.2: Equation explaining the step of calculation of the gradients and global average pooling of the Grad-CAM method.

Finally, the Rectified Linear Units (ReLU) technique is applied to highlight the features that contribute positively. Without this final step, too much of the feature map would be explained in the heatmap, resulting in poor localization performance.

This method could be used in our use case as an explanation of why certain waste products have been classified into their respective categories. These explanations however would mostly benefit the developer of the system trying to improve the performance of the object recognition software instead of giving an explanation to the workers, and therefore might not yield useful explanations in a cooperative sorting task.

Shapley Additive Explanations (SHAP)

Shapley Additive explanations (SHAP) was developed by Lundberg et al [51] as an attempt to provide a general and intuitive way to explain the importance and contribution of input features of machine learning models, by providing an importance weight to each feature. The method originated from a solution proposed by Lloyd Shapley to a problem in game theory, which is *"How should we divide a payout*

among a cooperating team whose members made different contributions?" [51]. This problem aligns with the machine learning setting, where instead of team members, the contribution of the different features is the target.

The authors argue that the way the values are distributed *"aligns with human estimation"*[51]. This argument is supported by two claims. Firstly, the importance weights can be considered contributions of different features to the output in terms of the features' actual value. In a reinforcement learning setting, this corresponds to a contribution to the actual reward received for the given action, which indeed aligns with human intuition. Secondly, the sum of the importance weights adds up to the total value given by the machine learning model, meaning that in the case of reinforcement learning, the reward can be completely broken down to the contribution of the features, or in the case of a reinforcement learning setting, to the contribution of state elements.

Although this method could indeed work well with standard problems machine learning solves in the economic fields, such as predicting prices or risks, where the different features are corresponding to easy-to-understand concepts, we argue that within robotics, where the different features correspond to the joints angles of the manipulator, the explanation might not end up being intuitive, and would yield less practical understanding of the underlying system.

One review [53] noted that using SHAP values could be a way to provide a general explainability model for DRL-based systems.

3.4 Explainable Reinforcement Learning techniques

Developing Explainable Reinforcement Learning techniques have two-fold goal [38]. One of the goals is to develop strategies that explain the actions and decisions made by the agent. The other goal is to develop new ways to implement RL, that allows for learning inherently explainable policies.

Two main surveys were found that assessed the state-of-the-art XRL techniques. Puiutta [54] divided the type of the explanation provided along two axes. On one axis, the explanations could be global, meaning that they aim to explain the structure of the model, how the whole model works, which puts the emphasis on trusting the model. The explanations can also be local, which focuses on explaining particular decisions. According to the author, local explanations tend to focus on increasing the user's trust in the predictions. The other axis relates to the structure of the model. Puiutta distinguishes between intrinsic models that provide an explanation during operation or training, or post-hoc models that provide an explanation of the trained model.

Autonomous policy explanation

Hayes et al. [55] outline a method that allows reasoning and question-answering over policies learned through RL. The authors implemented a natural language inquiry-answering system, that is able to provide meaningful explanations of the underlying policy. First, the algorithm determines the context the inquiry relates to by mapping it to a set of relevant states. The relevant states are encoded using Python decorator functions into a verbal representation of states and actions called behavior model. This way every step of the MDP is grounded in natural language representation and using boolean classifiers as predicates, the actions and states are mapped to string representations, which can be accessed and manipulated. The inquiries are based on pre-defined templates.

Although the method proposed by the authors is a novel and intuitive way to provide global explanations, two main issues arise regarding this solution. One is the assumption that the action-space of the agent and the state-space are intuitively understandable and presentable in a natural language format. This however is not always the case, as in case of manipulators, the state-space and action space could consist of joint positions and velocities, which has an inherent mathematical representation and might not convey useful information by itself. The other issue is the use-case of the explanations, as the algorithm is meant to explain its behavior after an action has been executed, and the explanation is used as means of fault diagnosis, in contrast with our scenario described in Chapter 1, where the robot should be able to provide explanations on an ongoing basis as to increase the productivity, thereby preferring an intrinsically explainable model.

Hierarchical Reinforcement Learning

Actions in real life, in many aspects, reflect a hierarchical structure, with several layers of abstractions and reasoning embedded. If one wants a robot that cooks dinner autonomously, it has to make some high-level decisions, such as choosing a recipe, then choosing and getting ingredients, placing them from one place to another, while it also needs to make low-level decisions relating to the movement of the joints. How to chop optimally without injuring oneself.
This complexity and gradual learning are present in humans as well, that learn from an early age, gradually how to move around and handle different objects, while at later ages they work their way up to the higher-level decision making.
This aspect of human learning and its application in modern machine learning is called Hierarchical Reinforcement Learning(HRF). The goal of this framework is to divide a complex task requiring multiple skills into several smaller and simpler subtasks. By doing this, a top level policy is trained to choose the best sub level

policy for a given sub task. Shu et al. proposed this framework for training an agent in Minecraft with the goal of stacking blocks on top of each other. Here the top level policy instructed the agent to stack blue game objects and the sub-policies consisted of sub-tasks such as find object, move object, and put object.

These human-defined sub-tasks are assumed to make the policies and decisions of the agent inherently more human-interpretable and thereby providing explainability to the system. Furthermore, this method of decomposing a complex task was shown to improve learning efficiency compared to a flat policy and additionally generalize unseen environments better. The greatest downside to employing this method is the weak human supervision that must be applied in order to divide the general task into smaller subtasks which can vary in difficulty from application to application [56].

Duan et al.[57] designed a HRL architecture to control a self-driving car. The architecture had only two layers, where first the low-level layer was trained, each policy trained to accomplish simple maneuvers, such as taking over or keeping in the lane. Once the low-level policies were trained, a high-level master policy was trained to choose the appropriate maneuver based on the current state. This way the high level policy could produce actions that are high level, and therefore interpretable for humans. All the networks were trained using fully connected layers, and their custom algorithm inspired by A3C (Asynchronous Advantage Actor-Critic). This solution's structure could be highly desirable in case of controlling manipulators, as low-level actions, such as joint angular velocities or tool center point positions do not carry inherent interpretability, whereas high-level actions could have predictive capabilities.

Beyret et al. [58] were applying explainable hierarchical reinforcement learning techniques to a robotic manipulator. As opposed to the previously mentioned work, where the high-level policy was choosing between actions given by pre-trained low-level policies, here the high-level policy was used to generate a trajectory and break the overall motion down to the easier task of reaching sub-goals. The policies were trained using DDPG combined with Hindsight Experience Replay to make the training sample efficient and apply sparse rewards.

Osa et al.[59] proposed a hierarchical reinforcement learning architecture for robotic grasping. In their work, the low level policies learn multiple grasping strategies that can be applied to different objects, while the high level policy chooses the appropriate low level policy (grasping strategy) given the object at hand.

Summary

This section presented some of the existing state-of-the-art explainable DRL techniques. Although several methods could prove to be a feasible solution, we chose hierarchical reinforcement learning as the most promising candidate for our use-case. The reason is in part due to its inherent transparency in terms of architecture, as the policies could be trained individually, and their output could be presented to the user as an explanation. Depending on the structure of the system, these explanations could be presented in a format that is understandable to laymen, therefore addressing the lack in literature as remarked by [44].

An additional, although related reason for the application of HRL is its ability to make movement patterns discrete, as each pattern is performed by a separate policy. This is relevant for explanations as [60] noted that in the case of an intentional action performed by the human, a prerequisite of the explanation consists of the classification of movement patterns as a specific action, performed by a specific agent.

The following chapter summarizes the relevant information presented in the report thus far, and combines it into a problem statement, which could be decomposed into research objectives and requirement specifications.

Chapter 4

Problem formulation

Through this project, a system is going to be developed. This chapter starts by presenting the research objectives that should be investigated to determine which aspects of the explainability should be implemented in the system, followed by the introduction of the requirements and limitations of the system, which together set a framework for the rest of this project.

4.1 Summary of related findings

The previous chapters presented some background information relating to both deep learning, reinforcement learning and theories of explanation, as well as introduced some of the state-of-the-art solutions.

Hierarchical reinforcement learning framework has been chosen as the basis of the solution. The gap in the existing literature addressing explainability for non-expert (layman) users is considered a problem that could be addressed with our solution, as the explanations provided during a cooperative sorting task would be addressed to the workers the sorting is carried out with. However, as mentioned by [44], an explanation might not even be needed for the task, as it might only take extra processing power from the users without adding much benefit. Therefore it has to be assessed if the explanation provides any extra improvements in the performance of the task.

4.2 Research objective

The focus of the remainder of the project is to explore the solutions to the following research question

How can a system with Explainable Hierarchical Reinforcement Learning be developed to increase the task performance of a cooperative sorting task?

Research objectives

- **RO 1 (Explanation effect)** Investigate whether there is an effect of explanations provided during a cooperative sorting task.
- **RO 2 (Explanation modality)** Investigate which modality should be used to communicate the explanation in order to obtain the greatest increase in task performance.
- **RO 3 (System design)** While considering the results of the above formulated objectives, design a system incorporating hierarchical reinforcement learning that is capable of performing the cooperative sorting task.

4.3 Delimitation

The main delimitation of the system is that it is a proof of concept, meaning that the main focus will be on a proof of concept, rather than a fully functional system for an end user.

Virtual environment

Due to the safety and performance issues related to training a reinforcement learning agent on a real robot (as detailed in Chapter 2), the development of the system is mainly taking place in a virtual environment. The potential to transport the agent to a real-life robot is considered while designing the system.

Available equipment

Aalborg University has provided a UR-3 manipulator that could be potentially used as a platform to transport the trained agent for real-life testing. The UR-3 is a 6-axis table-top collaborative robot with a reach of $0.5m$, that is designed to be able to perform collaborative pick and place operations, such as the sorting task [61]. The is equipped with a Schunk EGP 50-NNB parallel gripper, designed for small components. It has a stroke length of $8mm$ for each jaw, and is capable of exerting maximum gripping force of $215N$ [62]. The stroke length poses an additional limitation on the size of the objects the robot is able to manipulate with this type of gripper.

Therefore the system is limited to the sorting of aluminium cans, due to their simplicity in terms of shape and their availability both in real life and in simulation environments. However the system should be scalable to function with other types or multiple types of objects.

Additional equipment

As the focus of this project is mainly on the aspects related to robot control and explanations, additional elements of the system that would be required to provide the observation from the environment, specifically a depth camera with object recognition software is considered out of the scope of this project and its functioning is taken for granted. To provide the observation a simulated camera is used therefore with built-in object recognition software.

Explanations in light of the task

Considering the nature of the task poses a limitation on what kind of explanations are available. As mentioned in Chapter 2, the type of explanation needed depends on the audience. In this thesis the explanations are targeted at the "users" of the robot, the humans performing the collaboration (e.g. sorting task).

4.4 Requirements Specification

In order to be able to assess the performance of the system, requirements must be set. As the hierarchical reinforcement learning agent consists of multiple elements, two categories of specifications are determined. *Unit test specifications* determine requirements that could be tested separately, by examining parts of the system separately, while *integration test specifications* lists the requirements the system has to fulfill as an integrated whole.

Unit test specifications		
ID	**Requirement**	**Description**
UT-1	The system must be able to create and perform trajectory to a specific pose within the workspace within a success rate of 90%.	The generation of trajectories through reinforcement learning is required to fulfill the task of interacting with the objects within the objects. Failing to meet this requirement could lead to damaging of the robot.
UT-2	The system must be able to create a trajectory to a specific object to be grasped within the workspace within a time of 1 second.	The time constraint of the system is important to ensure a functional system for the user they interact with. Failing to meet this requirement could lead to inadequate performance and thereby counteract the human-robot interaction aspects of the system.
UT-3	The system must be able to grasp objects with a success rate of 90%.	In order to manipulate objects the system is required to grasp these objects on a variation of different positions. Failing to reliably grasp an object multiple times would lead to a much worse operational performance of the system.
UT-4	Timing requirement: The system must be able to time the grasping operation, adjusting to the speed of the conveyor belt.	A moving conveyor belt introduces a time constraint on the sorting task, making the timing of specific operations, such as grasping a crucial element of the operation.
UT-5	Explanation requirement: The system must be able to indicate where it's at in the decision process.	The system's inherent explainability should be utilized to inform the user of its intentionality. This is done to alleviate stress factors related to the user having to guess to behavior of the system.

Table 4.1

Integration test specifications		
ID	**Requirement**	**Description**
IT-1	Time requirement: The system must be able to perform the operation within 5 seconds.	As the applicability of the system to real-life scenario is crucial, a time limit has to be added.
IT-2	Sorting performance: The system must be able to perform the sorting operation correctly 90% of the time.	The sorting operation can be considered correct if the cans are successfully grasped and moved to the desired location without dropping.

4.5 Project structure

As the results of the investigations of RO 1 and RO 2 directly influence the design choices of the explainable reinforcement learning system (RO 3), first an experiment is conducted to answer these objectives.

The results of the experiments are then serving as a basis for the considerations of how explainability should be introduced and what modality should be used.

Following the experiment, the system is designed, parts of which are evaluated individually, and assessed whether they fulfill the unit test requirements described above, after which the parts are integrated into a single system and evaluated once again using the integration test specifications.

Chapter 5

Experiment

In order to explore the effects of explanations on the task performance and user experience, an experiment is conducted. This chapter presents the details of the different aspects and overall design of the experiment.

5.1 Introduction

To verify that implementing a system where explanations are provided by the robot during a collaborative waste sorting task is a worthwhile effort, an experiment is conducted using a virtual environment.

In order to design a situation where explainability is required, the conditions in which the necessity of explanations arises are examined. Malle et al. [63] describe three conditions for an explanation to be desired by a person. The person has to be 1) aware of the event that is occurring, 2) Have no complete understanding of the event, characterized with a certain level of uncertainty and 3) Find a personal explanation personally relevant.

In the case of a cooperative sorting task, all these conditions could be met, as the workers are aware of the situation and the task at hand, have a sense of uncertainty regarding which objects the robot will take, and could find it personally relevant in case the task performance is considered important.

The experiment, therefore, consisted of a cooperative sorting task, where the users, working together with the UR-3 manipulator, tried to sort different colored aluminium cans. The robot had a display connected to it, where the explanations were provided. The experiment was conducted in person at the Human-Robot Interaction Lab of Aalborg University. The participants consisted exclusively of volunteers.

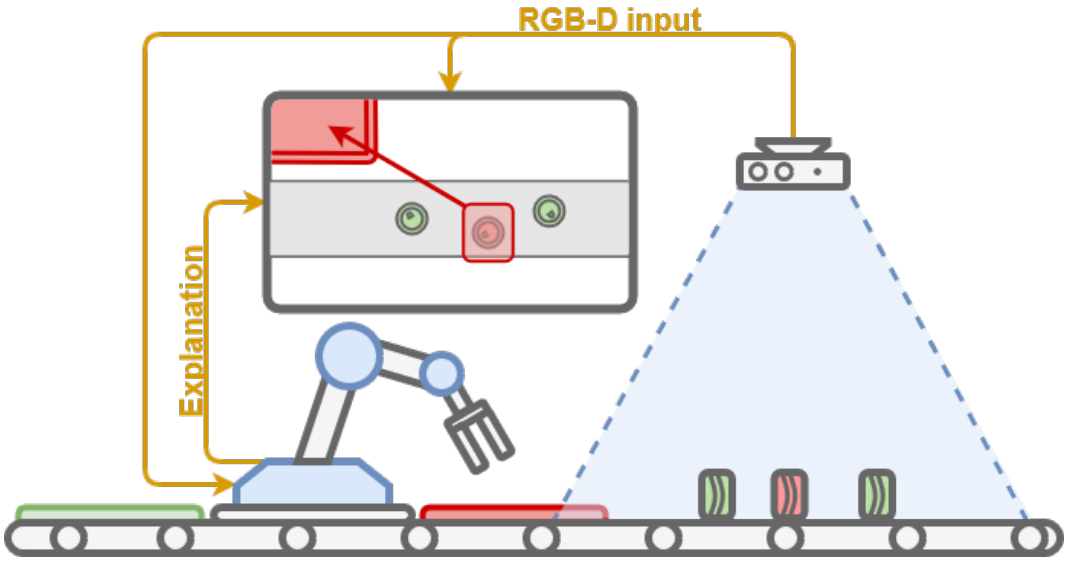


Figure 5.1: An overview of the system as it would be conceived in a real life setting. A depth camera would provide RGB-D images to the robot, which after running the necessary pre-processing and HRL algorithm would provide an explanation that is shown to the user on the display. As real life implementation is the end goal of the system, the environment created in the simulation follows this setup.

The exact experimental procedure with the implementation is detailed in the following section. The rest of this section provides an overview of the hypotheses proposed as well as the experimental design choices.

5.2 Hypotheses

The aim of the experiment is two-fold. One of the purposes is to find empirical evidence that could support the arguments, that providing an explanation has potential benefits. The second purpose is to compare the effects of different modalities and explanation types by which the explanations are displayed. Therefore the following hypotheses were proposed:

Hypotheses on explanation effect

- **H1 (Task performance):** The main aim of providing an explanation is to increase the task performance. Accordingly, the first hypothesis states that **providing an explanation improves the collective task performance**.
- **H2 (Cognitive load):** As explanations may reduce the uncertainty regarding the robot's actions, it seems plausible that in a collaborative task the

cognitive load participants feel while performing the task is also affected. The second hypothesis is, therefore, that **providing an explanation affects the cognitive load of the task**. The hypothesis is left two-tailed however as it is not certain if the explanations increase or decrease the cognitive load, as they might take additional cognitive power to process, in which case it would increase the cognitive load, but on the other hand it could also reduce the uncertainty and the stress associated with it, in which case the perceived cognitive load would be lower.

- **H3 (Effect on trust):** As recommended by [37], one of the main measures of the explanation is the amount of trust it induces, with an increase in perceived trust indicating higher explanation quality. Therefore we hypothesize that **providing an explanation increases the perceived trust users feel towards the robot.**

Hypotheses on explanation modalities and explanation types

In the collaborative sorting task in our use case, two different modalities are compared, namely *graphical explanations* and *written explanations*. These modalities also convey different types of information as categorized by [35]. Therefore the second set of hypotheses tries to explore the differences between the two modalities: **Effect of modality on task performance: We hypothesize that there is a difference between the effect of the explanations on the task performance (H4), cognitive load (H5) and perceived trust (H6).**

5.3 Experimental Design

The independent variable in this study is the explanation modality and type that is provided for the user. The screen that is connected to the robot is used to display the information in each condition. Inspired by the [38], independent-measures (between-subject) design was chosen, where participants performed the experiment in one of the following conditions:

- **Graphical explanation:** In this condition, the can of the robot's choice was highlighted on the screen by drawing a rectangle around it, as well as an arrow indicating which bin the robot intends to put the can in. The explanation in this condition can be considered as a "What" explanation, clarifying the action of the robot.

