University of Cologne
Faculty of Arts and Humanities
Department for Digital Humanities

Introducing Machine Learning Using Robots –

Design and Integration of Simple Neural Networks and the Q-learning Algorithm in the Robot Simulation Environment of Open Roberta Lab, Accompanied by the Development, Testing, and Evaluation of Complementary Teaching Materials

by

Viktoriya Olari

Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Arts in Information Processing

Thesis Supervisor: Prof. Dr Øyvind Eide

October 13, 2020

Cologne

Summary

The following master's thesis provides an approach to introducing machine learning to students using the block-based programming language NEPO in combination with educational robotics. The target group of the research study are students from primary to high school, representing beginners without any previous knowledge of machine learning.

After analysing the guidelines and methods for the introduction of machine learning in schools, as well as concrete proposals for artificial intelligence (AI) school curricula with a particular emphasis on machine learning, the author identified a large discrepancy between the requirements for introducing the topics of supervised, unsupervised, and reinforcement learning in schools and the solutions currently available on the educational landscape to do so. Most of the approaches which are currently available either remain a black box or are inaccessible to young students. Only a few approaches focus on making the underlying technical processes of machine learning tangible, which is crucial for enabling students to create the proper mental models and avoid misconceptions.

In order to close this discrepancy, and following the ideas of constructionism, the author developed three approaches to introduce machine learning using robots. (1) The Neural Network Playground allows the user to experiment with simple neural networks. The student can train the neural network by modifying the weights and directly observing the effects on the simulated robot. (2) The Q-learning Playground enables the student to tinker with the Q-learning algorithm by creating unique learning environments for the robot and playing with the parameters of the algorithm. Step by step, the student can debug the algorithm and explore how it is learning from the agent's perspective. (3) An unplugged activity introducing the k-means algorithm makes the unsupervised learning tangible.

The author accompanied all approaches with a curriculum and a series of learning materials. She then conducted and evaluated a user study with 24 children from primary, middle, and high school. The results underline the practical feasibility of the approaches: the children of all age groups perceived the topics as interesting and ranging from very easy to moderately hard to grasp. Thus, the research study proposes a solid concept for the introduction of machine learning to beginners which fundamentally differs from the currently available approaches and enriches the educational landscape. Future research can focus deeper on measuring the understanding of children, the increase in their knowledge or the effectiveness of the approaches and materials developed.

Acknowledgements

I am very grateful for my thesis supervisor, Prof. Dr Øyvind Eide, who supported me through the entirety of this research project with his expertise, regular discussions, and valuable feedback. I thank you for your commitment to my ideas, for sharpening my thinking, and for your constant help and advice. I would also like to thank Dr Jan Wieners, who supported me by sharing his expertise on gamification in the context of artificial intelligence and gave me feedback throughout the genesis of this master's thesis. The presentation of the Q-learning algorithm in your dissertation on SpoookyJS inspired me to develop the Q-learning playground and served as a basis for my implementations.

Special thanks to Kostadin Cvejoski, my colleague at the Fraunhofer Institute for Intelligent Analysis and Information Systems, who helped me assess the feasibility of my research project and constantly supported me in opening the black box of machine learning by sharing his expertise. I thank you for your valuable advice throughout the study and your belief in the success of the project, and for proofreading this thesis. I would like to acknowledge my colleagues at the Roberta-Initiative – Thorsten Leimbach, PhD Reinhard Budde, and Beate Jost – for supporting me with advice on Open Roberta Lab and discussing my methodology and the interim results of the study.

I would like to thank Güncem Campagna and Marc Bertram from the Codingschule junior, who made it possible to conduct the user study during COVID-19. Thank you for your willingness to test new approaches in digital education. I highly appreciate your open mindset and your constant support. I would also like to thank all 24 students who took part in the evaluation. You have motivated me to think about machine learning from a beginner's perspective. I would like to thank my friend, Tobias Hübner, for inspiring me with his approaches to teaching young students in technical subjects and for proofreading the final thesis.

Finally, special thanks go to my family. I would like to thank my husband, Iuri Olari, for his constant support at all times and at all levels. Thank you for sharing with me your critical views on my research, your infinite patience, and your technical advice. I could not have completed this master's thesis without your stimulating discussions and your support. I would also especially like to thank my uncle, Alexander Elfantel, who critically reflected on the graphic design of the machine learning extensions and the learning materials and supported me with advice and feedback from the designer's perspective. I am very grateful for my parents, Viktoriya and Oleg Lebedynski. Thank you for your love, understanding, and lifelong trust in me, which has encouraged me throughout my life to dare the impossible.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial intelligence
API	Application programming interface
AST	Abstract syntax tree
COVID-19	Coronavirus disease 2019
Fraunhofer IAIS	Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
ICT	Information and communication technology
IDE	Integrated development environment
LVQ	Linear vector quantisation
npm	Node package manager
STEM	Science, technology, engineering, and mathematics
UI	User interface
VPL	Visual programming language
ZDI	Zukunft durch Innovation

1 Introduction

Machine learning is becoming ubiquitous, not only in industry, society, and business, but also in the educational context. Scientists around the globe advocate that people of all ages, including children, should be familiarised with its basic concepts (Hitron, Wald, Erel, & Zuckerman, 2018). People interact with intelligent technologies even in childhood and are increasingly influenced by these technologies as they grow older. If they are not educated about smart devices and technologies, they often trust them too much and can be easily manipulated by them (Williams, Park, Oh, & Breazeal, 2019). However, how can a novice such as a child be introduced to machine learning, and how can the underlying principles of machine learning algorithms be made tangible to someone without prior knowledge of linear algebra and statistics?

Using the benefits of educational robotics and inspired by explanations of machine learning in children's books, the following master's thesis aims to examine these questions in a practical case study with the focus placed on school students as novice representatives.[1] The author's motivation stems from her extensive work educating young people in programming and exploring how to make digital technologies understandable to everyone from university students to primary school children.

The theoretical framework builds upon the guidelines and methods for the introduction of artificial intelligence (AI) and machine learning in schools, which date back to 1971 (Papert & Solomon, 1971). Recent advances in machine learning technologies have led to the increasing development of concrete proposals for AI school curricula with a particular emphasis on the technical aspect of machine learning (Clarke, 2019; Long & Magerko, 2020; Sloman, 2009; Touretzky, Gardner-McCune, Martin, & Seehorn, 2019; Wong, Ma, Dillenbourg, & Huan, 2020). These provide the basis for this research study.

There is a large discrepancy between the requirements for instruction in machine learning and the solutions currently available on the educational landscape to do so: While children at all school levels from primary to high school are expected to be able to cope with the central paradigms of machine learning – supervised, unsupervised, and reinforcement learning (Jatzlau, Michaeli, Seegerer, & Romeike, 2019; Kahn, Megasari, Piantari, & Junaeti, 2018; Michaeli, Seegerer, & Romeike, 2020; Williams, Park, Oh, et al., 2019) – most of the currently available learning materials and guidelines focus on

[1] Introducing a technical topic with the focus on a child helps educators consider reducing ideas to understandable and straightforward terms – an approach forwarded by physician Richard Feynman in his treatment of quantum physics (Feynman, Leighton, & Sands, 2011).

ethical content or the social impacts of machine learning (Blakeley & Breazeal, 2019; Kleeberger, Prost, & Sternkopf, 2019; Universität Paderborn, 2019). Only a few approaches focus on making the underlying technical processes of machine learning tangible for students (Lin, Brummelen, Lukin, Williams, & Breazeal, 2020; Williams, Park, & Breazeal, 2019; Williams, Park, Oh, et al., 2019). Most available approaches either remain a black box[2] or are technically so complex that they are largely unsuitable for school students (Jatzlau et al., 2019).

However, in order for students to create the proper mental models and avoid misconceptions, it is crucial that they understand the concrete processes (Hitron et al., 2019; Lin et al., 2020). If the children grasp the processes, they can understand that difficult moral dilemmas can arise through the use of machine learning technologies or, for example, that AI can increase the harmful power of authoritarian regimes as shown by Molnar (2020).

1.1 Research Questions and Methodology Overview

This research study aims to close this discrepancy between requirements and solutions for teaching by examining how the principles underlying machine learning can be made accessible to novices, such as young students.

First, the author analysed school curricula and recommendations regarding machine learning to determine the requirements for instruction in machine learning and identify the status quo. Furthermore, the author investigated existing possibilities for introducing machine learning topics to beginners. Based on the results and the gaps identified, the author developed proposals for how to meet the requirements. The first research question is thus as follows:

What are the specific curricular needs concerning machine learning in schools? What possibilities can be identified to meet these curricular needs? Where are the limits?

Second, the author established a theoretical framework for the proposals and implemented them in Open Roberta Lab, a visual block-based open-source programming platform. The author chose this platform because of its focus on teaching programming with robots to beginners and its advanced ecosystem, which includes a robot simulation. The author then designed the machine learning curriculum and developed the learning

[2] In this master's thesis, the author refers to the "black box" according to Jatzlau et al. (2019) as a metaphor for a process underlying a machine learning algorithm which remains hidden to the student. The reason for this is that the actual computing is performed by external services that do not provide access to the underlying machine learning models and algorithms.

materials. The second research question relates to the implementation and documentation of machine learning extensions and learning materials. It is as follows:

How can the previously defined proposals be anchored pedagogically and concretely implemented in Open Roberta Lab?

Third, to determine how the developed concepts appeal to the focus group, the author evaluated the results in a user study. Specifically, the author tested her developments with students from primary, middle, and high school. In the evaluation, she examined how children of different ages perceived the topics and whether they had difficulty understanding them. For this purpose, she developed a computer-based questionnaire based on a 5-point semantic differential scale and analysed the results and the oral feedback given by the students at the end of each session. Finally, she evaluated the observations of an observer during the sessions. The third research question is thus as follows:

How do the developed concepts appeal to students of different school grades? What help do students need in order to understand the machine learning concepts proposed?

1.2 Thesis Structure

Chapter 2 details the relevance of teaching children about machine learning and summarises previous studies on establishing machine learning in education. To this end, the author conducts a requirements analysis based on school curricula, guidelines and case studies. She discusses related work and learning approaches to meet curricular needs. The chapter concludes by highlighting the limitations and shortcomings of current efforts, which forms the basis for the design and development of new approaches in the course of this study.

Chapter 3 provides the theoretical background to the paradigms of machine learning and develops three approaches to close the gaps identified in Chapter 2 using robots and Open Roberta Lab. It introduces the idea of direct supervision as a representative of supervised learning, the Q-learning algorithm as a representative of reinforcement learning, and the k-means algorithm as a representative of unsupervised learning.

Chapter 4 presents the methodology from two perspectives. First, the pedagogical design principles are outlined. These, together with the evidence from Chapters 2 and 3, constitute a methodological framework. The chapter summarises the framework and presents concrete implications for designing machine learning extensions, curricula, and learning materials. Second, the chapter presents the framework for user study and evaluation as well as potential limitations. Finally, the chapter gives insights into the tools used and the management of the research project.

Chapter 5 outlines the design and technical implementation of machine learning extensions in Open Roberta Lab. The author first reviews Open Roberta Lab, including its initial project structure and system architecture. Second, she presents new block categories and blocks that she has implemented in Open Roberta Lab to enable the user to interact with machine learning extensions. Third, the author describes the actual extensions – the Neural Network Playground and the Q-learning Playground – their system architecture, user interface, and central workflows. The chapter concludes with reflections on technical challenges.

Chapter 6 covers the machine learning curriculum and the learning materials that accompany the extensions introduced in the previous chapter. The author also presents an unplugged activity designed to introduce the k-means algorithm as a third approach to introduce beginners to machine learning.

Chapter 7 reveals the evaluation results for machine learning extensions and materials gained in the user study with 24 children from primary, middle, and high school. The author describes the setup, participants, procedure, results of the questionnaire on children's perception of the machine learning topics covered, and overall feedback from students and the observer.

In Chapter 8, the results of the master's thesis are discussed. First, the overall course of the user study is reflected upon, and then special attention is paid to reflecting on the machine learning extensions and teaching approaches that were developed. In addition, the chapter critically discusses the user experience observed during the user study, the approaches of using simulated robots and visual programming languages, and the design of learning activities concerning the methodological framework outlined in Chapter 4. Second, the chapter summarises limitations and proposes recommendations for future research.

Chapter 9 lists the main findings as answers to three research questions posed in Section 1.1. Finally, it reviews and reflects on the entire research study in terms of the approaches selected and summarises the author's main contributions and their role for future research.

2 Background and Related Work

The introduction of machine learning into the educational context is increasingly becoming the focus of research and numerous case studies. This chapter first outlines the current curriculum needs with regards to machine learning in schools. It then discusses previous attempts to establish machine learning in education from two perspectives: (1) related work and learning approaches to meet curricular needs and (2) the limitations and shortcomings of current efforts.

2.1 Curricular Needs for Teaching Artificial Intelligence and Machine Learning

Researchers around the globe have invested considerable effort into the development of curricula and guidelines for AI education in schools. In their meta-study, Long and Magerko (2020) conclude that the requirements that children must fulfil in the future can be summarised by the term “AI literacy” (p. 2). They define AI literacy as “a set of competencies that enables individuals to critically evaluate AI technologies; communicate and collaborate effectively with AI; and use AI as a tool online, at home, and in the workplace” (p. 2). Although this definition emphasises the importance of social and ethical aspects of AI, concrete proposals for AI curricula devote special attention to the technological aspects of AI, especially to machine learning. Figure 1 illustrates learning as the third “Big Idea” of AI (Touretzky et al., 2019, What are the “Big Ideas” in AI?) which should be taught across all school levels.

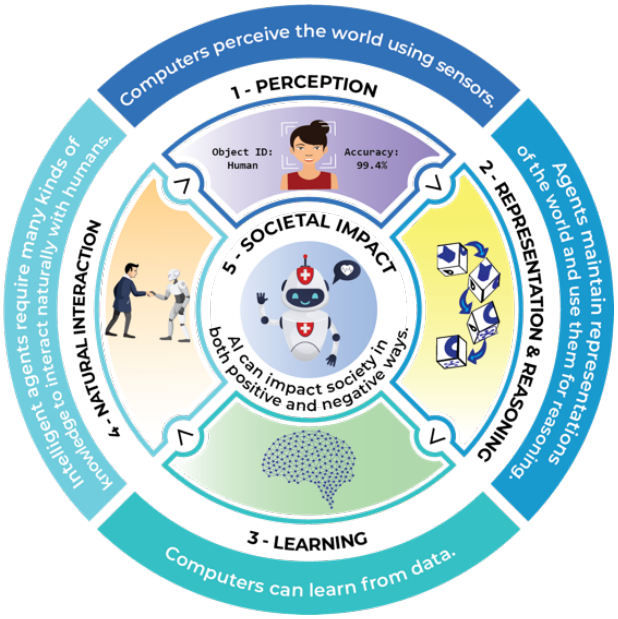


Figure 1: “Big Ideas” of AI (Touretzky, 2019).

The structure and precision of the individual topics vary across curricula. The following are several examples of teaching content and competencies that the children are expected to master.

Even young learners should understand the basic mechanics of AI systems, including such terms as “dataset”, “learning algorithm”, and “prediction” (Blakeley & Breazeal, 2019). Williams, Park, Oh, et al. (2019) suppose that students also need to be able to recognise how the computer learns patterns (supervised machine learning), how it uses previous knowledge for future decisions (knowledge-based systems), and how it can create things (generative music AI). They should explain simple chatbots and program applications enriched with AI mechanisms (Sloman, 2009). They should know what a neural network is and how it is trained (Burgsteiner, Kandlhfer, & Steinbauer, 2016; Sloman, 2009). By the end of primary school, students should be able to examine representations created by intelligent agents, modify simple perception-based applications that include simple AI elements, adapt object recognition applications and should have some experience with machine vision (Touretzky et al., 2019).

Children in middle and early high school must be competent enough to create AI programs using text- and block-based programming languages and even apply acquired AI concepts as potential solutions to real-world problems (Wong et al., 2020). They should comprehend intelligent agents, automata, decision trees, and program applications implementing simple reinforcement learning algorithms (Burgsteiner et al., 2016; Jatzlau et al., 2019). In addition, forwards and backwards propagation (Clarke, 2019) is as much a part of the agenda as basic knowledge in computer linguistics (Touretzky et al., 2019).

Kahn, Lu, Zhang, Winters, and Gao (2020) expect students in high school to understand technical terms and be able to program complex deep neural networks. Michaeli et al. (2020) suppose that children should be able to create programs based on unsupervised learning algorithms such as linear vector quantisation and reinforcement learning algorithms such as Q-learning (Jatzlau et al., 2019).

2.2 Meeting the Requirements: The Introduction of Machine Learning in the Educational Context

The brief extracts from the currently proposed curricula indicate that children in schools are expected to cope with multiple areas of machine learning. The following section presents the work related to helping teachers and students meet such curriculum requirements. Since the curricula involve the three paradigms of machine learning, the chapter is structured according supervised, unsupervised, and reinforcement learning.

2.2.1 Supervised Learning

Supervised learning is a machine learning paradigm which frequently occurs in the proposed curricula and case studies (Jatzlau et al., 2019; Michaeli et al., 2020). The spectrum of approaches developed for the introduction of supervised learning is accordingly extensive.

Touretzky et al. (2019) summarise various easy-to-use applications that are repeatedly considered by teachers. Through descriptive examples, the applications offer beginners and non-programmers the opportunity to experiment with areas of supervised learning such as image or text classification. In Teachable Machine (Google, 2020), for example, children can train a model to classify images, audio files, or body positions. Machine Learning for Kids (Lane, 2020) introduces children to the training of machine learning models to recognise text, numbers, images, or sounds. Recently, code.org, an online coding platform for beginners, launched tutorials to train a neural network to recognise images (code.org, 2020).

Although these examples are excellent resources for introducing machine learning to children, they all have one alarming aspect in common: The training of the model is hidden from the user, which can lead to oversimplified or inaccurate mental models of machine learning (Hitron et al., 2019; Michaeli et al., 2020). Hitron et al. (2019) and Lin et al. (2020) argue that in particular, the understanding of the concrete processes is crucial in order to create the proper mental models and avoid misconceptions.

Attempts are thus being made to open the black box and to look behind the scenes of supervised learning. Hitron et al. (2018) and Hitron et al. (2019) implemented a visual application for supervised learning to classify movements and examined children's understanding of it. They found that even 10-year-olds can understand the basic concepts behind the system.

Kahn and Winters (2017) and Kahn, Lu, Zhang, Winters, et al. (2020) developed extensions for the visual block-based programming language Snap! (University of California at Berkeley, 2020). The children look behind the scenes by programming deep neural networks to discover real-world data relationships. Figure 2 shows one of the examples for the programming of deep neural networks with blocks. The user can configure a model and training parameters using blocks prepared in advance. For instance, they can set the number of layers, choose the loss function and optimiser, and determine the learning rate.

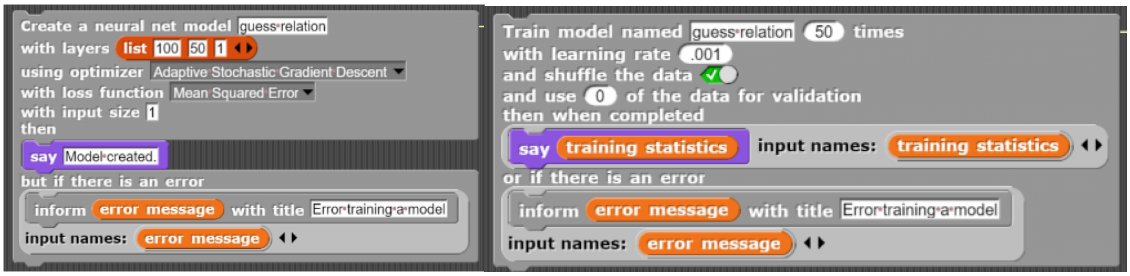


Figure 2: Blocks for creating a model and initiating the training proposed by Kahn, Lu, Zhang, Winters, et al. (2020, pp. 5–6).

Kahn et al. (2018) indicate that it is also possible to use a block-based approach for image recognition and speech synthesis. Based on a study with 40 high school and vocational students in Indonesia, Kahn and Winters (2017) and Kahn et al. (2018) claim that using blocks is a successful way to understand what happens inside the black box of supervised machine learning. Similar results were reported by Queiroz, Sampaio, Lima, and Lima (2020), who combined visual programming and a WiSARD neural network model to enable beginners to learn training and classification.

2.2.2 Unsupervised Learning

In contrast to supervised learning, little research on introducing school students to unsupervised learning has been conducted. Michaeli et al. (2020) presented an approach to introduce children to linear vector quantisation (LVQ), an algorithm that finds clusters in data sets. The target group are high school students. First, learners immersed themselves in unsupervised learning through an “unplugged” activity. Afterwards, they deepened their knowledge through the use of the block-based programming language Snap!. Since the authors have not yet evaluated the approach, there is no evidence of how children would react to it.

Currently, there are no studies that examine the teaching of unsupervised learning to primary or secondary school students. However, there is an approach originally developed for university students that the teacher might consider when introducing unsupervised learning in middle and high school. EduClust (Universität Konstanz, 2020) is a platform that provides a visual categorisation based on the clustering behaviour of the chosen clustering algorithm. The platform covers the most prominent clustering algorithms and can be used by students without time-consuming implementation efforts. Fuchs et al. (2020) considered a case in which this platform had been used for two years in computer science classes and report positive feedback from the students, who were willing to use EduClust in their learning routine.

2.2.3 Reinforcement Learning

Similar to unsupervised learning, the paradigm of reinforcement learning remains almost untouched in the school context (Michaeli et al., 2020). However, attempts have increasingly been made to introduce reinforcement learning into the classroom to meet curriculum requirements.

Kandlhofer, Steinbauer, Hirschmugl-Gaisch, and Huber (2016) carried out a proof-of-concept study focusing on children from kindergarten to university. Although the topics of machine learning appeared only briefly, the findings suggest that the researchers addressed the learning agents with high school children. The preliminary results indicate that the pilot implementations of the proposed concept succeeded, and the students gained a solid understanding of fundamental issues. A detailed evaluation is still pending.

Jatzlau et al. (2019) focused on Q-learning – a reinforcement learning algorithm that is fast and therefore well suited to use in a school setting. Similar to Kahn et al. (2018), they used the block-based programming language Snap! to program the agent and the algorithm itself. When analysing the blocks and structuring the programs, it is striking that the approach of Jatzlau et al. (2019) was able to reproduce the Q-learning algorithm exactly without having to adapt technical terms for the prospective audience. Figure 3 shows the programming of the model on the left side and a fragment for programming the Q-learning algorithm on the right side.

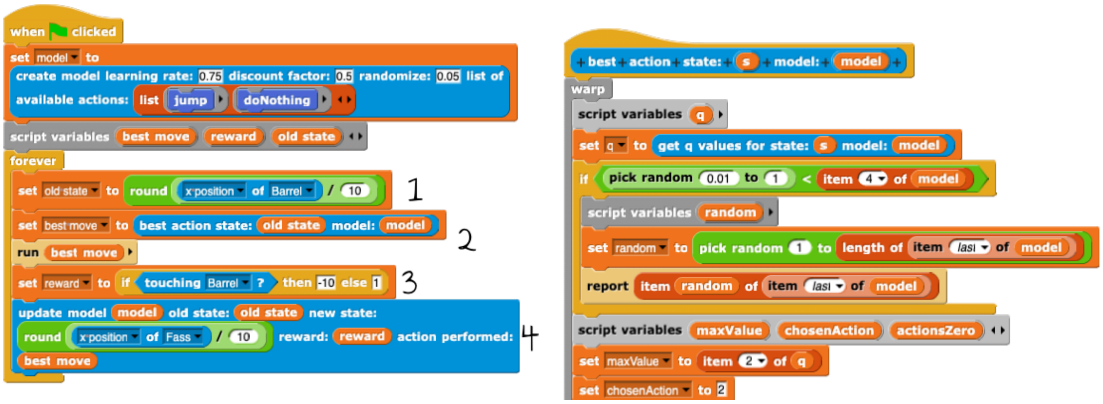


Figure 3: Programming of the Q-learning algorithm, an extract from Jatzlau et al. (2019, Chapter 4. A).

The researchers conducted a subsequent case study with a tenth-grade class, and their findings suggest that the children were able to look behind the scenes and inspect every programming block, thus tracking the program flow. During the learning process, the children observed how the Q-values changed in the Q-learning table. The entire process of reinforcement learning became visible and tangible. However, the researchers point out that the speed of the Q-learning algorithm can become a potential problem if

the teacher decides to use it in the classroom. If the children create complicated learning environments, it will take too long for the agent to learn, which would exceed the time of the classroom lesson.

Toivonen, Jormanainen, and Tukiainen (2017) also used Q-learning to introduce the machine learning paradigms, whereby the researchers combined the Q-learning with neural networks and educational robots. Although the target group was young adults, they found the reinforcement learning questions in the preliminary questionnaire challenging. The results of the post-test show that the participants' knowledge increased, and they achieved a deep practical understanding of reinforcement learning.

2.3 Learning Approaches for Introducing Machine Learning

This section discusses current learning approaches to introduce machine learning in the school context. Since it is common practice to introduce computer science topics in an unplugged and plugged manner, the chapter is divided into unplugged and plugged activities.[3] Finally, related work on the use of robots for teaching AI in schools, which involves both physical and computer-based interaction, is presented.

2.3.1 Unplugged Activities

Unplugged activities are popular in schools (Romero et al., 2019). These do not require any special preparations and can be carried out in almost any classroom. Teachers remain independent of the hardware and can utilise common learning materials including paper, scissors, coloured pens, glue, thread, and handicraft tools.

Educators often use unplugged activities in the context of machine learning a) to explain complex algorithms or mechanisms, for instance, before the students start the programming part, or b) to initiate a reflection or discussion on ethical or social issues. In the former, the teacher designs an intervention in which the learner can put themselves "in the agent's shoes" in order to give meaning to the agent's[4] argumentation process (Long & Magerko, 2020, p. 6). The students can then be involved in embodied simulations of the algorithm or hand-on physical experiments with the technology. In b), students are actively encouraged to take diverse perspectives and then debate them with their classmates.

---

[3] Plugged activities require the use of a computer with specific software, connected objects, or robots to teach the computer science subjects. Teaching topics unplugged is a paradigm which requires neither hardware nor software (Romero, Duflot-Kremer, & Viéville, 2019).

[4] An intelligent agent is a system that is able to act independently and autonomously to pursue and achieve individually relevant goals (Šalamon, 2011; Wieners, 2014). It perceives its environment through sensors and acts upon that environment through actuators (Russell & Norvig, 2016).

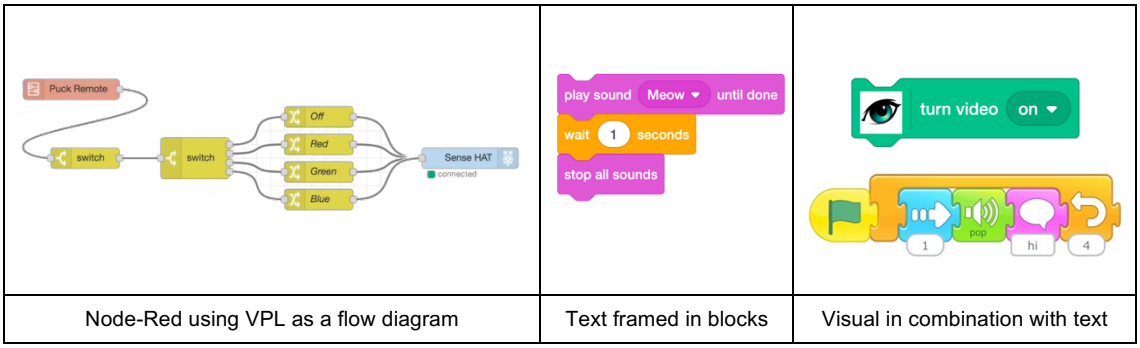
School children of all ages can participate in such activities. As Kandlhofer et al. (2016) note, even kindergarten children remember well how they traversed graphs in the role of the robot or found their way out of labyrinths a few days after the project day.

The number of instructional materials that is available for the unplugged teaching of concepts of machine learning is gradually increasing. Clarke (2019); Kandlhofer et al. (2016); Kleeberger et al. (2019); Lindner and Seegerer (2019); Seegerer, Lindner, and Romeike (2019) and Universität Paderborn (2019) are only some of the authors proposing diverse activities. From backwards propagation and deep learning to image classification, reinforcement learning algorithms, clustering mechanism, Turing tests, and offline games, there are virtually no limits to creativity when it comes to concepts that can be taught unplugged; even face recognition is feasible (Krueger, 2020).

Most unplugged activities are intended for use in a classroom with several students, and they are not suitable for young children to explore machine learning on their own. At this point, children's books provide an attractive way for learners aged 2 years and older to be introduced to machine learning independently (Dhoot, 2019a, 2019b, 2019c; Ferrie & Kaiser, 2019; Liukas, 2019; RocketBabyClub, 2018a, 2018b, 2019a, 2019b, 2019c). Such books, written and designed for young learners, present complex matters playfully, and they can also equally enhance adult learners' experience and contribute to their understanding of course content (Freeman, Feeney, & Moravcik, 2011).

2.3.2 Plugged Activities

The introduction of machine learning with plugged activities is often realised using visual programming languages (VPLs), which allow the user to create programs via graphical manipulation (Druga, 2018). Although some VPLs use flow diagrams for programming, many still use text framed in blocks or a visual in combination with text. Figure 4 shows examples of various VPL types.