- **Textual explanation:** In the textual condition (also called written condition), the explanation consists of displaying a short written message above the

cans that are either 1) chosen, by displaying a number above them representing their rank in terms of priority, or 2) not considered at all due to difficulty grasping, or perceiving a color that is not the robot's responsibility to sort, or 3) considered unreachable by the robot. These explanations could correspond to a combination of "what" and "why" explanations, as the number provides a clarification for the robot's actions, indicating *what* it's going to do, and the texts regarding the ability to reach the can in time or grasp the can provide a reason *why* a certain can is or is not considered.

- **Both graphical and textual explanations:** In order to explore if these explanations have any additive effects when combined, a condition is introduced where both types of explanations are simultaneously displayed on the screen.
- **No explanation:** This serves as a control condition, where no explanation is provided to the user, but the screen simply displays the view of the camera.

An example of the four conditions can be seen in Figure 5.3. The rationale behind choosing a between-subject design is to avoid the carry-over effects stemming from the familiarity with the sorting tasks and the robot's behavior, as well as to limit the effects of fatigue in participants. Within-subject design or mixed design however could have been conceivable and was briefly considered, mainly due to the reduction in the number of participants required and greater sensitivity to experimental manipulations.

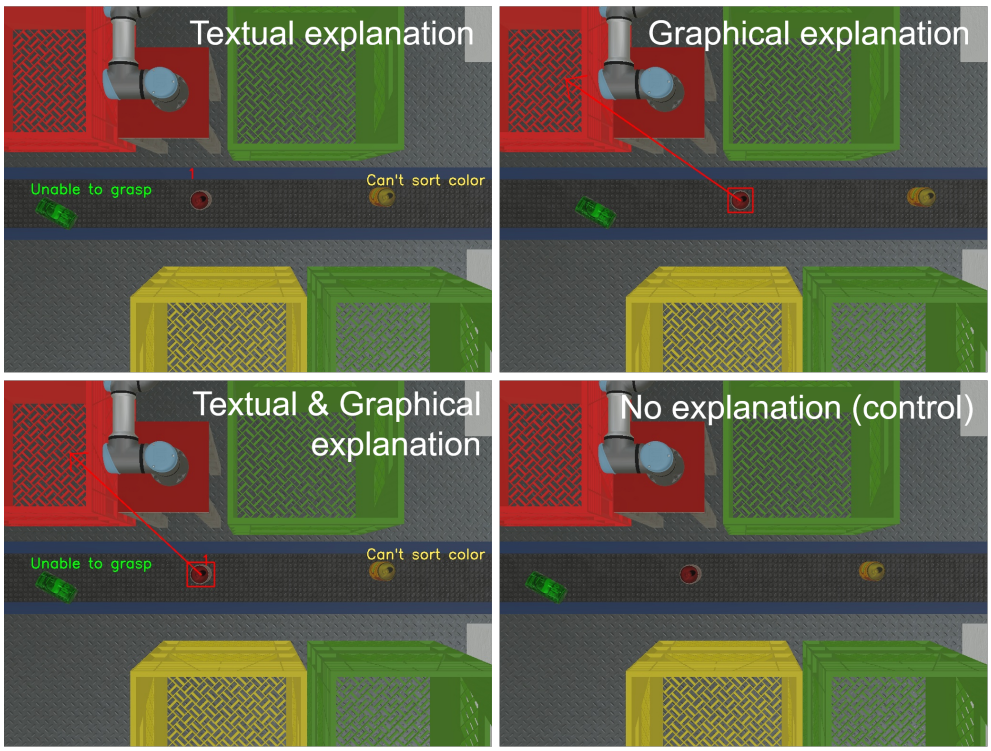


Figure 5.2: Sample images from the four different conditions of the experiment. These same example images were used for the introductions to the conditions.

Unsorted Cans

In both games the participants played, a total of 15 cans out of the 40 were either mislabelled, or laid down on the conveyor belt, both of which caused the robot to either not pick the can up, or to place it in an incorrect bin. The number of faulty cans was determined randomly, resulting in a total of 8 cans that were laid down (4 red, 4 green), and 7 cans that were mislabelled. During the first round of the game (the warm-up round), one yellow and one red can was labelled as green, one red can and two green cans labelled as yellow (and ignored by the robot), and one yellow can and two green cans labelled as red. For the second round of game one green can was labelled as red, and one was labelled as yellow, while two red cans and three yellow cans were labelled as green.

The mislabelled cans had an indication of the mistake in the *written* explanation condition by displaying the explanation in the color of recognition, while in the *graphical* explanation condition the rectangle and arrow was the indication of the mistake. Participants in the *all explanations condition* received both "hints".

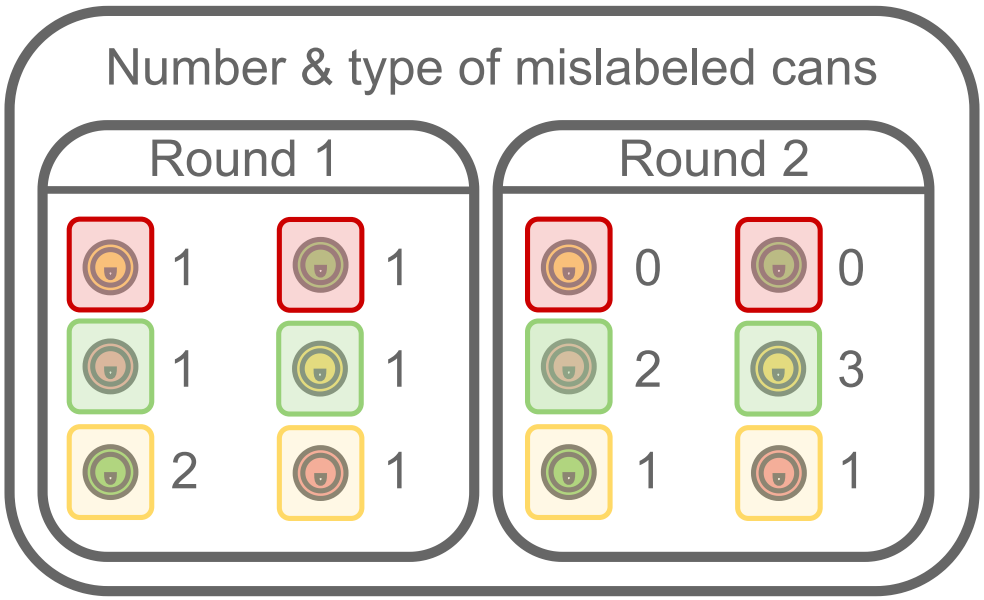


Figure 5.3: Each can had a 30% chance of being mislabelled. Although the mislabelled colors were assigned randomly, using the same random seed resulted in the same cans having assigned the same error.

Measurements

The dependent variables can be separated into two groups: subjective measurements reported by questionnaires and objective measurements recorded via variables within the simulation.

An additional inspiration for the experiment setup was gained from [40], where a conceptual model of the explanation process was discussed. In his model, the initial introduction to the system should bring about the generation of an initial mental model of the system as well as incite a certain level of trust, which then is updated after the exposure to the explanations. In an ideal case, these explanations would increase the users' understanding of the system, which would lead to an increased performance. The three main methods to assess the explanations according to the authors is by measuring the users' satisfaction, their understanding of the system and their performance. An adopted version of the process can be seen in Figure 5.4.

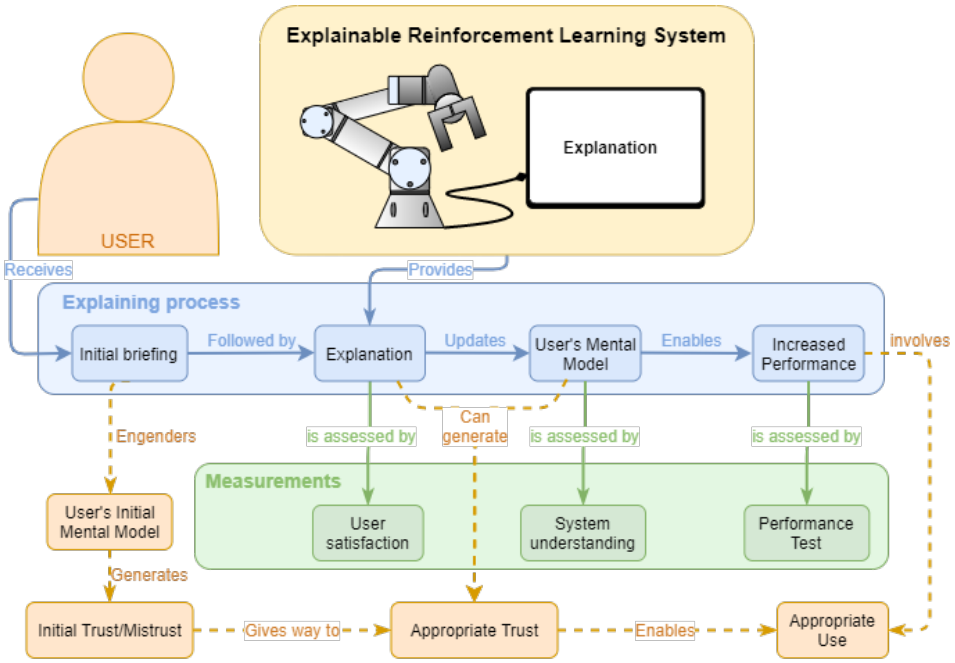


Figure 5.4: Conceptual model of the explanation process, partly adopted from [40].

Subjective Measurements

In order to examine the cognitive load the task imposed on the subjects, the NASA Task Load Index (NASA - TLX) was used. Although an official software version of the questionnaire is freely available, it was not compatible with the operating system of the iPad provided by the university, therefore a free online version [64] based on the original paper-and-pencil test was used for the experiment.

As a measurement of perceived trust, the shorter version of the Trust Perception Scale - HRI (TPS-HRI) [65] was used, which contained - instead of the original 40 items - only 14 items, which makes it ideal for a pre- and post-experiment administration.

Godspeed IV and Godspeed V [66] were used to measure *perceived intelligence* and *perceived safety* respectively. Following the author's suggestion, both Godspeed questionnaires were diluted with additional items in order to mask the intention of the survey. Most of the additional items were taken from the other three Godspeed questionnaires. The choice of perceived intelligence is due to its choice in related work on robot transparency [67], while perceived safety was chosen based on the intuition that explanations and understanding the robot's behavior provide a feeling of safety and an increase of trust.

Additionally, a demographic survey measured the participants' age, ethnicity,

gender, education and past level of experience with robots. All the pen and paper questionnaires used for the experiment can be found in A.

Objective Measurements

In order to increase the reliability of the measurements, additionally to the questionnaires, users were awarded scores within the game.

Additionally, meta-scores were also recorded for each game, meaning the correct and incorrect sorts of the robot and of the user, and the cans none of them picked up. The ratio of these variables (how many cans did the user allow the robot to pick) could also indicate the user's trust in the robot's abilities.

The recording of variables was slightly modified while conducting the experiment to include additional variables, however, these variables were only recorded during the second half of the experiment, for the *control* and *all explanations* conditions. The meaning conveyed by the analysis of these variables is discussed separately.

Participants

A total of 47 people participated in the experiment, where the first five of them took part in the trial run and served as a test-bed to the experiment procedure. An additional two participant's data could not be used due to technical difficulties and errors made in the experimental procedure, leaving 40 participants (26 male, 14 female) with a mean age of 28.1 (SD = 8.32) where 10 were randomly assigned to the first two conditions (*graphical explanation* and *textual explanation*), and once 20 participants were recruited, the following 20 were assigned to the *control* and *all explanations* conditions. This method was grounded in the concern that due to the restrictions due to COVID-19 to campus access and to social interactions in general, there might not be enough participants to examine all four conditions.

The participants were recruited either from the experimenter's social circles or by collecting volunteers on the university campus. The majority were of Danish (N=16) or Hungarian (N=8) nationality with 13 other nationalities being represented by 1 or two participants.

Previous experience with robots was reported by 45% (N=18) of the participants in terms of previously interacting with a robot more than 10 times, 32.5% (N=13) controller a robot more than 10 times, and 22% (N=9) programmed a robot before more than 10 times, with 45% (N=18) of the participants having reported some kind of programming experience regarding robots.

Participation was encouraged by offering muesli bars and a chance to participate in a draft for winning a 200 DKK worth of Aalborg gift card.

Apparatus

The experiment was conducted using a virtual environment developed in the open-source robot simulation software WeBots (version R2021a). A personal computer with Windows 10 operating system, an NVIDIA GeForce GTX 1080 Ti graphics card, 16GB RAM and Core i7-8700 GPU was used to run the software.

The NASA TLX questionnaire was administered using a freely available HTML-based online tool [64]. The questionnaire was accessed using Google Chrome browser.

The Godspeed, TPS-HRI and demographic surveys were provided in a pen and pencil format. The questionnaires can be seen in the Appendix A.

Procedure

The participants started out by filling the demographic survey, followed by an introduction to the task and scenario.

The introduction consisted of an interactive tutorial, where the users followed the instructions printed on the screen and learned about the mechanics of the game. The introduction text explained that the task of the participant is to sort yellow and green cans by placing them into the bin of their corresponding color, while the robot sorts green and red cans. The cans were arriving from a conveyor belt from the right, and the users were rewarded with a point for every correctly sorted can, while punished with a point deducted for every incorrectly sorted can. The participants were made aware that the robot's sorting also affects the score, as it gets double points for every correct sort, but loses double points for every mistake made. The cans could be sorted by clicking first on the can, then on the bin they wished to sort the can into. The cans that were not picked up by either the robot nor the user proceeded on the conveyor belt until they were out of the screen. The neglected cans were subtracted from the overall score at the end of the game when the collective points were also broken down as shown in Figure 5.5.

The users were given an example of the image the screen displays, which is identical to the ones seen in Figure 5.3.

Following the introduction, the users were instructed to fill out a TPS-HRI questionnaire keeping the robot displayed in the introduction in mind. The questionnaire was administered in a paper and pencil format. The participants then proceeded to the game, where each of them performed two sessions of collectively sorting with the robot 40 cans in total. The first run served as an introduction, where the participants familiarize themselves with the system and the robot's behavior. During the run, all questions were answered, trying to make sure all participants understood

Figure 5.5: An example of the score summary displayed at the end of each run.

the game procedure. The second run was increased in difficulty in terms of the frequency with which cans appeared.

After the second run was completed, all participants were asked to fill out the NASA-TLX online survey, followed by the Godspeed surveys and finally the second TPS-HRI questionnaire measuring the perceived trust post-interaction. An overview of the experimental procedure can be seen in Figure 5.6.

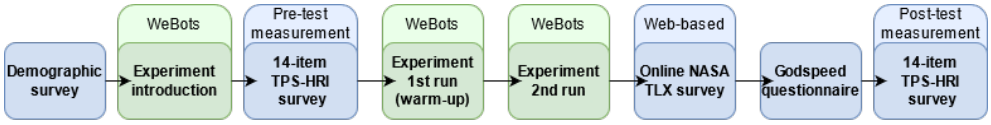


Figure 5.6: Overview of the experiment procedure. The differences between the conditions are exclusively within the simulation (marked as green).

5.4 Implementation

The experiment was implemented in the open-source simulation software WeBots. The scene consisted of both visual elements, such as a stack of pallets and a fire extinguisher, and functional elements, such as a screen, a conveyor belt, colored bins, a camera and a robot. The camera was only shown in the introduction and remains outside of the camera's field of view during the game. An example of the setup can be seen in Figure 5.5.

Camera & Display

WeBot's built-in Kinect camera was used as the depth camera for the experiment, placed above the conveyor belt. WeBots allows for the use of built-in recognition, as a *Recognition* node provides for each recognized object both the location on the image (middle of the blob) and object type, which then is parsed to find the object node within the simulation.

With the help of the *supervisor* node, all the necessary information about the cans within the view of the camera could be accessed and paired with the recognized objects. Using the open-source computer vision package *OpenCV* both the textual and graphical explanations are added to the image, which is then displayed on the built-in *Display* object of WeBots. The explanation was displayed according to the color the robot "believed" the object to be, which was assigned to be different from the actual color of the can 30% of the time to represent an error in the recognition.

The display is set to be a $1024cm \times 768cm$ long rectangular screen and is serving as a TV display that could be used in a real-life experiment. Although the size of the display is greater than a commonly available physical display would be, it was found to be an acceptable trade-off for balancing the camera resolution and readability of the display, as smaller display sizes were difficult to see and read.

Conveyor Belt And Can Placement

The conveyor belt was customized to fit the width of the conveyor belt provided by Aalborg University, which is $15cm$. As the simulation could only run at a speed of 0.5x real life, the conveyor belt and robot movement speed were adjusted accordingly.

The cans were generated randomly, controlled by four variables:

- *Seed*: The random seed is used as an input for the random number generator. Different seeds were used for each condition.
- N_{cans}: The number of cans the user had to sort. For both experimental conditions, 40 cans were used.

- F_{cans}: The frequency with which the cans would spawn onto the conveyor belt. The second run's frequency was set to 150% higher than the introduction run.
- S_{limit}: The spawn limit regulated how much was the minimum time that had to pass between the spawning of two consecutive cans. This variable was set to $960ms$, which corresponds to approx. $1.5s - 2s$ from the participant's perspective (accounting for the slow simulation) .

The can generation was furthermore influenced by two additional random number generators, where one was serving as an additional regulator to the spawning frequency to make it more unpredictable, and the other was used to select the type and location of the can that was to spawn on the conveyor belt.

Two different types of cans were spawned. Standing cans were able to be grasped by the robot while laying cans were not, and needed the user's intervention to be sorted.

The standing cans were spawned in five different locations along the width of the conveyor belt, spread out evenly in approx. $2.5cm$ intervals. The laying cans were spawned from a certain height so that upon landing on the conveyor belt they would roll around and settle at a point before entering into the user's field of view, and evoke a feeling of randomness.

Robot Programming

In order to increase the control over the environment, the robot was programmed with a custom algorithm, using a structure similar to state machines. The high-level overview of the robot's functions can be seen in Figure 5.7.

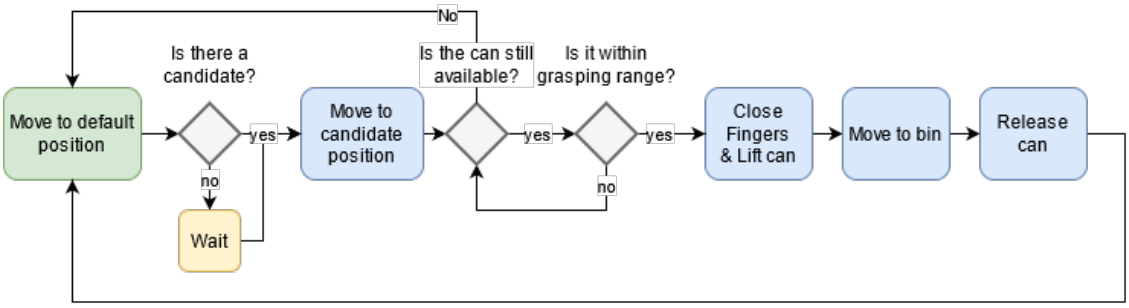


Figure 5.7: A high-level overview of the robot's workflow and inner structure. This loop runs once every time-step, which is set to $32ms$.

Moving To The Default Position, Waiting For The Can

The robot starts out by moving to a default position, where it waits for a can selected as a candidate to pick up. Candidates are chosen by the following criteria: 1) It had to be within at least $1.2m$ from the reach of the robot, 2) it had to have a recognized color of red or green (excluding cans with an error in recognition), 3) it had to be standing upright (excluding cans, that were tilted, representing cans it cannot grasp), 4) it had to be at a certain distance away from the previous candidate, to make sure the object can be grasped in time after sorting the previous candidate.

The $1.2m$ range mentioned in criteria 1) was determined in a trial-and-error fashion and adjusted according to the speed of the conveyor by choosing $max(kv_{conv}, 1.2)$, where k is a constant scalar and v_{conv} represents the velocity of the conveyor belt in m/s. The optimal value for k was found to be 0.8. The reason for selecting the highest of the two values was to make sure the robot reaches the given position in time.

Moving To The Candidates Position

Once a candidate has been recognized, the robot moves to a pre-programmed position according to the candidate's current location. These pre-programmed positions have been calculated with an inverse kinematics package called *ikpy*, however as running the inverse kinematics calculations online introduced an additional strain on the simulation which made running it at a reasonable speed infeasible, the positions were calculated in advance and stored in a dictionary. A total of 24 positions were recorded, 6 different positions along the width of the conveyor belt in order to represent the can's positions, 2 different positions along the length of the conveyor belt, and 2 different positions along the height, so that the robot can move to the lower position to grasp the can, and stay at the higher position before grasping, while waiting for the can and after grasping.

Monitoring The Cans

As it is possible for the user to sort the can the robot has selected as a candidate, the robot needs to constantly monitor if the selected can is still available. In case a candidate is removed, a new candidate is selected, and the position is adjusted accordingly. In case there are no other candidates available for the robot, it moves back to the default position, and waits there.

Grasping The Can

Unfortunately, due to the limitations of the built-in physics engine of WeBots the grasping of the cans relying solely on the *boundingObject* properties of the gripper and the cans, proved to be challenging, even when modifying the *contactProperties* of the surfaces to increase the friction, the cans seemed to slip out from between the fingers of the gripper.

To work around this problem, *Connector* objects were added to both the cans and the robotic gripper, which creates an unbreakable binding force between the objects, ignoring the physics engine. Setting the *distanceTolerance* property of the connector to $5cm$, the robot was able to lock the connector when closing the gripper, creating the sense of grasping. The *axisTolerance* and *rotationTolerance* properties were set to π radians in order to allow the connection from all the poses.

Scene Interaction

The players were able to modify the cans' location by first selecting a can, then selecting a corresponding bin. During the experiment, the viewpoint was locked, and the moving of objects and opening of menus was disabled to avoid unintended interactions. Furthermore, full-screen mode was enabled to hide all additional menus.

The selection of objects was managed by the *getSelected()* method of the *supervisor* node, which returns the ID of the object currently selected. The ID was used to determine if the participants selected a can or one of the colored crates. In case a crate was selected after a can, the can was teleported to the location of the crate, and removed from the game variables, leaving only the physical object on the scene, that could not be interacted with anymore (and so any occasional mistakes made could not be mended).

5.5 Experiment results

Reliability analysis

In order to validate the reliability of the measurements, Cronbach's Alpha is calculated for each of the questionnaires. According to [66], for perceived intelligence, the satisfactory internal consistency consists of values of 0.7 and higher, which indeed corresponds to the calculated value of our measurements ($\alpha = 0.70$). For perceived safety, however, the internal consistency seems unsatisfactory ($\alpha = 0.30$). This unreliability combined with the reasoning that it might not measure the perceived safety of the system the same way as it would in the case of working with a physical

robot, was considered a reasonable ground to omit these measurements from further analysis.

For the NASA TLX questionnaire, good internal consistency ($\alpha = 0.79$) was found, similarly to the pre-test and post-test trust questionnaires ($\alpha = 0.76$ and $\alpha = 0.86$ respectively).

The following sections present the results gained after analyzing the dependent variables.

NASA TLX questionnaires

As by nature the measurement of cognitive load is ordinal when using the NASA TLX questionnaires, the Kruskal-Wallis test was run on the overall weighted scores, but no statistically significant differences were found ($H(3) = 1.98, p = 0.58$). An

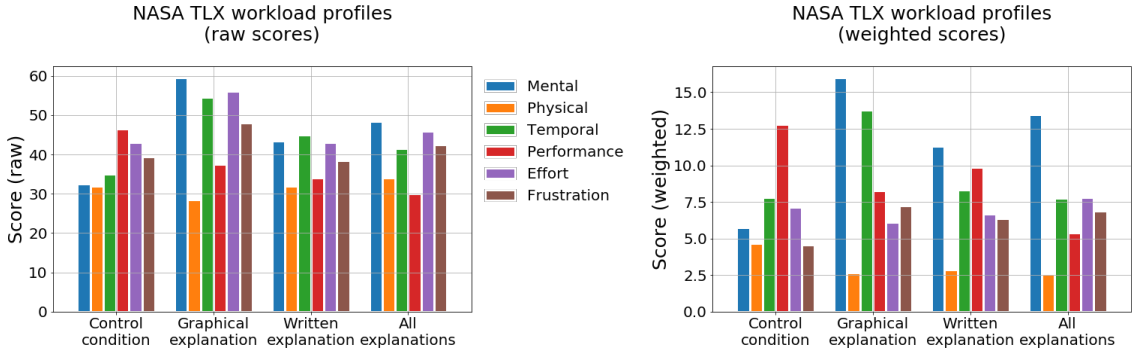


Figure 5.8: NASA-TLX scores broken down to individual dimensions. The weighted scores reveal that participants attributed much of the task load to the mental effort.

alternative, relatively common way to analyze the results however is to compare the weighted subscales of interests [68], which we also applied, as some of the sources of the workload such as physical effort have little to no interest when conducting a virtual sorting task (some of the participants indeed asked to make sure that they are to rate how physically demanding it was to click with a mouse). Therefore the five subscales were also analysed individually using the Kruskal-Wallis tests. The only statistically significant difference was at the "Mental demand" dimension ($H(3) = 7.95, p < 0.05$). After conducting post-hoc pairwise Mann-Whitney U-tests with Bonferroni corrected value of $\alpha = 0.0083$, statistically significant difference was found between the *"all explanations"* ($N = 10, Mdn = 12.83$) and *"control"* ($N = 10, Mdn = 4.17$) conditions ($U = 16.5, p = 0.006$) with a strong effect size ($r = 0.78$), and between the *graphical explanation* ($N = 10, Mdn = 16.65$) and *control* conditions ($U = 15, p = 0.005$) with a weak effect size ($r = 0.09$).

A discussion on these results can be found in the following section.

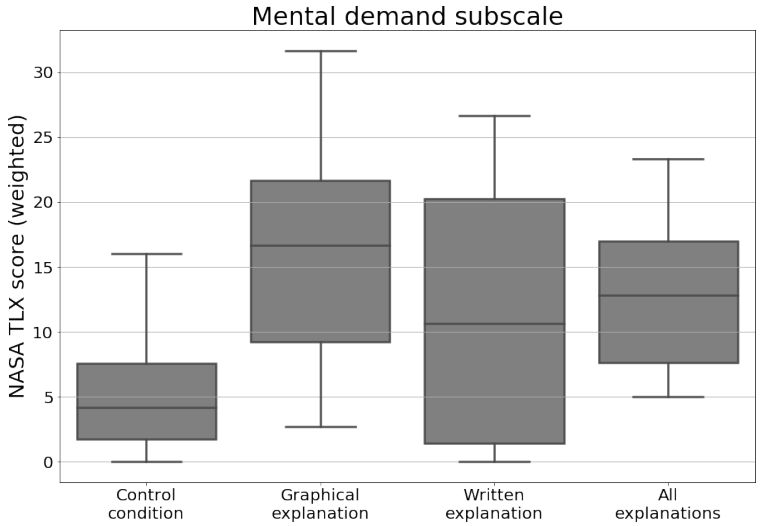


Figure 5.9: An analysis of the *mental demand* subscale of the NASA-TLX scores suggested that participants considered the task containing graphical explanations more demanding.

Perceived Intelligence (Godspeed IV)

The Kruskal-Wallis test revealed no statistically significant differences between the conditions regarding the perceived intelligence of the manipulator ($H(3) = 1.94$, $p = 0.58$).

Trust Scores (14-item TPS - HRI)

In order to calculate the final trust scores, the total scores were calculated for each participant, both for pre-test and post-test, and the change in trust between the two tests was taken as the final trust score. An illustration of the difference between the pre-test and post-test scores can be seen in Figure 5.10. A Kruskal-Wallis test was calculated, but revealed no statistically significant differences in the trust scores between the conditions ($H(3) = 4.34, p = 0.23$).

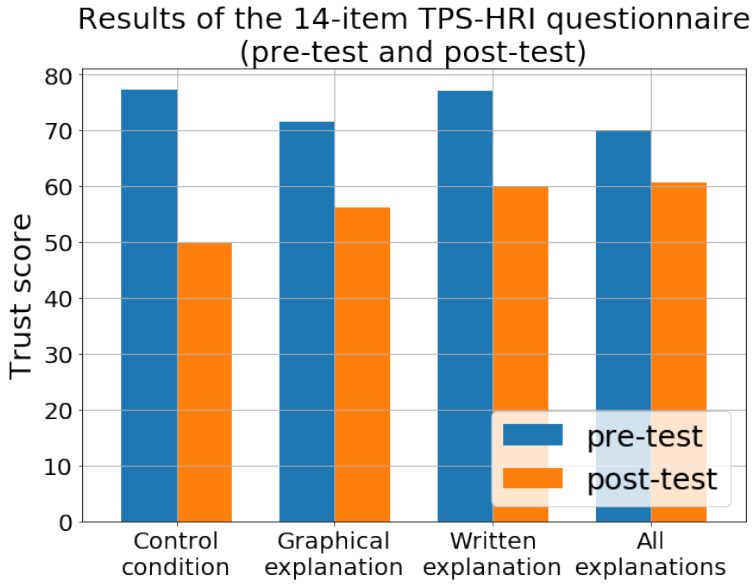


Figure 5.10: Scores of the Trust Perception Scale (TPS) - HRI. A drop in trust was measured across all conditions after the gameplay.

As all the 14 items of the questionnaire relate to different constructs. The individual trust scores were also assessed and considered individually, a compact summary of which can be seen in Figure 5.11. A Kruskal-Wallis test was re-calculated for each individual item, but the only significant difference ($H(3) = 9.23, p = 0.026$) was regarding the *"communicate with people"* question, where post-hoc Mann Whitney U-tests revealed a significant difference between the *Control* and *Graphical explanation* group ($U = 18, p = 0.000, r = 0.59$). It is also worth noting, that close to significant difference was found on the answers to the *provide appropriate information"* question ($H = 7.77, p = 0.051$, where post-hoc tests showed the significant difference between the *graphical* and *written* explanation conditions ($U = 18, p = 0.0078, r = 0.83$). Similarly to the previous post-hoc tests, a Bonferroni corrected α of 0.0083 was used to determine significance.

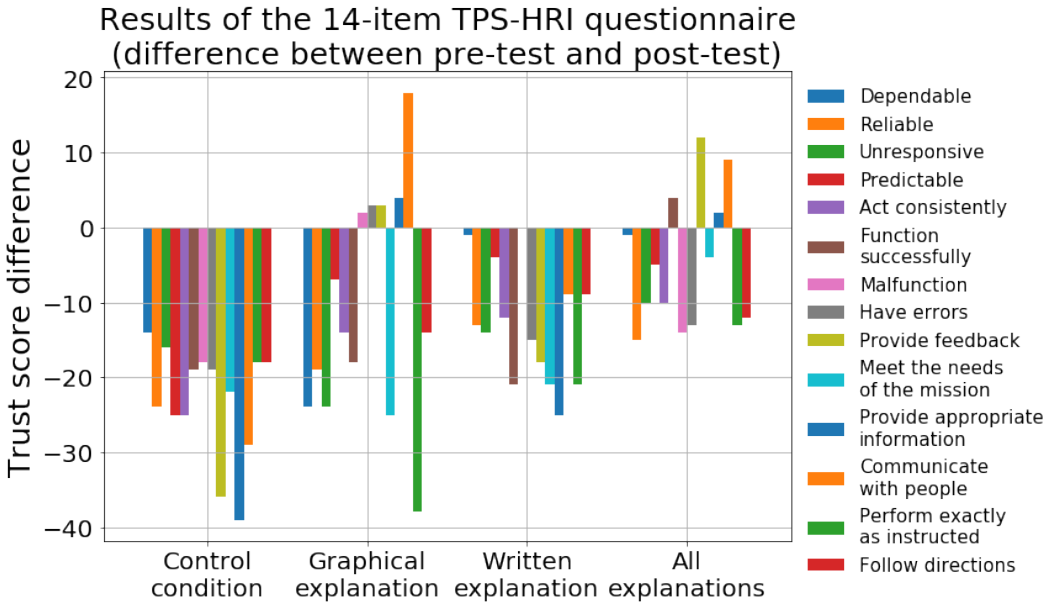


Figure 5.11: Scores of the individual items of the Trust Perception Scale (TPS) - HRI.

Game Variables

In order to be eligible for parametric tests, the game scores must not violate the assumptions of homogeneity of variance, and of normal distribution, which is tested using Levene's test and Shapiro-Wilk tests respectively. It must be noted though, that the limited sample size reduces the reliability of this test as both of these tests (and the alternative of the Shapiro-Wilk test, the Kolmogorov-Smirnov test) are sensitive to sample sizes [69].

The Shapiro-Wilk test revealed that the assumption of normality was not violated ($W = 0.97, p = 0.36$), therefore we do not have enough evidence to say that the sample is not normally distributed. This result was further confirmed by running the Kolmogorov-Smirnov test ($D = 0.078, p = 0.97$). The assumption of homogeneity of variance was also not violated, as a Levene's test revealed that there was not a significant difference between the variances of the conditions ($F = 0.79, p = 0.51$).

As the game score qualifies for parametric tests, a one-way independent analysis of variance (ANOVA) is performed. The results showed a significant difference between the conditions ($F(3, 36) = 3.170, p = 0.04$) with a medium effect size ($r = 0.46$). To discover the source of the differences, post hoc t-tests were run, using Bonferroni correction to control for Type I error. As a total of six tests were run, the Bonferroni corrected value of $\alpha = 0.0083$ was used. Only between the *control* group ($N = 10, M = 37, SD = 5.38$) and the *graphical explanation* group ($N =$

$10, M = 47.9, SD = 5.8$) was a statistically significant difference found ($t(18) = -4.12, p < 0.008$) with substantive effect ($r = 0.7$).

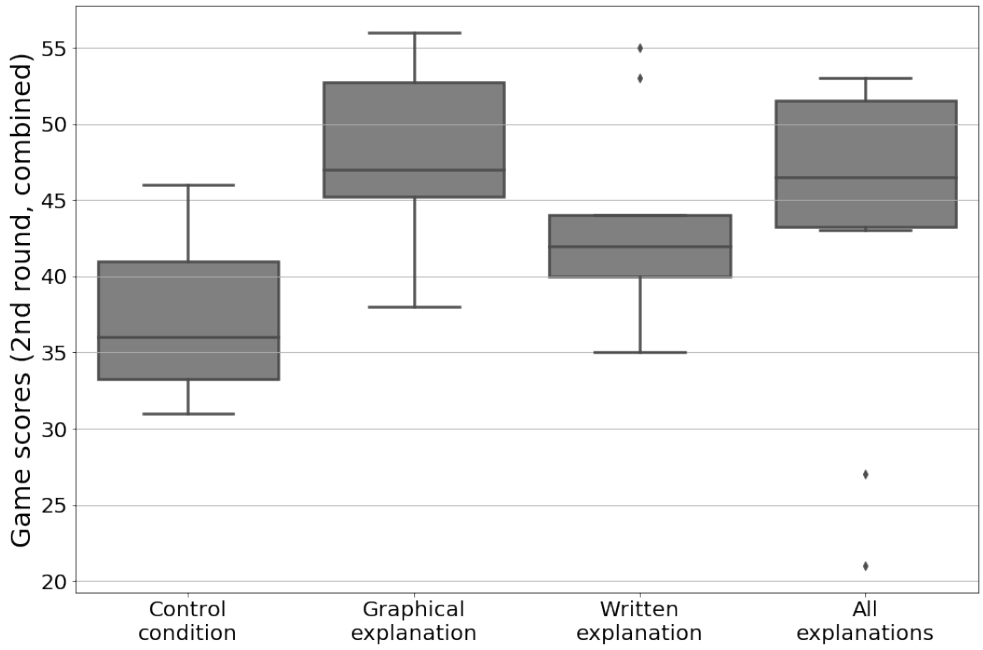


Figure 5.12: The combined scores (both from the user and the robot) achieved during the 2nd round of the sorting.

The number of cans the user allowed the robot to sort was also analysed throughout the conditions. Although this measurement by nature qualifies for parametric tests, Shapiro-Wilk test revealed that the data distribution violates the assumption of normality ($W = 0.884, p = 0.001$), and therefore Kruskal-Wallis test was applied, which revealed a statistically significant difference across the conditions ($H(3) = 7.80, p = 0.05$). Post-hoc Mann-Whitney U-tests however did not reveal significant differences between the conditions, although the p-values for the tests between the *control* and *graphical explanation* conditions, and the *graphical explanation* and *all explanation* conditions was close to the Bonferroni-corrected α value ($p = 0.032, p = 0.011$ respectively). The box-plot representation of the distributions can be seen in Figure 5.13.