Node-Red using VPL as a flow diagram	Text framed in blocks	Visual in combination with text

Figure 4: Examples of visual programming languages.

Visual block-based languages are user-friendly, and due to the following factors, educators increasingly adopt them in the school context (Kahn, Lu, Zhang, Winde, & Gao, 2020; Kahn & Winters, 2017):

(1) The description of the blocks in natural language makes the program easy to read.

(2) Using predefined blocks saves debugging time by eliminating typing and syntax errors.

(3) Browsing the language is easy, because the commands are already pre-sorted by category.

(4) The user assembles the program via drag and drop, which is fun.

(5) The students are motivated because they create applications intuitively.

The following platforms provide features for working on machine learning projects with blocks: Machine Learning for Kids (Lane, 2020), Teachable Machine (Google, 2020), TensorFlow Playground (Smilkov & Carter, 2020), Cognimate (Druga, Qiu, T.VU, Likhith, & Dale, 2020), Calypso for Cozmo (Visionary Machines LLC, 2020), Snap! (University of California at Berkeley, 2020), makeBlock (Makeblock Co., 2020), code.org (code.org, 2020), and eCraft2Learn (University of Oxford, 2020).[5] All these platforms are free, and web based.

Machine Learning for Kids and Teachable Machine introduce newcomers to training machine models to analyse pictures, poses, sounds, text, and numbers without writing any code. No previous knowledge of machine learning is required. Machine Learning for Kids uses APIs to access IBM Watson (IBM, 2020) and Teachable Machine the TensorFlow.js, a library for machine learning in JavaScript (TensorFlow, 2020). Once the model is trained, the user can employ it in educational coding platforms such as Scratch (Scratch, 2020a),[6] App Inventor (Massachusetts Institute of Technology, 2020),[7] or even their own Python integrated development environment (IDE).

A TensorFlow Playground invites users to tinker with a neural network by observing the training process. The user can add or remove additional neurons and layers and change the weights of individual links and biases of nodes. He or she can also

[5] The eCraft2Learn platform differs from other platforms as it only offers learners a single gateway to software, tools, coding platforms, and worksheets.
[6] Scratch is the most popular online coding platforms, which is being developed by MIT Media Lab. Scratch publishes monthly statistics which point out that in the month July 11 713 380 unique visitors from all over the world has programmed in Scratch (Scratch, 2020b).
[7] App Inventor is a platform for building mobile apps quickly using blocks.

experiment with parameters such as learning rate, activation function, regularisation, regularisation rate, and problem type.

The remaining platforms are block-based coding platforms that provide high-end APIs to various AI cloud services. Cognimate implements a collection of dedicated extensions that enable tinkering at home with AI devices and services such as Jibo, Alexa, Muse, Smart Lights and Plugs, Color Tracking, and Image Recognition (Druga, 2018). Code.org hosts tutorials for image classification.

Calypso for Cozmo and makeBlock provide a simple graphical interface on which to program the educational AI robots Cozmo and Codey Rocky. The AI functionalities of these robots are limited to recognising colours and images, showing some emotions on display, and moving and shifting objects.

Snap! is block-based programming for advanced learners. It stands out from the other examples because the user not only has access to potent APIs, but can also work with blocks to program machine learning algorithms without having to rely on complex APIs or cloud services. (Jatzlau et al., 2019; Kahn, Lu, Zhang, Winters, et al., 2020). Sections 2.2.1 and 2.2.3 describe some examples of the use of Snap! in more detail.

Most of the introduced platforms offer tutorials and instructional materials that the teacher can consult as a guide. Touretzky (2020) has compiled a list of publishers and initiatives that have already implemented the platforms in school education and provide lessons plans.

2.3.3 Using Robots and Robotic Simulators

Extensive empirical evidence indicates that children are excited when working with robots (Barker, Nugent, Grandgenett, & Adamchuk, 2012). Sklar, Eguchi, and Johnson (2002) and Sjödén, Lind, and Silvervarg (2017) suggest that the use of robots as teachable agents – robots that the students can teach – has strong motivational effects, because the robot is treated as a social character. These findings apply to children of all age groups, from kindergarten to university (Druga, 2018; Druga, Williams, Park, & Breazeal, 2018; Klassner, 2002; Klassner & Anderson, 2003; Parsons & Sklar, 2004; Williams, Park, Oh, et al., 2019).

Cooper, Keating, Harwin, and Dautenhahn (1999) and Li, Chang, and Chen (2009) mention Seymour Papert as the first person to teach AI concepts to school children and apply robots in education. Figure 5 presents the first educational robot, the Turtle, constructed by Papert and his team at MIT and evaluated with children over succeeding decades.

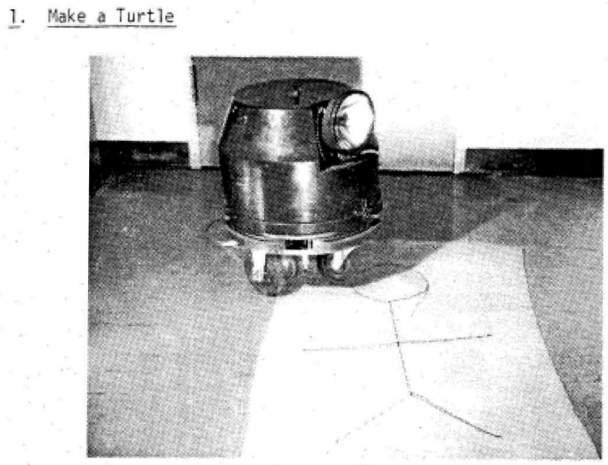


Figure 5: The first educational robot, constructed by Seymour Papert and his team at MIT (Papert & Solomon, 1971, p. 3).

The Turtle can obey simple commands from a computer and send signals back to the computer. It is extendable by “any sense organs one is clever enough to make” (Papert & Solomon, 1971, p. 3) – for instance, touch sensors, light-sensitive cells, and sound detectors. Papert (1993b) commented on the experience that he gained with the Turtle for educating in science and technology subjects: “The idea is that early experience with Turtles is a good way to ‘get to know’ what it is like to learn a formal subject by ‘getting to know’ its powerful ideas” (p. 138). The children thus learn by gaining experience. While playing with the Turtle, students learn thought interaction, creating and understanding an artefact (Michaeli et al., 2020; Wang, 2016).

While studies since the 1990s have confirmed the positive effects of the introduction of robots in education (Chin, Hong, & Chen, 2014; Sklar et al., 2002),[8] there is insufficient evidence regarding the use of robots to introduce AI and machine learning. Kumar (2004) conducted a three-year study with junior/senior level students who attended a course in AI, in which he used LEGO Mindstorms robots to teach blind searches, informed searches, expert systems, and game playing. The results suggest that students believe that robot projects were reasonable and helped them learn the underlying AI concepts. However, they rated such projects as much more time consuming than traditional projects. Klassner (2002) and Klassner and Anderson (2003) reported similar experiences with LEGO Mindstorms in the context of AI courses to teach the topic of intelligent systems.

[8] Discussing robots in education can have different meanings. Li et al. (2009) differentiate among using robots as learning materials, learning companions, and teaching assistants. In this research, robots are utilized as learning materials and as something that the children can teach (see Section 2.3.3).

Recently, Druga (2018); Druga et al. (2018) and Druga, T.Vu, Likhith, and Qiu (2019) conducted studies with children between 7 and 14 years of age on teaching AI with robots. The findings suggest that after interacting with smart agents through programming and teaching, the children changed their perception of smart toys and developed a better understanding of AI concepts. Williams, Park, Oh, et al. (2019) designed LEGO-built and virtual robots to teach kindergarten students basic AI concepts: knowledge-based systems, supervised machine learning, and generative music AI. The results indicate that even kindergarten and pre-school children can successfully gain an understanding of AI algorithms with the help of robots (Williams, Park, & Breazeal, 2019; Williams, Park, Oh, et al., 2019).

There is also evidence that robot simulators are as beneficial for the learning process as operating the real robot. Papert (1993b) asserted that “the Turtle in all its forms (floor Turtles, screen Turtles, and Dynaturtles) … is both an engaging anthropomorphizable object and a powerful mathematical idea” (p. 137). The simulators even have a decisive advantage: The time required for code-test-debug loops is considerably less than it is when working with real robot (Dodds, Greenwald, Howard, Tejada, & Weinberg, 2006), meaning that the simulation saves time during testing. However, simulations can also have disadvantages. Dodds et al. (2006) indicate that the use of simulation can lead to (1) the loss of the physical embodiment that attracts many students to learn AI with robots and (2) the loss of the unpredictability of physical interaction in the real world. They therefore propose using a simulator in combination with a robot hardware setup identical to that of the simulated robot.

2.4 Analysis and Summary of Shortcomings

Against the background of the curriculum and findings on the introduction of machine learning topics in the school context using robots, this section analyses the limitations and summarises the shortcomings of current efforts and learning approaches.

(1) There is a wide range of activities for supervised learning, but most of them follow the black-box approach.

The findings in Sections 2.2.1 and 2.3.2 indicate that there is a wealth of easy-to-use services that introduce beginners to supervised machine learning. Usually, these make use of a limited number of descriptive examples, such as image or sound classification. The main disadvantage of such applications is that the mechanisms underlying the training and classification remain hidden from the user. Increasingly, there are efforts to open the black box of supervised learning using VPLs. However, even then, they often only provide an interface to powerful high-end APIs. The

children play only with high-end robot systems and ready-trained models and have no opportunity to learn how the training is performed and what algorithms are working behind the scenes and how. When the learner uses the trained model, he or she does not discover why the model came to a given decision.

(2) Translating the underlying concepts into a block-based language opens the black box, but only for those who can understand this complexity.

Sections 2.2.1, 2.2.3, and 2.3.2 presented several examples of attempts to break through the black box of currently available machine learning applications. The idea of these approaches is to transfer the underlying concepts of machine learning, such as artificial neural networks and Q-learning, into the VPL Snap!. Although the authors argued that these approaches are child friendly (Kahn & Winters, 2017), intuitive, and straightforward (Jatzlau et al., 2019), such a representation of the algorithm is not suitable for young children due to its complexity and numerous technical details. Using a block-based programming language, therefore, does not necessarily mean reducing the complexity of the topic or making the topic more accessible.

(3) Introductory activities revolving around reinforcement and unsupervised learning are rare.

Section 2.2.2 presented a case study for the introduction of unsupervised machine learning with the LVQ clustering algorithm. However, the block-based approach used in this study is complex, focuses on high school students, and there are no studies on the introduction of unsupervised machine learning with the target group of young learners. Section 2.2.3 presented several case studies which focus on reinforcement learning. They all refer to the Q-learning algorithm as one that is well suited to an introduction to reinforcement learning. Again, the studies focus on undergraduate and high school students and do not apply to younger children. The authors discuss one of the main problems of introducing Q-learning in schools: the time the algorithm takes to calculate complex problems. Therefore, it appears challenging to teach complex problems in the classroom using Q-learning.

(4) Thematically, the approaches to introducing machine learning are sparse and do not reflect the complexity and breadth of the field.

In summary, the topics presented in Sections 2.2 and 2.3 that are suitable for young learners can be reduced to the following areas of machine learning: image, text, and sound classification; speech synthesis; the programming of intelligent toys; ethics and the social impact of machine learning on everyday life; offline activities for learning about AI; and some aspects of human–machine interaction. The concepts

underlying machine learning, such as deep neural networks or concrete algorithms (e.g., Q-learning and LVQ), are not sufficiently covered. These can be omitted, as the case studies in which they occur are oriented more towards high school than towards primary and middle school children.

(5) The use of blocks is promising.

Nonetheless, Section 2.3.2 indicates that the use of blocks provides the user with easy access and low entry barriers to programming. It motivates learners to create applications intuitively. However, using the advantages of visually based programming languages on the one hand and conveying the complexity of the topic on the other is a challenge. Long and Magerko (2020) also point out that it is crucial for designers to bear in mind that coding skills can be an entry barrier, especially for children who are still learning to read.

(6) The teaching materials focus on older children.

Although the number of instructional materials for young learners is growing, the vast majority of case studies mentioned in Sections 2.2 and 2.3 focus on high school or undergraduate students. However, Section 2.3.1 found that unplugged activities are often used in the context of teaching children complex or abstract topics and are successfully applied to children of all ages.

(7) Teaching materials are often not suitable for children to learn alone.

The teaching materials for introducing machine learning are suitable for use in schools, and are less so for self-directed learning. In addition, the design of the materials and applications is sometimes unintuitive. Therefore, the materials are not suitable for children without prior knowledge. In Section 2.3.1, it was suggested that children's books offer opportunities for young learners to reflect and experiment unplugged on machine learning topics.

(8) The use of real or simulated educational robots is useful for teaching AI topics.

Section 2.3.3 illustrates that there have been very few attempts to implement robots in the teaching of AI and machine learning, and even less research has been carried out with robots for teaching machine learning to young children. The results so far show that the use of robots for educational purposes is effective. The success has been demonstrated in a small number of case studies with kindergarten students and multiple studies with high school and university students.

3 Machine Learning Paradigms

Building on the requirements outlined for teaching machine learning paradigms in schools (Section 2.1) and the summary of the deficits (Section 2.4), this chapter answers the second part of the first research question and partly addresses the second research question: What possibilities can be identified to meet the curricular needs? How can the previously defined proposals be anchored in Open Roberta Lab?

This chapter presents the theoretical backgrounds of three paradigms of machine learning, which together form the essential framework for closing the gaps identified in Section 2.4. The paradigms are supervised learning, reinforcement, and unsupervised learning. Although Russell and Norvig (2016) suggest that the distinction among the paradigms is not always clear and Mohri, Rostamizadeh, and Talwalkar (2018) present other learning scenarios, this chapter adheres to this distinction, as it is widely accepted (Ertel & Black, 2018).

The chapter applies the theoretical framework and successively extends it to three concrete approaches to address gaps and curriculum needs using robots and Open Roberta Lab: direct supervised learning, Q-learning, and k-means algorithms. The practical implementation of the approaches is described in Chapters 5 and 6.

3.1 Supervised Learning and Neural Networks

This section provides a brief overview of supervised machine learning and supervised neural networks, before discussing how these concepts can be realised in Open Roberta Lab.

3.1.1 Overview

Supervised learning is a machine learning paradigm in which the agent learns from a labelled set of examples and can then generalise to unseen points in the future (Mohri et al., 2018). The engineer thus provides the data and defines the loss function on which the model is trained (Nguyen & Zeigermann, 2018).

Formally, the goal of supervised learning is to learn the mapping function f, which entails understanding how the input x should be matched with output y using available data. Russell and Norvig (2016) formalise the task of supervised learning as follows:

> Given a training set of N example input–output pairs
> $(x1, y1), (x2, y2), \dots (xN, yN)$,
> where each y_j was generated by an unknown function $y = f(x)$,
> learn a function h (hypothesis function) that approximates the true function f,

where x and y do not need to be the numbers but can be any values.

When the output y is a number, then the learning problem is called regression. If the output y is one of the values in a finite set, then the learning problem is called classification. Learning involves a search through the space of possible hypotheses for a hypothesis that performs well with new examples beyond the training set.

Supervised learning does not imply the use of neural networks, for instance, in supervised tasks in which simple linear regression is possible. However, neural networks and supervised training are used for applications with supervised problems involving a large amount of input data or building models for classification (Brabazon, O'Neill, & McGarraghy, 2015).

A neural network consists of layers of simple processing units called nodes or neurons. There are three types of layers: input, output, and hidden layers (Khishe & Parvizi, 2020). The nodes are connected by weights or signals (edges; Brabazon et al. (2015)). There are several types of weight gain. For instance, feedforward weight gain means that the signals move in one direction – from input to output. The output of each layer does not affect that layer (Khishe & Parvizi, 2020).

In supervised training, neural networks receive data in the form of inputs and estimated outputs. Outputs or targets are a specification of how the neural network should respond to the input (MacKay, 2003). As the network is trained, the weights are changed until the difference between the network output and the desired output is acceptable (Khishe & Parvizi, 2020).

Exemplary problems for supervised learning include pattern detection; text, speech, and object recognition; recommendation systems; and machine translations (Khishe & Parvizi, 2020; Nguyen & Zeigermann, 2018).

3.1.2 Supervised Learning in Open Roberta Lab: Direct Supervision and Simple Neural Networks

A barely explored but promising possibility for introducing supervised learning and neural networks for young students is to limit the neural network to its essentials. By using the sensors and actuators of a robot as input and output nodes, one can demonstrate the components of a neural network and their functionality.

This idea is derived from Braitenberg (1986) and briefly discussed by Leimbach and Breuer (2012) in the context of introduction to the basics of AI using components of simple robotic systems such as various sensors and simple actuators. Braitenberg (1986) advocates that to simulate intelligent behaviour, it is sufficient to connect a simple sensor directly to a motor. The resulting behaviour is the result of the connection: The sensor

values are the input values for the motors. Different types of connections between the sensors and motors lead to different behaviours – fear, love, anger, or rage.

Figure 6 demonstrates a simple connection between a motor and a light sensor. The higher the value of the light sensor, the faster the motor rotates. The behaviour of the vehicle allows the observer to suspect that the vehicle seems to like the light, because it continually moves towards the light source.

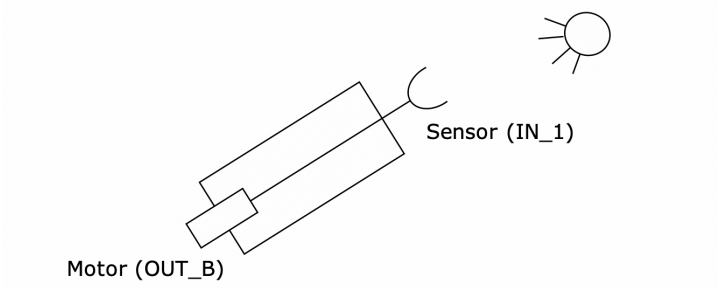


Figure 6: The value of the light sensor serves as the input and is transmitted directly to the motor (Leimbach and Breuer (2012, p. 14).

Such straightforward examples illustrate well the idea of an input–output connection, which Braitenberg (1986) expands by introducing the concept of a threshold device. Although Braitenberg never called the threshold device a neuron, a threshold device resembles a neuron in its functionality – it models a connection between input and output and has a threshold value. If the sum of all values applied to the inputs exceeds this threshold value, a value of 1 is output; otherwise, the value 0 is output.

In Open Roberta Lab, Braitenberg experiments can be adapted and extended. The sensors of the simulated LEGO EV3 robot,[9] such as light, ultrasonic, or colour sensors, can be used as input nodes, and LED, motor, display, and sound can be used as output nodes. A similar approach was followed by Toivonen et al. (2017), who used sensor values as inputs for a machine learning algorithm, while the behaviour of the robot represented the output.

When compiling the program, the neural network can be created by directly linking the input and output nodes to each other. The user should then be allowed to regulate the strength of the connection – that is, the weight. By modifying the weights and observing the results directly from the robot's behaviour, the students can experience a training

[9] Although there are multiple other robots which work well in the simulation, LEGO EV3 is the only driving robot in Open Roberta Lab that can be used for experiments after Braitenberg at the time of writhing this master's thesis.

process of the neural network via direct supervision.[10] Figure 7 shows a prototype of a simple neural network on the left side and the behaviour of a robot on the right side.

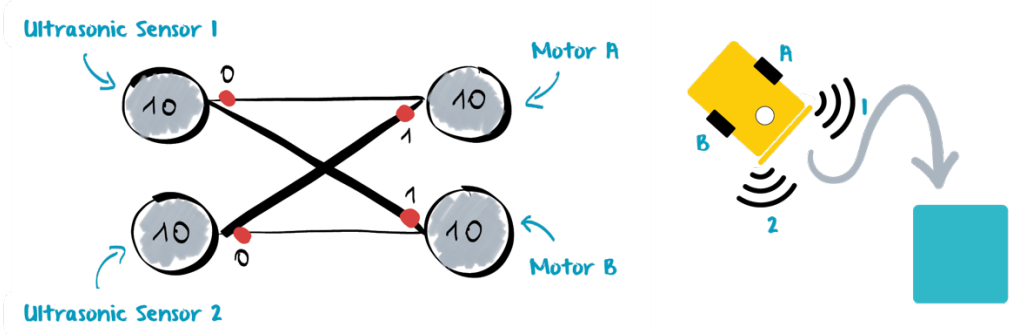


Figure 7: Mock-up of a trained simple neural network and the behaviour of the robot (author's representation).

The adaption of Braitenberg experiments and their extension with additional input and output sensors should make it possible to explain supervised learning problems even to young students. While the students receive immediate feedback from the network configuration, they should understand how the robot learns. At the same time, the students should be able to discover practical components of neural networks such as nodes, layers, links, and weights. Immersion in the training process should enable the children to focus on the underlying processes of supervised learning and thus help open the black box discussed in Sections 2.3.2 and 2.4. The findings of Toivonen et al. (2017) suggest that students without previous knowledge can understand the basics of neural networks through such an approach and can transfer the theory-based knowledge about neural networks into a more practical form.

3.2 Reinforcement Learning

This section defines a brief theoretical framework for reinforcement learning and then discusses the possibilities of introducing reinforcement learning to young students with Open Roberta.

3.2.1 Overview

In reinforcement learning, the agent learns by feedback or the reinforcement it receives after each step or a sequence of steps. Reinforcement can be both positive and negative (Russell & Norvig, 2016). The agent is a learner and decision-maker facing an unknown Markov decision process.[11] It has an explicit goal, can perceive the environment and can

[10] The process of adjusting the weights until the robot behaves as desired is in this work called "direct supervision", in which the students are involved in the training process of the neural network and imitate it by adjusting the weights manually.
[11] For the Markov decision process, see, e.g., Feinberg and Shwartz (2012); Russell and Norvig (2016).

choose actions to explore the environment or to exploit it with the help of available knowledge. The agent can also influence the environment (Sutton & Barto, 2018).

Reinforcement learning problems involve learning how to map situations to actions so that the agent can maximise a numerical reward for a situation or sequence of situations. The main elements of reinforcement learning are, besides the agent and the environment, a policy, a reward signal, a value function, and an optional model of the environment. A policy defines the agent's behaviour at a certain point in time (Sutton & Barto, 2018). A reward signal – a number – is sent to the agent after each time step. The objective of the agent is to use observed rewards to learn and develop an optimal policy for the environment, one that maximises the expected reward (Kober & Peters, 2014). While a reward is immediate and has an effect for a single step, a value function specifies the long-term desirability of environment states (Sutton & Barto, 2018). Figure 8 shows an interaction process between an agent and the environment in reinforcement learning.

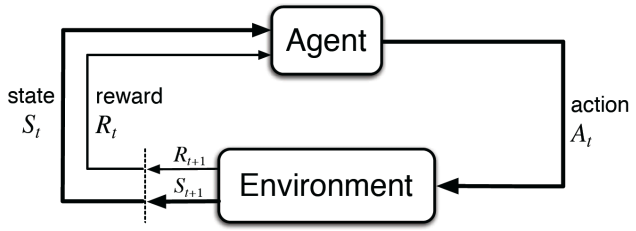


Figure 8: Interaction between an agent and the environment (after Sutton and Barto (2018, p. 54).

At each time step t, an agent receives a representation of the environment $state$, where $S_t \in S$ and performs an $action$, where $A_t \in A(S_t)$. S is a set of possible states, and $A(S_t)$ is a set of actions available in state S_t. Thereafter, the agent receives a $reward$, where $R_t \in \mathcal{R} \subset \mathbb{R}$, and the learning process starts from the beginning, with S_{t+1}.[12]

In practice, reinforcement learning is used in environments in which the agent is supposed to learn how to behave successfully and in which it is almost impossible to provide all the rules. For instance, in a game context, the agent learns while playing. If it loses the game, it receives penalty points; if it wins, it receives a reward (Russell & Norvig, 2016).

3.2.2 Reinforcement Learning in Open Roberta: The Q-learning Algorithm

Section 2.2.3 indicated that Q-learning is a reinforcement learning algorithm that can be successfully applied in the educational context because of its high speed and small problem spaces (Jatzlau et al., 2019). Since its introduction by Watkins (1989), the algorithm

[12] The algorithm is reproduced after Sutton and Barto (2018).

has been intensively investigated and has become popular (Wieners, 2014). Based on these insights, the Q-learning algorithm is introduced and adapted in this work in Open Roberta.

Q-learning is a model-free algorithm (Watkins & Dayan, 1992), which means that the agent does not attempt to create a model of how the world works (Millington & Funge, 2018). Compared to model-based algorithms, the effort needed to implement a model-free algorithm is moderate.

The agent learns in several steps. According to the policy available $\pi(s)$, the agent starts in a new state s and attempts an action a in order to progress to the next state. Depending on whether the agent is exploring or exploiting, the next state s' is either taken randomly or based on the highest Q-value from the following step. The agent then evaluates the consequences in terms of the reward it receives, and in terms of the value q of the state taken. By trying out all the actions in all the states, the agent learns which steps are best overall, as judged by the long-term discounted reward (Watkins & Dayan, 1992), and updates the Q-table, which holds the Q-values. The rule for Q-learning is as follows:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma\, max_{a' \in A(s')}\big(Q(s',a')\big)),$$

where $Q(s,a)$ is a current Q-value for action a from state s, and $max_{a' \in A(s')}\big(Q(s',a')\big)$ refers to the maximum reward from the next state s' as a result of action a that the agent expects based on its knowledge of its current environment. r is a reward for action a from state s. α is a learning rate, and γ is a discount rate in the interval $0 \leq \gamma \leq 1$. α determines how much older Q-values are included in the update process, and γ reduces the impact of the subsequent environmental reward compared to the previous reward (Wieners, 2014). α and γ are set only once and remain constant throughout the learning process.

In Open Roberta Lab, the Q-learning algorithm can be implemented for a simulated LEGO EV3 robot. The robot can learn to find an optimal path from a start to a finish state in a given environment. By using blocks, the children can be given the opportunity to independently program applications based on the Q-learning algorithm. In contrast to the solution proposed by Jatzlau et al. (2019), the terminology and parameters of the algorithm should be simplified to be accessible to young students. To open the black box and understand how exactly the learning takes place, the learning process can be visualised sequentially. This requires a learning environment that the children can understand, in which the robot can represent the agent that is learning. After the learning process is

completed, the agent calculates the optimal path based on the best Q-values from the start to the finish state and then follows the path in the environment.

Algorithm 1 shows a simplified Q-learning algorithm (adapted from Wieners (2014); Xu, Wu, and Zhao (2015) that can be implemented in Open Roberta Lab.

Algorithm 1: Q-learning

1: Q table of values for states → actions.
2: s, a, r the previous state, action, and reward, initially null.
3: t, e, q time, episode, q-value.
4: trade-off exploration-exploitation by possibility of RHO.
5: trade-off continue-change state by possibility of NU.
6: α, γ learning rate, discount rate from 0 to 1.
7: **while** $t > 0$ and $e > 0$ **do**
8: randomly choose a float number rho from 0 to 1.
9: randomly choose a float number nu from 0 to 1.
10: **if** $nu < NU$ **then**
11: $s \leftarrow$ choose random state.
12: **end if**
13: **if** $rho < RHO$ **then**
14: $\mathrm{r}, \mathrm{s}' \leftarrow$ explore another action a.
15: **else**
16: $\mathrm{r}, \mathrm{s}' \leftarrow$ exploit current optimum action a.
17: **end if**
18: $\mathrm{q} \leftarrow$ get $Q(s, a)$
19: $a' \leftarrow$ get best action for s'
20: $maxQ \leftarrow$ get $Q(s', a')$
21: $\mathrm{q} \leftarrow (1 - \alpha) \times q + \alpha \times (r + \gamma \times maxQ)$
22: $Q(s, a) \leftarrow q$
23: $s \leftarrow s'$
24: decrease t by time elapsed since the beginning of this iteration.
25: decrease total number of episodes e by one.
26: **end while**

3.3 Unsupervised Learning

This section provides a brief introduction to unsupervised learning and examines the k-means clustering as an algorithm that can be introduced in Open Roberta.

3.3.1 Overview

In unsupervised learning, the agent learns patterns in the input data, even though no explicit feedback is provided (Russell & Norvig, 2016). In contrast to supervised learning,

in unsupervised learning, the agent receives inputs $x_{1,}x_2 \ldots, x_n$ – that is, unlabelled training data only (Mohri et al., 2018). The task of the machine is then to find structures in collections of data or groups and to categorise the data (Michaeli et al., 2020).

A classic example of an unsupervised learning problem is clustering (Ghahramani, 2004), in which a set of data points must be partitioned into similar homogenous subsets (Aggarwal & Reddy, 2013; Mohri et al., 2018).

Unsupervised learning problems can be approached using a variety of methods. Ghahramani (2004) points out that almost all work in unsupervised learning can be considered in terms of learning a probabilistic model of the data. The machine estimates a model that represents the probability distribution for a new input x_n given previous inputs. The learner model is then $P(x_n|x_1 \ldots, x_{n-1})$. If the order of inputs is irrelevant or unknown, the machine can build a model of data with $x_{1,}x_2 \ldots, x_n$ as independent points. The central idea in probabilistic models is to model data from a generative process. Generative models are fundamental because they try to understand the underlying process through which a cluster is generated (Aggarwal & Reddy, 2013).