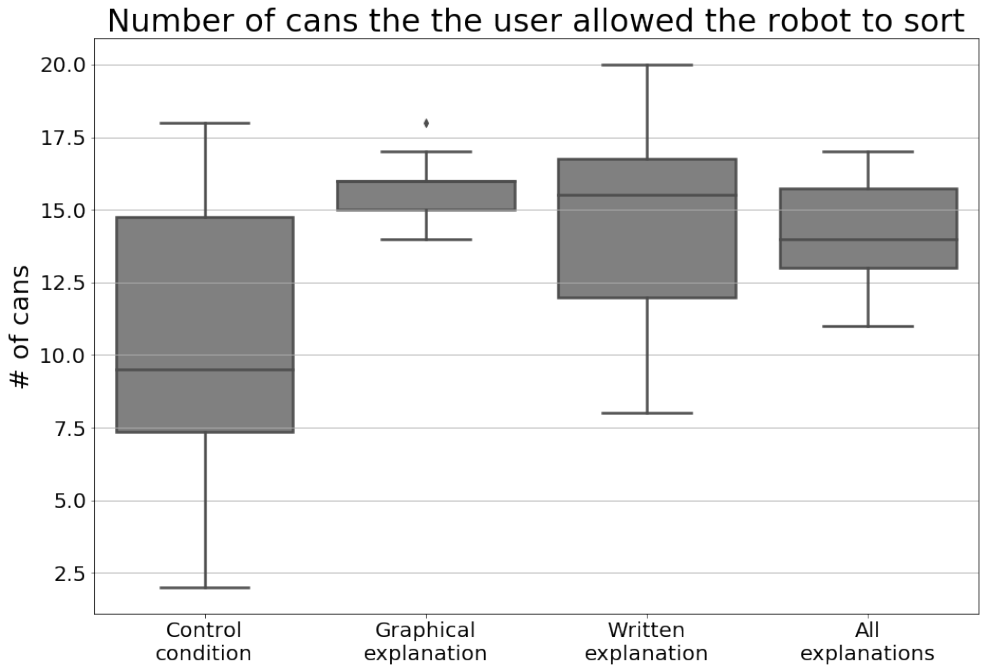


Figure 5.13: The number of cans the user allowed the robot to pick represented graphically for each condition.

5.6 Additional Game Variables

Rationale Behind The Additional Game Variables

It has been realized while conducting the experiment, that some of the potentially valuable game variables have not been recorded. To amend this mistake, the data recording was modified to include two new dependent variables, the *mistakes prevented* and *picking distance*. These variables however are only analyzed for the *control* and *all explanations* conditions, as for the other two conditions only three samples are available for each.

Players in the condition where explanations were available had a chance to deliberately prevent mistakes from happening. The overall count of green and red cans that were prevented from incorrect sorting is measured. Although yellow cans are also counted, they are not considered in the post-hoc analysis, as users could sort them without needing to consider what the robot is planning to do, it does not affect the score negatively. The prevention can be measured as a raw count, although it can be considered more indicative as a ratio of misidentified (prevented) and correctly identified cans picked, as the users who aimed at maximizing the .

Although the *picking distance* could be analyzed individually, as it can be indicative of the performance (how fast the participants picked the cans), for our purposes these variables were used to classify the cans. The conveyor belt is divided into four zones, each representing an area where the user's sorting can be influenced by different effects.

- The **"itchy finger" zone** is the zone on the conveyor belt that falls between the edge of the camera's field of view and the user's field of view, meaning that the cans are already visible to the user, but are not yet visible to the robot. Cans that are sorted in this zone are sorted without the knowledge of the robot's explanation. The significance that can be derived from counting the amount of red and green cans sorted in this zone is the indication that the user might not have considered the extra points that could be earned by letting the robot sort the cans. Furthermore, the mistakes prevented in this zone are not considered, as the user did not have access to the explanation to deliberately prevent a mistake. The zone stretches over $0.22m$, and the cans take approx. $2s$ in real-time to pass through.

- **Prevention zone** starts from the point on the conveyor belt where the cans enter into the field of view of the camera and their corresponding explanations appear on the screen. The zone ends where the robot is able to grasp the can. Participants that sort the green and red cans in this zone have a chance to consider and act upon the explanations. The zone corresponds to a $0.8m$ long stretch, through which the cans travel for approx. $6s$.

- The **robot operation zone** is the $0.4m$ long stretch where the robot is able to pick cans up. In the implementation for the experiment, the robot was picking cans at the two ends of the zone.

- The **cleanup zone** is the last visible part of the conveyor belt. The cans that were not picked in the previous zones and got outside of the robot's reach and the camera's view have a chance to be picked up again by the participants. These cans are also not counted among the prevented mistakes, as the explanations are neither visible nor relevant at this point.

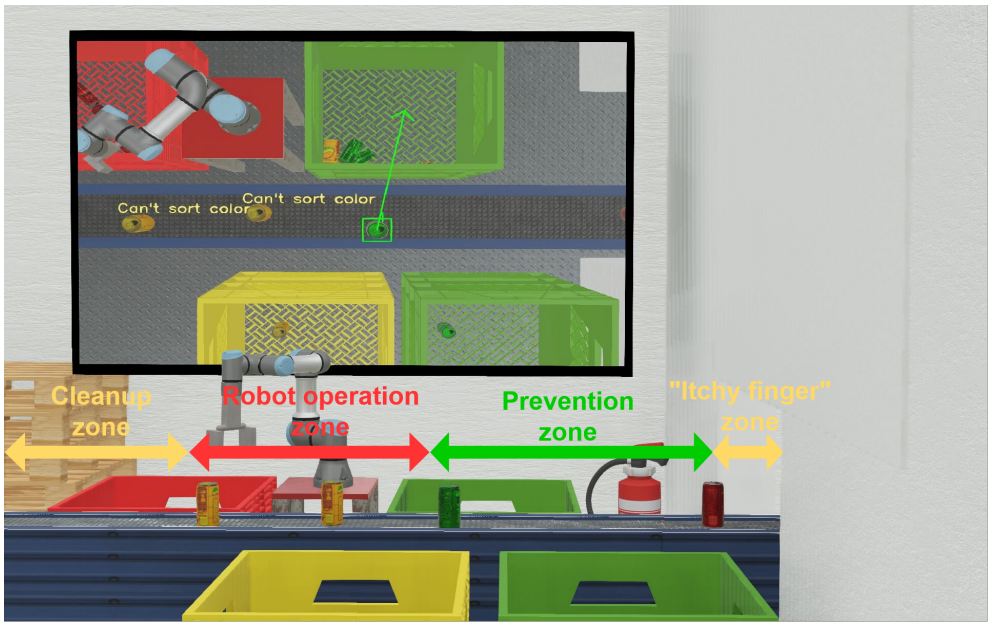


Figure 5.14: The game area is divided into different zones based on the available information to the participants and the robot's capabilities.

Additional Results

The *control* and *all explanation* conditions were compared on the number of mislabelled cans sorted ($N = 10, M = 10.2, SD = 2.34$ and $N = 10, M = 11.9, SD = 2.331$ respectively). The raw number was corrected by subtracting the green and red cans sorted by the user in the *"itchy finger"* and *"cleanup"* zones. Shapiro-Wilk test showed no statistically significant differences from the normal distribution ($W = 0.913, p = 0.072$), and the Levene's test did not show a significant difference in the variances ($F = 0.019, p = 0.89$). Therefore an independent t-test was run, which however did not show significant differences between the conditions ($t(18) = 1.625, p = 0.122$).

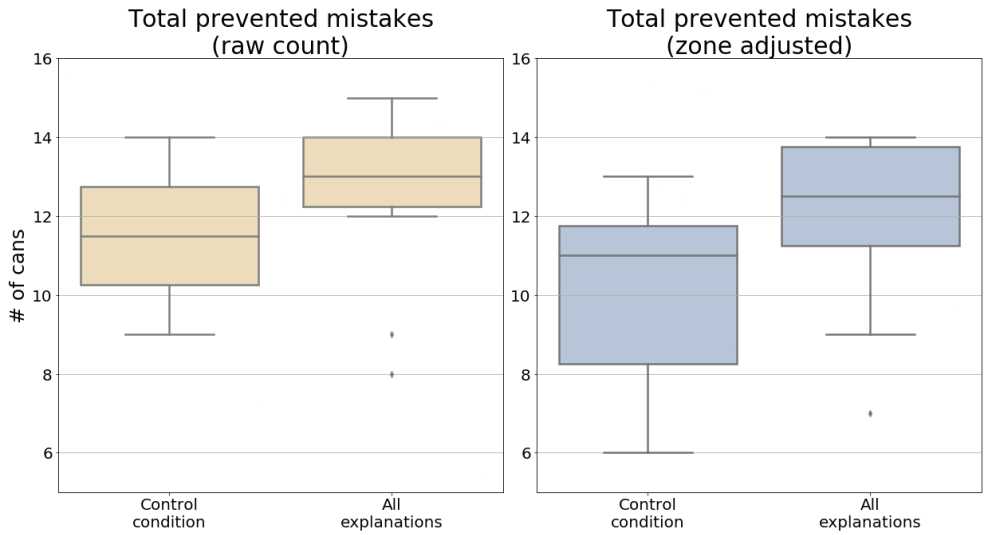


Figure 5.15: On the left: The total number of mislabelled cans sorted by the participants in each condition, without correcting for the two zones where the prevention has no significance. **On the right**: The raw number of cans was adjusted by subtracting the red and green cans sorted in the cleanup and "itchy finger" zones.

5.7 Experiment Discussion

General Discussion

Between-group design has a lower sensitivity to the experimental manipulations, as a within-subject design could be beneficial. Uncontrolled game flow. The user's decisions influenced the robot's behavior and therefore the flow of the game. Some participants mentioned after the experiment that they did not notice or pay attention to the screen and the explanations displayed on it, and still achieved a good score by "accidentally" preventing the mistakes and choosing the right cans to maximize the score. Although these cases were rare, a custom questionnaire would've been beneficial that could've attempted to measure these effects.

Mental Demand (NASA TLX)

The low ratings on the mental demand in the *Control* condition compared to the condition with *graphical* explanation aligns with the conclusions of [44], according to which explanations take cognitive power to process and act upon, which might result in a lower overall task performance, questioning the justification of integrating explanations into the system. As we can see below however, in this case positive effects of the explanations have been found, which could justify the effort of incorporating

them into the system.

Trust scores (TPS- HRI)

Although no significant differences were found in the trust scores calculated from the 14 items collectively, the individual analysis of the 14 items gave some intriguing insights. Besides revealing, that the participants in the *graphical* condition considered the robot more communicative and reported that it provides appropriate information, a relatively great drop in the *"perform exactly as instructed"* and *"meet the need of the mission"* points was also noticed. Our speculation is that the greater transparency and clarity of the actions also highlight the shortcoming of the system (which are discussed later in Chapter 8), bringing them to the user's attention. If this line of reasoning is correct, it could mean that when comparing explanations in a system that performs in a more reliable manner, the increase in trust induced by the graphical explanation could be significant.

Furthermore, when both explanations were displayed, the sub-scales of the meta-analysis show lower values than with a single explanation, indicating that the combination of these types of explanations do not have additive benefit, and "might do more harm than good". This finding aligns with the concerns reported by [29] and [39], according to which too much explanation can cause an information overload and decrease the effectiveness and user satisfaction.

Game Variables

Although some of the game variables were unfortunately not recorded for all participants, which would have resulted in a more accurate analysis, still a statistically significant increase in collective scores have been found for participants in the *graphical explanation* condition. However no significant differences were found between the performance measures of the two modalities.

5.8 Experiment Conclusion

We have conducted an experiment to 1) examine how explanations affect the cognitive load, trust and task performance, 2) compare two different modalities by which explanations can be displayed. The analysis of the results has shown that participants who got *graphical explanations* presented to them achieve significantly higher scores when compared to the control group, but these explanations do not differ significantly from the *written explanations* in terms of performance measured by collective scoring.

Furthermore, a meta-analysis of the sub-components of the 14-item TPS-HRI questionnaire implied that participants might consider the *graphical explanation* more communicative, while also providing more appropriate information.

Analysis of the *mental demand* subscale of the NASA TLX test however has also shown a significant increase in terms of cognitive load when comparing participants' responses from *graphical explanation* and *all explanations* groups to those of the *control* condition.

The analysis indicates an apparent trade-off between cognitive load and task performance, although due to the small sample sizes further investigation is needed to verify these results.

For the purposes of this report, the *graphical explanation* was chosen as the preferred explanation type due to its substantial ability to increase performance, and due to the indications that the explanation type is preferred by the users.

Moreover, the results of these experiments suggest an answer to both **RO-1** and **RO-2**, as an effect of explanations has been found, while a preferred modality is also identified.

The following section, therefore, proceeds to solve **RO-3**, by presenting the design of the explainable reinforcement learning agent that incorporates these findings to produce a system that could be capable of increasing overall performance by providing graphical explanations.

Chapter 6

System Design

This chapter presents the underlying principles of the system design as well as an outline of the system's architecture. Each level of the design is detailed with their corresponding environments.

6.1 Use Case Specification

As a use case, collaborative sorting of aluminium cans resembling the experiment setup 5. In a real-life scenario, an RGB-D camera will capture a life image of the incoming cans on the conveyor belt. Based on this the position, orientation and color will be identified for each can. The system will use this information to make a trajectory to a specific can, grasp it and sort it into the bin designated to the color. Meanwhile, the intentionality of the system will be displayed on a screen as a graphical explanation to the user. The graphical explanation is an arrow pointing from the selected can to the intended bin. The robot will be controlled with the use of HRL with policies tasked with sub-tasks of moving to a can, grasping a can, and sorting a can in the bins. Higher level policies will control the other of the lower policies and picking cans to be sorted. All this will work while a human cooperates with the system of sorting the cans. This use case is designed to resemble a real-world sorting task, however, it is simplified for the purpose of lower system complexity. A drawing of this use case can be seen in Figure 5.1. Based on this use case, a system architecture must be designed, which is presented in the following section.

6.2 System Architecture

The proposed system design is a hierarchical framework consisting of three levels of policies, which can be seen in Figure 6.1. The High level policy is a dueling DQN

agent given the task of picking cans based on the position of all the cans. When a can is picked, The environment supervisor sends the distance, orientation and color the picked can to mid level policy. This policy is given the task of picking the correct order of commands to complete the task. It can choose to either move to a can, grasp a can, time the grasping, or move to a bin. The low level consists of two different SAC policies and two DQN policies which are trained to complete the tasks chosen by the mid level. If all these policies are executed in the correct order they will each return a reward to the mid level policy along with a success signal. When the mid level has received a success signal from all policies it will send a success signal to the high level policy and this policy will then pick another can repeating the process. The design of each of these policies will be explained in the following sections after a brief introduction to the simulation software, WeBots and the RL tool kit, OpenAI Gym.

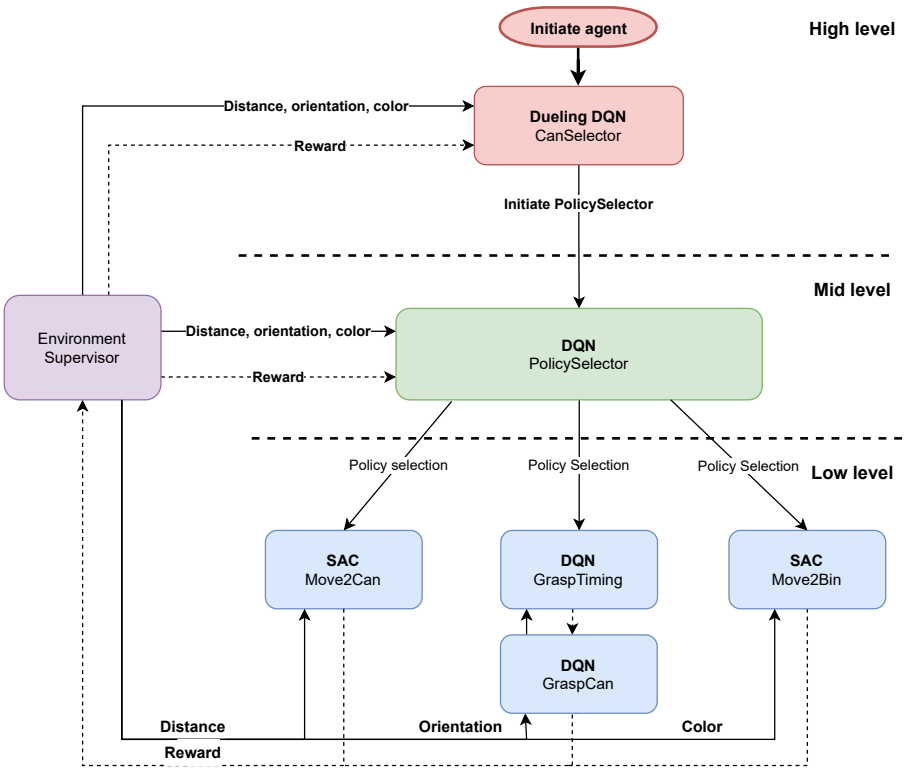


Figure 6.1: Architecture of the hierarchical system design consisting of three levels of policies. Low level policies: Move2Can, Move2Bin, GraspCan and GraspTiming, Mid level policy: PolicySelector, High level policy: CanSelector

6.3 Simulation

As previously mentioned in section 2.4, training a reinforcement learning agent on a real robot in a real-world environment is not an ideal solution both because it is usually a sample-inefficient way to gather training data, and because the robot could cause harm in both the environment and itself.

For these reasons, we are developing training and developing our system in a simulated environment, for which the simulation software WeBots [70] is chosen. WeBots is an open-source robot simulation software, that is used as an educational tool for academic purposes by some of the world's most prestigious universities, such as MIT, Caltech, Harvard University and Oxford University, and as an R&D tool by companies such as Sony, Honda, Toyota, or NASA [71].

Webots comes with a fully integrated environment for modelling, programming and simulation of robotic applications. Furthermore, it provides a large asset library of actuators, sensors and robots such as the UR3 from *Universal Robots* specifically used for this master thesis. The robot controller can be programmed using languages such as C, C++, Python and built within the native Webots compiler. All control is exclusively programmed in python3 to take advantage of the machine learning framework *PyTorch*. In the following sections a detailed description of the simulation environment, robot modelling and control programming will be given.

6.4 OpenAI Gym

As a part of the hierarchical reinforcement learning, individual agents must be trained to enable the correct behavior of the system. For the design of these training algorithms, the general structure of the open-source development kit Gym created by OpenAI [72] is used. For the purpose of this project, the designed structure consists of three parts, an initiation of the environment, a step function, and a reset function. All these parts can be seen in Figure 6.2, where the gym environment is a part of the Reinforcement learning algorithm. In this figure, the learning is initiated and the reset function is executed, which resets the simulation's physics. Subsequently, the step function is computed repeatedly until a terminal state is reached. The supervisor is a method within Webots responsible for resetting the physics, setting joint velocities and extracting the state of the environment. The initiation, reset environment and step functions will now be explained.

In the initiation function all the needed variables are defined. The most important part of these is the action and state space representation. The action space is defined as the actions the agent can take to influence the environment e.g. the joint velocities of the UR3. The state space is important information about parts of

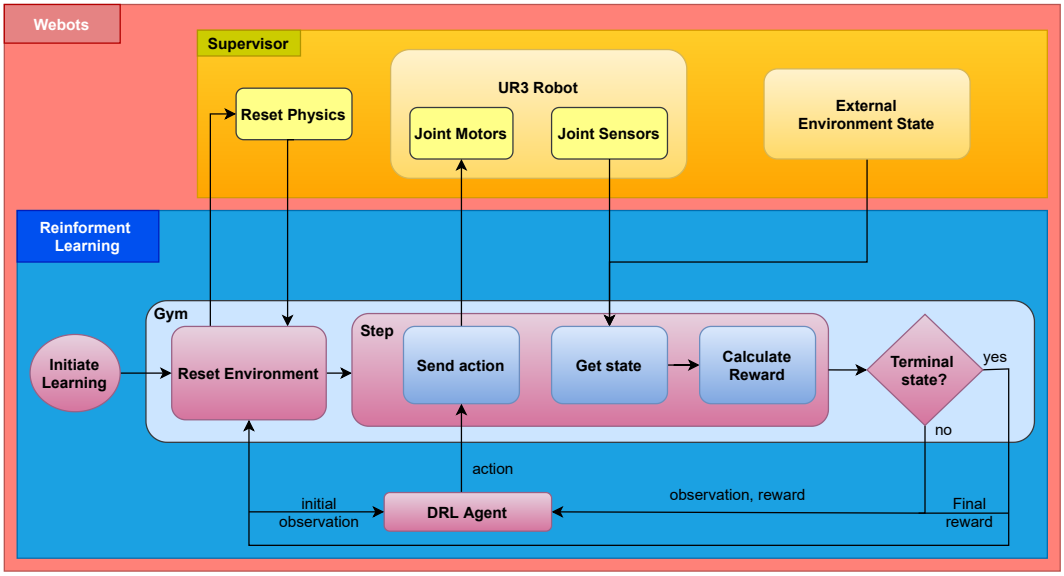


Figure 6.2: The architecture of the developed deep reinforcement learning framework within Webots.

the environment influencing the agent e.g. the distance to a goal or the pose of the UR3.

In the step function, an action is first chosen by the agent, the state of the environment is gathered and the reward is calculated. The reward is based on a function designed to inform the agent of how it should be behaving. An example of this could be that the agent gets a positive reward for actions that move it closer to the goal and a negative if it moves further away. From a perspective of supervised learning, the reward can be seen as a way of dynamically labeling good actions. After the reward has been calculated a binary value indicating if the agent has reached a terminal state is returned, and if it has not, it executes the step function again.

If the agent has reached a terminal state, the reset function is executed. Here the environment is reset to an initial state, which in this case includes resetting all positions and physics in the simulation.

6.5 Low Level Policies

The low level policies are responsible for the actions that would be considered low-level from a human's point of view, and that would not contribute directly to an explanation. This level consists of four policies, each with different learned behavior. The Move2Can, GraspCan and Move2Bin are tasked with moving the can, grasping the can and manipulating the can, respectively, while the GraspTiming policy uses

the GraspCan policy and learns to time the grasping operation. The setup of each of these low level policies will be explained in the following sections.

Move2Can

Move2Can is designed to control the UR3 to reach an adequate pose of the end-effector, enabling potential subsequent grasping of a can. For this reason, the UR3 is placed on a table in an upright pose and a random goal is created relative to the base of the robot with a position of -0.4 m to 0.4 m in the X orientation, 0.4 m to 0.45 m in the Y orientation and at a height of 0.15 m. The training environment can be seen in Figure 6.3.

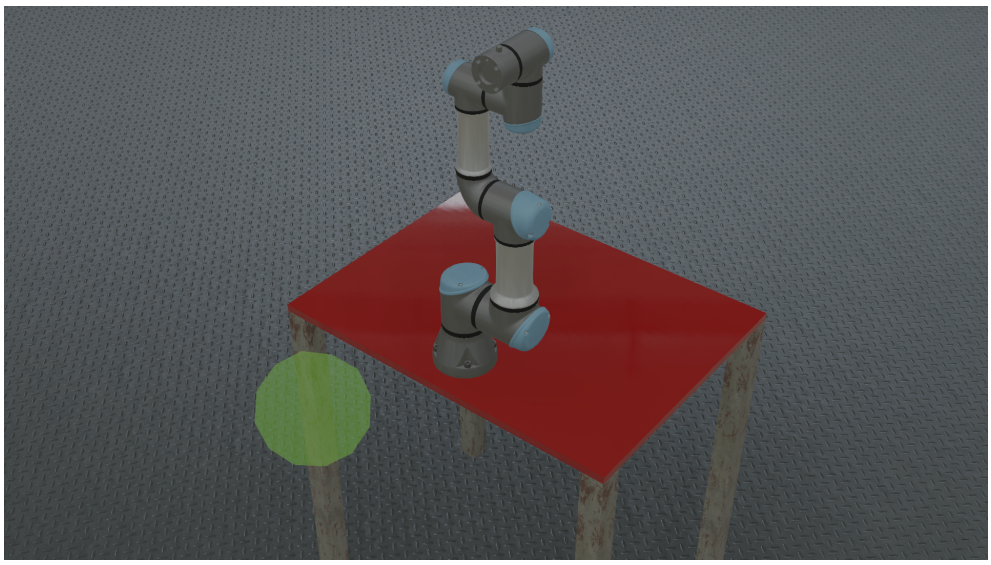


Figure 6.3: The environment designed to train the position and orientation control of the end-effector. The goal is represented as a green sphere.

An action space of the agent is created consisting of the joint velocity control of the UR3 model except for the end-effector joint. The state space is the joint position and velocities, the position of the goal, the position of the end-effector and the distance from the end-effector to the goal. The reward function is a dense reward for moving closer to the goal and a sparse negative reward for either colliding with an object or a positive reward for getting within 2 cm of the goal.

$$r(s_t, a_t) \begin{cases} r_{arrive} & = 1000 \\ r_{collision} & = \text{-}300 \\ 100 \cdot (d_{t-1} - d_t) & \end{cases} \tag{6.1}$$

Here the reward r for a given time t in a state s_t with an action a_t is given as a positive reward of 1000 for arriving at the goal. A negative reward of $r_{collision} = -300$ is given if the UR3 collides with an object. Finally, progression towards the goal is rewarded by $d_{t-1} - d_t$ and multiplied by a scaling factor of 100.

Move2Bin

The Move2Bin environment is created to train the agent to control the UR3 to move to a location of a bin. For this reason, the UR3 is placed on a table in a pose resembling the joint position values after a grasp. Two bins are placed in the proximity of the robot and one of these is moved within reach of the robot if the color associated with the color is picked. At this stage, the bin is picked randomly and set as the goal. This environment can be seen in the following Figure 6.4.

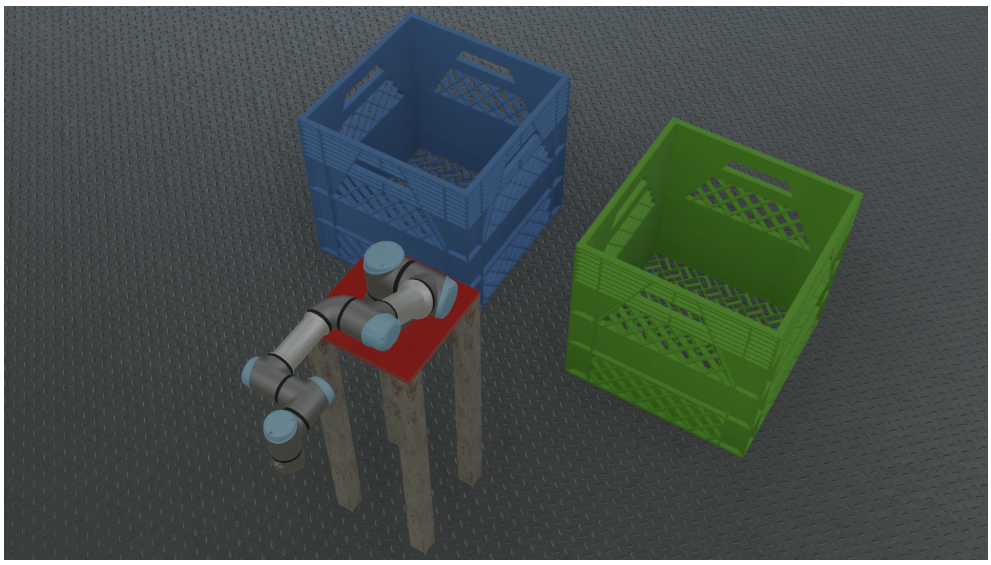


Figure 6.4: The environment designed to train the agent to move to one of the green or blue bins. When a bin is picked it will move into reach of the robot.

An action space of the agent consists of the joint velocity control of the UR3 model except for the end-effector joint. The state space is the joint positions and velocities, the position of the end-effector, the distance from the end-effector to the goal and the position of the goal.

The reward function is a dense reward for moving closer to the goal and sparse negative reward for either colliding with an object or a positive reward for getting within 5 cm of the goal.

$$r(s_t, a_t) \begin{cases} r_{arrive} & = 1000 \\ r_{collision} & = \text{-}300 \\ 100 \cdot (d_{t-1} - d_t) & \end{cases} \tag{6.2}$$

Here the reward r for a given time t in a state s_t with an action a_t is given by a positive reward of 1000 for arriving at the goal. A negative reward of -300 if the UR3 collides with an object. Finally, progression towards the goal is rewarded by $d_{t-1} - d_t$ and multiplied by a scaling factor 100.

GraspCan

Robotic grasping is an essential part of robotic manipulation, and can be a challenging task.

For the simulation environment a custom gripper was designed as the end-effector to enable grasping of the cans. It was designed to resemble the Schunk EGP 50-N-N-B parallel gripper [62], with the addition of two fingers of dimensions 7 cm (height) x 2 cm (width) x 6 cm (length). A parallel gripper was chosen since it has a simple control scheme and is considered adequate for grasping the cans of the system. The force of two motors controlling the fingers are set to 215 N in order to match the Schunk EGP 50-N-N-B parallel gripper. Position control is used to adjust the position of the fingers. This gripper can be seen in Figure 6.5.

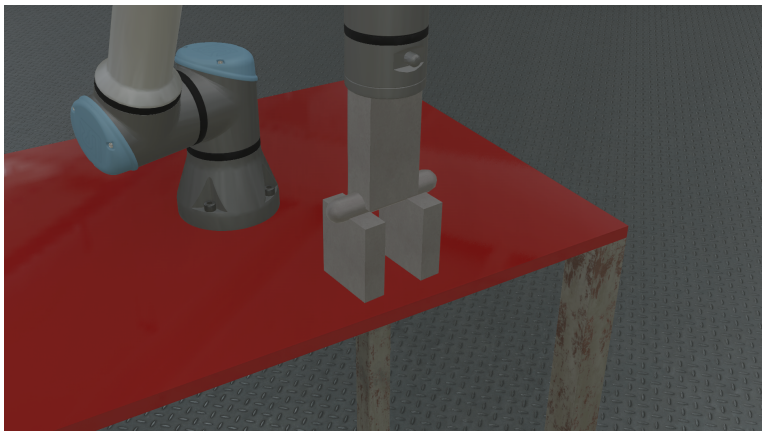


Figure 6.5: The gripper designed to enable grasping of the cans

In our setup, the training of the grasping consisted of finding the right angle from which cans lying in different orientations could be grasped. The training setup is inspired by [49], meaning that instead of using motor commands as actions, the grasping operation was embedded in a primitive action, which consisted of the fingers

opening, the tool center point (TCP) moving down $10cm$, the fingers closing, and moving the TCP back up. Once the movement ended, approx. $3s$ of waiting time was ended before concluding if the grasping was successful, as an attempt to filter out "lousy" grasps, where the can falls out of the fingers shortly after grasping. Between each sub-action of the primitive action, an approximately $0.2s$ of buffer time was inserted to ensure that the physics engine of the simulation is capable of performing the calculations necessary to carry out the actions.

The can was reset to a random position after every grasping attempt, and each episode lasted for 100 attempts.

The state space consisted of the pose of the can (provided by the *supervisor* node, and the distance of the can on the x-y plane relative to the TCP, while the action space consisted of 8 discrete actions, each corresponding to a multiple of a $22.5°$ turn from $0° - 180°$. Due to the symmetry of the gripper, only angles up to $180°$ are considered.

Following the reward design of [49], a reward of 1 was given for each successful grasp, while no reward was given otherwise.

The difficulty of the environment was continuously increased, where initially only the rotation of the can was changed with it's position kept constant (the conveyor belt was not moving).

After successfully learning to maximize the reward in the environment, the displacement of the can relative to the TCP was incrementally increased with the conveyor belt still kept still. The reason for the displacement was to discover the limits of the accuracy needed in the positioning of the TCP for the *Move2Can* policy.

Once the grasping policy has been successfully trained and optimized, an additional policy called *GraspTiming* was trained in order to perform the grasping operation with a moving conveyor belt.

GraspTiming

The timing of the grasping operation is of crucial importance. As attempts of incorporating this timing element into the *GraspCan* policy have been unsuccessful, a separate network is created on top of the *GraspCan* policy. Ideally the algorithm should learn when to initiate the grasping based on the distance of the target object and the speed with which it is moving.

The environment was built upon the *GraspCan* policy's environment discussed above. The *GraspTiming* policy takes the can's distance and the speed of the conveyor belt as an input, and outputs an integer representing the amount of timesteps t the robot should wait before initiating the grasping motion. The *GraspCan* policy was applied to produce the appropriate angle for the grasping, and similarly to the previously described environment, a reward structure of 1 was used for every successful reward, while a negative reward proportional to the distance of the can at the downward position of the grasping was given.

The can was reset to a random position along the conveyor belt after every grasping attempt, where the distance was sampled from a uniform distribution between the values of $0.5m and 1.5m$ away from the TCP. Similarly to the previous environment, each episode lasted for 100 grasping attempts.

Due to the discrete action space a DQN architecture was chosen, similarly to the *GraspCan* policy. The difficulty of the environment was also incrementally increased, with the conveyor belt speed first set to a constant of $0.5\frac{m}{s}$, and the can set to fixed orientation in order to avoid rolling, as it was observed during the first training attempts to disrupt the training process.

Furthermore, due to a limitation of the simulation software, the conveyor belt speed could not be varied at every episode during the training process, as it caused the software to crash. Once the correct reward structure is found however, a variation in the speed of the can should be introduced in order to increase the flexibility of the grasping process.

6.6 Mid Level Policy

The policies trained in the Move2Can, GraspCan, Move2Bin and GraspCan environments are used in the training of the mid level DQN, PolicySelector. The goal of this policy is to choose the correct low level policies, in the right order, such that a can is moved to, grasped and manipulated to a bin. The trained agent is allowed to repeat policies if it fails, however, it will be penalized for cycle time. Additionally, the agent can choose to request a new can, if there is no acceptable low policy to select.

The training environment is a combination of each of the low level training environments for each of these policies to work correctly and can seen in Figure 6.6.

The action space is four discrete actions picking one of the three low level policies or picking a new can, which is provided by the high-level network discussed below. The state space is the position of the end-effector, the distance from the end-effector to the can and if a can is picked or not.

A sparse reward is given when a can is placed in a bin:

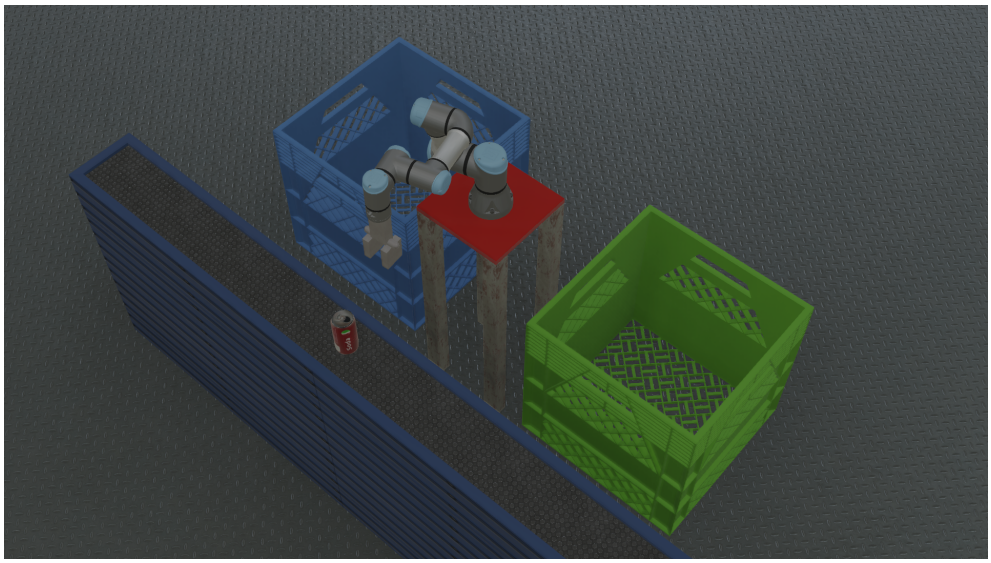


Figure 6.6: The environment designed to train the mid level DQN agent

$$r(s_t, a_t) \begin{cases} r_{success} & = 1 \\ r_{fail} & =\text{-}1 \end{cases} \tag{6.3}$$

where it gains a reward of 1 if it manages to put a can in a corresponding bin, and a negative reward of -1 if the can is incorrectly sorted, r_{fail}. A dense small negative reward is awarded every time step to incentivize quick results. A new action is selected after the previous selected action returns a "done" flag, meaning that the action has been executed. The success of the previously selected action is accessible to the policy from the environment.

6.7 High Level Policy

The high level policy, CanSelector, is trained using a Dueling DQN agent. The goal of this policy is to pick an appropriate can based on pose and color. Furthermore, since a Dueling DQN agent is used, the output of the advantage function for each action can be displayed to the user, in order to provide written explanations of the agent in the full implementation of the system. The simulation environment consists of the UR3 model on a table, six cans and the conveyor belt. The cans are colored red, green and blue with two of each and placed horizontally on the conveyor belt. The training environment can be seen in Figure 6.7.

The action space consists of seven discrete actions representing six potential cans to select and a "standby" action to not select a can. These cans can be selected

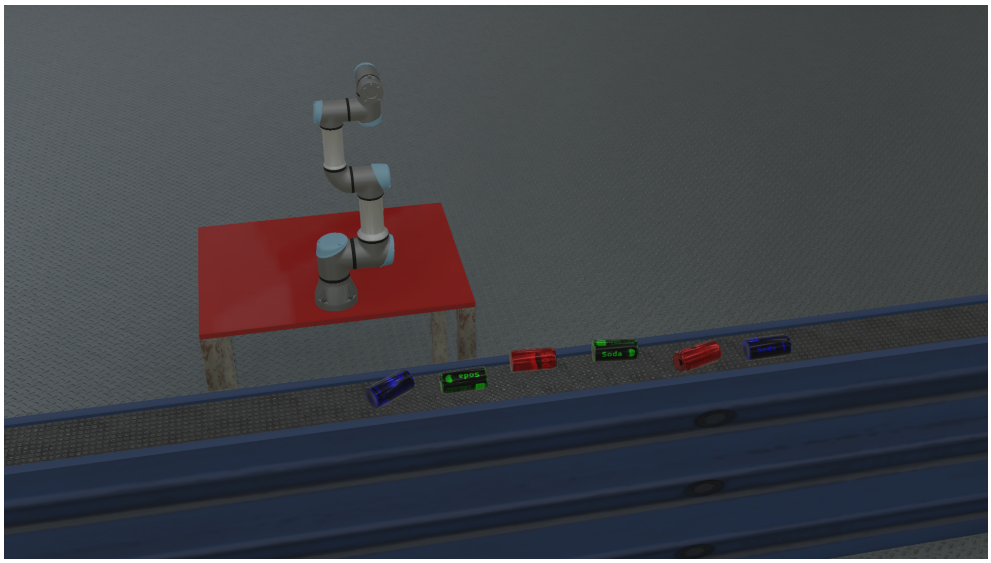


Figure 6.7: The environment designed to train the high level Dueling DQN agent

within a zone of 2.2 m from the robot, along the conveyor belt, representing the range of the conveyor belt covered by the camera. The state space is made up of the distance from the UR3, the full orientation and the color of each can.