The problems of unsupervised learning include classification, outlier detection, collaborative filtering and recommendation systems, dynamic trend detection, monitoring, social network analysis, communication, and efficient data compression.

3.3.2 Unsupervised Learning in Open Roberta: K-means Algorithm

This section presents the k-means algorithm as a possibility for introducing unsupervised learning in Open Roberta. This algorithm is chosen because it is one of the most popular clustering algorithms that is easy to understand and to implement (Hamerly & Elkan, 2002).

K-means uses an iterative refinement technique: The algorithm operates alternately with assignment and update steps, whereby an initial set of k centres $m_1, \ldots, m_k$ (Mirkes, 2011) is given. First, the representative points are selected as the initial cluster centres. Each point in the data set is then assigned to the nearest cluster centre based on a selected proximity measure.[13] Once all points are assigned to centroids, so that the clusters are formed, the centroids are updated. The last two steps of the algorithm are repeated until the centroids do not change.

[13] There is a wide range of proximity measures for computing the closest point centroids that can be used with the k-means clustering algorithm, including the Manhattan distance, the Euclidian distance, and Cosine similarity. The Euclidian distance is the most popular choice. For more information about proximity measures and objective functions, see Aggarwal and Reddy (2013).

Algorithm 2 provides an outline of the basic steps of the k-means algorithm according to Aggarwal and Reddy (2013), which can be anchored in Open Roberta.

Algorithm 2: k-means clustering

1: Select K points $m_1, \dots, m_k$ as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** convergence criterion is met.

An implementation of the k-means algorithm in the robot simulation environment of Open Roberta Lab is barely possible, since there are scarcely any use cases within the given ecosystem. However, the k-means algorithm can easily be adopted in Open Roberta as an unplugged activity. For this purpose, teaching materials should be developed to help the children understand how to group a set of items according to the k-means algorithm described in Algorithm 2.

4 Methodology

This chapter completes the answer to the second research question from a methodological perspective by suggesting how the proposals should be anchored pedagogically and concretely implemented in Open Roberta Lab. First, the design principles that underly the machine learning extensions and learning materials developed in the practical part of this work are presented. Second, the user study design and the methods used to evaluate the extensions and materials that are developed are explained. Third, the project management framework and tools used in this research are outlined.

4.1 Design Principles

Educational research has shown that hands-on experience positively impacts learning (Kandlhofer et al., 2016). Hence, to encourage children to tinker with machine learning from a technical perspective, the design of the extensions and learning materials is pedagogically oriented towards constructivist and constructionist theories and connectivist framework. The following provides a brief overview of the fundamental principles of these frameworks.

4.1.1 Constructivism, Constructionism and Connectivism

Multiple research studies about teaching with robots and VPLs are grounded in constructivism and constructionism (Cooper et al., 1999; Kandlhofer et al., 2016; Lister, 2011; Moro, Arlegui, Pina, & Frangou, 2007; Wang, 2016). The teaching of AI and machine learning is also often geared towards constructionist and constructivist principles (Druga, 2018; Druga et al., 2018; Hitron et al., 2019; Jatzlau et al., 2019; Michaeli et al., 2020; Queiroz et al., 2020; Williams, Park, Oh, et al., 2019).

Constructivism is a theory of learning developed by Jean Piaget. At its core, it sees children as active builders of their knowledge (Piaget, Fatke, & Kober, 2016): Instead of receiving information passively, children learn about the world by actively interacting with it. Resnick and Robinson (2017) summarised the critical principle of constructivism as “Children don’t get ideas, they make ideas” (p. 37).

Papert (1993a, 1993b) and Papert and Solomon (1971) expanded Piaget’s cognitive theory and developed a constructionist approach based on the theory of constructivism. The main idea of this theory is that children construct their knowledge most effectively when they are actively involved constructing things in the world (Papert & Harel, 1991). As children construct things in the world, they construct new ideas in their heads, which motivates them to construct new things in the world (Resnick & Robinson, 2017). When learners participate in constructing an artefact and interacting with it, the construction of

knowledge is more effective (Michaeli et al., 2020; Queiroz et al., 2020). Engagement with the design of artefacts and modelling enhances the learning of complex systems through systematic exploration (Hitron et al., 2019).

Siemens (2005, 2014) outlined the limits of constructivism, describing it as a learning theory that focuses only on the individual, and suggested a new learning theory of connectivism to describe how learning happens in the digital age (Siemens, 2005, 2014). Although there is criticism and debate as to whether connectivism is a learning theory or merely a pedagogical view (van Pløn Verhagen, 2006), the connectivist idea of knowledge and learning is enriching in the context of this research. Connectivism postulates that knowledge must be accurate and up to date, emphasising the role of the currency of knowledge in today's world. It also stresses that one of the core skills of the modern learner is the ability to recognise connections among domains, ideas, and concepts, whereby learning and knowing are constant, ongoing processes (Siemens, 2005).

4.1.2 Four P's of Creative Learning

The Four P's of Creative Learning (Resnick & Robinson, 2017) is a modern framework that engages young students in creative learning experiences. The framework is based on constructivist ideas and focuses on the intersection of emerging technologies, activities and strategies (Sakulkueakulsuk et al., 2018). The core values of the framework and its guiding principles are the four P's: projects, passion, peers, and play.

Learning through making and working on projects that matter is a key idea of the first P – *projects*. Here, children create things and tinker with what they are interested in. By continuously going through the Creative Learning Spiral of imagining, creating, playing, sharing, and reflecting, they develop their ability to think, organise, refine, and reflect on their ideas (Fuste, 2018; Resnick & Robinson, 2017). As children work on projects that grab their attention, they build on their interests and are willing to work harder and longer. This is the central idea of the second P – *passion*.

Resnick and Robinson (2017) also took up the idea of working with *peers* – the third P – as an effective didactic approach. Peer learning is an evidence-based instructional method that is theoretically well founded. The effectiveness of working with peers in the form of peer tutoring, peer mentoring, peer mediation, peer counselling, and other forms have been proven in several research studies (Büttner, Warwas, & Adl-Amini, 2012; Hattie, 2008; Lebedynska, 2017; Zeneli & Tymms, 2015).

The concept of *play*, the fourth P, is centred on creativity. Resnick and Robinson (2017) emphasised that playing is not just about laughing or having fun, but is much more about "being a mischief maker" (p. 128) – about experimenting, taking risks, and

testing boundaries. Resnick and Robinson (2017) also made a crucial distinction between two types of playful environment: playpen and playground. The first is a restrictive environment in which the children have limited space to experiment, whereas the second is designed to allow them to move, explore, and collaborate. If the aim of play is to educate creative thinkers, the instructional playpen environments should remain a stepping-stone, not a final destination. Instead, the learning environment should be more oriented towards a playground style, with metaphorically low floors and wide walls, so that the children can make decisions about what to make and how to make it (Resnick, Martin, Sargent, & Silverman, 1996; Resnick & Silverman, 2005).

4.1.3 Derivation of Guidelines for Machine Learning Extensions and Curriculum Design

All three approaches described in Section 4.1.1 can be adopted to develop creative learning experiences with machine learning for young students in the practical part of this work. The extensions and learning materials should promote the idea of an active learner who gains knowledge by making and constructing things. The extensions and materials should also encourage learners to investigate the currency of machine learning paradigms and help establish connections between machine learning ideas and concepts and other fields such as robotics.

Teaching materials and extensions should be developed based on the Four P's of Creative Learning outlined in Section 4.1.2 as a practical guide for designing curricula oriented towards constructionist ideas. The approach developed in the practical part of this work should encourage creativity, while at the same time supporting students in working on projects based on their passions, in collaboration with peers and in a playground-style atmosphere.

The ideas of play and playful learning described in Section 4.1.2 should be incorporated in the extensions wherever possible. Some structured activities for learners should be designed to help students to get started with exploring of machine learning extensions. However, the aim should be for these structured activities to serve as a stepping-stone, not a final destination. The students should be empowered to play with machine learning technologies and create something that interests them, following the ideas of constructivism and constructionism presented in Section 4.1.1.

Two extensions for Open Roberta Lab are planned based on the evidence and shortcomings summarised in Section 2.4, the technical considerations described in Sections 3.1.2 and 3.2.2, and the educational reflections presented in Sections 4.1.1 and 4.1.2: the Neural Network Playground and the Q-learning Playground. The playgrounds should

build upon the needs described in Section 2.1 and should be mainly oriented towards two “Big Ideas” of AI – perception and learning (Touretzky et al., 2019) – as practical guidance for designing AI curricula. According to Touretzky (2019), perception is one of the most significant achievements of AI which enables computers to perceive the environment by interpreting sensory signals. Learning is another machine learning technology that stimulates significant advances in many areas of AI. By playing with machine learning algorithms, the children should learn how machine learning algorithms enable a robot to create its own representations of the world using data that is either provided by students or acquired by the robot itself.

Students should be able to work with simulated robots in Open Robert Lab, following the demonstration of the motivational characteristics and effectiveness of using robots for educational purposes in Section 2.3.3. The involvement of robots should help children understand perception as a process in which sensors are used to observe the environment. At the same time, even the youngest students should be able to immerse themselves in the challenges of supervised, unsupervised, and reinforcement learning through interaction with the underlying algorithms and processes of machine learning. In this way, robotic systems should help establish a link between seemingly abstract learning content and reality, in which the robot either does or does not behave as expected.

The Neural Network Playground should allow the user to experiment with simple neural networks. The learner should be able to train the neural network by modifying the weights and directly observing the effects on the simulated robot, thereby grasping the concept of “direct supervision” outlined in Section 3.1.2.

In the Q-learning Playground, the student should be able to tinker with the Q-learning algorithm described in Section 3.2.2 by creating unique learning environments for the robot and playing with the parameters of the algorithm. Step by step, the student is expected to debug the algorithm and explore how the learning takes place from the agent’s perspective.

To make the unsupervised learning tangible for young students, the k-means algorithm described in Section 3.3.2 is adapted as an unplugged activity. Access to unsupervised learning unplugged is designed first to test a different type of hands-on activity and second as a control instance. The intention is to determine whether children perceive this topic of machine learning differently in terms of interest and difficulty if it is introduced unplugged instead of plugged.

All three approaches should be accompanied by a curriculum that introduces young learners to the various machine learning paradigms. Such an approach will reflect the complexity and breadth of the field addressed in Section 2.4.

By transferring the design principle of embodied interaction, described in Section 2.3.1, to digital space, students can be virtually put in the agent's shoes. The students should then immerse themselves in the behaviour of the simulated robot and put algorithms into practice in order to understand them and take a look behind the scenes. Such an approach should promote transparency towards the explainable AI (Long & Magerko, 2020), which is problematised in Section 2.4.

In order to accommodate children across school levels, approaches should be based on the "low floors and wide walls" principle described in Section 4.1.2. The examples should vary in their level of difficulty. Concerning the usability of and storytelling in materials and extensions, inspiration can be drawn from the graphic design and storytelling of children's books, mentioned in Section 2.3.1. Castella (2018) also proposes concrete principles for designing materials for children. In order to spark and maintain children's interest, learning materials should be colourful, have a similar logic, contain exciting task descriptions, and relate to a child's life. The extensions and materials should be appealing and straightforward: Design for children must be for everyone and allow room for exploration.

With all these possibilities, it is expected that children will be able to construct knowledge through the interaction between creating an artefact and understanding it. The use of blocks and robots should assist children in gaining a deep understanding of the technical aspects of different machine learning paradigms problematised in Section 2.4. This should make the underlying principles of machine learning transparent for students of all school levels, from primary to high school. The currency of knowledge as a key principle of connectivism described in Section 4.1.1 should be fostered.

4.2 User Study Design

In order to answer the third research question, the extensions and materials that are developed were tested with children. The sampling should include representatives across school levels – primary, secondary, and high school.

The user study was conducted in a three-day block with one session per day. On the first day, the approaches were evaluated with the high school children, grades 7–9, on the second day with the primary school children, grades 3–4, and on the third day with the middle school children, grades 5–6. All children had prior knowledge of working in Open Roberta Lab with LEGO EV3 robots, as they had participated in the introductory

session the day before. All three sessions took place as part of the summer holiday programme of the school administration office and the ZDI network MINT Düsseldorf, organised in cooperation with the non-profit coding initiative Codingschule junior (Codingschule gGmbH, 2020). All interested children were able to register for a session online. There were no requirements for participation. The first come, first served principle was used if there were more registrations than places. As this work was carried out in Germany, thus, all materials and extensions were developed in German.

At the beginning of each session, the children's knowledge of machine learning and AI was informally pre-assessed. The modules were then completed in the order described in Section 6.1. Finally, the children filled out short questionaries.

Due to regional measures implemented in response to COVID-19, restrictions were in effect when testing the extensions and materials. The children were not allowed to work in tandem or in groups. The limitations of all activities to individual work made it impossible to follow the peer learning approach, described in Section 4.1.2, as an education strategy.

The Q-learning Playground, introduced in detail in Section 5.4, was restricted to three maps, and the students were allowed to set as many obstacles as they wished. Although the extension was technically designed to allow students to create and upload environments on their own under certain conditions, it was necessary to restrict the activity in order to eventually achieve comparable results.

4.3 Evaluation Methods

A computer-based questionnaire with six items based on a 5-point semantic differential scale was developed to measure the children's perception of the machine learning topics. The goal of this was to understand how the children felt about the approaches and whether they experienced difficulties understanding them. The questions were formulated so as to allow the children to give personalised feedback on the difficulty of and their interest in the topics.

The semantic differential scale was chosen because it allows the rapid measurement of attitudes and performs well with few items (Salkind, 2006). The questions were as follows:

1. How interesting did you find the topic "Supervised Learning and Neural Networks"?
2. Was the topic "Supervised Learning and Neural Networks" difficult to understand?

3. How interesting did you find the topic “Unsupervised Learning”?
4. Was the topic “Unsupervised Learning” difficult to understand?
5. How interesting did you find the topic “Reinforcement Learning”?
6. Was the topic “Reinforcement Learning” difficult to understand?

To answer, the children indicated a number on a scale from 1 to 5 between two pairs of adjectives: “Very Uninteresting” (1) to “Very Interesting” (5) for questions 1, 3, and 5 and “Very Difficult” (1) to “Very Easy” (5) for questions 2, 4, and 6. In the evaluation, the responses to individual items were summed and then averaged to provide an overall score. A higher number thus reflects a more positive evaluation (Salkind, 2006).

In the questionnaire, the children were also asked about their general attitude towards further involvement with the topic AI and machine learning. The children could respond “Yes”, “Maybe”, or “No” and provide brief written feedback (one sentence) about what they took with them from the project day.

Figure 9 demonstrates an example of the question from the questionnaire. Before the student was asked to answer the question, she or he could view the image used in the presentation of a particular machine learning topic during the session to help the student recall the topic. All individual topics in the questionnaire were illustrated in this way.

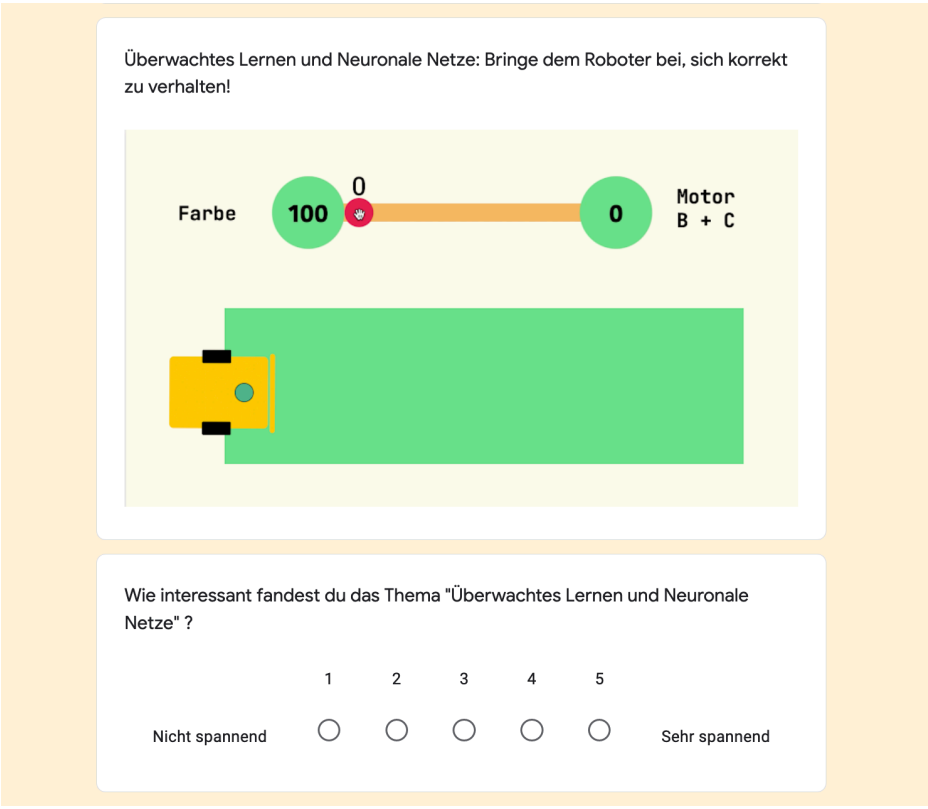


Figure 9: An item from the computer-based questionnaire developed to measure the children’s perception of the topic “Supervised Learning”.

The sessions were videorecorded, and an observer logged the activities of all three days. Parts of the videos – the introduction with the children's oral pre-assessment and the final feedback from the children about the session and what they learned about AI and machine learning – were transcribed for evaluation purposes. In order to take into account the ethical and legal aspects, all data collected was kept confidential and personal information made anonymous.

4.4 Tools and Project Management

A variety of tools were used to manage this work. In the following, these and the workflows pursued in this project are presented.

Source code version control with GitHub, IDEs IntelliJ and WebStorms, and Chrome DevTools

The Open Roberta Lab project's source code is distributed across several repositories on GitHub. For this work, two of them – `openroberta-lab` (Open Roberta, 2020c) and `blockly` (Open Roberta, 2020a) – were required and forked in the public account of the author (Olari, 2020b).

Extension development was managed through the GitHub platform using the Git version control system. The extensions were developed iteratively according to the idea of agile software development (Schmidt, 2015). In order to improve the traceability of additions made in the course of this work, new feature branches were created in each of the forked repositories: `feature/aineuralnetworks` in `openroberta-lab` and `feature/ai` in `blockly` (Olari, 2020a, 2020e).

In order to provide transparency about the development progress, code changes were commited daily. The changelog can be found in Appendix A.3, and the concrete code changes can be viewed in the git history of the projects.

GitHub issues and project management tools were used to manage various tasks. In `vlebedynska/openroberta-lab` (Olari, 2020e), two projects were created to manage the development of the extensions: "AI Extension: Reinforcement Learning" and "AI Extension: Neural Networks". In `vlebedynska/blockly` (Olari, 2020a), an "AI Extension" project was created to organise the UI design of the blocks.

Figure 10 shows the exemplary structure of the project "AI Extensions: Reinforcement Learning". It is divided into three stages: "To do", "In progress", and "Done". Depending on their status, the issues are stacked in the corresponding columns. Each issue includes a brief description. Some of issues contain subtasks. An exemplary description is shown

for Issue #19 on the right side of Figure 10. Each time a change is committed, it is tagged and assigned to the related issue.

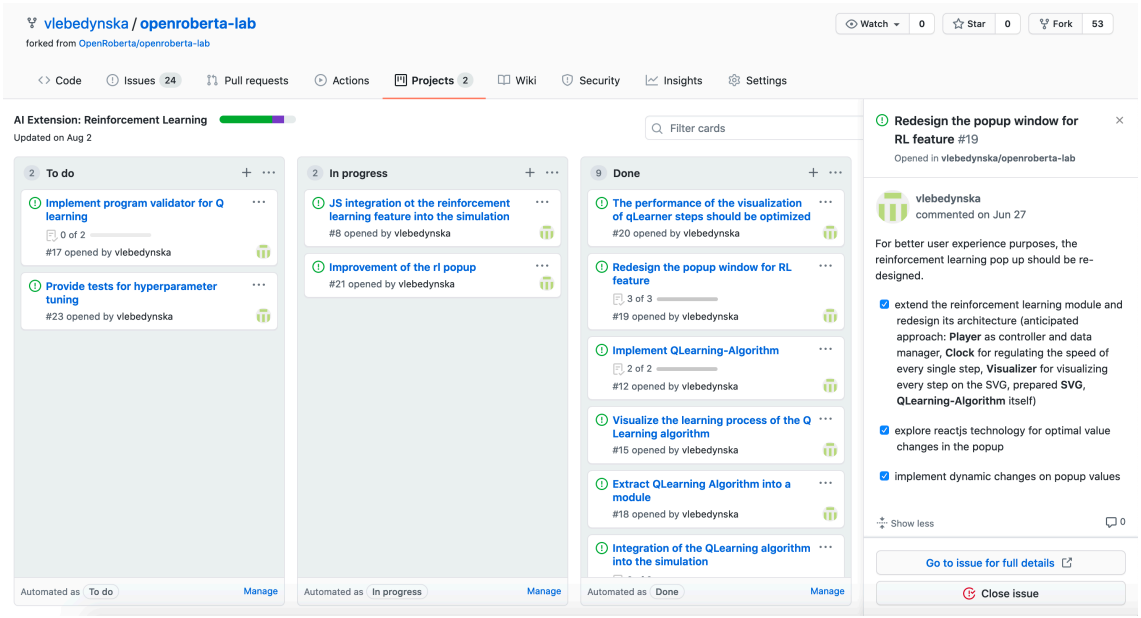


Figure 10: GitHub project “AI Extension: Reinforcement Learning” shows an example of the project structure and individual issues.

Two IDEs were utilised for the development: IntelliJ IDEA (JetBrains, 2020a) and WebStorm (JetBrains, 2020b). Although it was also possible to use IntelliJ alone, a WebStorm provided a better environment for developing and debugging TypeScript code.

The author also used the development tools provided by the Chrome browser for debugging, structure analysis of HTML, JavaScript and CSS sources, and performance testing.

Adobe InDesign, Illustrator, and Fresco

Adobe InDesign (Adobe, 2020c) is used for the design of learning materials. For drawing illustrations, Adobe Illustrator (Adobe, 2020b) and Adobe Fresco (Adobe, 2020a) are applied. Adobe Illustrator is also used to prepare the environments for the Q-learning Playground. It offers the possibility of drawing illustrations and saving them in SVG format, which is then implemented and processed in the Q-learning extension. The workflow of implementing and processing an SVG file in Q-learning extension is discussed in detail in Section 5.4.4.

5 Machine Learning Extensions: System Design and Implementation

Based on the approaches proposed in Sections 3.1.2 and 3.2.2, as well as the methodological framework outlined in Section 4.1.3, this chapter presents the design and technical implementation of new blocks and two machine learning extensions in Open Roberta Lab.

First, to enable the reader to understand the integration of new blocks and machine learning extensions into the existing Open Roberta Lab ecosystem, the author provides an overview of Open Roberta Lab and its initial project structure and system architecture. The overview focuses on processes relevant to the simulation environment of the LEGO EV3 robot, which the author has extended with new blocks and machine learning features. Although simulation environments are also offered for other robot systems, LEGO EV3 was chosen because it was the only driving robot and could be extended with examples of the simple neural networks based on ideas of the Braitenberg experiments described in Section 3.1.2.

Second, the author presents new block categories and blocks that she has implemented in Open Roberta Lab to enable the user to program simple neural networks and applications based on the Q-learning algorithm. In order to fill each block with functionalities, for instance, to enable the Neural Network block to create a neural network, the author has made several additions in the Open Roberta Lab project. Since there is limited documentation for the execution workflows in Open Roberta Lab, the author makes the execution process transparent for the reader by describing the simplified workflow using the lifecycle of a block implemented by the author as an example. In this way, the author demonstrates the central points in which she has made additions to Open Roberta Lab project while implementing functionalities for each of the blocks.

Third, the author describes machine learning extensions that she has developed to allow the user to experiment with machine learning technologies, as described in Sections 3.1.2, 3.2.2, and 4.1.3: the Neural Network Playground for training simple neural networks and the Q-learning Playground for visualisation of and interaction with the Q-learning algorithm. The author describes the system architecture, user interface, and central workflows of these extensions. Reflections on technical challenges close the chapter.

Before reading this chapter, the reader should note three remarks: (1) The reader will find that most figures and screenshots depict blocks and extensions in German. This is because the user study was conducted in German. If the reader would like to observe the blocks and extensions in English and view the demo of the blocks and extensions in use, he or she can follow this link:

`www.figshare.com/s/9bf7608f9408ea2f8da8`, accessed on 09.09.2020.

(2) If the reader would like to test the extensions, he or she can install Open Roberta Lab locally on the Raspberry Pi. The author has prepared an image[14] that the reader can download under this link:

`www.figshare.com/s/e92bb50916b8556eb603`, accessed on 12.10.2020.

(3) The reader should also note that this chapter only documents the most relevant processes and additions to the Open Roberta Lab project. However, if the reader desires more detailed information about the timing of development, he or she can refer to a comprehensive commit history maintained by the author and attached in Appendix A.3. If the reader wishes to examine specific code changes, he or she can refer to the respective repositories in the author's public account on GitHub (Olari, 2020a, 2020e). There, the additions, deletions, and changes are presented. The reader can also view the commit history tagged in individual issues (Olari, 2020c, 2020d).

5.1 Investigating Open Roberta Lab: System Overview

The development of new components required extensive additions throughout the entire Open Roberta Lab project. In order to achieve the set goal, the structure of the Open Roberta Lab project and central workflows had to be examined in detail. At the time of writing, Open Roberta Lab did not provide much documentation on processes and workflows. Therefore, the following three sections summarise the essential findings and aspects relevant to further understanding of the development of new UI blocks and new machine learning features.

5.1.1 User Interface

When visiting the Open Roberta Lab environment (Open Roberta Lab, 2020), the user is asked to select one of the available robots to continue. Figure 11 illustrates the user interface after selecting the LEGO EV3 robot.

[14] The image must be copied to an SD card and inserted into the SD card slot of a Raspberry Pi. Open Roberta Lab is started at boot time. Now the user can access Open Roberta Lab from any device on the same local network by entering the address http://orlab.local in the browser. See Open Roberta (2020b) for more detailed instructions.

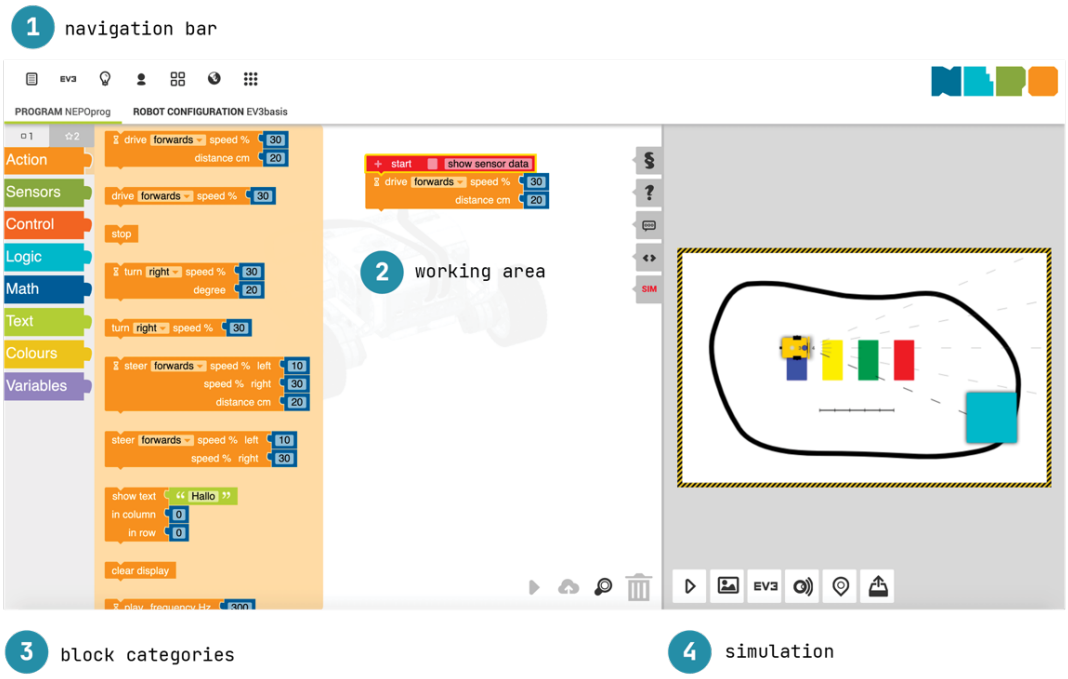


Figure 11: The user interface of Open Roberta Lab (Open Roberta Lab, 2020).

Via the (1) navigation bar, the user can save, export, and import the program and can open a source code editor. Additionally, the user can switch the robot system and connect to the real robot, get help, log in to the personal area, open a gallery of the published programs of other users, and change the interface language.

Under the (3) block categories, the user finds programming commands in the form of blocks available for the selected robot. The block categories presented in Figure 11 are similar for all implemented robot systems.

In the (2) working area, the user assembles the program from singular commands listed in block categories. The commands to be executed must be appended to the red “Start” block. The program that is developed can be executed either on a real robot by pressing the “Start” button at the bottom of the working area or in the simulation. The commands are executed in the order in which they are connected in the program from top to bottom. The commands are executed linearly, so there is no possibility for the parallelisation of processes.

The (4) robot simulation can be opened by pressing the “SIM” button located on the right side of the working area. The robot simulation imitates the behaviour of the real robot, although not all commands and sensors of the real robot are available. In the simulation for the LEGO EV3 robot, the user can run the program on the simulated robot, change and load custom backgrounds, interact with the simulated robot brick, access the current measurements of the plugged sensors, and return the robot to its initial position.

5.1.2 Project Structure

The source of Open Roberta Lab is stored in the GitHub project Open Roberta (Open Roberta, 2020d). For this research, two repositories – `openroberta-lab` (Open Roberta, 2020c) and `blockly` (Open Roberta, 2020a) – were copied to the author's public repository (Olari, 2020a, 2020e).

`openroberta-lab` is the central repository of the Open Roberta Lab project. The installation description and an overview of the needed libraries can be found in the `README.md` file of the project. The `openroberta-lab` project implements back-end functionalities for robot systems and is mainly written in Java, Python, and C++. It also contains front-end functionalities for the web appearance of Open Roberta Lab and the robot simulation environment in JavaScript and TypeScript. Apace Maven is used for package management. The JavaScript library `require.js` is used as a file and module loader.

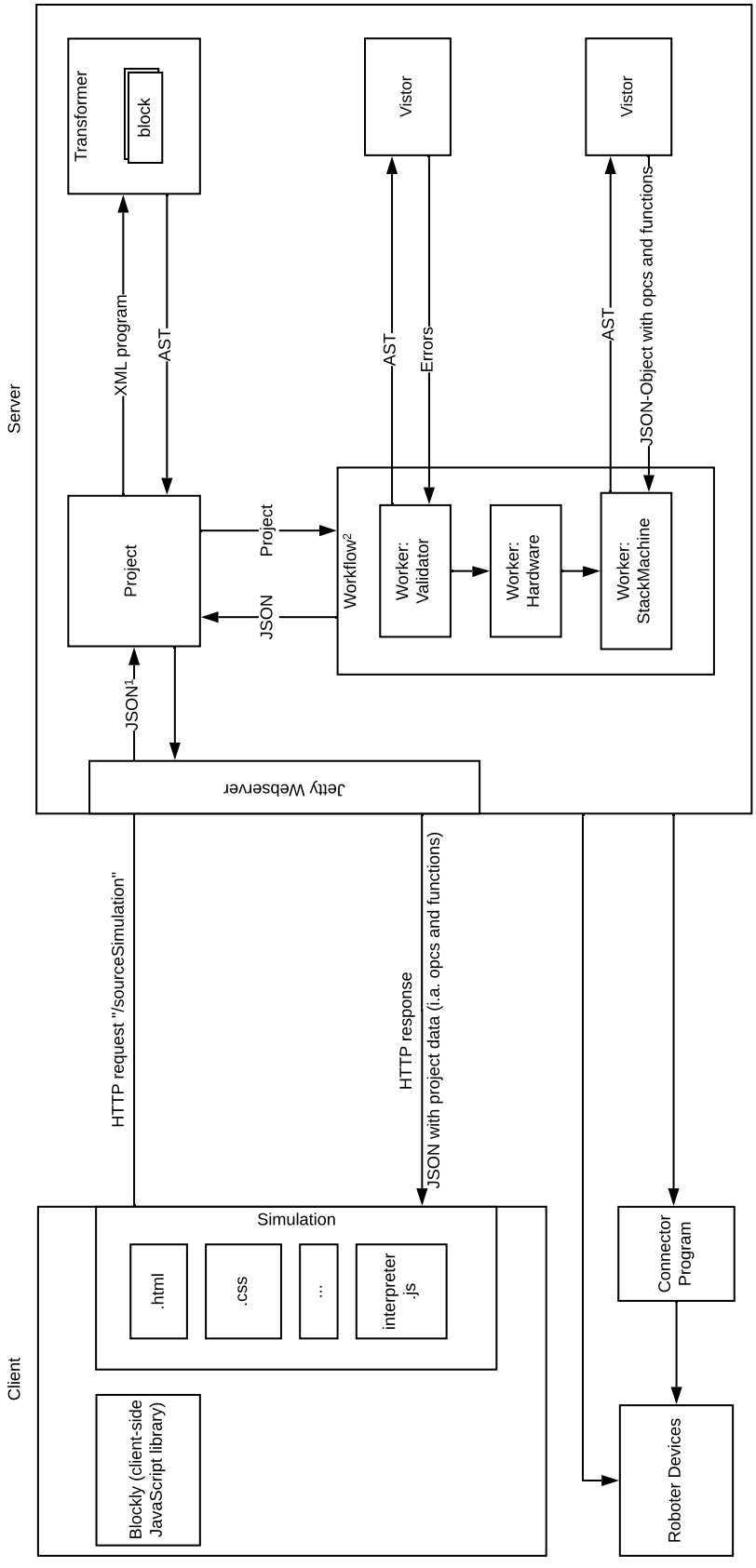
Blockly is the client-side JavaScript library that provides a web-based visual programming editor (Google Developers, 2020c). The Blockly editor of Open Roberta (Open Roberta, 2020a) allows the defining of command blocks,[15] which can be exported as a compressed JavaScript file and used in the local `openroberta-lab` project. Blocks can be accompanied by translations into different languages by adding the corresponding meanings to the JSON file of the respective language.

Currently, for the LEGO EV3, Open Roberta Lab offers eight block categories with 48 command blocks in beginner mode and 11 block categories with 100 command blocks in expert mode.

5.1.3 System Architecture

Only limited documentation is available for architecture and workflows in Open Roberta Lab (Open Roberta, 2019a, 2019b, 2020e). However, it was vital to understand the overall system architecture and central processes in order to implement the functionalities for the individual blocks. For this reason, the author reverse engineered the project. **`Error! Reference source not found.`** shows the result, presenting an excerpt from the schematic overview of the Open Roberta Lab structure relevant for machine learning features. Specifically, the figure visualises the system components and their interdependencies involving the robot simulation environment.

[15] Block types, their appearance and the compatibility of the blocks with each other are defined by Google Developers (2020a).



[1] Project data (robot's name, password, language, XML program, robot configuration XML, Token etc.)
[2] Workflow can change the project. Project object can be manipulated in workflow steps.

Figure 12: Simplified system overview of the Open Roberta Lab project involving the simulation environment.

Client Blockly

On the client-side, the Blockly library is responsible for the definition and visualisation of the programming blocks. When the user starts the simulation in Open Roberta Lab, the program composed by the user is transmitted to the server via an HTTP request as JSON data for further processing. JSON data contains not only the program itself in XML format, but also other project data such as the name and configuration of the robot.

Server

On the server-side, the Jetty Java library runs the web server and provides the interface to the backend of the project. When the JSON data arrives on the server, the block types from the XML file are mapped by `Transformer` to the corresponding Java classes using the axillary JAXB[16] library. The output of the `Transformer` step is an abstract syntax tree (AST)[17] component, which is returned for further processing.

Among other project data, the AST itself also passes through the workflow pipeline. In the case demonstrated in **Error! Reference source not found.**, the workflow is determined by data transmitted via an HTTP request. The pipeline in **Error! Reference source not found.** is composed of three workers: `Validator`, `Hardware`, and `StackMachine`. However, depending on the workflow, the pipeline may defer. A worker performs tasks sequentially and can manipulate the project data.

(1) The `Validator` worker sends a request to the visitor[18] encoding the element class to check the validity of the AST component. This step returns an error list if any errors are found. In case of an error, the workflow procedure is aborted.

(2) The `Hardware` worker verifies that the hardware for the robot to use is correctly configured and installed.

(3) The `StackMachine` worker then sends the project data to the block visitor, which assembles the operations and functions defined in respective AST components into a JSON object.

Simulation Client

The client extracts the JSON object and processes functions and operations in `interpreter.interpreter.js`. Programs for Open Roberta Lab simulation are conceptionally implemented as a stack machine. Each program operation is executed depending

[16] JAXB – allows the mapping of Java classes to XML representations.
[17] The AST is a detailed tree representation of the Java source code (Vogel, Scholz, & Pfaff, 2020).
[18] Gamma, Helm, Johnson, and Vlissides (2019) describe the visitor as a pattern that allows a new operation to be defined without changing the classes of the elements on which it operates.

on its operation code (opc) and can use the stack to exchange the data with other operations on the program stack. `robotMbedBehaviour.js` is the main file in which the functions for the simulation of the robot's behaviour are defined.

5.2 New Blocks and Categories for Machine Learning Playgrounds

With blocks, the user creates programs in Open Roberta Lab. In order to enable the user to program neural networks and applications based on the Q-learning algorithm, new blocks had to be developed. To this end, Open Roberta Lab was extended with the addition of a new "AI" category, consisting of two subcategories – "Neural Networks" and "Reinforcement Learning". The general name "AI" was chosen in order to allow new subcategories to be added in the future.

This chapter briefly outlines considerations for creating new blocks for machine learning features, followed by a description of the functionalities of new blocks and a presentation of the execution workflow on an exemplary lifecycle of a block. The lifecycle illustrates the process of how a program written in blocks reaches the simulated robot.

5.2.1 Considerations for Designing of New Blocks

In order to enable children to construct knowledge through the interaction of creating an artefact and experimenting with it – one of the design principles defined in Section 4.1.3 – the author created a set of 16 command blocks. The following considerations guided the author:

(1) The blocks should be easy to grasp, so that even young students do not experience difficulty understanding them. The complexity of the technical terms thus had to be reduced. Instead of using abstract vocabulary, the author simplified the language so that even children without previous knowledge could understand the blocks. If a term could not be simplified, for instance, the term "Neuron" in Figure 13, it had to be explained in the learning materials or by the facilitator in the introduction to the topic. The appearance of blocks also had to be simplified. For instance, lists used as layers in neural networks can be simplified, as demonstrated in Figure 13. These measures should promote the design principle of "low floors and wide walls", introduced in Section 4.1.3.

Traditional block presentation	Simplified version
	

Figure 13: Traditional presentation of the List block in Open Roberta Lab and its simplified version.

(2) The blocks should allow the students to create unique artefacts. For instance, users should be able to program neural networks with different types of input and output neurons. To make this possible, the author was able to reuse different types of sensors and actuators of the LEGO EV3 robot, enriching the user experience through a multifaceted approach to the topic. The user should also be able to create networks of different levels of complexity.

While creating applications based on the Q-learning algorithm, the user should be able to build learning environments of varying complexity and experiment freely with the parameters of the Q-learning algorithm. These opportunities should encourage the student's creativity by giving them room to experiment – another design principle defined in Section 4.1.3.

(3) In order to keep implementation costs within limits, if possible, the author should reuse the solutions provided by Open Roberta Lab. For instance, she was able to adapt the existing sensors and actuators as input and output neurons and reuse lists to represent the layers in the neural network.

5.2.2 AI Blocks

Figure 14 shows the user interface extended by the two subcategories – "Neural Networks" and "Reinforcement Learning".

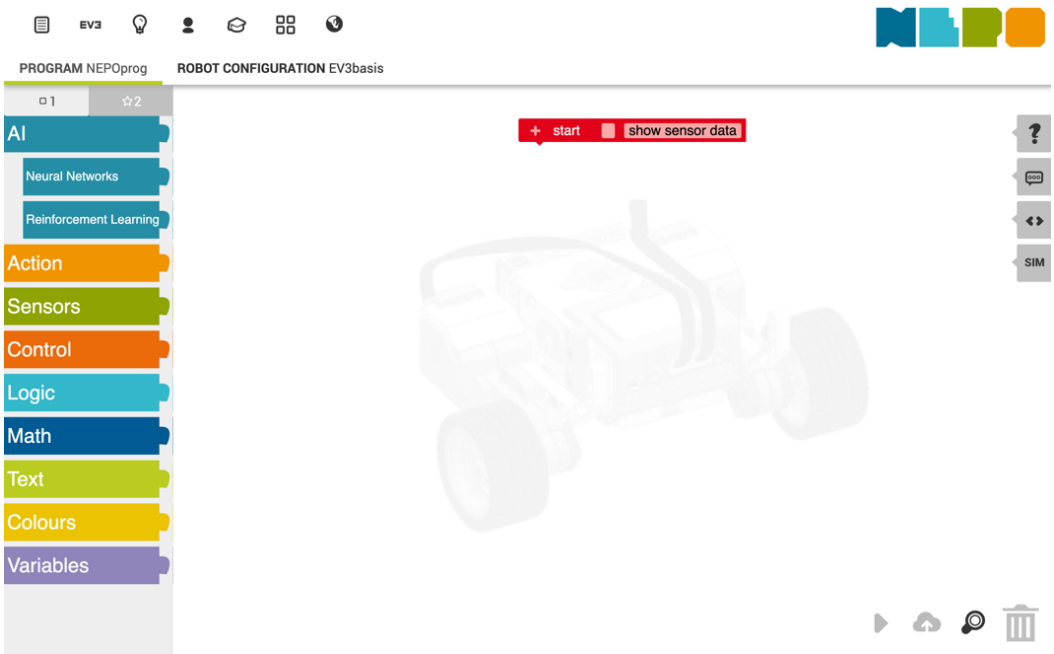
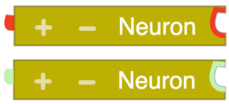


Figure 14: The extended user interface of Open Roberta Lab.

Ten new blocks were defined and implemented for the block category "Neural Networks". Table 1 presents them and explains their purpose.

Since the neural network is a new feature in Open Roberta Lab, the author had to define new data types: `InputNode` and `OutputNode`. All input neurons are of type `InputNode` and can be recognised on a red connection point. All output neurons are of type `OutputNode` and have a turquoise connection point. In order to implement the input and output neurons, existing functionalities of the selected sensors and actuators were adapted and modified. In the first versions of blocks, the input and output nodes had an additional opening for a threshold value, which the author also implemented in the backend. However, it turned out that the threshold value made the appearance of the block complex, so the author decided to leave it out for the initial run with students. It is possible that in the future, it could be made available again as a block for expert users.

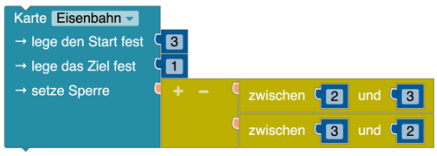
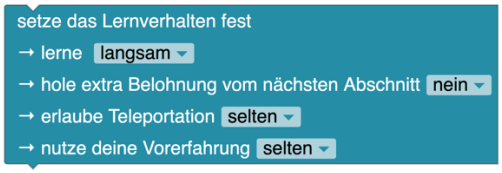
Table 1: New blocks developed for the subcategory "Neural Networks".

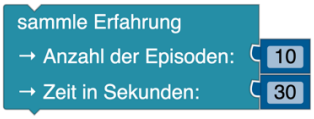
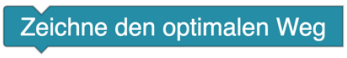
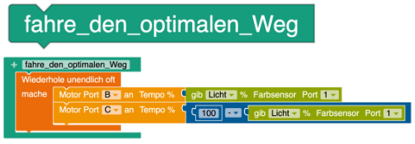
Block appearance	**Description**
	The neural network itself –has two openings. The left opening is for the input layer, and the right opening is for the output layer.
	The neural network layer – can be used as an input or output layer. The user can adjust the number of neurons by pressing the “+” or “–“ symbol.
	Ultrasonic sensor input neuron – uses the data supplied by the ultrasonic sensor plugged into the robot. It returns the distance to the next obstacle in cm.
	Light input neuron – uses the data provided by the robot’s colour sensor. It returns the light intensity on a scale of 0 to 100.
	Colour input neuron – uses the data from the robot’s colour sensor and returns 1 if the robot detects the selected colour and 0 if not. There are 8 colours implemented.
	Colour channel input neuron – uses the data from the robot’s colour sensor and

	returns a value from 0 to 255 for a selected channel. The value is then scaled from 0 to 100 for processing purposes.
	LED colour output neuron – uses the LED of the robot as output.
	Motor output neuron – uses the selected motor as output.
	Text output neuron – uses the robot's screen as output.
	Sound output neuron – represents the sound frequency that is sent to the robot's speaker.

Six new blocks were designed and integrated for the category "Reinforcement Learning". Table 2 outlines their appearance and functionality.

Table 2: New blocks developed for the subcategory "Reinforcement Learning".

Block appearance	**Description**
	Configures the learning environment. The user selects a map and defines the start and finish station. He or she can also place obstacles between two stations, which means that the robot cannot pass this section.
	Configures the learning behaviour of the robot. The user sets up the following: - α by choosing between different learning speeds - γ by receiving an additional reward from the next step

	- the NU value by allowing the robot to teleport to the station that is not directly connected to the previous station - the RHO value by allowing the robot to accept the use of its previous experience
	Starts the Q-learning algorithm. The user must specify the number of episodes and the time in seconds.
	Draws the optimal path based on the best Q-values on the way from the start to the finish station.
	Sets the robot to the start position in the simulation and lets it follow the line of the optimal path. This block is defined as a function, so it also requires a function body to be placed in the working area.

5.2.3 Lifecycle of One Block

A general overview of the program execution workflow is given in Section 5.1.3. In this section, the implementation details of the workflow process are presented with an example. The author describes the lifecycle of the block `ai_neural_network`, which demonstrates where in the existing ecosystem of Open Roberta Lab the author has inserted her additions. The lifecycle presented in the following is simplified, because it would go beyond the scope of this section to outline all the additions. However, the process is valid for all new blocks implemented by the author.

The necessary steps in the lifecycle of a block are as follows: (0) create a block in Blockly and integrate it into the Open Roberta Lab front-end; (1) create a program in Open Roberta Lab by assembling the blocks and starting its processing; (2) map blocks to corresponding Java classes; (3) parse blocks to AST components; (4) process the blocks to stack machine operations; (5) send the stack machine program to the simulation environment; (6) unpack the stack machine program and execute each operation in

the simulation. In the following, each step is described for the block `ai_neural_network`.

0) Before the new block `ai_neural_network` can be used, it must be implemented in the Blockly library (Olari, 2020a). The code snippet shown in Figure 15 demonstrates the block definition and presents the resulting program block. It should be noted that while the code snippet provides insight into the initialisation function of the block, it does not outline all the additions associated with the block definition, as they are distributed throughout the Blockly project.

```
/**
 * This block defines the simple neural network with an input and an output layer.
 */
Blockly.Blocks['ai_neural_network'] = {
    init : function () {
        this.setColour(Blockly.CAT_AI_RGB);
        this.appendDummyInput().appendField( 'Eingabe: ');
        this.appendValueInput("INPUT_LAYER").setCheck("Array_InputNode");
        this.appendDummyInput().appendField( '   ');
        this.appendDummyInput().appendField( 'Ausgabe: ');
        this.appendValueInput("OUTPUT_LAYER").setCheck("Array_OutputNode");
        this.setInputsInline(true);
        this.setPreviousStatement(true);
        this.setNextStatement(true);
        this.setTooltip(Blockly.Msg.NEURAL_NETWORK_TOOLTIP);
    }
}
```

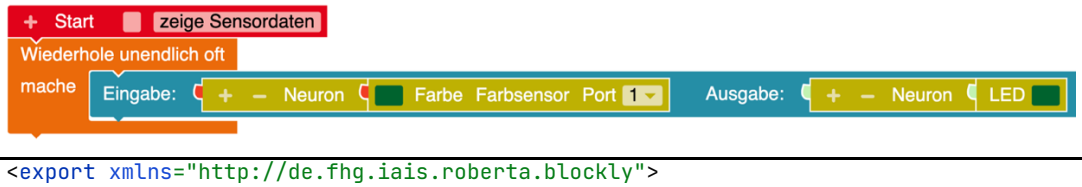
Figure 15: Code snippet for defining the `ai_neural_network` block in Blockly (top) and its visual representation (bottom).

After defining the colour of the block and label text, the `setCheck()` function sets the input type of the block, which can be passed as an inline input field[19] `INPUT_LAYER`. The input `INPUT_LAYER` allows the insertion of only blocks of type `Array_InputNode`. For the input `OUTPUT_LAYER`, the `setCheck()` ensures that the second input field only accepts blocks of type `Array_OutputNode`. Other local function calls inside the `init()` method define the appearance and functionality of the `ai_neural_network` block.

In order to use the defined block in the `openroberta-lab` project, it must be exported to a JavaScript file `blocks_compressed.js` by executing a Python script provided by the Blockly project. The generated file must then be manually moved to the `openroberta-lab` project.

[19] If the inline input fields are set to true, the holes in the block are arranged horizontally. Otherwise, they are arranged vertically.

1) When the user creates a program with the `ai_neural_network` block and compiles it by pressing the “Start" button, Blockly creates the representation of the program, including all blocks in XML format. This representation is subsequently transmitted to the back-end of Open Roberta Lab for further processing. Figure 16 demonstrates a simple program containing the `ai_neural_network` block and its representation in XML format.




```
<export xmlns="http://de.fhg.iais.roberta.blockly">
 <program>
  <block_set xmlns="http://de.fhg.iais.roberta.blockly"
             robottype="ev3" xmlversion="2.0" description="" tags="">
   <instance x="136" y="60">
    <block type="robControls_start" id="3-|B8P@m_T~q/T_Nd|*p" intask="true"
           deletable="false">
     <mutation declare="false"></mutation>
     <field name="DEBUG">FALSE</field>
    </block>
    <block type="robControls_loopForever"
           id="}faa=TbV.la1Ri3xi:gx" intask="true">
     <statement name="DO">
      <block type="ai_neural_network" id=",Z,j/{BU1-#Fh@hto.g+" intask="true">
       <value name="INPUT_LAYER">
        <block type="ai_easy_list" id="WDLKJ?#2,hy=!]q/#vgM" intask="true">
         <mutation items="1" list_type="InputNode"></mutation>
         <value name="ADD0">
          <block type="ai_nn_input_node_coloursensor_color"
                 id=";[@0}b)`K.He`Gb~Y4O!" intask="true">
           <field name="COLOUR">#00642e</field>
           <field name="SENSORPORT">1</field>
          </block>
         </value>
        </block>
       </value>
       <value name="OUTPUT_LAYER">
        <block type="ai_easy_list" id="Fmf`QA0Gq]vRkdEcPCy(" intask="true">
         <mutation items="1" list_type="OutputNode"></mutation>
         <value name="ADD0">
          <block type="ai_nn_output_node_led" id="SG+aSO/dR.-o|j.a5Wv7"
                 intask="true">
           <field name="OUTPUTNODE">#00642E</field>
          </block>
         </value>
        </block>
       </value>
      </block>
     </statement>
    </block>
   </instance>
  </block_set>
 </program>
 <config> … </config>
</export>
```

Figure 16: Representation of the Blockly program in XML format.

2) In the Open Roberta Lab back-end (the server component in Section 5.1.3), the blocks included in the XML program are mapped to the Java classes by their types

as defined in the corresponding configuration file in YAML format. The block type `ai_neural_network` is mapped to the Java class `de.fhg.iais.roberta.syntax.ai.AiNeuralNetwork`. The mapping is shown in Figure 17.

```
AI_NEURAL_NETWORK:
  category: STMT
  implementor: de.fhg.iais.roberta.syntax.ai.AiNeuralNetwork
  type: [ai_neural_network]
```

Figure 17: Configuration snippet for mapping the block type `ai_neural_network` to the Java class `AiNeuralNetwork`.

The Java class `AiNeuralNetwork.java`, attached in Appendix A.2, implements a method `jaxbToAst()` for transforming the block data from an XML source into the Java object representing an AST component. Due to the nested structure of the block, the transformation takes place recursively. Affected are the child nodes of the `ai_neural_network` block: two blocks of type `ai_easy_list`. Like any visitable AST component, the `AiNeuralNetwork.java` implements the visit method `acceptImpl()` used by visitor implementation.

3) Figure 18 demonstrates an example of how `EV3StackMachineVisitor` assembles the stack machine program, which consists of single operations from `aiNeuralNetwork` AST component. The stack operations for neural network layers are created by calling the `accept()` method for the AST component of each layer – `listNNInput` and `listNNOutput`. Finally, *`PROCESS_NEURAL_NETWORK`* operation is appended to the stack machine program.

```
@Override
public V visitAiNeuralNetwork(AiNeuralNetwork<V> aiNeuralNetwork) {
    aiNeuralNetwork.getListNNInput().accept(this);
    aiNeuralNetwork.getListNNOutput().accept(this);
    JSONObject o = mk(C.PROCESS_NEURAL_NETWORK);
    return app(o);
}
```

Figure 18: Code snippet of `visitAiNeuralNetwork` method.

4) The result of `EV3StackMachineVisitor` is a stack machine program in the form of a JSON object. The steps of the simulation workflow are completed, and the stack machine program is transferred to the simulation requestor.

5) In the simulation environment, the stack machine program is processed by the stack machine interpreter, defined in the file `interpreter.interpreter.js`. The interpreter runs through all operations. In the program section of the neural network, the

layers data are prepared before the actual processing of the neural network takes place. The data exchange is realised using the stack. When the interpreter reaches the stack operation *PROCESS_NEURAL_NETWORK*, as shown in Figure 19, the output and input layer data are on the stack. This data is popped from the stack, and afterwards, the function `processNeuralNetwork` is invoked with the loaded layers' data.

```
case C.PROCESS_NEURAL_NETWORK: {
    var outputLayer = s.pop();
    var inputLayer = s.pop();
    n.processNeuralNetwork(inputLayer, outputLayer);
    break;
}
```

Figure 19: Code snippet for *PROCESS_NEURAL_NETWORK* operation in `interpreter.interpreter.js`.

The `processNeuralNetwork` function is an entry point to the Neural Network Playground and is implemented in the `interpreter.robotMbedBehaviour.js` file. This function is responsible for creation and processing of the neural network that the user sees and interacts with in the simulation. As this function is large, it is presented in greater detail in Appendix A.1, and the content of this function is described in more detail in Section 5.3.4.

At this point, the program is running in the simulation on the robot, and the lifecycle of the block ends here. If the user recompiles the program by pressing the "Start" button, the lifecycle starts from Step 1.

5.3 Neural Network Playground

Once the neural network, including the user interface (UI) displayed in the pop-up, is generated, the user can experiment with the neural network in the Neural Network Playground. This section describes the author's considerations for the design of the Neural Network Playground and presents the solution. It outlines the workflow the user encounters, describes the user interface, and provides insights into the system design and implementation details.

5.3.1 Considerations for Feature Design

This section presents technical considerations for the design of the Neural Network Playground. In terms of content, the considerations are based on the idea of integrating supervised learning into Open Roberta Lab presented in Section 3.1.2.

In the course of the development of the Neural Network Playground, the author designed several mock-ups, of which Figure 20 presents several. The underlying idea of all mock-ups was to enable the user to train the neural network live: The user should get

direct feedback on how the training takes place by observing the behaviour of the robot. A direct causal relationship between changing a parameter of the neural network and observing the effect on the behaviour of the robot should enable even the youngest user to understand how the neural networks are trained and what the problems of supervised learning are.

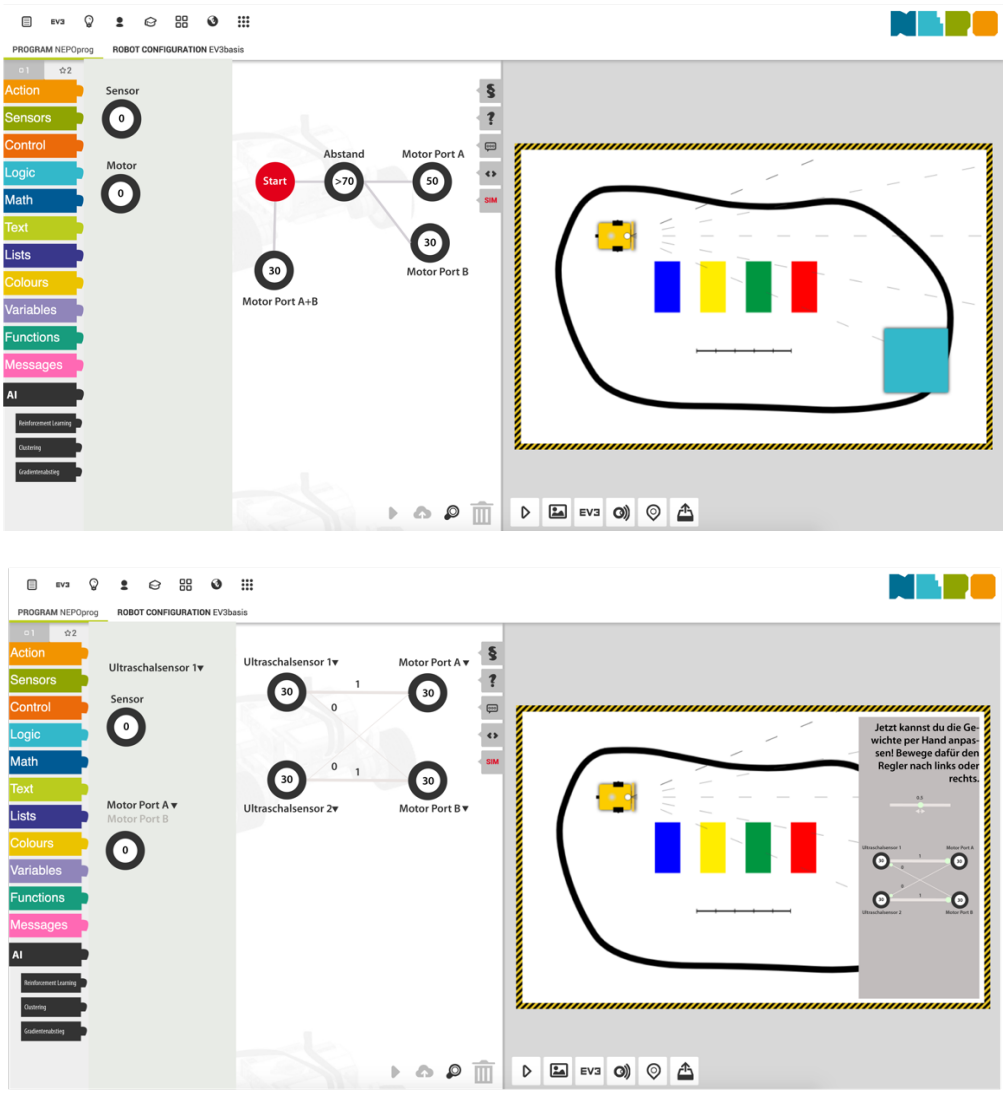


Figure 20: First mock-ups for the Neural Network Playground.