The reward function is designed based on "dummy" functions representing the lower policies. A composite reward is constructed based on:

- The distance along the conveyor belt: the closer the selected can, the higher the reward. This reward aims to incentivize the time-effectiveness of the operation. This reward component should be acquired from the mid-level policy, by measuring how long each selection takes to complete.
- Orientation of the object: some objects could be harder to grasp than others. In the dummy function a simple case was used, where standing cans represented hard-to-grasp objects, and choosing them was penalized with a negative reward. In an integrated system, this reward component would be derived from the success of the GraspCan low level policy.
- The color of the can: A fixed reward scheme was developed for the different colors. This component of the reward reflects that some objects could be more valuable than others by their intrinsic value, although it can also represent a time element, as some objects might need more time to sort (i.e. the destination for the sorting is further away, therefore taking more time to access it). Similarly to the first reward component, the reward associated with the color (e.g. type) of the object is also derived from the mid-level policy.

The overall reward function therefore can be seen at equation 6.4

$$r(s_t, a_t) = \alpha r_{color} + \beta r_{distance} + \gamma r_{orientation} \tag{6.4}$$

where α, β and γ represent scalar values that vary the importance of the different components.

As the graphical explanation developed for the experiment only needs a target, the output of this policy need only be parsed with the available cans and passed to the function generating the visual effects of the explanation.

Chapter 7

Results

This chapter presents the results of the training and testing processes, assessing parts of the system by checking if the specification requirements for unit testing are fulfilled.

Low level Training And Hyper Parameter tuning

Hyper parameter tuning has been used to increase the performance and training speed of this agent. First the agent of the Move2Can policy seen in section 6.2 is tuned, since it is assumed to be the hardest to train, however the reward graph for each policy will be presented at the end of this section. For the purpose of this project, the hyper parameters including the learning rate, batch size and network size are tuned. The standard hyper parameter values from the original paper [73] are used as a baseline along with two additional versions of each of the parameters are presented. The mean score of the last 100 episodes will also be shown, where a score above 1100 is considered perfect play. All versions have been trained for 5000 episodes.

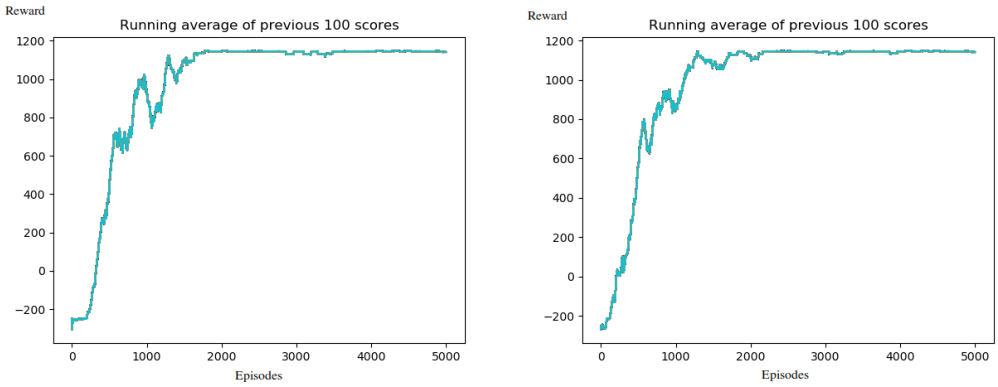
The different versions of training are shown in Table 7.1 and can be seen that baseline and version 4 achieved acceptable results for solving the task. The training graph for each of these can be seen in Figure 7.1.

A video of each of the versions can be seen on YouTube. Here it can be noticed that version four has trouble finding the correct pose and the control is consistent compared to the baseline version. For this reason, the baseline version is chosen as the agent for the Move2Can policy.

The Move2Bin policy has been trained for 5000 episodes with identical hyper-parameters as the Move2Goal policy. The training graph can be seen in Figure 7.2

	Learning rate	Batch size	Network size	Mean reward
Baseline	0.0003	512	256,256	1169
Version 2	**0.0001**	512	256,256	1142
Version 3	**0.0005**	512	256,256	1017
Version 4	0.0003	**256**	256,256	863
Version 5	0.0003	**1024**	256,256	602
Version 6	0.0003	512	**128,128**	995
Version 7	0.0003	512	**512,512**	982

Table 7.1: Training results of Move2Can agent with different hyperparameters. Baseline is the original and bold numbers are the changed setting for SAC. The mean reward is the last 100 episodes of training.



(a) Baseline **(b)** Version 4

Figure 7.1: Training graph of the two successful versions of the hyper parameter tuning presented in Figure (a) and (b) with a learning rate of 0.0003 and 0.0001, respectively.

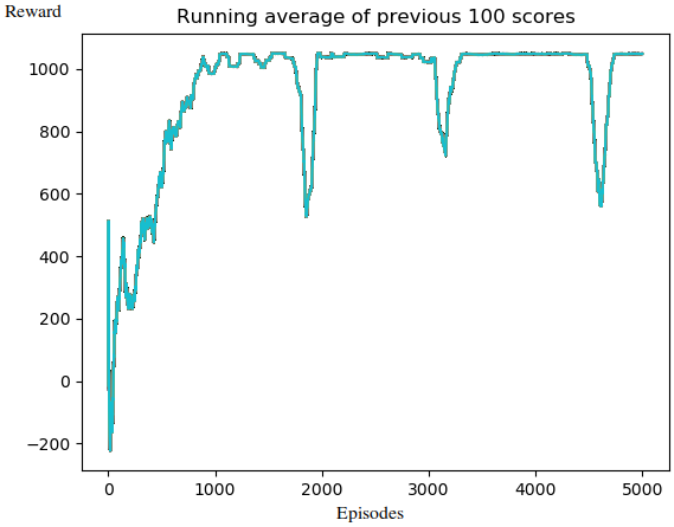


Figure 7.2: The environment is designed to train the agent to move to one of the green or blue bins. When a bin is picked the robot will move into the proximity of the selected bin.

This agent managed an average score of 1049 which is considered adequate for solving the Move2Bin policy. A video of of the Move2Bin training can be seen on YouTube.

GraspCan

Initially the algorithm could not achieve a higher than a maximum of 55% success rate. The reason for this however was that the discount factor γ was left at the default setting of 0.99, meaning that the expected rewards from the future states were accounting for 99% of reward acquired, while the current state only contributed 1%. Amending for this mistake, and lowering the discount factor drastically improved the performance of the algorithm, converging in less than 40 episodes (see Figure 7.3).

Since the performance of the algorithm was sufficient (>99% success rate), no further hyperparameter tuning was conducted, and a final layer size of 128 was used for both layers, with a learning rate $\alpha = 0.003$ and discount factor $\gamma = 0.00$, as the cans are independent from each other, the expectation of rewards from the future state should not be taken into account in this scenario and therefore are completely ignored.

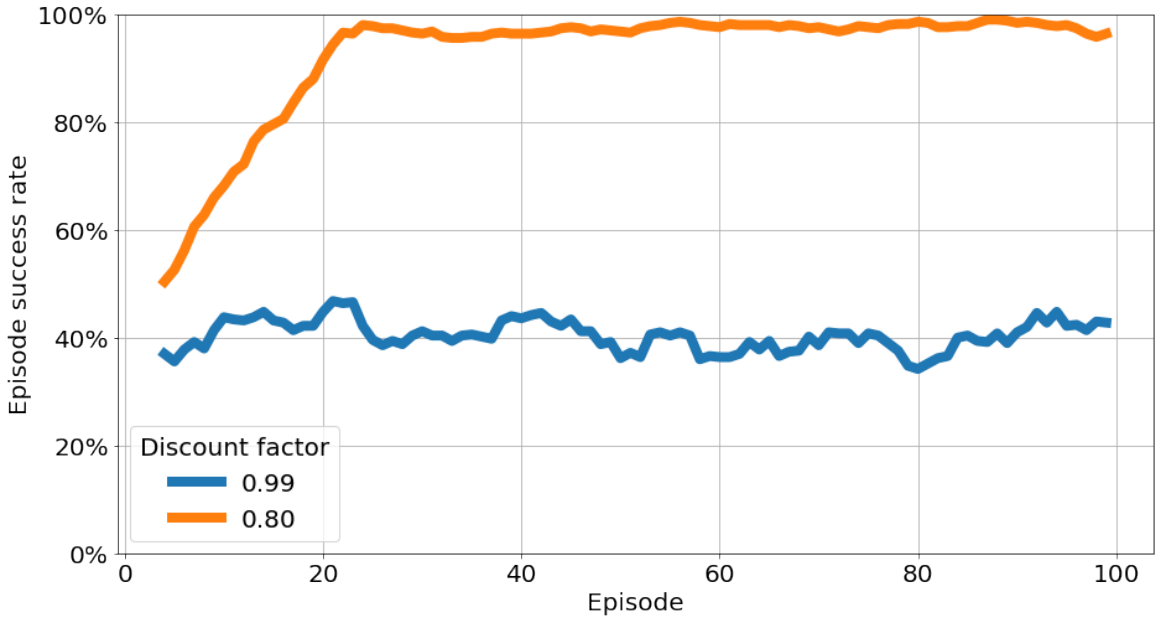


Figure 7.3: Similarly to the GraspCan environment, a one-shot learning yields a more stable training

Additionally, the effect of displacement (measured from the TCP) was assessed

by randomly placing the can at a position within a range of 1, 3, 6 and 8cm away from the TCP position projected onto the plane of the conveyor belt. The results show that the requirement specifications of 90% success rate can be fulfilled by placing the can up to 6cm from the TCP position (Figure 7.6). This result means that considering the Move2Can policy, the error margin of the distance from the destination could be increased to 3cm. This however was not evaluated due to time constraints, and therefore left for future work.

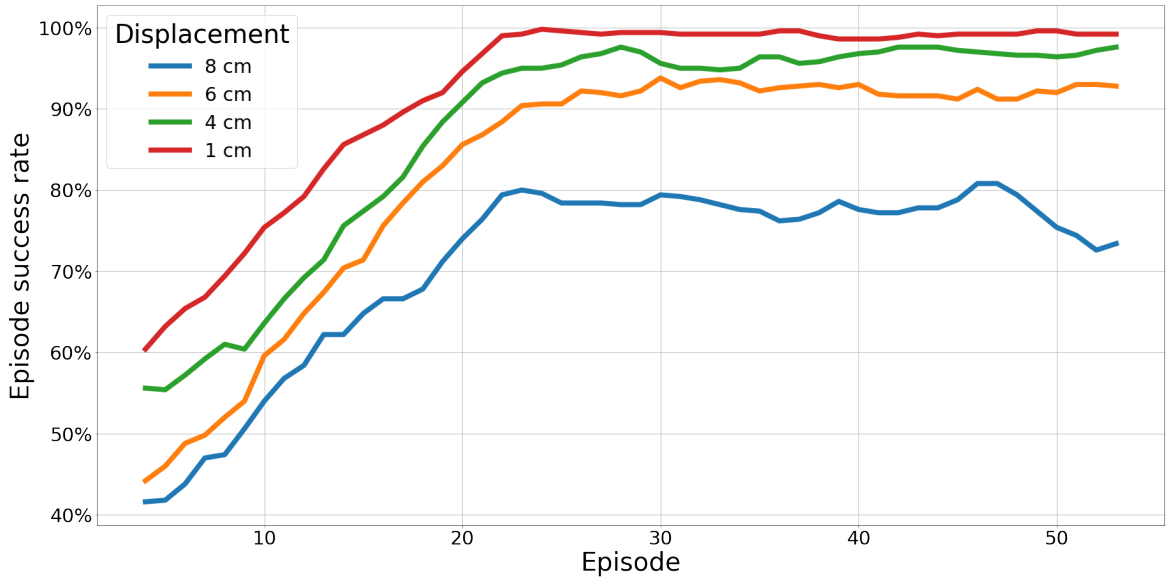


Figure 7.4: Similarly to the GraspCan environment, a one-shot learning yields a more stable training

GraspTiming

The timing of the grasping has been trained on a DQN architecture with two fully connected layers with 512 neurons in each layer. Different layers sizes were experimented with, however a maximum successful grasping rate of 20% was not surpassed. The most consistent training process was achieved with the discount factor γ of 0, reducing the learning process to one-shot learning.

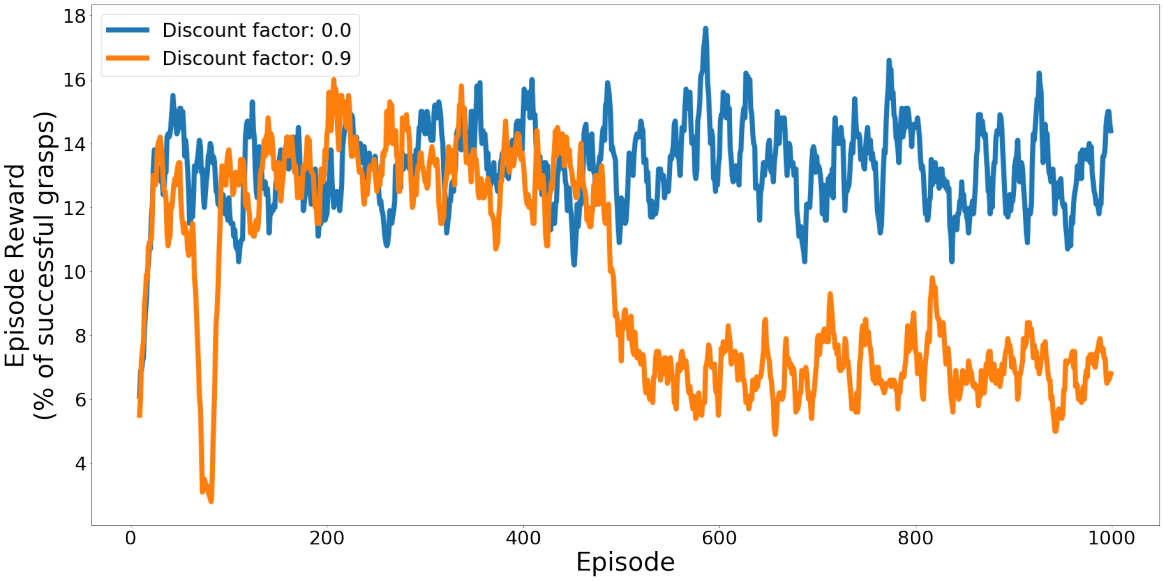


Figure 7.5: Similarly to the GraspCan environment, a one-shot learning yields a more stable training

Although reducing the discount factor stabilized the training, it did not result in an improved performance. Hyper-parameter tuning on the batch size, learning rate and the layer size did not succeed either, which indicates that the problem lies within the setup of the environment. Some of the possible reasons for this failure is discussed in the next chapter.

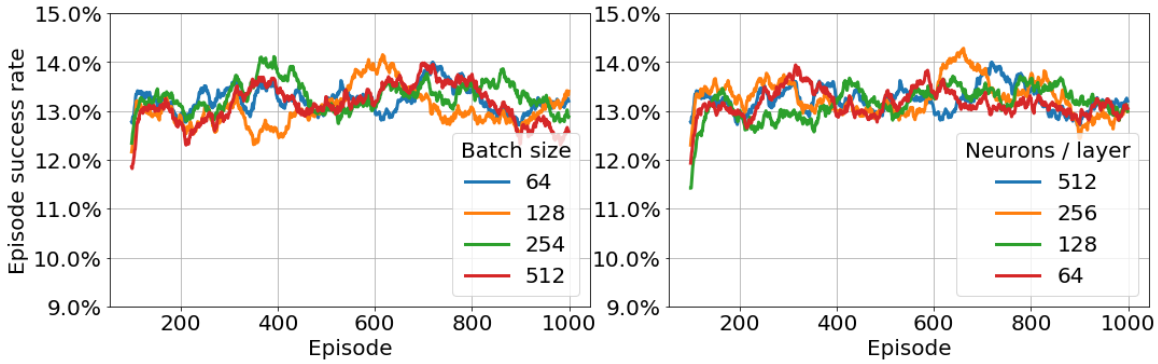


Figure 7.6: An attempt to train the network with different batch sizes and layer sizes did not result in an increase in performance.

Mid Level Policy

Although a proof-of-concept implementation of the policy was implemented, in order to test the algorithm, the low-level policies need to function well-enough. However, due to the current limitations posed by the GraspTiming policy and further limitations of time, the mid-level policy was not tested and its performance evaluation is left for future work.

High Level Policy

As described in the previous chapter, a proof-of-concept implementation of the high-level policy was developed, where the policy is able to learn which cans to pick based on the outputs of functions serving as "dummy" policies.

In order to evaluate the actual working performance of the policy, similarly to the mid-level policy, the lower level policies need to be functioning. The testing of this policy therefore is left for future work.

7.1 Unit Test Evaluation

The low level policies are executed individually in order to perform unit testing on the system and thereby assess the performance of the system against the requirements from section 4.4.

UT-1 - The system must be able to create a trajectory to a specific pose within the workspace within a success rate of 90%.

To verify this requirement low level policies, Move2Can and Move2Bin, were tested throughout 100 episodes. Move2Can and Move2Bin managed to reach a success rate of 98% and 100%, respectively. Requirement **UT-1** is thereby deemed fulfilled.

UT-2 - The system must be able to create a trajectory to a specific object to be grasped within the workspace within a time of 1 seconds.

To verify this requirement the low level policy Move2Can was tested throughout 100 episodes. Move2Can managed a time to execute a trajectory of 0.72 seconds. Requirement UT-2 is thereby deemed fulfilled.

UT-3 - The system must be able to grasp objects with a success rate of 90%.

As mentioned in the previous section, due to the decrease in the discount factor the system is able to grasp objects (cans) with a success rate of close to 100%. This requirement was however only fulfilled with stationary objects, further evaluation

is needed therefore, once the necessary low level policies are successfully trained. Requirement UT-3 is thereby deemed partially fulfilled.