Both mock-ups show on the left side the idea of programming and training the neural network in Open Roberta Lab with round blocks, whereby the user can edit the values of single blocks and the weights of single connections. These mock-ups had two decisive disadvantages. First, the robot was not able to change its behaviour live; the users had to compile the program after each change to see how the change influenced the robot's behaviour. Thus, it would be not directly obvious what would happen if the user changed the weight. Second, it would not be possible to define round blocks, because the Blockly library does not support such serious adaptions in the design of blocks.

In order to address the second issue, the author considered adapting an open source solution available on the market, for instance, TensorFlow Playground (Smilkov & Carter, 2020), whose interface is shown in Figure 21. However, adapting the TensorFlow Playground does not solve the first issue and in addition may cause difficulties for young students: the input fields above the tensors are small and may be difficult to hit with the mouse for young students. Potentially, this could affect the user experience of primary school children, who also do not know how to cope with decimal numbers, in which the weights should be specified. Finally, and most importantly, the implementation efforts were difficult to estimate.

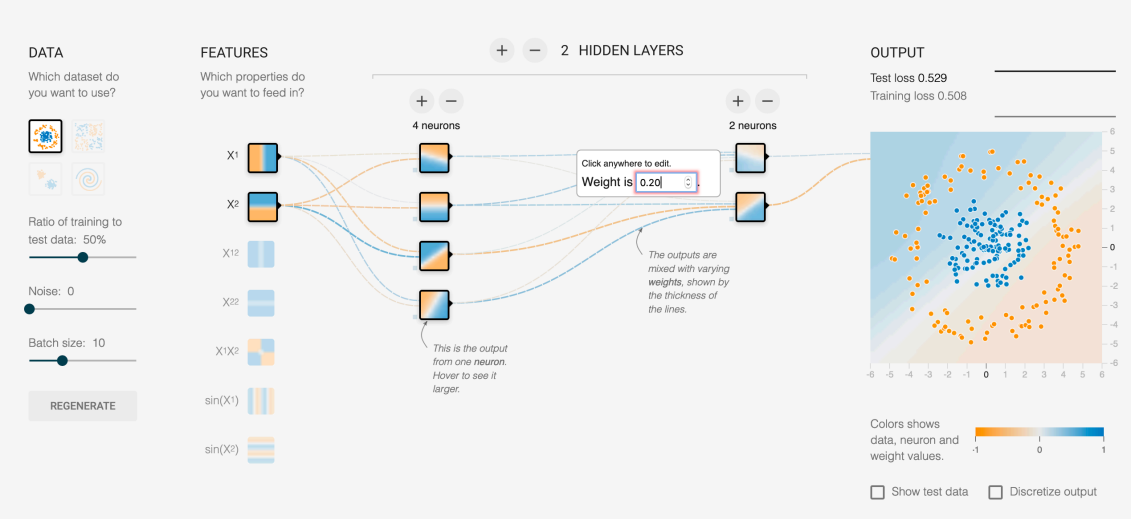


Figure 21: TensorFlow Playground as an idea to be implemented in Open Roberta Lab (Smilkov & Carter, 2020).

Therefore, the author decided to separate the programming and the training from each other. The programming was redesigned to be done with blocks as usual, while the training redesigned was to take place in a new simulation extension. The extension should be able to dynamically create a neural network from the program that the user has compound in blocks and allow the user to train the network and observe the effects directly, that is, to change the weights between the single connections and directly observe the result on the robot. This idea was realised in a grey pop-up on right side of the second mock-up in Figure 20.

In order to enable the students to easily operate the Neural Network Playground, the author made a series of decisions which are presented in a following list. It should be noted that the author did not formulate all these ideas at the beginning of the research study. Instead, the decisions were elaborated through several iterations of the development process of the Neural Network Playground:

1) The students should focus on the essentials of the neural networks as proposed in Section 3.1.2. For this purpose, the hidden layers should initially be left out. The students should explore and understand what is happening between two neurons of the input and output layer. However, the extensions should have the technical opportunity to be extended by the hidden layers.

2) In order to give the students room to experiment and playfully discover the topic, it should be possible to create the networks with any number and different types of input and output nodes.

3) The weight changing should be realised as a slider: activating the link with a click and moving the regulator back and forth should allow the user to directly observe the result of his or her action – that is, the change in the robot's behaviour. In the course of developments, the author noted that activation of single links may be difficult for the user, because the connection lines are thin. Having the slider above the neural network, as shown in the second mock-up in Figure 20, is also cumbersome, as the user has to switch between the activation of the link on the neural network and the slider. Instead, it would be more intuitive for the singular connections to have regulators, enabling the user to change the value of the link by moving the regulator directly placed on the link.

4) Since it can be useful in some cases, the user should have a possibility in the Playground to change weights in the neural network without its immediate execution. This could be realised by providing the possibility of pausing and starting the training of the network without the need to recompile the whole program. The user should be able to pause the simulation, configure the network, and then restart the reconfigured network on the robot.

5) The current value of the link should be expressed by changing the thickness of the line – the thicker the line, the stronger the link. This would help the young students see at first glance which connections are stronger than others. In addition to the thickness of the line, the value of the slider can be shown on the regulator. Being able to see the exact value of the link at first glance may be helpful for older students.

6) When the neural network is created, the initial values of the links should be 0. The students should then explore on their own what will happen if they move the slider towards 1.

7) In order to provide more transparency in the underlying processes, the current values of the input and output nodes should be shown inside the nodes. The nodes should also be labelled, so the user can easily make a connection between the program that

he or she has compounded in blocks and the neural network that is created. Depending on the sensor or actuator, the nodes can have different colours. For instance, if red colour from the colour sensor is used as the input block, then the input node in the Playground can be coloured red. Regarding the sliders in the neural network, the question arises as to whether to show the value for the active link only or to outline the values of all links at the same time. The Playground should be simple and clear but at the same time provide enough information to make the processes transparent.

8) The design of the Playground should be appealing and easy to understand. It should be oriented towards the focus group of school students. The dimensions of the Playground should be sufficiently large for easy operation. The design should also match with the teaching materials developed.

9) To keep the dependencies simple and the solution technically feasible, the usage of additional source code libraries should be avoided if possible. The author should reuse the elements already available in Open Roberta Lab to maintain the design consistency.

5.3.2 Workflow

Based on feature design considerations described in Section 5.3.1, the author established the workflow demonstrated in Figure 22.

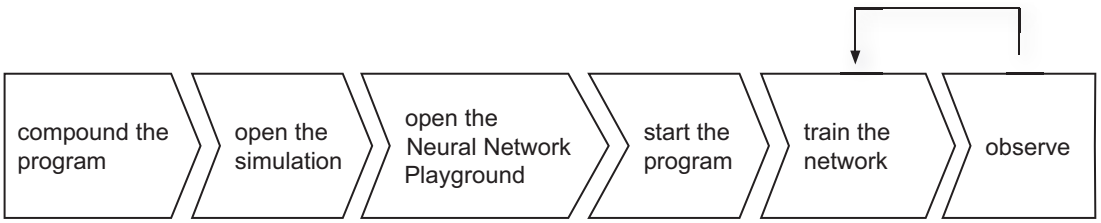


Figure 22: Process workflow for training the neural networks in the Neural Network Playground.

First, the user must compound the program using blocks. Then, he or she opens the simulation and then the Neural Network Playground. The program can then be started in the simulation. When the program is compiled, the neural network is (re-)created dynamically. The user may train it and observe how the robot's behaviour changes.

5.3.3 User interface

Taking into account the feature design guidelines from Section 4.1.3 and the considerations outlined in Section 5.3.1, the author developed the Neural Network Playground. Figure 23 demonstrates its appearance in Open Roberta Lab.

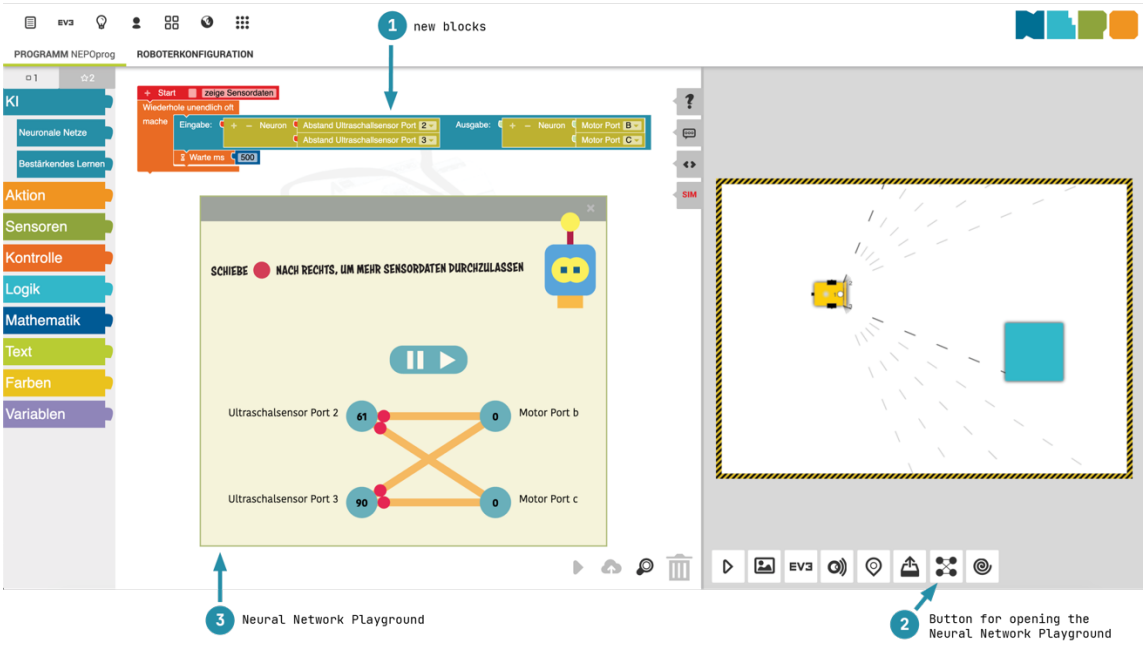


Figure 23: User interface of the Neural Network Playground.

In (1), a simple program for programming the Neural Network is demonstrated, (2) shows the button to open the Neural Network Playground, and (3) displays the Playground itself.

At the top of the Neural Network Playground, the user finds a brief description as a hint for what he or she can do: “Move the red circle to the right to allow more sensor data to pass through”. The user also sees the head of the mascot, which is used in the learning materials and ensures visual recognisability. Below the task description is a control bar, which allows the robot simulation to be stopped and started. This enables the user to configure the neural network first and then test it on the robot. Changing the weights between two nodes directly influences the behaviour of the robot without the need to pause the simulation. The current simulation state (paused/playing) is highlighted by the yellow background colour of the corresponding button.

The neural network UI is created dynamically and takes up most of the Neural Network Playground. The nodes of each layer correspond with the blocks set in the program and are labelled. Inside each node, the current value of the node is displayed. The value range of a node is between 0 and 100. Node colour depends on the node type: the nodes based on the colour sensor receive the colour that the user selected in the block. Other nodes have the default blue background.

The link between the nodes changes the thickness when the slider is moved. The value range of a link is between 0 and 1. The closer the value of the slider is to 1, the

thicker the line and the stronger the connection between two nodes. Unless it is zero, the current value of the link is shown above the slider.

Figure 24 illustrates the program with the colour input nodes. It also shows what the neural network pop-up looks like when the link between the first input node from the top and the first output node from the top is set to a value of 0.8: The link between these nodes is thicker than other links whose values are set to 0. The robot in the simulation is on the red square. Therefore, its red input neuron shows a value of 100. The values from the green channel node and blue channel nodes are 0, because the red square in the robot simulation is a pure colour without the addition of blue and green. The value of the input node is transmitted proportionally to the first output node, which means that with the current configuration of the neural network, the robot rotates the motor on port b at a speed of 80.

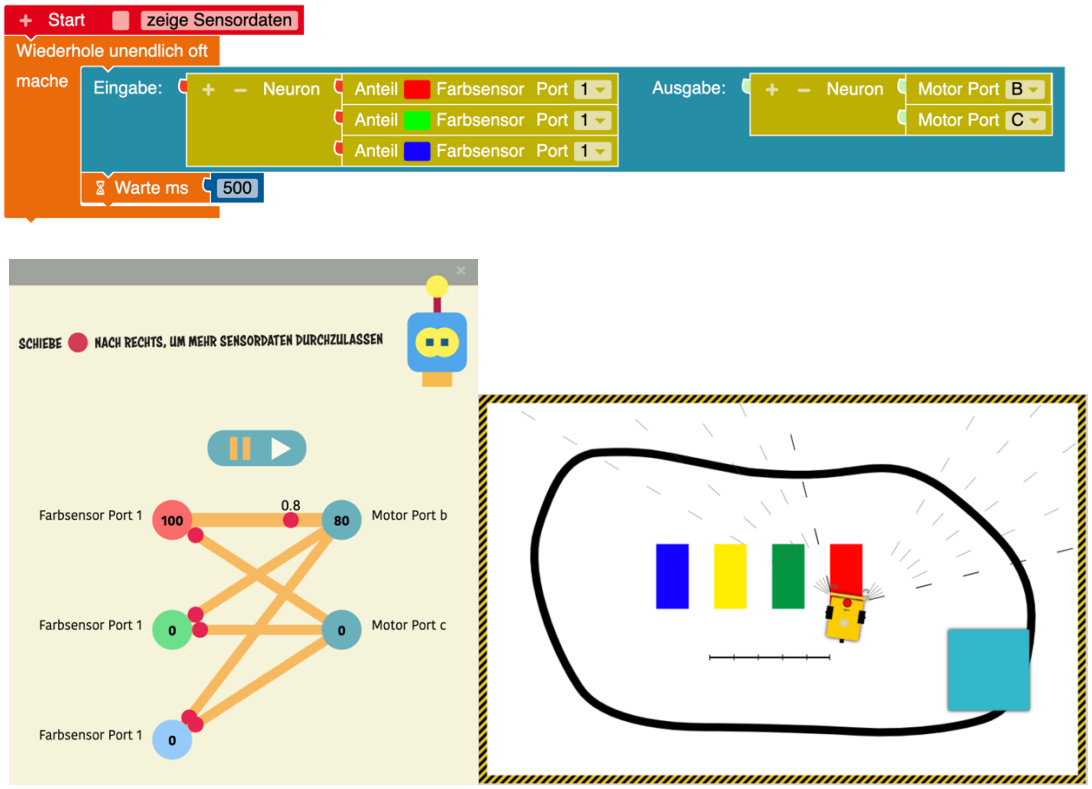


Figure 24: Adjusting the weights in the neural network.

5.3.4 System Architecture and Selected Implementation Details

This section presents the system architecture of the Neural Network Playground and outlines implementation details. The reader should note that only the most important implementation details are presented.

The Neural Network Playground operates completely client-side. It is written in TypeScript, which is a syntactical superset of the JavaScript (TypeScript, 2020). In comparison to JavaScript, TypeScript provides type safety and modularity, which helps prevent

error occurrence at the compile time and the runtime of the Neural Network Playground. One drawback of the TypeScript which the author took into account was that TypeScript requires compilation step before the application can be executed.

The Neural Network Playground is designed according the object-oriented approach and has a modular structure. The structure orients on the model–view–controller design pattern (Osmani, 2012), which leads to the separation of the user interface, the data models, and the underlying functionalities of the Neural Network Playground. In order to achieve the encapsulation of the components, the author implemented the event listener/event dispatcher pattern for the data exchange. To draw the UI of the neural network, the author decided to use SVG – vector-based graphics in XML format (w3schools.com, 2020). In order to manipulate the `SVG` objects more easily, the author used the `SVG.js` library (Fierens, 2020).

The following classes were implemented:

- `AiNeuralNetworkModule` acts as an interface between the Open Roberta Lab simulation and the Neural Network Playground extension.
- `AiNeuralNetworkImpl` implements the model of the neural network.
- `AiNeuralNetworkUI` implements the UI for the neural network model.
- `LinkImpl` implements the model of the link between neuron nodes.
- `LinkUI` implements the UI for the link.
- `NodeImpl` implements the model of the neuron node.
- `SVGSliderImpl` implements the model of the svg slider.
- `Player` provides the pause and restart functionality for the robot simulation.
- `Draggable` provides the functionality to drag the regulator on the link.
- `SVGUtils` includes diverse helper methods for usage by svg components.

The models of `LinkImpl`, `NodeImpl`, `AiNeuralNetworkImpl`, and `SVGSliderImpl` classes were abstracted in, respectively, `Link`, `Node`, `AiNeuralNetwork`, and `SVGSlider` interfaces. The model of `AiNeuralNetwork` supports a multi-layer structure.

Figure 25 shows in simplified form how the Neural Network Playground is initialised. The author omitted loops and some event dispatching/listening constructs for clarity purposes. The diagram also shows only calls of the most vital local functions which are necessary to understand the process. The diagram design is based on the suggestion of Balzert (2011). It shows parameters in the functions only in the most important cases.

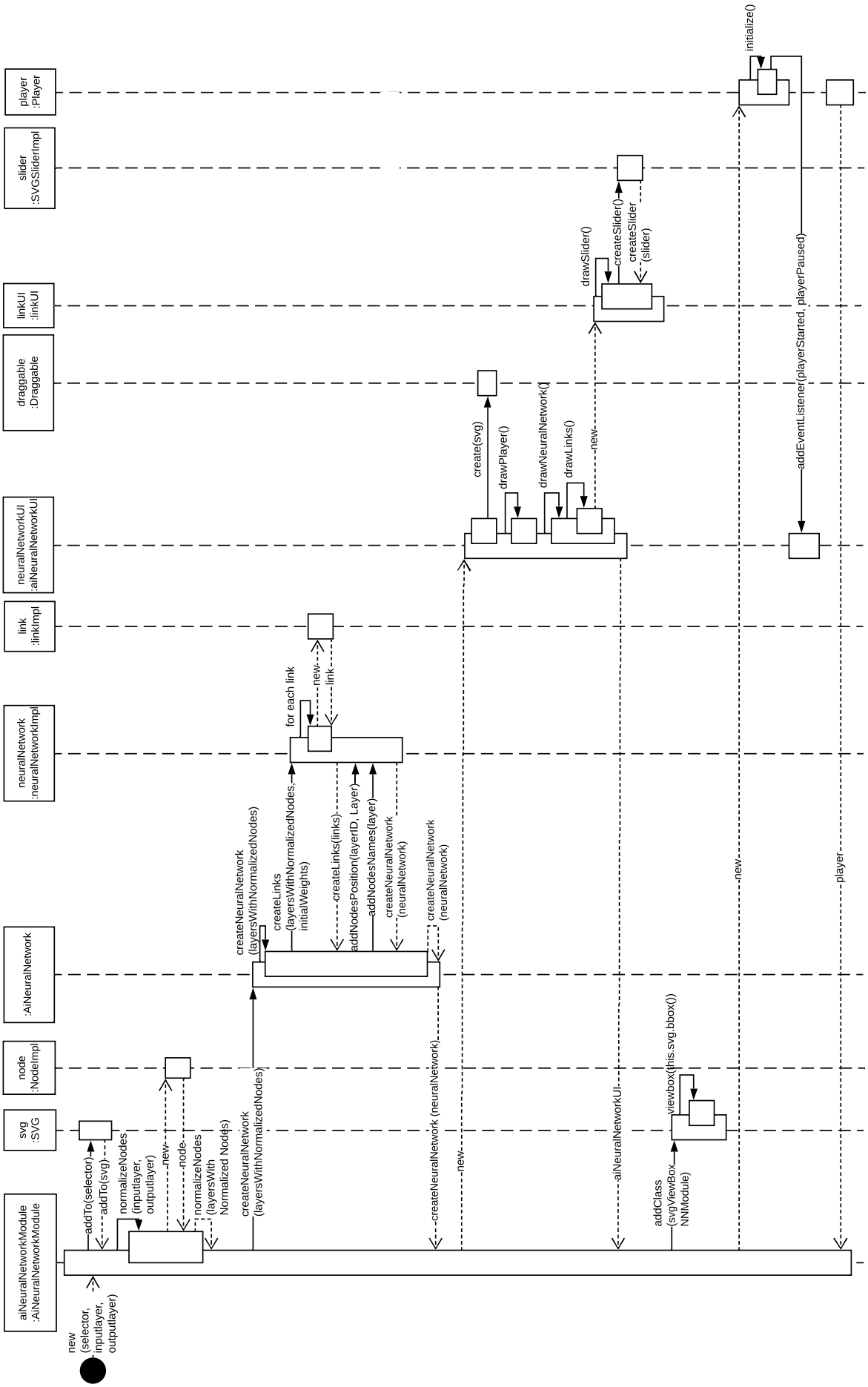


Figure 25: Simplified sequence diagram for creating the Neural Network Playground.

`AiNeuralNetworkModule` is instantiated as a singleton by the `processNeuralNetwork()` function, described in Section 5.2.3, and is an entry point into the Neural Network Playground. It initialises the Neural Network module UI as the svg object – the instance of the `svgdotjs` module from `SVG.js` library that the author used to simplify the manipulation of the `svg` objects. The `AiNeuralNetworkModule` then pre-processes the input and output node values derived from the stack program for simulation by converting them to neural network nodes. The values of the input nodes are normalised to the range from 0 to 100. The instances of the class `Node` are created. Subsequently, `AiNeuralNetworkModule` initialises the neural network by creating an instance of `AiNeuralNetworkImpl` and of the corresponding UI object `AiNeuralNetworkUI`.

`AiNeuralNetworkImpl` implements the core functionalities of the neural network. Here the functionality for creation of the neural network, including the creation of links between the nodes and the calculation of the neural network output, is defined. `AiNeuralNetworkUI` is responsible for the visual presentation of the neural network and handles user's activities. On initialising, it creates a module global instance of `Draggable`, draws the `Player`, and initialises the `LinkUIs` for each link between two nodes. `Draggable` implements `mosemove`, `mouseup`, and `mousedown` event handling by dragging the regulator on each link. `LinkUI` contains an instance of `SVGSliderImpl`. Furthermore, it implements functionality for updating the thickness of the link on its weight change and the event handling on the link activation and deactivation. `SVGSliderImpl` is responsible for creating a slider, calculating the current slider value, setting the new slider value upon external update, and updating the position of the slider regulator and the text above the slider regulator upon slider value change. `SVGSliderImpl` also implements the `closestPoint()` function, defined in the `SVGUtils` class, which is responsible for keeping the slider regulator on path while dragging. Details on the implementation of the Draggable and closestPoint algorithm can be found in Chapter 5.5.

In order to dynamically customise the Neural Network Playground's UI size depending on the total number of nodes in the input and output layer, the `AiNeuralNetworkModule` adds a CSS class `svgViewBoxNNModule` and sets the size of the Playground's HTML element to the svg `bbox`. `Bbox` is a minimum bounding box that can be drawn to include all elements inside the `svg`. Finally, `AiNeuralNetworkModule` initialises the instance of `Player` which implements functionalities to control the execution of the simulation.

Once the Neural Network Playground is created and the simulation is running, which means that the stack program is executed and the player running state is set to true, the data of the input nodes is permanently actualised. The values of the output nodes are

updated respectively. On the `dragmove` event, that is, if the user moves the regulator of one of the links, the `sliderValue` is recalculated depending on the current position of the regulator on the path, and the corresponding output node's value is updated. Then, the `sliderValue` property setter function that dispatches the `sliderValueChanged` event is called. The event listener implemented in the slider in the `LinkUI` receives the `sliderValueChanged` event and changes the thickness of the line.

The `processNeuralNetwork()` function defines the robot's behaviour depending on the value set for each output node. If the output node is a motor, the motor speed is set to the value given in the output node. If the output node is text, the text value of the output node is shown on the display of the LEGO EV3. If the output node is sound, the frequency based on value of the output node is played. If it is an LED, the LED of a robot is switched on or off depending on the value of the output node.

5.4 Q-learning Playground

Q-learning Playground is the extension that the author developed in order to foster the understanding of a novice of reinforcement learning, as described in Section 3.2.2. This section describes the author's considerations for the design of the Q-learning Playground and presents the solution. It outlines the workflow that the user encounters, describes the user interface, and gives insight into the system design and the implementation details.

5.4.1 Considerations for Feature Design

In terms of content, the Q-learning Playground is based on the methodological framework defined in Section 4.1.3 and on considerations about the Q-learning algorithm presented in Section 3.2.2. This section gives insight into concrete technical design considerations.

In the course of the development of the Q-learning Playground, the author designed several mock-ups, some of which are shown in Figure 26. The graphic on the top left shows how the Q-learning algorithm can be configured via blocks. The user may dynamically define the environment by defining the number of rooms that the robot can visit and setting the start and finish rooms. The user can also determine the policy by defining the possible directions of movement between rooms for the learning agent. For all rooms that are directly connected to the finish room, the agent should get a maximum reward. As shown in the mock-up, learning environments such as a soccer field can be used.

After the user has configured the algorithm and compiled the program, the learning environment on the right should be dynamically created, and the robot should start

learning by moving from one room to another and actualising its Q-value matrix. While the robot learns, the user can observe the learning procedure.

Figure 26: Mock-ups of the Q-learning Playground.

Two pictures at the bottom of Figure 26 visualise an alternative to the soccer playground. While the robot learns, it explores the environment, so that an increasing amount of the environment become visible. The aim of the learning procedure for the robot is to find the optimal path from start to finish in the created environment by calculating the highest Q-values on the way from start to finish. When the learning procedure is finished and the optimal path is calculated, the robot should drive this optimal path.

In order to achieve the objectives, the author considered the following concrete design suggestions:

1) In order to open the black box of Q-learning, the user should be given the ability to define all parameters of the algorithm: α, γ, RHO, NU, number of episodes, and total amount of time.

2) The user should be able to influence the learning environment structure by defining rooms through setting the walls and customising the start and finish via blocks.

This should foster beginners' creativity and inspire their imagination, as indicated in Section 4.1.3.

3) The transparency of the learning procedure should be provided by visualising how the robot learns. Insight into the learning procedure can be granted by animating the paths that the agent is visiting and highlighting the paths that have higher Q-values.

4) The result of the learning procedure should be visualised. For this, the optimal path from start to finish should be calculated based on the highest Q-values. The robot should be able to drive the optimal path.

5) The visualisation of the learning procedure should be separated from using the robot's knowledge in the simulation environment. This would help the beginners separate the learning process from the real world of the robot, in which it uses the knowledge gained. In order to realise this, the learning visualisation should take place in a separate window. The result – the optimal path – should then be transferred in the simulation, enabling the robot to drive the optimal path.

6) The robot should be able to move in the simulation following the calculated optimal path. For this, a new GPS sensor should be implemented or other solution strategies such as programming the line follower considered.

7) To design the learning environment, the SVG format can be used. At the beginning, the author evaluated several technologies that can be used for the visualisation of the learning process. She decided to use the SVG format because it is scalable and editable in a simple way and because animations for learning progress visualisation are straightforward to implement.

Based on these considerations, the author implemented the first prototype, as demonstrated in Figure 27. For this, she designed first drafts of Q-learning blocks, designed the Mars landscape by adding the nodes which can be visited, and set the goal for the robot to find the best way from quadrants A to I.[20]

[20] As the technical implementation remains the same for the final solution, this procedure is discussed in Section 5.4.4.

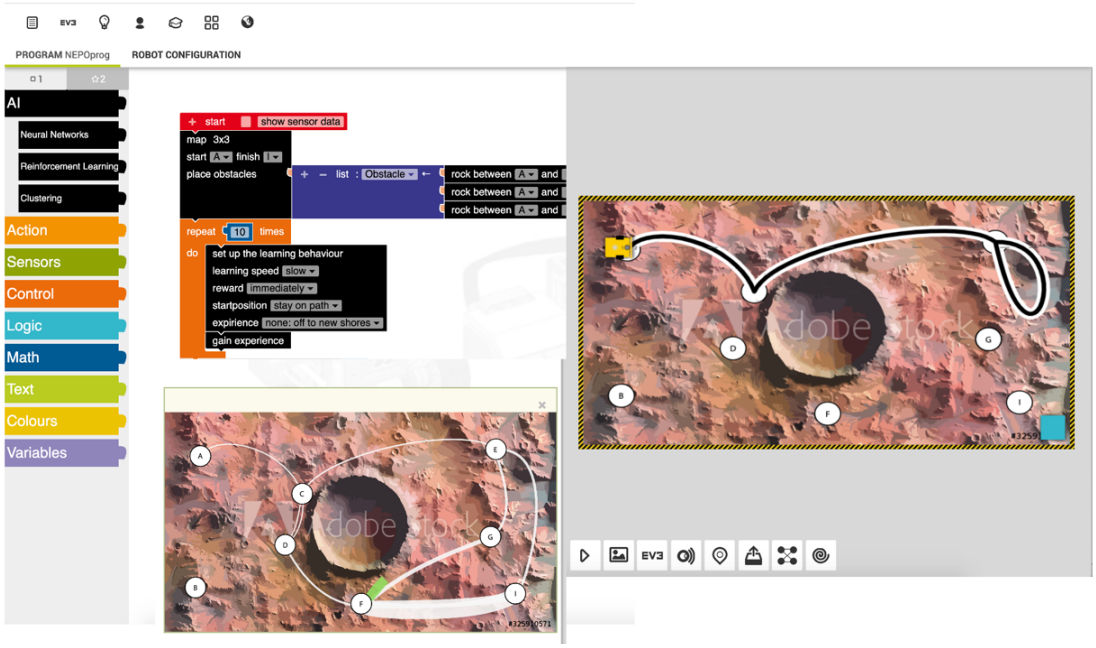


Figure 27: First implemented prototype for experimenting with the Q-learning algorithm.

While observing how the robot learns, the author encountered two problems and considered refinements for designing the final solution.

1) The first issue concerned the visualisation of the learning procedure, shown in the bottom window in Figure 27. The author decided to visualise the learning procedure by animating the route that was visited by moving the green line from the start to the finish node – as in the example from F to G. The Q-value of the road was visualised through the line thickness – the thicker the line, the higher the Q-value and the more likely the robot is to choose this way at the end. However, with this solution, the users had no ability to look deeper into the learning process: They only saw the green line moving fast from one node to another. The lines gained in thickness over time. The user took up the role of a passive observer without having the ability to stop the algorithm and show what was happening at each step.

   These issues should be addressed in the final solution; transparency in the learning procedure can be fostered by providing the user with more possible ways to control the learning process step by step, for instance, by implementing the control unit to pause and slowly execute the algorithm. Instead of line thickness, the representation of the Q-value can be realised with colour or additional icons such as stars. Visualisation of walls between the rooms should be improved. In the current solution from Figure 27, the walls are invisible; in the final solution, they may be visualised as rocks or stop signs.

2) Experience with the Mars landscape has shown that the learning environment does not necessarily have to consist of rooms – a quadrat matrix, as it is usually visualised, for instance, by Wieners (2014). For the Q-learning problem, any graph would be well suited, as shown by Xu et al. (2015), which opens more possibilities to design landscapes of several levels of difficulty. This point should be considered in the final design.

5.4.2 Workflow

Taking into account the considerations outlined in Section 5.4.1, the workflow presented in Figure 28 was established.

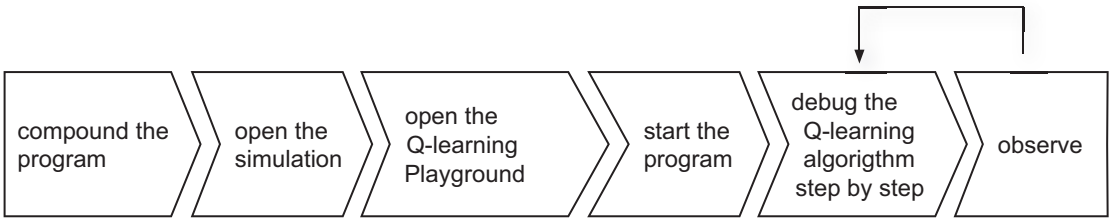


Figure 28: Workflow for starting the Q-learning Playground.

The user first compounds the program, configuring the learning environment and the parameters of the Q-learning algorithm. He or she then opens the robot's simulation and launches the Q-learning Playground. After pressing the "Start" button, the learning environment is loaded with the dynamic content based on the program data entered by the user. The user can then observe the agent's learning and debug the algorithm step by step.

5.4.3 User interface

The final version of the Q-learning Playground visualises, aside from the learning process, the parameters previously defined by the user in corresponding blocks. It also gives the user an opportunity to control the algorithm progress step by step.

Three versions of the Q-learning environment were developed: "Railway", "In the forest labyrinth", and "In the city". Figure 29 presents these.

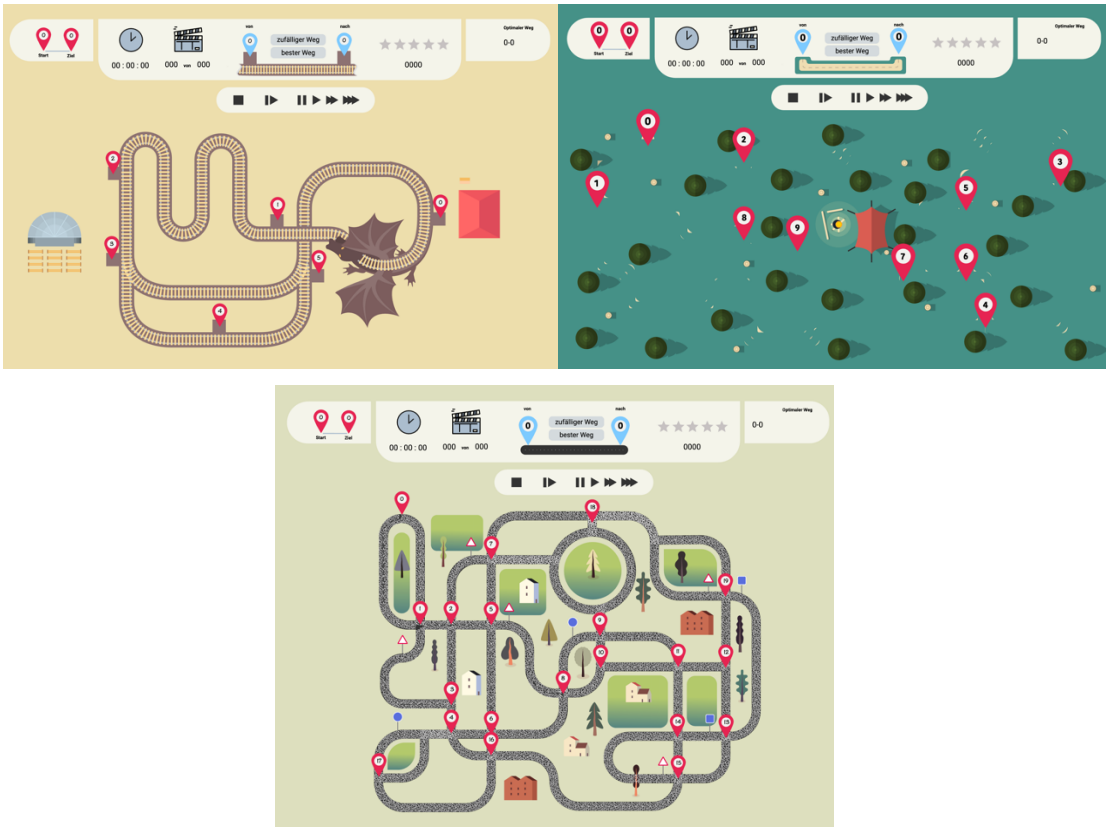


Figure 29: Three environments in the Q-learning Playground.

The environments have similar optics, with the same statistics and control panel above the map; the main distinction is the appearance of the map. Based on the number of nodes, the “Railway” is the easiest map, with only six nodes, “In the forest labyrinth” is a map of medium difficulty, with eight nodes, and “In the city” is the most difficult map, with twenty nodes. Visualisation of the Q-value also varies among the maps. Although the Q-value is visualised on every map in the statistic panel through stars – the higher the Q-value, the more stars the section receives – the “In the forest labyrinth” and “In the city” environments provide additional visual feedback. In the map “In the forest labyrinth”, the paths are initially hidden and are revealed slowly – the higher the Q-value, the more visible the path. In the map “In the city”, all streets are dirty first, and the higher the Q-value, the cleaner the street becomes.

Figure 30 represents the Q-learning Playground “Railway” in more detail.

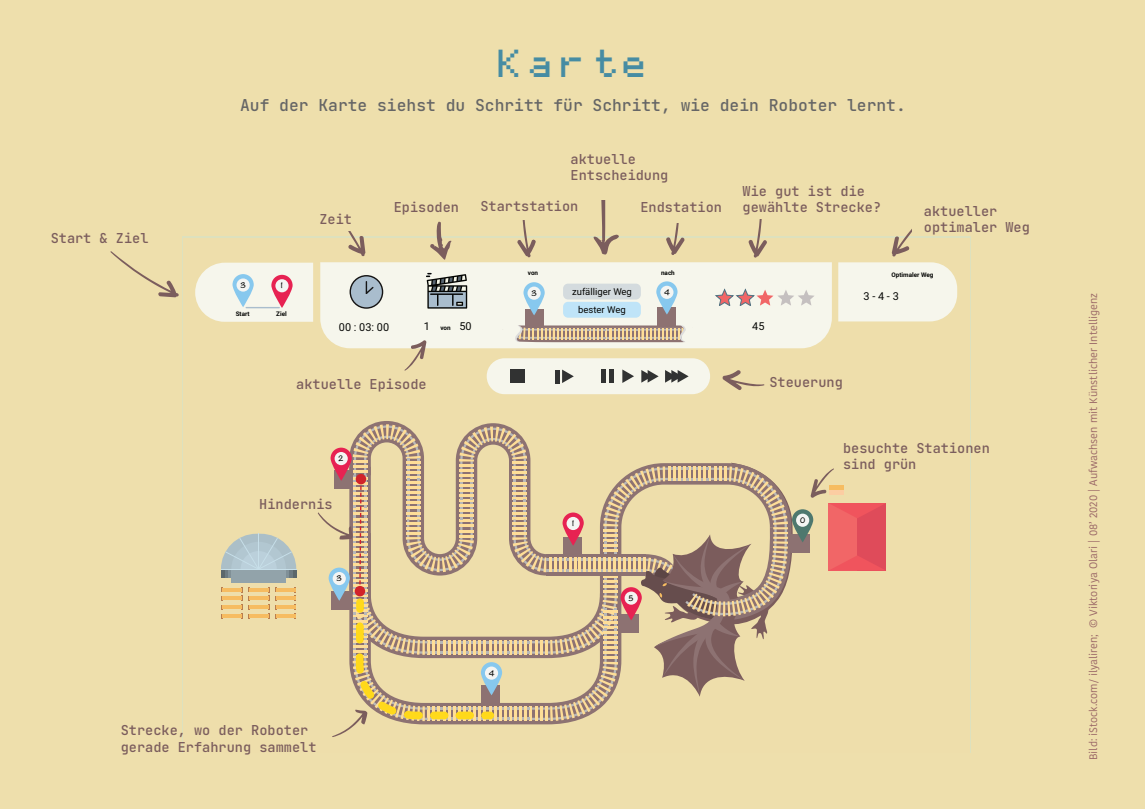


Figure 30: Q-learning environment "Railway".

At the top of the map, the user finds the dynamic statistic panel that changes sequentially on every Q-learning step. In the middle, the statistic panel visualises how much time is left, the current episode, and the current start station. Then, the current decision of the agent is shown - either to find the way to the next station or to exploit the knowledge and take the best way. The end station and the Q-value of the current segment in the form of five stars complete the panel. The presentation of the Q-value is always relative depending on the highest Q-value in the current step. Figure 30 demonstrates exemplary how the Q-value of the current step changed in comparison to the experience from the previous time the agent was exploring this way – it grew on one star.

The left side of the statistics panel displays the start and finish nodes, and the right side of the panel displays the current optimal path based on the segments with the highest Q-values on the way from the start to the finish node.

The control panel is located below the statistic panel. It enables the user to control the execution of the Q-learning algorithm. The user may stop, pause, and play the Q-learning algorithm at three distinct speeds: normal speed, twice as fast, and three times as fast. By pressing the second button on the left, the user also has an opportunity to execute the Q-learning algorithm step by step.

The nodes on the map have three states: not visited (red), visited (green), and currently active (blue). If the user has set obstacles in the program, they are drawn dynamically between the two nodes. In the map presented in Figure 30, one obstacle is set: between the second and the third node.

The current active segment between two nodes is animated: the active nodes are highlighted and the active segment from the start to the finish station is dynamically drawn as a yellow dashed line.

5.4.4 System Architecture and Selected Implementation Details

This section presents the system architecture of the Q-learning Playground and outlines the implementation details. The reader should note that only the most important implementation details are presented.

The Q-learning Playground operates completely client-side within a web browser and is written in TypeScript. It is designed according the object-oriented approach and has a module structure which orients on the model–view–controller design pattern and leads to the separation of the user interface, the data models, and the underlying functionalities (Osmani, 2012). In order to guarantee the encapsulation, the event listener/dispatcher pattern is implemented for the data exchange. It uses the SVG format to design the user interface and the `SVG.js` library to manipulate objects on the Playground. The CSS technology is used, for instance, to design the elements of the control panel such as Q-value stars or to manage the change of colours in nodes. Furthermore, the Q-learning Playground uses the async execution concept in order to implement workflows in a non-blocking way.

Generation of the Q-learning Problem

Before the author presents the classes and the implementation details, the reader should note the procedure for the preparation of the learning environments introduced in Section 5.4.3. It is vital for the reader to understand this process, because it outlines the basics for creating the Q-learning problem.

In order to create the Q-learning problem dynamically from the learning environment, the appropriate SVG files were prepared by the author. While designing the environments graphically in Adobe Illustrator (Adobe, 2020b), the author drew invisible paths in places in which an action between one node and another was possible. For instance, on the map shown in Figure 30, the author drew invisible paths on the railway between the following nodes: 0–1, 1–0, 1–2, 2–1, 2–3, 3–2, 3–5, 5–3, 3–4, 4–3, 4–5, 5–4, 5–0, and 0–5. Every single path was named according the schema `path-startnode-finish-node`, for example, `path-0-1` for the first path. Adobe Illustrator added this name as the

`id` attribute to the respective path. While implementing the Q-learning algorithm, the ids fitting to the x-path expression `[id^="path-"]` were filtered out and pre-processed to dynamically create a list of possible actions and states. The code snippet in Figure 31 outlines the function responsible for this.

```
public getActions(): Array<Action> {
    let listOfPaths: Array<Action> = new Array<Action>();

    let allPaths: List<Element> = this._svg.find('[id^="path-"]');
    allPaths.each(function (item) {
        let idName: string = item.attr("id");
        let tokens: string[] = idName.split("-");
        listOfPaths.push({
            startState: {
                id: parseInt(tokens[1])
            },
            finishState: {
                id: parseInt(tokens[2])
            }
        });
    });
    return listOfPaths;
}
```

Figure 31: Code snippet showing the creation of problem actions by parsing the path ids extracted from the SVG element.

With this, the author had everything needed to generate the Q-learning problem.

Overview over the Classes Implemented

The Q-learning Playground is realised within the `QLearningAlgorithmModule` created in the `interpreter.robotMbedBehaviour.js` as soon as the Q-learning block for defining the Q-learning environment is used in a Blockly program.

The following classes were implemented:

- `QlearningAlgorithmModule` acts as an interface between the Open Roberta Lab simulation and the Q-learning Playground extension.
- `ReinforcementProblem` defines the Q-learning problem, including the functionalities to access the states and execute actions.
- `QValueStore` implements functionalities to store, query, and update the values in the Q-table. It also includes the functions for the creation of the optimal path.
- `QLearningAlgorithm` implements the Q-learning algorithm.

- `qValueLookup` implements the `qValue` interface and is a helper class to cache the section's previous Q-value.
- `Key` is a helper class for the `qValueLookup` class. Its aim is to generate a unique key consisting of a source and a target state for a section.
- `Visualizer` implements the visualisation of the Q-learning process. It takes actions from the user and dispatches events for further processing. It also implements the `ProblemSource` interface, because the visualizer can provide states and actions of a Q-learning problem based on paths of the SVG element stored in a visualizer, as shown in Figure 31.
- `Svglookup` is a class helping reduce performance when searching the element in the corresponding document for a section by mapping the path SVG element to its section key.
- `PlayerImpl` implements the `Player` model.
- `TimerImpl` implements the `Timer` model.
- `RlUtils` implements the helping function for generating the rewards and problem.
- `Utils` implements various help functions, for instance, for different convert and normalise functions.
- `HyperparameterTuning` is a test class implementing the execution of the Q-learning algorithm without user interface in order to test the Q-learning algorithm using all possible parameter combinations. The test input values are created by permutating all possible parameter values for each test run.

To provide type safety, anonymous objects were avoided by defining interfaces such as `QLearningStep`, `ProblemState`, and `ProblemState`.

Architecture of the `QLearningAlgorithmModule`

As soon as the Q-learning block for defining the Q-learning environment is used in a program, it undergoes the lifecycle described in Section 5.2.3. In the sixth step of the lifecycle, the stack machine interpreter processes the *`CREATE_Q_LEARNING_ENVIRONMENT`*, operation which calls the `createQLearningEnvironment()` function defined in `interpreter.robotMbedBehaviour.js`. The `qLearningAlgorithmModule` is then initialised.

Serving as the interface to the `qLearnigModule`, `robotMbedBehaviour` defines three further methods that correspond to the Q-learning blocks described in Section 5.2.2: `setUpQLearningBehaviour()`, `runQLearner()`, and `drawOptimalPath()`.

Figure 32 demonstrates the interface between `robotMbedBehaviour` and `qLearningModule`. It also gives insights into the main components of the `qLearnigModule` and their interdependencies.

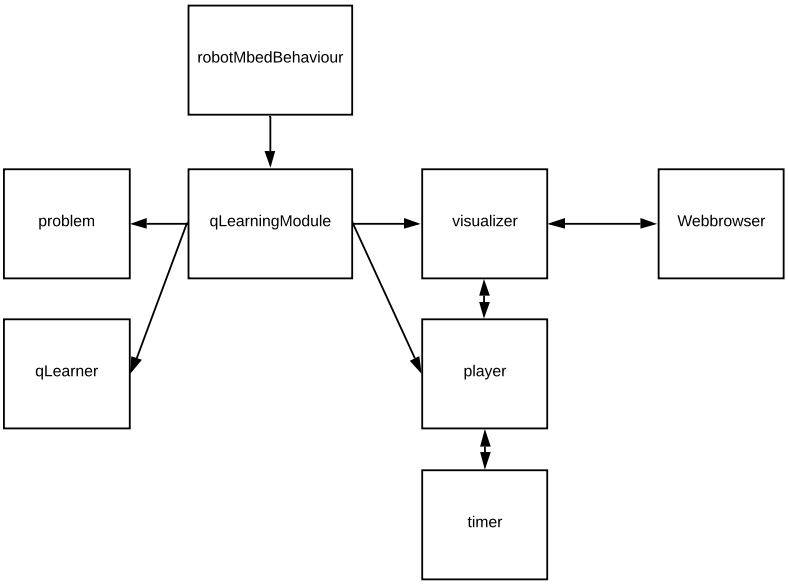


Figure 32: Main components of the `qLearningModule` (simplified presentation).

In order to understand how exactly the components interact with each other, the author designed a sequence diagram of the Q-learning Playground presented in Figure 34. The notation of the diagram is based on Balzert (2011). The reader should note that the author simplified the sequence diagram for clarity purposes: loops are omitted or an alternative presentation chosen, and the event listener/event dispatcher concept as well as functions parameters are only depicted if they contribute to understanding. The diagram shows only calls of the most vital local functions necessary to understand the process.

The instance of the `QLearningAlgorithmModule` is created in the `interpreter.robotMbedBehaviour.js` as soon as the Q-learning block for defining the Q-learning environment is used. Depending on the map selected by the user, the function `createQLearningEnvironment()` loads one of three maps, previously prepared in SVG format by the author. Then, the `QLearningAlgorithmModule` is initialised. To avoid blocking code, the Q-learning environment is created asynchronously using the implementation of the promise concept.

During the creation of the Q-learning environment, `qLearningModule` instantiates the `visualizer`, which creates `svglookup` and `qValueLookup` objects. They serve as caching mechanism in order to optimise performance at runtime. Then, the `qLearningModule`

converts the obstacles defined by the user in his or her program to `notAllowedActions`. The path-ids are converted to actions as described at the beginning of this chapter. Subsequently, not allowed actions are filtered out from all the possible actions and the matrix of all possible states and actions holding rewards as the matrix values is created. The `qLearningModule` initialises the Q-learning problem with previously defined `statesAndActions`. With this, the processing of the Q-learning environment block finishes.

If the user used the block to set up the learning behaviour and the block that starts the learning process, the processing continues. An instance of the `QLearningAlgorithm` is created based on the already defined problem and α, γ, RHO, and NU values. The `qLearningModule` subsequently starts the learning process until the episodes are used up and the learning is finished. The data of every `qLearnerStep` as shown in Figure 33 is stored along with the corresponding optimal path in an array named `qLearningSteps`. With this, the data for all steps is already given before the user is presented with a graphical visualisation of the learning process.

```
let qLearningStep: QLearningStep = {
    newState: newState,
    nu: nu,
    qValueNew: qValueNew,
    qValueOld: qValueOld,
    rho: rho,
    state: state,
    duration: duration,
    stepNumber: this.stepNumber,
    highestQValue: this.qValueStore.highestQValue
}
```

Figure 33: Code snippet showing the structure of the data stored after each `qLearnerStep`.

Then, the `qLearningModule` initialises the `player` as an instance of the `PlayerImpl` class, which in turn initialises the `timer` as an instance of `TimerImpl` class. Upon initialisation, the `player` passes the initial values such as the start and finish node, total amount of time, and number of episodes to the `visualizer` for visualisation of these values in the Q-learning Playground. The `player` registers itself to the `visualizer` as the event listener in order to be notified if the `visualizer` dispatches events (`playerStrated`, `playerStopped`, `playerPaused`, `playerStartedForOneStep`) that come from the user interaction with the control bar on the Q-learning Playground. When the `player` receives events from the `visualizer`, it delegates them to the `timer`; this process is omitted in the sequence diagram for clarity purposes. While `timer` is running, the

`visualizer` visualises all entries from the `qLearningSteps` array, one entry each timer tick.

If the program that the user compounded contains the draw optimal path block, the `drawOptimalPath()` function is called. In this function, the `qLearningModule` creates a binding to the `timer`'s stop event, and as soon as the `stop` event occurs, the function `drawOptimalFinalPath()` is executed. This function draws the optimal path as a white–black line on the Playground. After the line drawing process is completed, the SVG file containing the drawn optimal path is transmitted to the `scene` and put as the scene background. The robot is set on the start of the optimal path, and if the user added the block to drive the optimal path, then the robots starts moving.

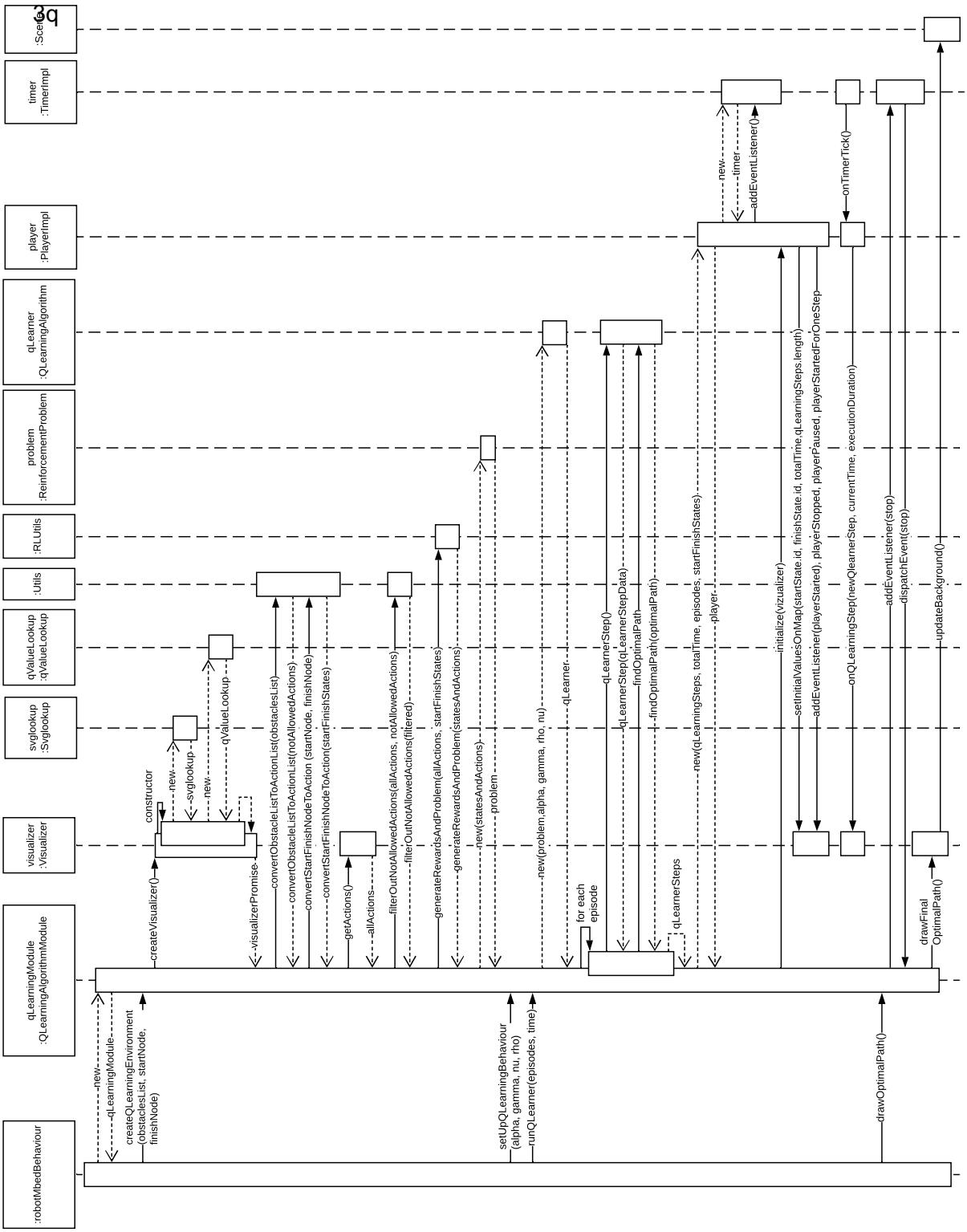


Figure 34: Simplified sequence diagram of the Q-learning Playground.

5.5 Technical Challenges

This section presents some reflections on the development process, selected technical difficulties, and solutions to these.

Initially, the author invested much time into understanding the concepts used in the source code of Open Roberta Lab, such as the API between front-end and back-end or visitor pattern for individual workflow workers. Although the wiki for the Open Roberta Lab project on GitHub platform offers some information on Blocky and the system architecture, overall, the documentation is patchy. With very few exceptions, neither the internal structure of the project source code nor the general concepts are documented. The source code documentation such as JavaDoc is missing. The only way for the author to get familiar with the source code and to be able to extend it was to explore it for herself, which was very time-consuming.

While defining new blocks as briefly described in Section 5.2.3, the author had to deal with the generation of Blockly blocks using the python script provided by the Blockly project in Open Roberta Lab. Although the script worked stably over the entire period of the research, the generator script execution aborted with an error one week before the user study. After very time-consuming research conducted after the error, the author found that the error occurred because the executed python script used the online compiler of the closure library (Google Developers, 2020b), which no longer supported the older closure library source code used by the Blockly project in Open Roberta Lab. The author solved this problem by manually adding changes to the last successfully generated source code file, which is not an optimal solution if the Playgrounds will be developed further in the future.

One of the most time-consuming tasks while implementing the Neural Network and the Q-learning Playgrounds was the proper binding of the `SVG.js` and `react.js` libraries considered for use for the Playground's implementation. Based on considerations outlined in Sections 5.3.4 and 5.4.4, the author used TypeScript and developed her code in JetBrains IDEs with native support from the node package manager (npm; npm (2020) to manage the TypeScript sources. However, in order to integrate the author's implementations into the Open Roberta Lab project, the RequireJS (require.js, 2020) framework as the JavaScript module loader was required. The porting of sources provided by the npm to the RequireJS framework was a demanding task: The author had to manually search for automatically downloaded library sources and then copy them manually inside the Open Roberta Lab project. The dependency declarations within the JavaScript sources compiled by the TypeScript compiler also needed to be adapted manually.

One of the most challenging implementations was the visualisation of the neural network link. To achieve the functionality described in the third and fifth considerations in Section 5.3.1 and 5.3.4, the link should be implemented as a slider showing the current value of the link weight and allowing the user to change the weight value by moving the slider regulator back and forth. During the implementation, the author encountered multiple issues. In the following, two of them are presented:

1) As JavaScript does not provide the native implementation of a drag-and-drop feature and the usage of external libraries was challenging and should be minimised, as mentioned in Section 5.3.15.3.4, the author had to implement this feature for the link slider herself, which was challenging. The author encountered problems in which the slider regulator did not move properly, with the user having to position the mouse exactly in the middle of the regulator in order to move it. To solve this problem, the author implemented a class `Draggable`. As outlined in Section 5.3.4, the `AiNeuralNetworkUI` object contains an instance of the class `Draggable` in which each `LinkUI` instance is registered. The `Draggable` fires on the `mousedown`, `mousemove`, and `mouseup` events the `dragstart`, `dragmove` and `dragend` events, respectively, with the corresponding `LinkUI` instance as the event source. In the first implementation, the drop action was not fired consistently, because the `mouseup` event was only fired if the mouse was inside the regulator. To ensure the proper functionality and to get the `mouseup` event in any case, the `Draggable` instance had to listen the `mouseup` event globally. Therefore, its `mouseup` events listener was expanded from the regulator element to the root document instance.
2) The next challenge was to keep the regulator while dragging at the correct position on the slider element to which it belongs. In order to find out the best matching point on the path depending on the mouse position, the author implemented the closest point algorithm. This algorithm distributes points on the path at equal distances and identifies the one with the shortest distance to the current mouse position as the result.