UT-4 - Timing requirement: The system must be able to time the grasping operation, adjusting to the speed of the conveyor belt.

As discussed in the above section, the GraspTiming policy could not improve further than approx. 15% success rate. This requirement is thereby deemed unfulfilled.

UT-5 - Explanation requirement: The system must be able to indicate where it's at in the decision process.

As the explanation interface developed for the experiment is fully capable of presenting graphical explanations on a display, and can be partially reused, this requirement needs only the high-level policy to be completed and integrated with the low-level policies. This requirement is thereby deemed partially fulfilled.

7.2 Integration Test Evaluation

Due to limitations on time, and the failure to fulfill UT-4 and UT-3, the integration tests are not performed and therefore left unfulfilled. Combining parts of the system into an integrated whole and assessing it against the specifications are also left for future work.

Chapter 8

Discussion

This chapter provides a discussion on multiple aspects of the report. First some of the experiment results are expanded and reflected upon, then the system design is evaluated on the requirement specifications in light of the results presented in the previous chapter, followed by the description of directions for future work and possible improvements.

8.1 Discussion Of The Experiment

One of the crucial elements of the effect of explainability that the current experiment design does not capture is the familiarity with the system. Ideally, workers who sort waste would be highly familiar with both the task and the robot, and would possibly utilize the explanations in a different way, or would prefer different kinds of explanations than the participants, who do not possess such expertise.

Although the experiment was designed in a way to incentivize participants to let the robot sort, and therefore introduced a bias by favoring conditions with explanations, the additional score was designed to reflect the potential benefit of having a robot perform the sorting task. Nevertheless, it could be argued that the score measurement was not really a measure of task performance, but a measure of "who let the robot play more", with the control condition being in an obvious disadvantage. Therefore the question of *"Is an explanation really necessary?"* is only answered with a yes, because the experiment was designed in a way that made the explanation necessary to perform well. This problem however could be addressed by changing the experiment setup in a way that the user would have to "pay" for the robot's mistakes by having to perform an "expensive" operation, like walking a certain distance to get around the table to pick up the object missed by the robot. This change might also be reflected in measurements such as the NASA-TLX, as the prevention of mistakes

could in a scenario like this decrease the physical labor the user has to perform.

Another question is that of the error rate of the robot, which is most likely to be related to the drop in the trust score differences between the pre-test and post-test questionnaires that was observed throughout all the conditions. We speculate that the drop could be related to disappointment (and sometimes frustration) felt, that an error rate of 30% could cause. Furthermore, we assume that on top of the error manifesting in mislabelled data, some of the difference is likely stemming from the "miscalculations" that come from the implementation of the algorithms. In some cases when the robot moved down to pick up a can (in the front position), but realized that the can had moved too far, it had gone to the other (rear) pickup position where it could reach the can. It was observed, however, that upon witnessing the "miscalculation", some participants took the can before the robot could try to pick it up again. If the same situation occurred in the rear position, the robot could not amend for the mistake, and the can was left unpicked. Although the explanation immediately changed when the robot "realized" that the can is getting out of reach, the user's trust in the explanation could've dropped significantly due to these unplanned errors.

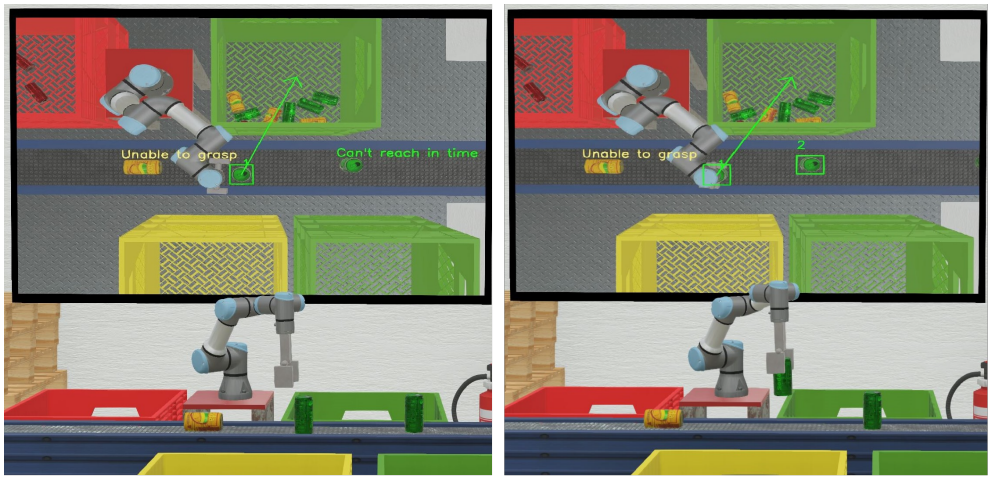


Figure 8.1: Upon successfully grasping the can, the estimates of reaching the other can has been updated. This change in effect could have had a confounding effect, as the explanations displayed were subject to change and were not final.

These errors lead to another question regarding the experiment design, as due to the several decisions involved, it is really difficult to control the user's experience. As no solution seemed to solve the issue, this lack of rigorous control was accepted as a drawback of the design. A probable, although time-consuming solution could be to control the can placements by designing the flow of objects manually.

One of the benefits, however, of designing an experiment in parallel with the actual HRL system was that manually programming the robot to perform the sorting task provided valuable insights into the nature of the task, helping the design process of the system. An example of this is the insight gained about the time-sensitivity of the grasping operation, which was not considered during the initial design considerations. It was only included in the design after conducting the experiment and noticing how sensitive an issue the timing of the grasping is.

People figured out that the robot doesn't take the can's that are lying It was also observed that the virtual nature of the task introduced a bias towards people who had more experience with coordinating mouse movements, such as participants who frequently play video games. Some people had a real struggle trying to click on the cans while the conveyor belt was moving, while others did so with ease. Conducting the experiment within a virtual reality setup however might solve this issue, with the added benefit of the possibility to record the eye movements of the participants, which could indicate how often the user looks at the screen.

8.2 Discussion Of The System Design

The low level policies of the system design were tested and largely successful. The training of the GraspTiming policy however did not yield satisfactory results. The solution could be further simplifying the environment until the bottleneck could be pinpointed, which in this case would consist of limiting the random placement of the cans from a range of $1m$ to a smaller, and more manageable distance, and increasing it gradually. An investigation into the field of curriculum learning that deals with the gradual learning steps within reinforcement learning could provide useful advice.

The Move2Can and Move2Bin had high success rates and low execution time, however, the end-effector's angle of orientation in many cases were tilted from the ideal orientation used to train the GraspCan policy. This can be seen in the following figure 8.2.

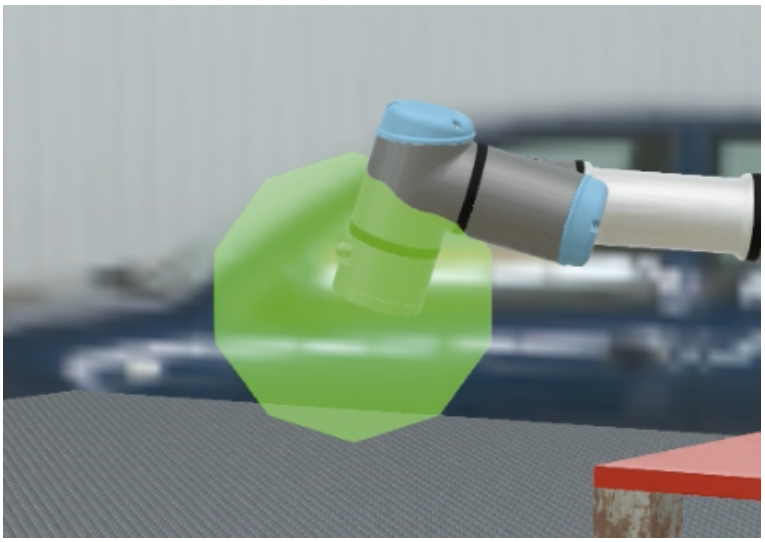


Figure 8.2: The tilting of the end-effector when running the Move2Can policy when reaching the goal.

This offset would have a large impact on the GraspCan policy, as it would make grasping cans harder to execute. Further development should be made to reduce this angular error before the full integration of the system can be developed.

The system design was heavily relying on the *supervisor* class of *WeBots*, which substituted the observations that in a real-life setting would have been provided by a depth camera and an object recognition algorithm. This would be a great part of a potential implementation of the system in the real world and further research should be made into computer vision algorithms like YOLO, that could enable this functionality.

Most control of the UR3 model was made in joint space since the inverse kinematics packages experimented with were too computationally heavy to use within a training process. Further research should be made into using faster forms of inverse kinematics since this could help to simplify the DRL training of the system. Instead of having 6 actions for each joint of the UR3, only 3 actions would be needed for the Cartesian displacement of the end-effector, with the added cost of having to account for accessible work-spaces and singularities.

8.3 Future Work

Additional analysis could be run on the data collected during the experiment, comparing for example the amount of cans that were missed by both the participant and the robot in each condition (as an indicator of performance), or further analysing

the types and colors of cans that were sorted in each zone. The demographic data was also only briefly considered, and additional effects or correlations might be discovered there.

Future work includes combining the individual system components to produce an integrated system. Any updates on the status of the system will be published on the repository and can be found at [74]

Running an experiment with the functioning system however would require the availability of the *written* explanations, which, in turn, would require that Shapley Additive Explanations to evaluate the state-space of the high-level layer and provide estimations of the contribution of each element of the state-space. These explanations were also partially implemented, however as the experiment analysis suggested a preference towards the graphical explanations, the focus shifted from the written explanations, and are also left for future work.

Once a functioning system has been developed, an extension of the learning algorithm that could utilize the actions of the user as feedback could be adapted to improve the cooperation performance. Such a system could aim at learning which types of objects are preferred by the user or the robot and thereby dynamically learn to adapt.

Sim2Real

Sim2Real refers to the method of going from a simulation to a real-life environment which is the future final goal of this project. The problem arising from adapting the designed simulation to the real world should be researched. This problem is also known as the reality gap and is a common barrier to real-world implementation. To bridge this gap to reality, mainly three methods are suggested in the literature. The first is system identification, where specific important aspects of the real world are modeled and taken into account when training in the simulation. In future work of this project, further research should look into things such as the time delay of the system response or system's state construction in continuous control, which will differ from discrete time steps of the simulation. The second method is called domain adaptation where the model is trained again in the real world by the use of transfer learning which fine-tunes the model to better fit the complexity of the real world. For this project, it would mean that all the policies would be trained individually again in the real world. The third to bridging the gap to reality is domain randomization. Here the goal is to randomize simulation to capture aspects of the real world when training. For the future research and development of this thesis, this would require more complex simulation environments for all the policy levels in the system. [46]

Chapter 9

Conclusion

This chapter summarises the project, outlining the obtained results and relating it to the problem formulation described at the outset of the project.

The main focus of this thesis revolved around the concept of explainability in reinforcement learning. As a use-case, a cooperative sorting task was chosen, in a waste recycling setting. Hierarchical reinforcement learning was chosen as the basis of the system architecture, which was expanded by the possibility to provide explanations.

Different modalities and types of explanations were investigated within the framework of an experiment that utilized an interactive virtual environment created within the WeBots simulation software. Four conditions were used to examine the effects of graphical and written explanations, comparing them both to each other, to their combined effect and to a control group.

Although a low sample-size prevented us from reaching solid conclusions, the results suggest that graphical explanations could be preferred over written explanations both in terms of increasing task performance, and in terms of user preference. Further studies are required to confirm these findings, for the purposes of this thesis however the results were considered when designing the system, and the focus of explainability was narrowed to graphical explanations.

The hierarchical reinforcement learning system designed within this thesis consists of three layers, where low-level policies are concerned with low-level motor actions that do not directly contribute to the explanations. The mid-level policy was designed to navigate between the low-level policies, learning the right policy to choose as an action at each time-step, while the high-level policy was designed to choose from the available cans based on the performance of the low-level system.

The training and implementation of all but one of the low level policies were successful, achieving the performance requirements specified in Chapter 4. The mid-level policy and high-level policies were implemented as a proof-of-concept, but were not evaluated as an integrated system; the integration and evaluation of the system are left for future work.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search", *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] J. Ker, L. Wang, J. Rao, and T. Lim, "Deep learning applications in medical image analysis", *Ieee Access*, vol. 6, pp. 9375–9389, 2017.

[4] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: A new paradigm to machine learning", *Archives of Computational Methods in Engineering*, pp. 1–22, 2019.

[5] S. Russell and P. Norvig, "Artificial intelligence: A modern approach", 2020.

[6] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey", in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, 2017, pp. 8–13.

[7] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.", *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[8] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai", *Information Fusion*, vol. 58, pp. 82–115, 2020.

[9] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning", in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89. DOI: 10.1109/DSAA.2018.00018.

[10] T. Hester, M. Quinlan, and P. Stone, "Rtmba: A real-time model-based reinforcement learning architecture for robot control", in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 85–90.

[11] A. Y. Ng, H. J. Kim, M. I. Jordan, S. Sastry, and S. Ballianda, "Autonomous helicopter flight via reinforcement learning.", in *NIPS*, Citeseer, vol. 16, 2003.

[12] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations", in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2074–2081. DOI: 10.1109/ROBOT.2010.5509621.

[13] T. Hester, M. Quinlan, and P. Stone, "Generalized model learning for reinforcement learning on a humanoid robot", in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 2369–2374.

[14] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics", *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[15] *Zenrobotics*. [Online]. Available: https://zenrobotics.com/.

[16] *Waste robotics*. [Online]. Available: https://wasterobotic.com/en/about/.

[17] *Amp robotics*. [Online]. Available: https://www.amprobotics.com/about-us.

[18] W. Flower, *What operation green fence has meant for recycling*, 2016. [Online]. Available: https://www.waste360.com/business/what-operation-green-fence-has-meant-recycling.

[19] Eurostat, *Eu exports of recyclables to china fallen sharply*, 2020. [Online]. Available: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/DDN-20200709-01.

[20] Xinhua, *China to ban all imports of solid waste from 2021*, 2020. [Online]. Available: http://www.china.org.cn/business/2020-11/27/content_76956187.html.

[21] C. Staub, *Coronavirus pandemic disrupts recycling sector*. [Online]. Available: https://resource-recycling.com/recycling/2020/03/17/coronavirus-pandemic-disrupts-recycling-sector/.

[22] C. Writh, *Amp robotics partners with waste connections to deploy ai-guided recycling robots*, 2020. [Online]. Available: https://www.amprobotics.com/newsroom/amp-robotics-partners-with-waste-connections-to-deploy-ai-guided-recycling-robots.

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

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning An introduction (Second Edition)*. MIT Press, 2018.

[25] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation", in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, IEEE, 2019, pp. 590–595.

[26] A. C. Scott, W. J. Clancey, R. Davis, and E. H. Shortliffe, "Explanation capabilities of production-based consultation systems", STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, Tech. Rep., 1977.

[27] H. A. Simon, "What is an "explanation" of behavior?", *Psychological science*, vol. 3, no. 3, pp. 150–161, 1992.

[28] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling, "Explainable agents and robots: Results from a systematic literature review", in *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1078–1088.

[29] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences", *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[30] D. A. Norman, "Some observations on mental models", *Mental models*, vol. 7, no. 112, pp. 7–14, 1983.

[31] A. Theodorou, R. H. Wortham, and J. J. Bryson, "Why is my robot behaving like that? designing transparency for real time inspection of autonomous robots", in *AISB Workshop on Principles of Robotics*, 2016.

[32] J. Tullio, A. K. Dey, J. Chalecki, and J. Fogarty, "How it works: A field study of non-technical users interacting with an intelligent system", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2007, pp. 31–40.

[33] A. Falcon, "Aristotle on Causality", in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Spring 2019, Metaphysics Research Lab, Stanford University, 2019.

[34] M. M. A. de Graaf and B. F. Malle, "People's explanations of robot behavior subtly reveal mental state inferences", in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019, pp. 239–248. DOI: 10.1109/HRI.2019.8673308.

[35] S. Stange and S. Kopp, "Effects of a social robot's self-explanations on how humans understand and evaluate its behavior", in *Proceedings of the 2020 ACM/IEEE international conference on human-robot interaction*, 2020, pp. 619–627.

[36] H. Johnson and P. Johnson, "Explanation facilities and interactive systems", in *Proceedings of the 1st international conference on Intelligent user interfaces*, 1993, pp. 159–166.

[37] D. Gunning, "Explainable artificial intelligence (xai)", *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, no. 2, 2017.

[38] D. Gunning and D. Aha, "Darpa's explainable artificial intelligence (xai) program", *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[39] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong, "Too much, too little, or just right? ways explanations impact end users' mental models", in *2013 IEEE Symposium on visual languages and human centric computing*, IEEE, 2013, pp. 3–10.

[40] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable ai: Challenges and prospects", *arXiv preprint arXiv:1812.04608*, 2018.

[41] J. M. Schraagen, P. Elsasser, H. Fricke, M. Hof, and F. Ragalmuto, "Trusting the x in xai: Effects of different types of explanations by a self-driving car on trust, explanation satisfaction and mental models", in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, SAGE Publications Sage CA: Los Angeles, CA, vol. 64, 2020, pp. 339–343.

[42] A. Holzinger, A. Carrington, and H. Müller, "Measuring the quality of explanations: The system causability scale (scs)", *KI-Künstliche Intelligenz*, pp. 1–6, 2020.

[43] J. Brooke, "Sus: A "quick and dirty'usability", *Usability evaluation in industry*, vol. 189, 1996.

[44] N. Xie, G. Ras, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated", *arXiv preprint arXiv:2004.14545*, 2020.

[45] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning", 2015.

[46] H. Dong, Z. Ding, and S. Zhang, *Deep Reinforcement Learning Fundamentals, Research and Applications*. Springer, 2020.

[47] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, "A framework for efficient robotic manipulation", *arXiv preprint arXiv:2012.07975*, 2020.

[48] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning", in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2020, pp. 1461–1466.

[49] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 4238–4245.

[50] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, *et al.*, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model", *Annals of Applied Statistics*, vol. 9, no. 3, pp. 1350–1371, 2015.

[51] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", *arXiv preprint arXiv:1705.07874*, 2017.

[52] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization", *International Journal of Computer Vision*, 2019.

[53] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning", *Knowledge-Based Systems*, vol. 214, p. 106 685, 2021.

[54] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey", in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2020, pp. 77–95.

[55] B. Hayes and J. A. Shah, "Improving robot controller transparency through autonomous policy explanation", in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, IEEE, 2017, pp. 303–312.

[56] T. Shu, C. Xiong†, and R. Socher, "Hierarchical and interpretable skill acquisition in multi-task reinforcement learning", 2017.

[57] J. Duan, S. E. Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data", *IET Intelligent Transport Systems*, vol. 14, no. 5, pp. 297–305, 2020.

[58] B. Beyret, A. Shafti, and A. A. Faisal, "Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation", *arXiv preprint arXiv:1904.06703*, 2019.

[59] T. Osa, J. Peters, and G. Neumann, "Hierarchical reinforcement learning of multiple grasping strategies with human instructions", *Advanced Robotics*, vol. 32, no. 18, pp. 955–968, 2018.

[60] B. F. Malle, *How the mind explains behavior: Folk explanations, meaning, and social interaction*. Mit Press, 2006.

[61] *Universal robots launches ur3*, Accessed: 2021-06-01. [Online]. Available: https://www.universal-robots.com/about-universal-robots/news-centre/universal-robots-launches-ur3/.

[62] *Schunk egp 50-n-n-b gripper*, Accessed: 2021-06-01. [Online]. Available: https://schunk.com/jp_en/gripping-systems/product/39759-0310960-egp-50-n-n-b//.

[63] B. F. Malle and J. Knobe, "Which behaviors do people explain? a basic actor–observer asymmetry.", *Journal of Personality and Social Psychology*, vol. 72, no. 2, p. 288, 1997.

[64] K. Vertanen, *Nasa-tlx in html and javascript*, online, Accessed: 2021-05-20. [Online]. Available: https://www.keithv.com/software/nasatlx/.

[65] K. Schaefer, "The perception and measurement of human-robot trust", 2013.

[66] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots", *International journal of social robotics*, vol. 1, no. 1, pp. 71–81, 2009.

[67] S. Tulli, F. Correia, S. Mascarenhas, S. Gomes, F. S. Melo, and A. Paiva, "Effects of agents' transparency on teamwork", in *Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, D. Calvaresi, A. Najjar, M. Schumacher, and K. Främling, Eds., Cham: Springer International Publishing, 2019, pp. 22–37.

[68] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later", in *Proceedings of the human factors and ergonomics society annual meeting*, Sage publications Sage CA: Los Angeles, CA, vol. 50, 2006, pp. 904–908.

[69] A. Field and G. Hole, *How to design and report experiments*. Sage, 2002.

[70] Webots, *Http://www.cyberbotics.com*, C. Ltd., Ed., Open-source Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com.

[71] *Webots portfolio: They use webots*, online, Accessed: 2021-04-29. [Online]. Available: https://cyberbotics.com/#portfolio.

[72] *Gym*, Accessed: 2021-06-02. [Online]. Available: https://gym.openai.com/.

[73] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor", *arXiv preprint arXiv:1801.01290*, 2018.

[74] A. P. Madsen and M. Adamik, *Explainable hierarchical reinforcement learning for a cooperative sorting task*, https://github.com/Nitrow/P10-XRL, 2021.

Appendices

Appendix A

Experiment

Participant ID:

Demographic Survey

Age: _____

Gender:
- ◯ Male
- ◯ Female
- ◯ Other

Nationality: _____

Education (select highest completed)**:**

- ◯ Elementary School
- ◯ High School
- ◯ Bachelor's Degree (BSc)
- ◯ Master's Degree (MSc)
- ◯ Doctoral Degree (Ph.D.)
- ◯ Prefer not to say

Occupation: _____

Have you ever interacted with a robot?

- ◯ Yes
- ◯ No

How many times?
- ◯ 1-2 times
- ◯ 2-5 times
- ◯ 5-10 times
- ◯ 10+ times

How many movies, television shows or computer games have you watched / played that featured robots?

◯ around 1 or 2 ◯ between 2-5 ◯ between 5-10 ◯ More than 10

Have you ever controlled a robot?

- ◯ Yes
- ◯ No

How many times?
- ◯ 1-2 times
- ◯ 2-5 times
- ◯ 5-10 times
- ◯ 10+ times

Have you ever programmed a robot?

- ◯ Yes
- ◯ No

How many times?
- ◯ 1-2 times
- ◯ 2-5 times
- ◯ 5-10 times
- ◯ 10+ times

If you would like to participate in a draft of winning a gift card worth of 200 DKK, please write the email address you would like to be notified on below:

Participant ID:

Please rate your impression of the robot on these scales:

Incompetent	1	2	3	4	5	Competent
Mechanical	1	2	3	4	5	Organic
Irresponsible	1	2	3	4	5	Responsible
Unconscious	1	2	3	4	5	Conscious
Foolish	1	2	3	4	5	Sensible
Unpleasant	1	2	3	4	5	Pleasant
Ignorant	1	2	3	4	5	Knowledgeable
Artificial	1	2	3	4	5	Lifelike
Unintelligent	1	2	3	4	5	Intelligent
Inert	1	2	3	4	5	Interactive

Please rate your emotional state on these scales:

Anxious	1	2	3	4	5	Relaxed
Happy	1	2	3	4	5	Sad
Quiescent	1	2	3	4	5	Surprised
Bored	1	2	3	4	5	Intrigued
Agitated	1	2	3	4	5	Calm
Angry	1	2	3	4	5	Peaceful

ID:	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
What % of the time will this robot be ...											
Dependable	○	○	○	○	○	○	○	○	○	○	○
Reliable	○	○	○	○	○	○	○	○	○	○	○
Unresponsive	○	○	○	○	○	○	○	○	○	○	○
Predictable	○	○	○	○	○	○	○	○	○	○	○
What % of the time will this robot ...											
Act consistently	○	○	○	○	○	○	○	○	○	○	○
Function successfully	○	○	○	○	○	○	○	○	○	○	○
Malfunction	○	○	○	○	○	○	○	○	○	○	○
Have errors	○	○	○	○	○	○	○	○	○	○	○
Provide feedback	○	○	○	○	○	○	○	○	○	○	○
Meet the needs of the task	○	○	○	○	○	○	○	○	○	○	○
Provide appropriate information	○	○	○	○	○	○	○	○	○	○	○
Communicate with people	○	○	○	○	○	○	○	○	○	○	○
Perform exactly as instructed	○	○	○	○	○	○	○	○	○	○	○
Follow directions	○	○	○	○	○	○	○	○	○	○	○

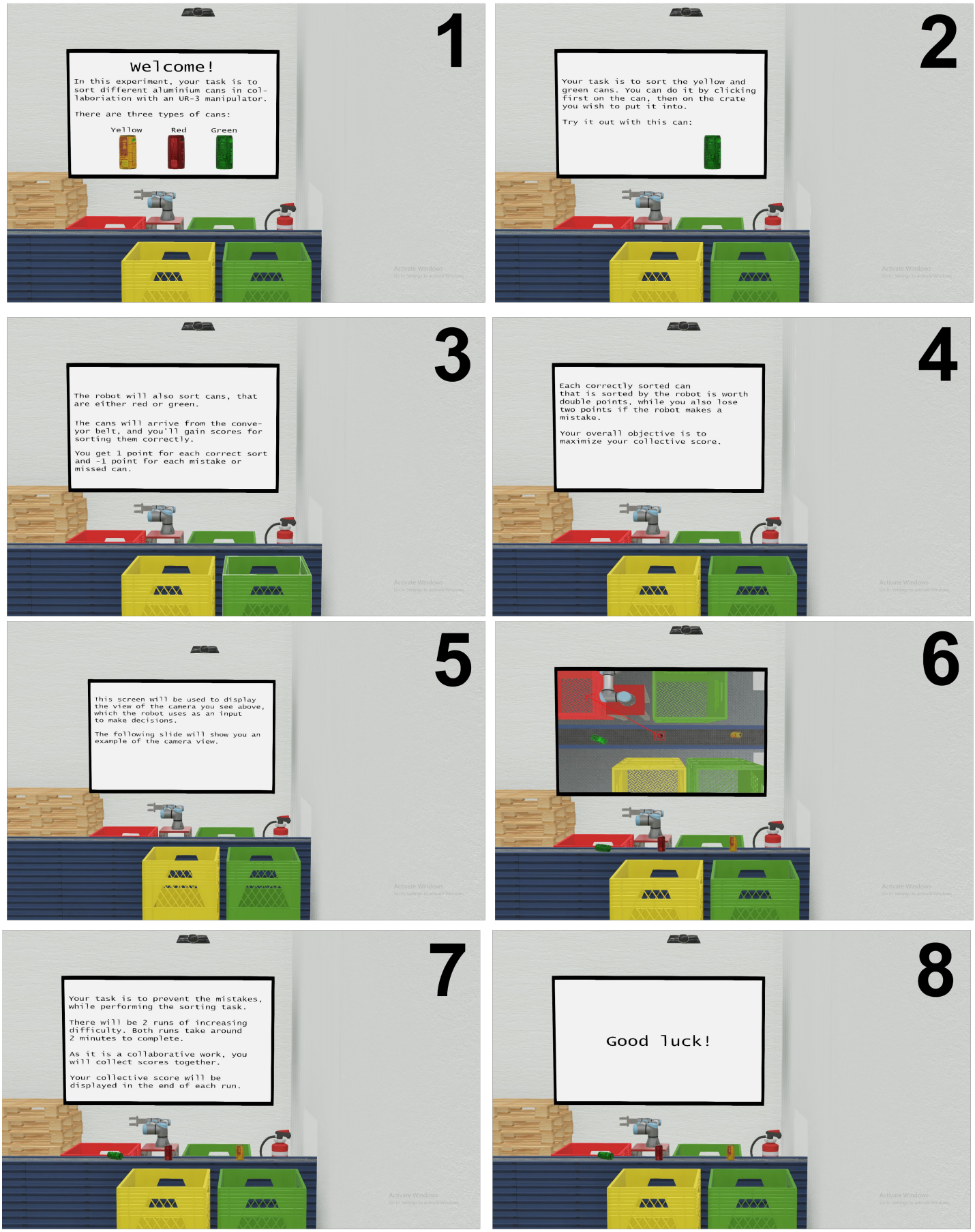


Figure A.1: The slides used to display the introductory texts. Frame 6 displays the *visual explanation* experimental condition.

Appendix B

Training And Hyperparameter Tuning

In this appendix the reward graphs for all training will be presented. Graphs B.1, B.2, B.4, show the learning of the Move2Can training, graph B.5 showing the Move2Bin training and

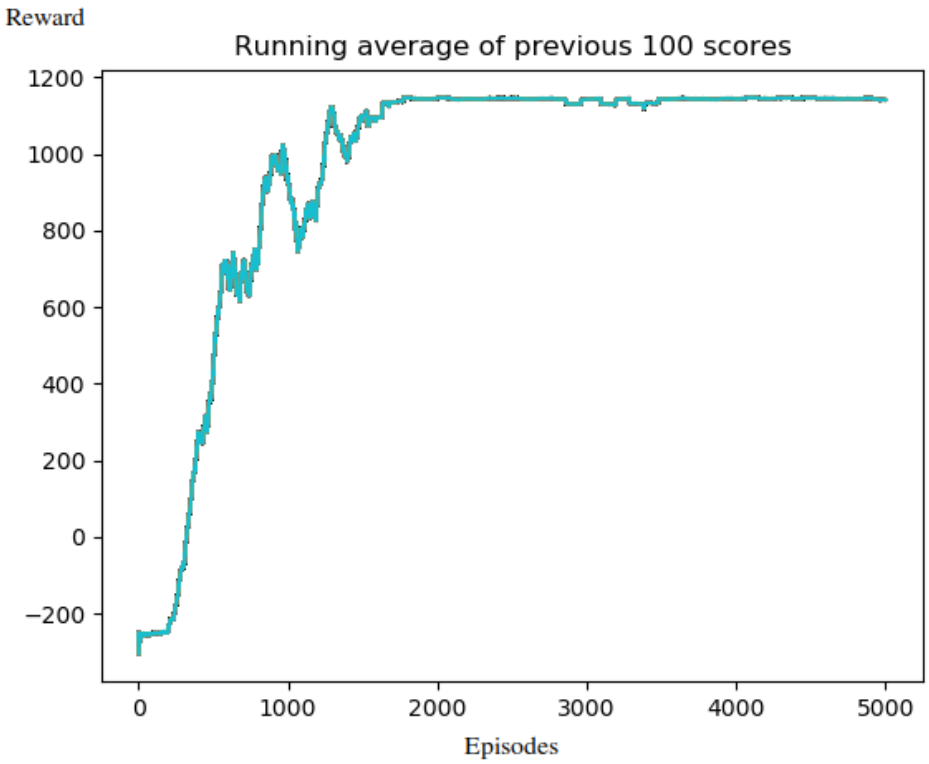
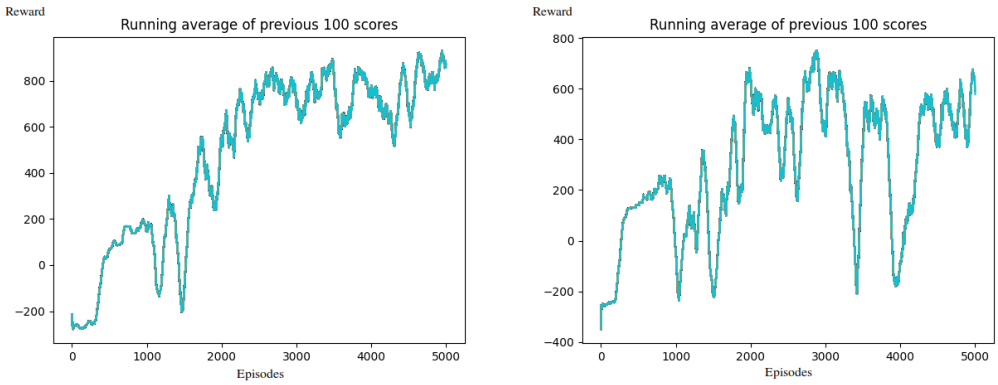
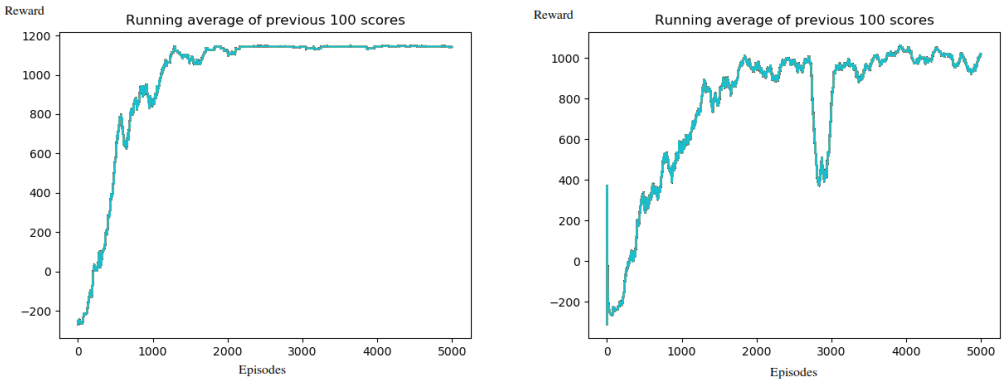


Figure B.1: Move2Can training with baseline version. Learning rate 0.0003, batch size 512 and network size 256,256



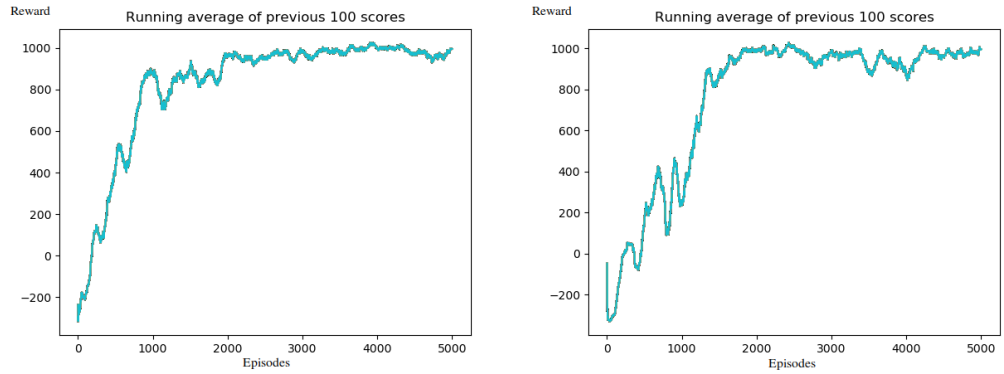
(a) Batch size 256 **(b)** Batch size 1024

Figure B.2: Move2Can training with different batch sizes compared to baseline version.



(a) Learning rate 0.0001 **(b)** Learning rate 0.0005

Figure B.3: Move2Can training with different learning rates compared to baseline version.



(a) Neural network of 128,128 **(b)** Neural network of 512,512

Figure B.4: Move2Can training with different neural networks compared to baseline version.

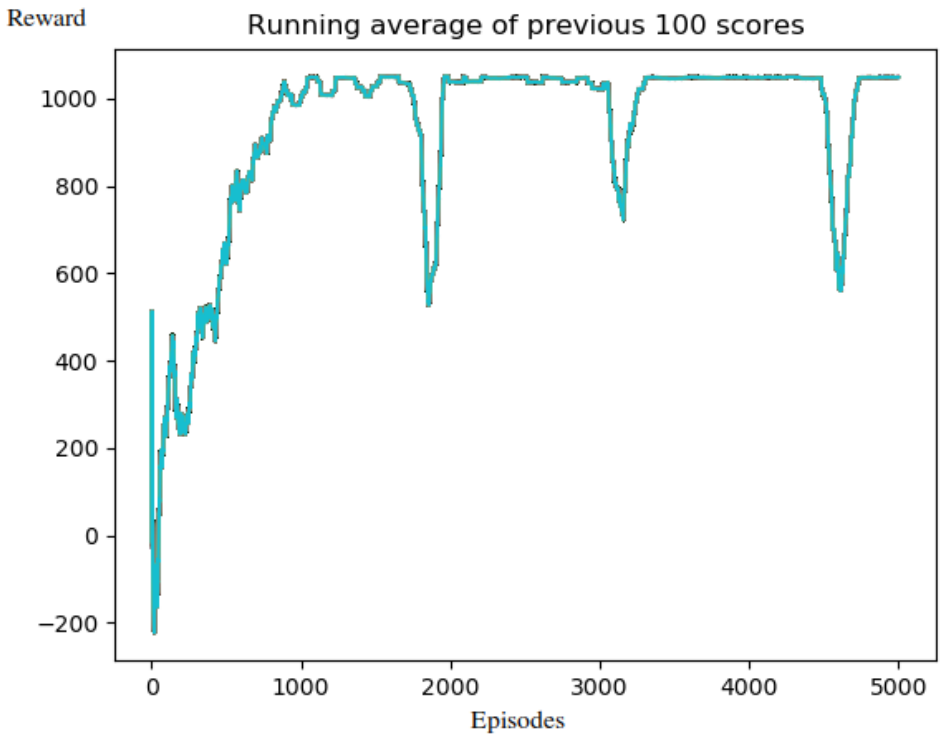


Figure B.5: Move2Bin training with hyper parameters of learning rate 0.0003, batch size 512 and network size 256,256