Another issue concerned the entire user interface of the Neural Network Playground. As the user can program the neural network to contain many nodes within one layer, the Neural Network Playground UI component can grow vertically. For this, the author considered either scaling the entire UI component to fit the screen dimensions or providing scroll options. The author decided to provide the scroll option, because only the vertical direction is affected. However, the issue of how to deal with the neural networks that are much larger than the screen remains, because the scaling will make the components small and therefore unsuitable for children.

While implementing the Q-learning Playground, the author encountered several difficulties.

1) Previously in this chapter, the author described difficulties importing the sources provided by the npm into the RequireJS framework used by Open Roberta Lab. This issue tremendously affected the choice of technologies used in the development of the Q-learning Playground. To provide performance implementation of the Q-learning Playground, the author considered using the `react.js` framework (react.js, 2020b) instead of the `SVG.js` library. The learning environment realised in the `SVG` format could be processed by the JSX (react.js, 2020a) parser offered by the `ReactJS` library. However, it turned out that the `ReactJS` library including `JSX` sources cannot be integrated into the RequireJS framework with reasonable effort. Thus, the author had to abandon the intention to use `react.js` in the Q-learning Playground implementation and proceed with the usage of the `SVG.js` library.

2) As described in Section 5.4.4, the simulated robot drives the optimal path computed by the Q-learning algorithm. In order to provide the correct behaviour, the robot must be positioned at the correct x, y coordinates and rotated to the correct orientation (Θ angle based on east-orientation). The calculation of the Θ angle implied trigonometrical calculations, which turned out to be a challenging task: In some cases, the robot was given the orientation mismatch of 180 degrees. The solution was to add π in certain ranges, depending on the start orientation of the optimal path.

3) After the implementation of the Q-learning Playground, the author conducted the manual tests via UI in order to gain the information of how the single parameters influence the optimal path. However, the test did not show any clear results. Therefore, the author decided to develop parameter tuning tests, which could be executed automatically by permutating all possible parameter values without UI. The test data was saved and analysed. The author found that the α and γ parameters have no clear effect on the results individually, but an effect of the parameters in combination with each other is clear. Indeed, both parameters seem to compensate for each other.

5.6 Summary

In this chapter, the author outlined how she addressed the limits of existing approaches to introducing machine learning for novice identified in Section 2.4. To this end, she presented solutions that she has developed for Open Roberta Lab, thereby partially

answering the second research question of how the approaches proposed in Sections 3.1.2 and 3.2.2 can be concretely implemented in Open Roberta Lab.

In order to enable the reader to follow the additions the author made to Open Roberta Lab, she demonstrated how the Open Roberta Lab project is structured. The author then described the main workflows relevant to the simulation environment of the LEGO EV3 robot, which she extended with new blocks and machine learning features.

The author presented 16 new command blocks that she defined and implemented in Open Roberta Lab. With these blocks, students can create simple neural networks and applications based on the Q-learning algorithm. In order to help the reader understand how the blocks are implemented in the ecosystem of the Open Roberta Lab project, the author described implementation details using the example of the `ai_neural_network` block.

The author then introduced her major developments: The Neural Network Playground, which enables students to tinker with neural networks and supervised learning, and the Q-learning Playground, which allows students to experiment with the Q-learning algorithm. For both the Neural Network Playground and Q-learning Playground, she outlined what technical considerations underlying these features. She then demonstrated the user interface for each and presented an overview of the new classes. With the help of the sequence diagram for each feature, the author clarified her implementations in detail.

Finally, the author presented the reflections on the development process. She presented selected technical challenges in the implementation of the Neural Network Playground, especially the implementation of the visualisation of a link. Then, she outlined the difficulties in implementing the Q-learning Playground, including the integration of external libraries, the calculation of the robot position and testing of all possible parameter combinations of Q-learning algorithm, in order to understand, how single parameters influence the optimal path.

6 Conception of the Machine Learning Materials

Based on the design guidelines in Section 4.1, the author developed a series of teaching and learning materials to accompany the machine learning extensions presented in Chapter 5 to help both educators and children grasp the machine learning paradigms that underly the extensions.

In total, the author designed a machine learning curriculum which serves as a guideline for teachers, a set of the Neural Network Cards for the introduction to the Neural Network Playground and the topic of supervised learning, and a set of worksheets and learning cards for the Q-learning Playground to introduce students to the topics of reinforcement learning. An unplugged activity to introduce unsupervised learning with the k-means algorithm closes the series.

Although the Neural Network Cards and Q-learning Cards are structed activities which help the young students get started, the aim was for them to serve as a steppingstone, not a final destination. The materials should enable the participants to play with machine learning technologies and make something that interests them, in accordance with the ideas of constructionism (Michaeli et al., 2020; Papert & Harel, 1991; Queiroz et al., 2020).

This chapter introduces the materials which the reader finds in their original size in Appendices A.6 and A.7. It should be noted, that the materials were developed in German, as they are to be used with German-speaking children. For the graphic design of the materials, the author took inspiration from explanations of machine learning topics in children's books, as described in Section 4.1.3 and based on Castella (2018) and the D4CR Association (2020).

6.1 Machine Learning Curriculum

A machine learning curriculum consists of a lesson plan based on four modules and a presentation. The lesson plan, which is attached to Appendix A.4, is tailored to six school lessons of 45 minutes, from which the teacher can shorten or widen it if required. The aim of the machine learning curriculum is formulated as follows (Appendix A.4):

> Students investigate how robots learn, think, and feel. They investigate what is meant by the term "artificial intelligence (AI)" and learn about three main areas of AI – supervised, unsupervised, and reinforcing learning – in a practical way. They explore when people describe a machine as intelligent and strengthen their newly acquired knowledge by dealing with the development and configuration of simple, descriptive AI applications in Open Roberta Lab.

Module 1 “How does your robot learn?” – Introduction to Artificial Intelligence and Machine Learning

The lesson starts with getting to know each other and a discussion in plenum. The teacher questions the students about the how the machines learn: “Do the machines learn at all?”; “If you have already written a program, does that mean that the computer/your robot has become smarter?”

After the discussion, the teacher performs a Braitenberg experiment using a Calli:bot robot as described in Section 3.1.2 and asks the children again “Is this behaviour of the robot intelligent?” After the discussion, the teacher holds a short input lecture introducing artificial intelligence and machine learning. The teacher pays particular attention to illustrating the topic with examples that have a connection to the children’s everyday lives. He or she then briefly introduces the three areas of machine learning – supervised, unsupervised, and reinforcement learning – and starts deepening the supervised learning topic.

Module 2 “Teaching your robot” – Introduction to Supervised Learning and Neural Networks

The educator explains that in order to teach the robot something, the children have to train its neural network. He or she then explains what the neural networks are and how they can be trained. For this, the educator can use examples provided in the presentation (Appendix A.5). Figure 35 shows one such example.

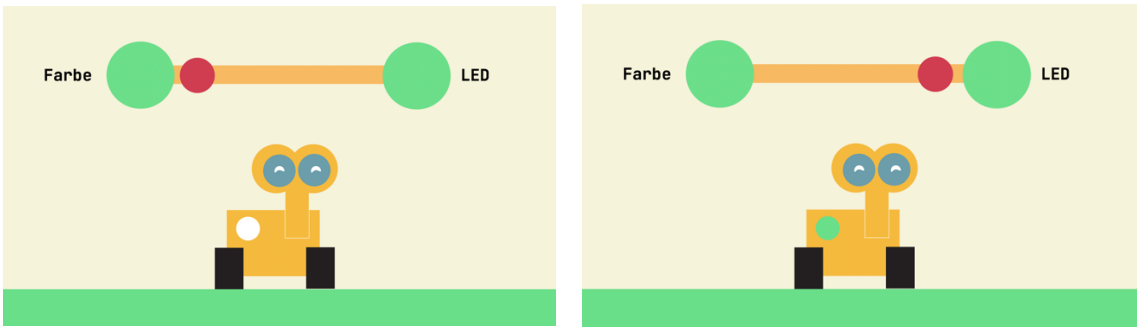


Figure 35: Simple illustration showing the basic functionality of the neural network on the AI-robot.

In Figure 35, the author reduced a neural network to the essentials – an input neuron, a link, and an output neuron. The idea is to show the causal relationship of how the input neuron influences the output neuron by moving the regulator back and forth. The robot in Figure 35 is placed on the green surface, and its colour sensor recognises the green colour. However, as shown in the left-hand illustration, the LED does not light up. The reason for this is that the value of the link is 0. If the regulator is moved to the right, as

shown in the right-hand illustration, the value of the link changes to 1 and the LED lights up.

After this brief introduction, the students are supposed to take up the role of teacher for the simulated robot. The children receive the Neural Network Cards and can work with them alone or in groups. The teacher goes around and helps where necessary.

Module 3 “Let your robot learn from experience” – Introduction to Reinforcement Learning

After the lunch break, the children are introduced to reinforcement learning. In order to maximally retain students’ attention, they are asked to keep their laptops closed during the introduction, as it is crucial for them to understand how the Q-learning algorithm works and what the task is before beginning to program the Q-learning algorithm. The teacher distributes the information and worksheets and lets the children read them. Before the children open their laptops, the teacher conducts the first trial in plenary, so the children have an example of how to document the first experiment in the Q-learning Card. The teacher may use the slide from Figure 36 for this. The description of the exercise can be found in Section 6.3.

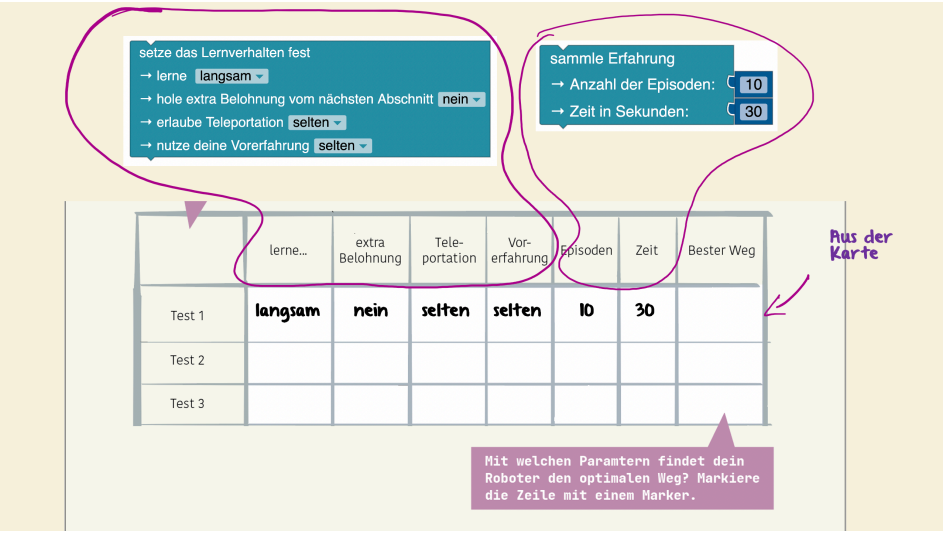


Figure 36: Slide that explains filling out the Q-learning card.

The children then have time to play with the parameters of the algorithm on the Q-learning Playground and to conduct the experiments. The results are discussed after the children have finished the trials.

Module 4 “Can robots learn autonomously?” – Introduction to Unsupervised Learning

In the last 40 minutes, the teacher introduces clustering and the k-means algorithm, showing an experiment and letting the children actively take part. The curriculum also

includes links to further instructional materials. If time remains, the teacher can use external instructional materials to further explore machine learning topics.

6.2 The Neural Network Cards

While designing the learning materials accompanying the Neural Network Playground, special attention was paid to the high consistency among the Neural Network Playground, learning materials and the teacher's presentation, so that the children can easily recall and transfer their knowledge.

Following the constructionistic ideas for designing the task as a problem-solving activity (Kandlhofer et al., 2016), the author developed a set of nine double-sided learning cards in the DIN-A5 format. Figure 37 shows an overview of the result. The complete set showing the Neural Network Cards from their front and back side is attached in their original size to Appendix A.6.

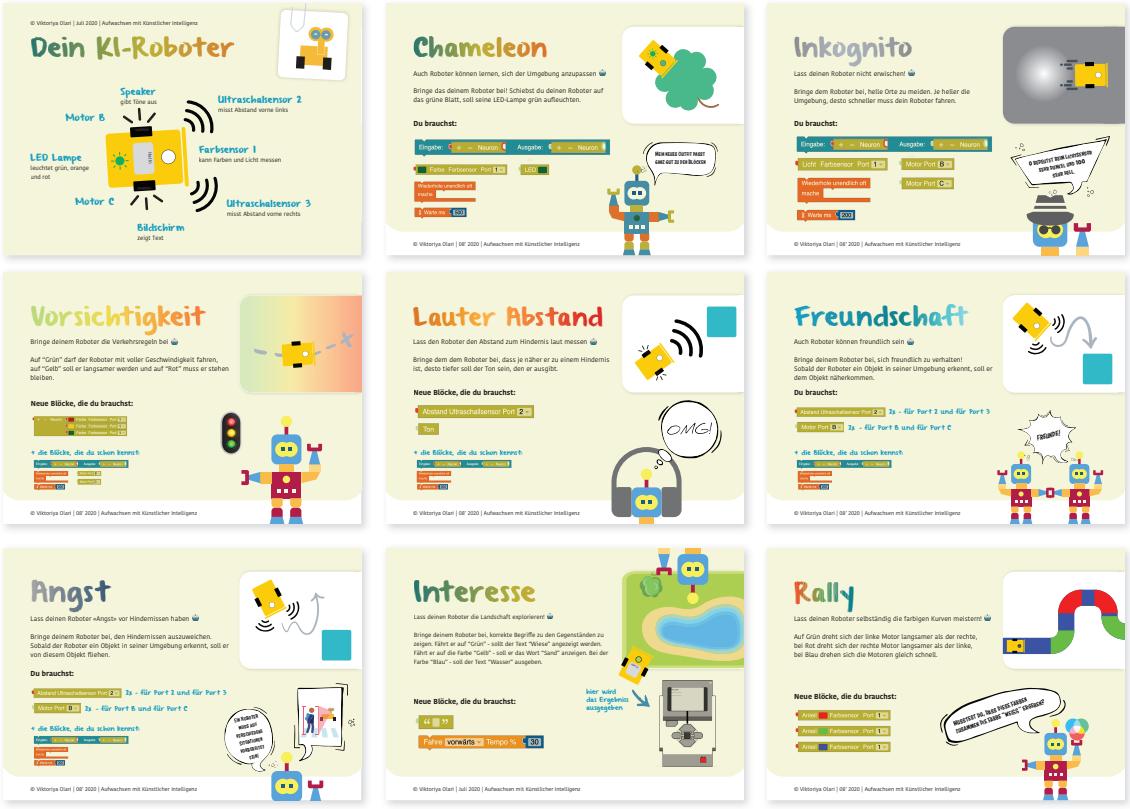


Figure 37: Overview on the Neural Network Cards (front sides only).

The cards are designed in comic style, for as Castella (2018) points out, characters and stories in comic style give kids a starting point for their imagination and motivate them to be creative. This is noticeable in the colours, selected fonts, and character designs. The colours are bright, optimistic, and playful: The background varies from soft white to mellow beige, and block colours are cosy and warm, yet vibrant. Adding the hand drawn illustrations and sketching the neural networks as shown, for example, in

Figure 38, evokes whimsical humour and emotions. Following the storytelling approach, the tasks and hints in the learning cards are problem based and written as short stories in order to be appealing to the student.

The cards adapt the principle of “low floors and wide walls” introduced in Section 4.1.3 by enabling the student to start in a straightforward manner. However, the level of difficulty increases with each card. With the first learning card, which is introductory, the child meets the central figures of the learning cards: the AI robot and its helping friend – a robot that is always there to lighten the atmosphere and give hints. On the front page, the AI robot presents itself and its current setup. On the back, the helping friend refers to the configurations of Open Roberta Lab, which are necessary for the further work with the learning cards.

Eight subsequent cards each deal with one topic. Table 3 gives an overview of the structure of the learning cards and the variety of input and output neurons used in each card.

Table 3: Overview of the structure of the learning cards and the corresponding input and output nodes.

Learning Card	**Input**	**Output**
Chameleon	1 x colour, colour sensor	1 x LED
Incognito	1 x light, colour sensor	2 x Motor
Caution	3 x colour, colour sensor	2 x Motor
Loud Distance	1 x distance, ultrasonic sensor	1 x Sound
Friendship	2 x distance, ultrasonic sensor	2 x Motor
Fear	2 x distance, ultrasonic sensor	2 x Motor
Interest	3 x colour, colour sensor	3 x Text
Rally	1 x R channel, colour sensor 1 x G channel, colour sensor 1 x B channel, colour sensor	2 x Motor

Each card is similarly structured. Figure 38 shows the front and back of the Neural Network Card “Friendship”.

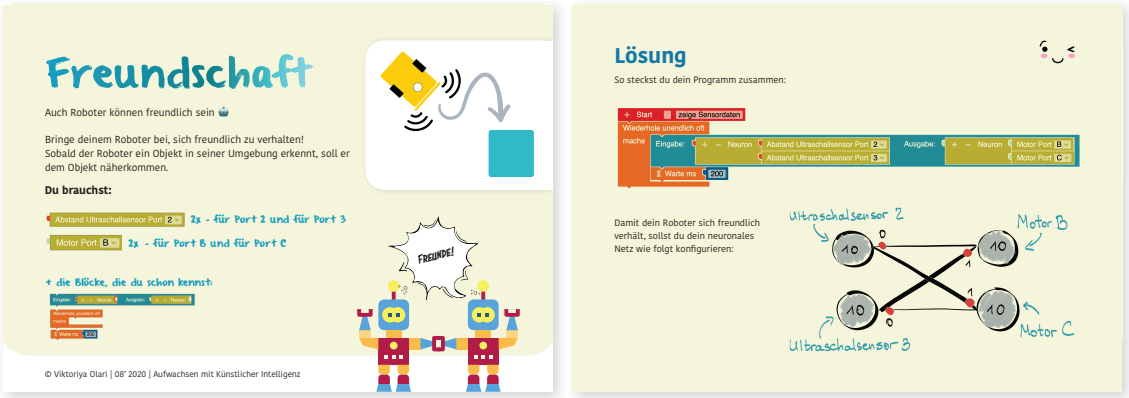


Figure 38: Front and back side of the Neural Network Card "Friendship".

On the front, the student finds a simple task description and a quick preview of the desired behaviour. For instance, in the learning card “Friendship”, the robot should move towards the blue square. The front side also contains the blocks that the student should use to fulfil the task, with their number decreasing on every next learning card in order to achieve increasing difficulty. The back of the card presents the solution – the final program and the correctly configured neural network, supported by a short comment. The back side of some cards contains additional tasks.

6.3 The Q-learning Cards and Supporting Worksheets

The Q-learning Playground is accompanied by a series of learning materials, including information sheets and worksheets. If the children work each for themselves or have questions, they can consult the information sheets. Worksheets help them get started with the Q-learning Playground and provide them with a straightforward structure for conducting the first experiments.

The aim of instructional materials is to support students understanding of how the Q-learning algorithm works by making it comprehensible and tangible. The instructions and problem-based tasks help students to gain the practical understanding that changing the parameters of the algorithm influences the learning environment and learning behaviour of the robot. With this, the students also gain theoretical knowledge of the functioning of the algorithm and understand what the optimal path is and on which criteria it is drawn.

While playing with the algorithm and observing and evaluating the learning procedure, the students put themselves in the agent’s shoes. They immerse themselves in the behaviour of the simulated robot, and in doing so, they gain insights and look behind the scenes. They discover how the robots are trainable and that they are not perfect.

The information in worksheets has been formulated as short stories whenever possible and was adapted with young students in mind. The tasks are based on the problem-

solving methods. Figure 39 shows the information sheets and worksheets, and the following overview explains the individual materials in more detail.

1) Program: Let your robot learn (Appendix A.7.4) – explains how to program applications based on the Q-learning algorithm with blocks and gives some hints that the user can experiment with.
2) Flow diagram: This is how your AI-robot learns (Appendix A.7.5) – introduces the Q-learning algorithm step by step.
3) Q&A: Reinforcement learning (Appendix A.7.2) – summarises possible questions and answers that the student may have about the operation of the Q-learning algorithm and reinforcement learning. The explanations in the Q&A are written in short, simple language, taking into account possible difficulties of the student.
4) Observation card (Appendix A.7.6) – is a worksheet that the students fill out. It serves as a basis for reflections on how the Q-learning algorithm works and whether the child believes that there are optimal parameters.
5) Map (Appendix A.7.3) – illustrates the Q-learning environment and all elements that the user sees and can interact with. The statistics and navigation bar are explained in detail.

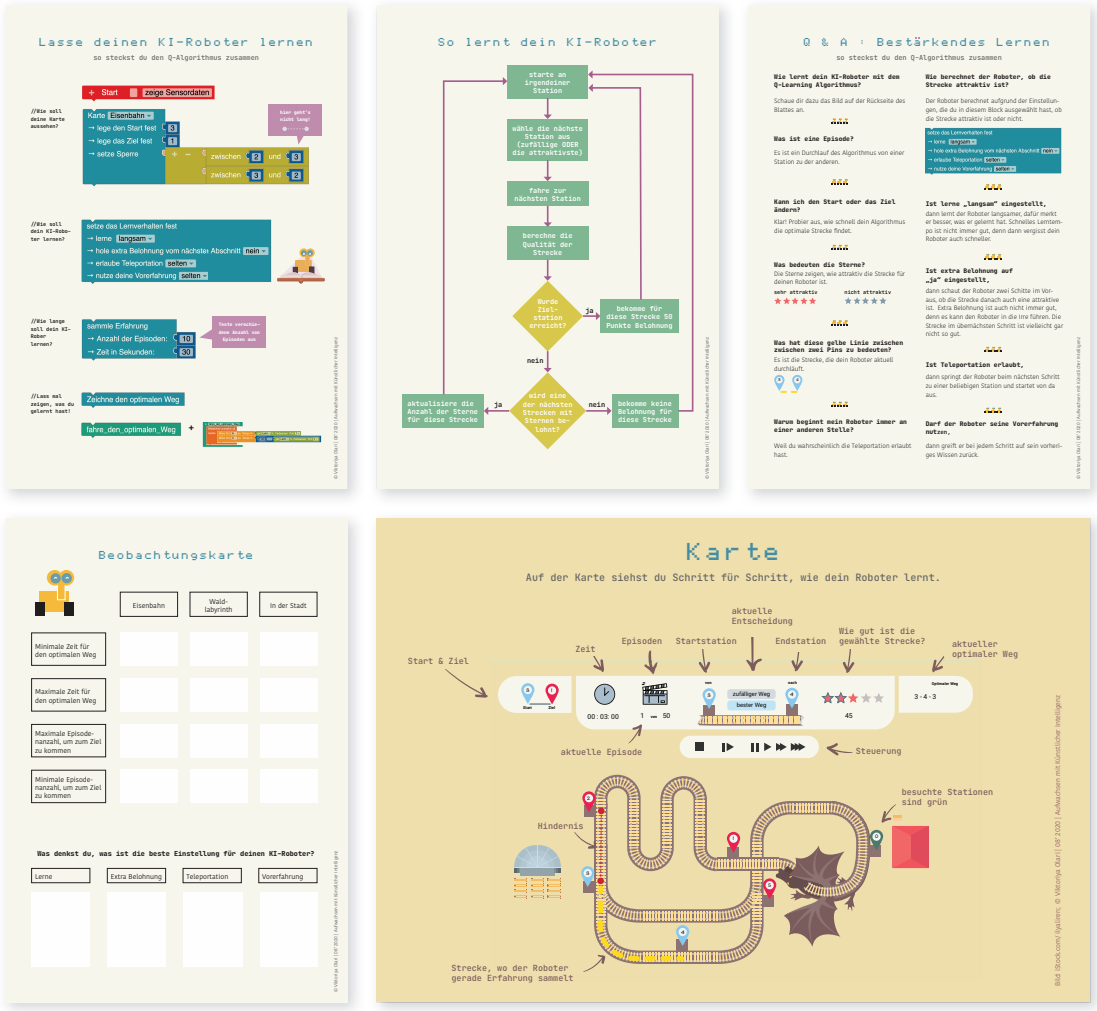


Figure 39: Instructional materials for the Q-learning Playground.

Furthermore, the author developed three Q-learning Cards (Appendix A.7.1) corresponding to the respective environments described in Section 5.4.3. The Q-learning Cards support students in conducting the experiments with the Q-learning algorithm. The students document the parameters they set in the Q-learning algorithm for each iteration and describe their observations of the optimal way that the robot found. The idea of experiments is based on a concept called hyperparameter tuning or optimisation, which refers to the selection of the best values to minimise or maximise the given function (Das & Cakmak, 2018). In the first learning card, the students are asked to complete three trials, and in the second and third, five trials. Figure 40 illustrates the cards.

Figure 40: Q-learning Cards.

Each of three Q-learning Cards has a similar front and back. The front illustrates the Q-learning environment, accompanied by the start state, finish state, and obstacles that the child can use for the trials. The back includes a table, in which the students take observation records. The students can also record their guesses for the optimal path before they begin conducting the trials.

Figure 41 shows the back of the Q-learning Card for the first learning environment “Railway”.

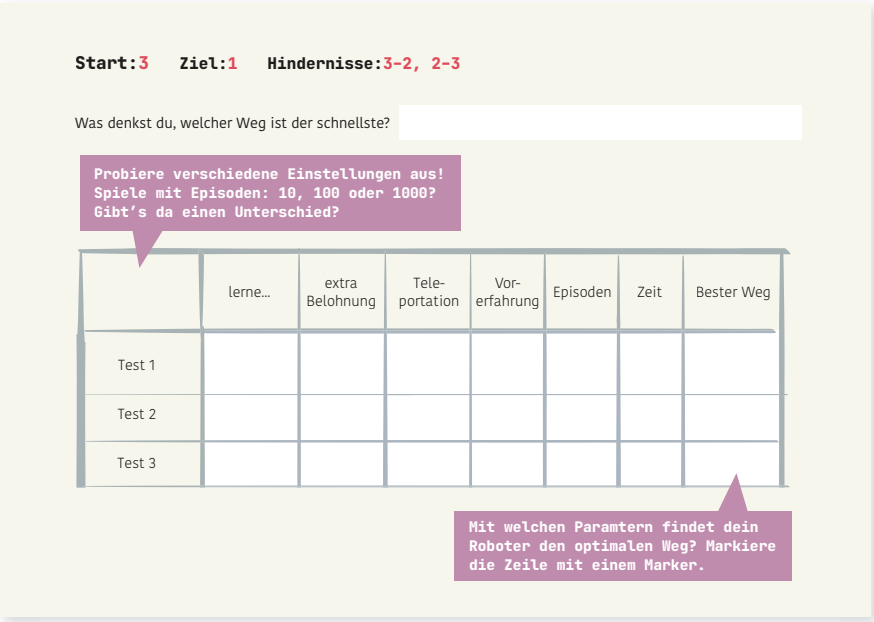


Figure 41: The back of the Q-learning Card "Railway".

In the table, the student can note following values: α, γ, NU, RHO, episodes, and time. After the learning process is finished and the robot drives the optimal path, the students can record this path in the last column.

Although the Q-learning materials are characterised by their uniform design with soft colours, colourful pictures, and hand-drawn illustrations, the design of the instructional materials for the Q-learning Playground differs from the design of the materials for the

Neural Network Playground: Both the fonts and the colours and characters are different. This is intentional, because the materials cover distinct topics.

6.4 Unplugged Activity Introducing the K-means Algorithm

Introducing the topic of unsupervised learning with the k-means algorithm is designed as an experiment and an analogous exercise. The unplugged activities are recommended by several studies (Seegerer et al., 2019; Wong et al., 2020), and unplugged activities to teach machine learning concepts are not uncommon (Jatzlau et al., 2019; Michaeli et al., 2020). This section describes the activity procedure based on the k-means algorithm outlined in Section 3.3.2.

The facilitator first displays a set of items and asks students how they would group the objects. Figure 42 shows the first illustration.

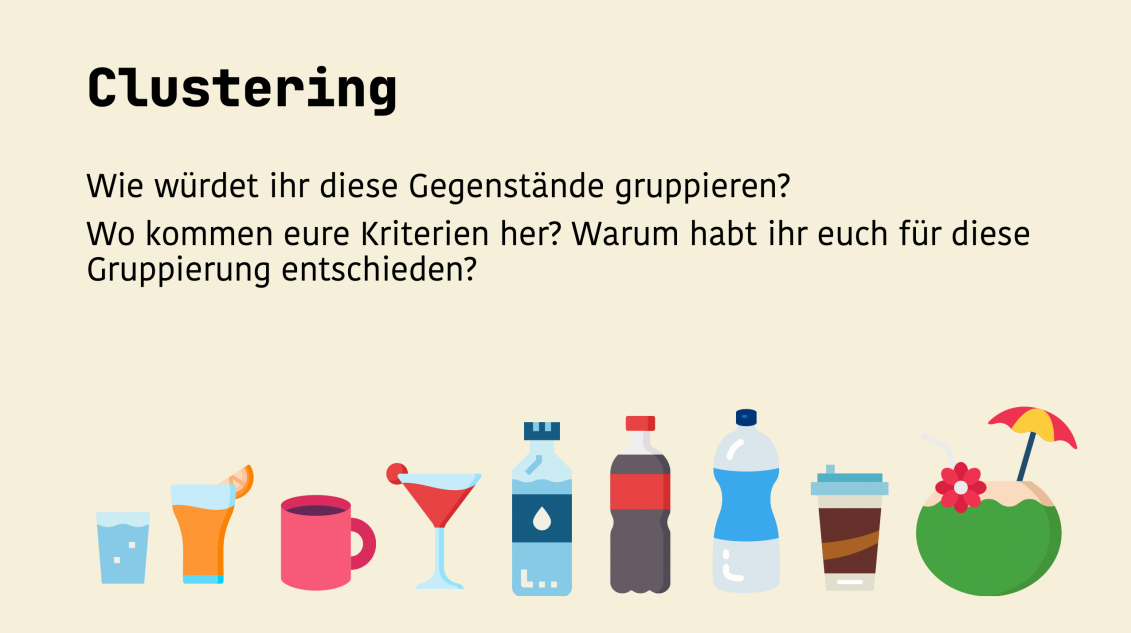


Figure 42: Clustering – the introductory slide.

After a brief plenary discussion, it becomes apparent that there are numerous possible criteria for grouping items: for example, by colour, material, content, or form. The facilitator poses the next questions to the students: “Where do your criteria come from?”, “Why did you choose this grouping?”

After the questions have been discussed in small peer groups and then in plenary, the educator suggests putting him- or herself in the role of a robot and sorting the items as the robot would do it. The children should pay particular attention to what criteria the facilitator used to sort the objects and for the sorting method – how exactly did the teacher sort the objects?

The facilitator sorts the objects according to the k-means algorithm. He or she first sticks a few post-it notes on random objects to form cluster centres. The items remaining in the set are then compared with each cluster centre according to a criterion known only to the facilitator. After comparing each item with the cluster centres, the facilitator places the items behind the cluster centre that he or she believes fits best.

When all items from the set have been sorted into the cluster centres, the teacher discusses with the children what they think the criterion for the sorting process was. The clustering procedure is then reviewed in plenary, and the teacher explains the k-means algorithm in detail. He or she presents Figure 43, which depicts the k-means algorithm step by step.

Figure 43: The k-means algorithm, step by step.

The children are then invited to select a criterion on their own and to sort the items according to the k-means algorithm in peer groups or in the front of the class.

7 Evaluation

This chapter outlines the results of the user study conducted from 5 to 7 August 2020 at the Heinrich-Hertz-Berufskolleg in Düsseldorf. It first presents the setup and participants, then provides insight into the procedure and presents the results of the questionnaire and the students' feedback.

7.1 Setup

The following preparations for the testing of the Playgrounds and teaching material were undertaken:

1. All extensions were successfully finished by the time of the test days. Open Roberta Lab was installed on a Raspberry Pi so that the Lab could be operated in a local network. A WiFi-router was installed and used as the WiFi-hotspot to provide access for the students' laptop clients to the Open Roberta Lab webserver. The second photo from Figure 44 shows this setup.

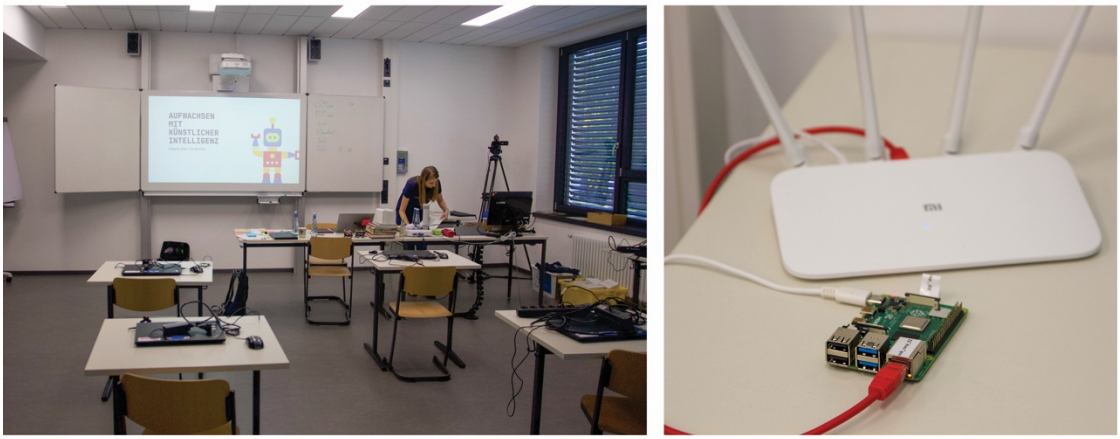


Figure 44: Classroom and hardware setup for all three sessions.

2. It was agreed that ZDI would provide all participants with the necessary hardware. Each participant had access to a laptop. The classroom had all the essential technical equipment, such as a projector, a sufficient number of sockets, and WLAN access. The first photo in Figure 44 depicts the classroom and hardware setting.
3. The day before the first session, all laptops were checked and set up for testing. For a smoother session, the web link to Open Roberta Lab and the link to the final questionnaire were saved in the favourites bar of each client's web browser. A folder with the backgrounds required for experimenting with the Neural Network Playground was downloaded to the desktop of each laptop. A short last test was performed to check that the Open Roberta Lab application remained stable even

if all laptops accessed it simultaneously during the compilation of their programmes.

4. Due to measures implemented in response to COVID-19, the materials used by the children in one session could not be reused in another session. They therefore had to be printed out individually for each child in each session.
5. The documentation equipment – two cameras with tripods and a clip-microphone – was borrowed privately.

7.2 Participants

A total of 24 children participated in the user study. On the first day, the extensions were tested by the high school children – seven boys in grades 7–9. On the second day, the primary school children – five girls and four boys in grades 3–4 – examined the extensions and materials. On the third day, the middle school children – one girl and six boys in grades 5–6 – participated in the study. Figure 45 shows the individual classroom settings.

Figure 45: Classroom setting for grades 7–9, 3–4, and 5–6.

All children had previous knowledge of working with Open Roberta Lab and real LEGO EV3 robots, as they had participated in the introductory session the day before.

The pre-assessment showed that some of the children had a vague idea of what AI is. Most of the primary school children could not define AI at all. One of the primary school students said, “Artificial intelligence are robots”[21]. In comparison, the answers of the middle and high school children were more differentiated:

- “AI – a being that can have its own personality and can make its own decisions” (high school student)
- “AI is when the machines themselves solve problems” (high school student)
- “Artificial intelligence is a new method of programming new machines” (middle school student)
- “AI is as intelligent as humans are intelligent, only for computers and robots” (middle school student)
- “AI is how the robot thinks” (middle school student)

[21] All citations of children were originally in German and were translated by the author for the purposes of this work.

Only one student associated AI with learning: “AI is a program that learns from its own mistakes and becomes smarter by itself” (high school student).

7.3 Insights in the Procedure

Each session lasted six school hours (one school hour = 45 minutes) and was conducted in a block with short breaks. The author led all three sessions. On the second and third days, four trainees observed the sessions. They were allowed to support children if they wished. Due to limitations described in Section 4.2, the children were not allowed to work in tandem or in groups. Therefore, all activities were limited to individual work.

The children’s knowledge of machine learning and AI was pre-assessed. Then, a Braitenberg experiment was conducted and discussed with the children. Figure 46 shows the author conducting the third experiment after Braitenberg (1986) that is called “Love” or “Friendship”. Calli:bot, the robot used by the author, is attracted by the objects the author presents on the table. The greater the stimulus, that is, the closer the object is, the lower the speed of the robot.

Figure 46: Author conducting the third experiment after Braitenberg (1986) with a Calli:bot robot.

Most children of all grades were confident that the robot did not behave like an intelligent creature. One of the middle school children said, “I don’t think it is an intelligence, because it only drives because you have programmed it to do so”. Another middle school student remarked that the robot was not intelligent because of his previous experience with the robot: “I don’t believe it is an artificial intelligence either, because we did something similar yesterday”. Both primary and high school children were also critical. One child said, “It’s not intelligent by not crashing into the box!” (primary school student), another explained his point at length:

> The robot has the equipment for it [autonomously driving], but it was just too stupid to know what to do with the sensors etc. Intelligent beings have to learn first,

> learning is the keyword, but they already have eyes, nose, mouth, everything – and they know roughly what to do with them. It [the robot] did not know how to use them" (high school student).

However, there were also doubtful voices: "So [. . .] it depends [. . .] If you have programmed it, it is not really intelligent. If you programmed only a part of it, then it is intelligent" (primary school student) and "Yes, it is intelligent enough to stop in front of something before it collides with it" (secondary school student).

After the discussion about the experiment, the modules were completed in order. In the first module, "How does your robot learn?", the author held a short input lecture that introduced AI and machine learning. Directly after the first module, the second module, "Teaching your robot", was undertaken. As planned, the children taught the robot various behaviours via direct supervision. In all three grades categories, everything went according to the plan described in Section 6.1. The children discovered hands-on components of neural networks such as nodes, layers, connections, and weights. All children in all age groups started with Neural Network Cards but were then encouraged to contribute and test the limits of what they could teach the robot.

Children from primary school coped well with the topic but had to be supported much more intensively than children from other school grades. During the session, the author had the impression that the children did not fully understand how direct supervision works. Although the idea of a sensor directly connected to an actuator is simple and was quickly understood by the students, the author had to explain in detail several times the whole process from the blocks to the effects of changing weights in the neural network on the behaviour of the robot. Some of primary school students also had motor difficulties using a mouse, and it was thus not easy for them to manage all of the workflow process outlined in Sections 5.3.2 and 5.4.2. Most of the middle school children and all high school children coped well with the topic on their own, and some criticised the complicated workflow process. The boys in the middle and high school grades were very competitive and were impatient. They wanted to complete the tasks on the cards as quickly as possible. This sometimes led to restlessness and a charged learning atmosphere. The author had to refer them several times to additional tasks that they had failed to complete due to inattention.

The vast majority of children stumbled over the two neural network cards "Fear" and "Friendship". Although these cards had completely different task descriptions, they depicted the same blocks that were necessary to solve the task. The students wondered how the same blocks could lead to opposite behaviour. When they turned the cards over,

they found that it was the configuration of the neural network that led to completely different behaviour in the robot.

The Neural Network Card “Rally” was an immensely enriching experience for all of the children. The author erred in the illustration of the neural network configuration, which led to a situation in which the children could not look for the right solution on the back of the card, and they had to master the correct training entirely on their own. The children of all grades spent much more time on this task than expected and were very excited when they were able to train their network accurately.

Overall, the children did not have much time to tinker with personal projects, as the task with the Neural Network Cards occupied all their time. Two children from primary and middle school created projects that went beyond the Neural Network Cards. Figure 47 shows some impressions from the second module and the children’s creations. The first picture on the right illustrates an example of the project, in which one student from the middle school group went beyond the tasks from Neural Network Cards and experimented on his own. The idea of his project was for the robot to drive autonomously through the area while simultaneously turning its LED to the colour detected by the colour sensor. He further developed his project so that the robot could display additional text while driving. Two photos in the bottom right corner show some insights in the brief input lecture to neural networks and training the networks via direct supervision. The remaining photos show children working with Neural Network Cards.

Figure 47: Documentation of the impressions from the second module and creations of the children.

In the third module, “Let your robot learn from experience”, the children were introduced to reinforcement learning and the Q-learning algorithm. After the students in the

first session were distracted by the opened laptops with Open Roberta Lab during the introductory phase, the students in the second and third sessions were asked to keep their laptops closed. On the second and third days, the author also changed the order of the modules because the experience from the first day indicated that reinforcement learning was too difficult to be introduced in the afternoon. Reinforcement learning was then carried out directly in the morning. These measures worked well, and children from the second and third session were attentive.

Before the children had started with the practical part, the result of the Q-learning algorithm – the optimal path – was problematised. The children's opinion of what the optimal path for humans is and what the optimal path for the robot would be was discussed. All three groups of students came to a similar conclusion: The optimal path for the robot from point A to B is the path that the robot can travel in the minimum amount of time. However, the optimal path for a person may vary depending on various factors such as whether there is an ice cream parlour on the way or the beauty of the path.

As planned, the children explored the Q-learning algorithm using Open Roberta Lab and analysed how a robot learns through rewards. The children from all school levels were able to create unique learning environments for the robot and experiment with the parameters of the algorithm. They all observed and analysed the learning and reasoning process on the Q-learning Playground step by step. The primary and middle school children were very diligent in experimenting and documenting their experiences with reinforcement learning on the observation cards. In contrast, the older students had hardly used the materials. Most quickly changed the parameters, wanting to observe the result and readjust the program as quickly as possible, if necessary. The author also had the impression that the children sometimes focused on less important things, such as how the learning agent moves in the Q-learning Playground, rather than looking at the navigation bar and observing how it really learns and how the statistics change step by step.

Figure 48 shows how the children explored and experimented with the Q-learning. In the top-right photo, the child creates a unique Q-learning environment and explores how the algorithm operates. The first photo on the left shows the introductory phase. The other pictures demonstrate the work in progress.

Figure 48: “Let your robot learn from experience”: Documentation of the third module.

The author did not have to provide much support for any group of children. The students’ questions related to clarifying the task or the process flow. Primary school children coped just as well with the exploration of the Q-learning algorithm as middle school and high school children. Sometimes, the students experienced cases in which the robot failed in learning and could not find a way to the target. Then, they were visibly disappointed and tried to correct the algorithm in the next iteration. Overall, the students of all school grades were inquisitive and motivated to explore.

In the last module, “Can robots learn autonomously?”, the children were introduced to the k-means algorithm through the unplugged activity. The author prepared a set of several vessels and presented them on the table. As planned, she discussed how the robot would group the objects presented on the table without any previous knowledge. She then sorted the vessels according to the k-means algorithm without explaining what criterion she used for grouping.

The children made assumptions about the grouping criterion. After discussing the grouping criterion and explaining the sorting principles, the students grouped the items themselves and let others guess the grouping criteria. Overall, all children of all school ages participated very actively in the discussion. Regardless of the school grade, they could cope with the topic without any particular difficulties.

Figure 49 shows the unplugged activity. In the first photo, the vessels are not yet grouped. The second photo shows the grouping process that was conducted in the plenary, and the third photo shows the result of the clustering.

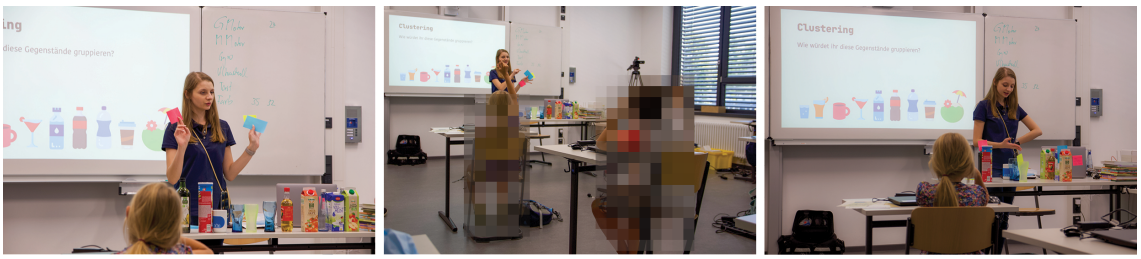


Figure 49: Exploring k-means clustering in an unplugged activity: Documentation of the fourth module.

At the end of each session, the children filled out short questionaries and gave oral feedback on the session.

7.4 Feedback and Questionnaire

Overall, children of all school grades were motivated during the sessions and used their time until the end of the sessions and even beyond to tinker with the tasks. The observer noticed that the children's attention was high during the entirety of the sessions and that they were all firmly committed to their projects. Several students praised the illustrations in the presentation and the design of learning materials. Many asked if it would be possible to continue working on their projects from home.

The questionnaire examined whether the learning experience with the extensions developed for machine learning promotes children's understanding of the underlying concepts of machine learning. The students could answer how interesting and how difficult they found the particular topics. At the end of the questionnaire, the children were also asked what they thought AI and machine learning are and whether they were motivated to continue working on machine learning. Insights into the results are given below.

Figure 50 shows the distribution of the questionnaire results. The x-axis illustrates the total number of responses. The y-axis shows three topics divided by the class grades. The first topic, from module 1, is not considered, because it was only an introductory unit. The graph displays the absolute number of the answers, with 0 representing the middle of the scale, that is, the value number 3.

In the descriptions that follow directly after the graph, the scores are averaged for each school grade and topic. The scores reflect the tendency of the answers for each topic per group of children: the higher the number, the more positive the children's evaluation (see Section 4.3). The descriptions also provide insights into the observations that the observer had made on the respective topics.

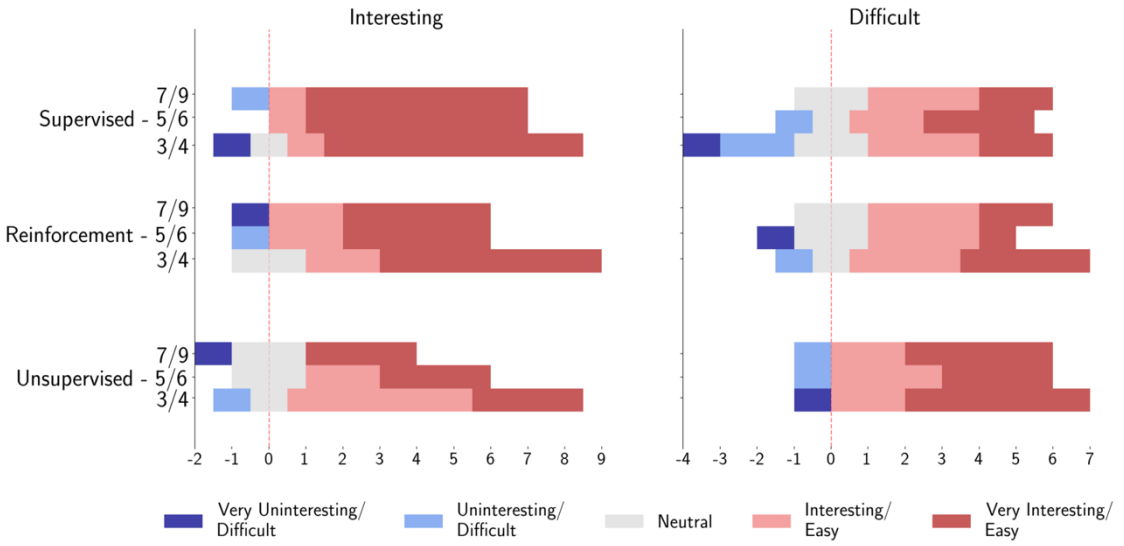


Figure 50: Participants' attitudes towards the topics supervised, reinforcement, and unsupervised learning.

7.4.1 Perception of Supervised Learning

The topic of supervised learning from Module 2 was the most difficult one for the primary school children, with an average score of 3.3. Children from middle school perceived it to be easier, with an average score of 4.28, followed by the high school students, at 4.0. The middle school children also found the topic of supervised learning to be the most interesting compared to other groups. The average score here for grades 5–6 was 4.57, followed by grades 3–4 at 4.3 and grades 7–9 at 4.0.

The observations suggest that the children of all grades were engaged and motivated by tinkering with neural networks and teaching the robot through direct supervision. Most of the children completed only the task with the Neural Network Cards. Only a few children then had time to tinker with applications based on their ideas. The feedback from the students in middle and high school was that the explanations were easy to follow. They also recommended improving some points in the user experience, such as the design of the simulation backgrounds and button locations.

7.4.2 Perception of Reinforcement Learning

The participants of all age groups found the topic reinforcement learning as they programmed the Q-learning algorithm to be interesting to very interesting. The average score of participants from grades 3–4 was 4.4 and those from high school 4.14. The middle school children found the topic to be the most interesting, with an average score of 4.42. However, at the same time, they found reinforcement learning to be the most challenging, with an average score of 3.42 for difficulty. High school children perceived the topic with 4.0 points more difficult than the primary school children with 4.2 points.

The observer stated that the children of all age groups spent very different amounts of time creating learning environments. Some children spent much time creating increasingly difficult environments, while others were interested in testing. The older children had less motivation to carry out the experiments and were often more distracted than the middle and primary school students.

7.4.3 Perception of Unsupervised Learning

The greatest level of interest in the topic of unsupervised learning, introduced by the unplugged activity, was shown by middle school children, with an average score of 4.14. The lowest level of interest was shown by high school children, at 3.71, followed by primary school children, at 4.0. The average score for difficulty varied from easy to very easy in all three groups: 4.4 for primary school, 4.14 for middle school, and 4.28 for high school children.

The observer noticed that the children were attentive while the facilitator conducted the experiment. They also actively participated in the discussion about the experiment afterward.

7.4.4 Student Motivation and Feedback

On average, 75% of the participants indicated that they would continue to work on the topics, and 25% indicated that they might want to continue working on the topics. None of the children gave negative feedback by indicating that they did not want to continue working on the topics. The distribution of responses varied considerably across age groups. While 100% of high school students responded that they would like to continue working on machine learning, only 60% of the primary school children indicated this, with 40% indicating that they might want to continue working on AI and machine learning. The middle school children were between these groups: 71.4% answered that they would like to continue working on the topic, and 28.6% answered “maybe”.

Overall, the feedback from the participants at the end of the sessions was highly positive. In the feedback round, almost all children reported that they had an enriching session and had fun. When asked which topic the children liked best, the children’s answers were divided between supervised and reinforcement learning. Only one student, from the high school group, indicated that unsupervised learning was the most exciting topic: “I liked unsupervised learning best because you could observe how AI solved problems on its own”.

One participant explained his experiences with reinforcement learning: “I found reinforcement learning to be very interesting, mainly because it improves by checking which way is the better one. [. . .] AI is a bit more complicated than I thought, is really something

that big . . . can be tricky". Another participant reflected on his experiences with the supervised and reinforcement learning and referred to the moment when the robot could not find its way out despite its knowledge: "So, I take it from this day . . . I take all these ways with me [. . .] I still can't describe [. . .], but it's in any case, it's independence and that it [robot] can do something by itself without help, yes and also as an example it can say 'no', which everybody is afraid of". There was also some critical feedback. One high school participant remarked, "I didn't like the topic with supervised learning so much because I have the feeling that the tasks could also be solved with 'if-then' queries".

In the open question of what the children took from the session and what AI and machine learning are, the answers were more differentiated than in the pre-assessment. Although there were general answers such as "It is a very extensive and interesting area" (high school student), "AI is what humans program in robots" (primary school student), "The brain of the robot" (middle school student), "Things that are invisible" (middle school student), and "I have learned a lot about AI, for example, that you can even find it in the online shop" (middle school student), there were also more answers associating AI with learning, including the following:

- "Artificial intelligence is artificial learning" (primary school student).
- "AI is not smart until you start it, then it gets smarter" (high school student).
- "A programme that learns independently and makes independent decisions" (high school student).
- "AI is a programme that solves problems and accomplishes tasks independently" (high school student).
- "It is fake intelligence" (primary school student).

7.5 Summary

In this chapter, the author presented the user study that she conducted with 24 children from primary, middle, and high school in order to test the machine learning extensions presented in Chapter 5 and the teaching materials proposed in Chapter 6. With this, she addressed the third research question on how the developed concepts appeal to students of different school grades and whether they need help in understanding the proposed concepts.

The author gave insights into the setup, the groups of participants and the procedure. Then, she presented the results of the evaluation, in which she examined how the children of different ages perceived the topics and whether they had difficulties in understanding them.

Overall, the children of all age groups perceived the topics very easy to moderately hard to grasp. Younger students noticed the direct supervision challenging, whereas Q-learning and k-means algorithms were much more accessible. The vast majority of high school children could cope with all topics without particular difficulties.

The author also presented oral feedback from students on the sessions. Overall, the students perceived the sessions positively, and the majority of the students would like to continue work on the topics. Compared to the beginning, many more students associated the term learning with the term AI at the end of the sessions.

The observer noticed the high motivation of the students to train the neural networks and less motivation to document the experiments with the Q-learning algorithm. During the module on unsupervised learning, all groups of students actively participated in the unplugged activity on the k-means algorithm.

8 Discussion

Based on the curricular requirements for the introduction of machine learning in schools presented in Section 2.1 and the gaps summarised in Section 2.4, this thesis aimed to propose, implement, and evaluate new possibilities to open the black box of machine learning from a technical perspective for students of different ages. These approaches should reflect the thematic complexity and breadth of the field.

Overall, it can be concluded that the author has succeeded in this endeavour. The author developed, realised, and evaluated three possibilities for the introduction of machine learning: the Neural Network Playground, presented in Section 5.3; the Q-learning Playground, introduced in Section 5.4; and an unplugged introduction to clustering, presented in Section 6.4.

To open the black box even for young students, the author used the benefits of educational robotics outlined in Section 2.3.3 and the visual block-based programming language presented in Section 2.3.2. In developing the extensions and the curriculum for machine learning, she derived design principles from constructivism, constructionism, connectivism, child-oriented graphic design, and playful learning presented in Section 4.1. The proposed approaches reflect the currency and thematic complexity of the field, which comprises three main areas of machine learning (Russell & Norvig, 2016): supervised, unsupervised, and reinforcement learning.

The author addressed not only the black-box approaches to supervised learning currently used for education purposes and discussed in Section 2.4, but also the sparse activities for reinforcement and unsupervised learning. The positive feedback from students on the learning materials presented in Chapter 6 indicated that the author could fill the gap in the lack of materials for young students to learn about machine learning, providing children resources to learn even on their own.

The results of this research provide new insights into a barely explored approach on how machine learning can be introduced to a novice. They have shown that children from primary to high school could successfully experience the technical part of machine learning in practice by experimenting on Playgrounds and completing four accompanying modules of machine learning. The students taught the robot by training simple neural networks and explored how the robot can learn with rewards by experimenting with the Q-learning algorithm. They also familiarised themselves with unsupervised learning by exploring the k-means algorithm in an unplugged manner. The results of the questionnaire, presented in Chapter 7, demonstrate that the approaches chosen to introduce

supervised, unsupervised, and reinforcement learning could raise the interest of students and be accessible even to young children in primary school.

In the following, the results are reflected against the background on research questions, related studies, developed extensions, and selected teaching approaches. The author places results in the broader research context and discusses limitations and considerations for future approaches to the introduction of machine learning with robots.

8.1 Reflections on the User Study

This section reflects on the third research question: how the implemented concepts appealed to the students and what help they needed to understand the concepts offered. The user study is discussed as a test phase for the approaches and materials developed.

The reaction of students from different school grades to the extensions and the developed materials was overall highly positive. As explained in Section 7.4, the vast majority of the children perceived the topics to be both very interesting and easy to understand. Even the primary school children were able to delve into the topics of machine learning. The students' feedback at the end of the sessions indicates that they gained extensive insights into machine learning and AI, and the vast majority indicated a desire to continue working on the topics.

These results are consistent with the evidence presented by Lin et al. (2020); Williams, Park, and Breazeal (2019); Williams, Park, Oh, et al. (2019), who also focused on the introduction of machine learning to young students by putting the student in the agent's shoes, using robots as teachable agents, and designing activities that were based on constructivist theories.

In contrast to Jatzlau et al. (2019); Kahn, Lu, Zhang, Winters, et al. (2020); Kahn et al. (2018); Kahn and Winters (2017), who used the VPL Snap! to teach children machine learning, it was shown that with simplified technical vocabulary, a block-based programming language can be used not only by high school children, but even by children in primary school. The results of the questionnaire indicate that the children perceived the topics similarly, regardless of their age.

As can be deduced from Section 7.3, the level of support provided by the author to the children while working on the topics was moderate and varied according to the age of the children. On the topic of supervised learning and neural networks, the youngest students needed more intensive support than the middle and high school children. For other topics, the level of support was comparatively low.

Although it was not the aim of this study to measure the increase in children's knowledge, the author briefly pre-assessed what the children knew about AI and

machine learning at the beginning of the session. At the end of the session, she interviewed the children again. The results shown in Section 7.4.4 suggest that children's knowledge about AI and machine learning increased. Considerably more children associated learning with AI, and the answers were more differentiated than during the pre-assessment.

Despite the vibrant sessions and the learning atmosphere during the user study, the author notes ambiguities in the process and results. One remaining question is whether the children could grasp the machine learning concepts in such a way that they were able to build correct mental models of underlying machine learning principles, as claimed by Hitron et al. (2019) and discussed in Sections 2.2.1 and 4.1.3. Young students in particular did not find supervised learning intuitive, giving an average score of 3.3 for the difficulty of the topic in the questionnaire.

As described in Section 7.3, many children stumbled with the Neural Network Cards "Fear" and "Friendship". This can be a sign that children misunderstood direct supervision and neural networks. The children had not directly recognised that not only the blocks, but also the configuration of the neural network, were decisive for the behaviour of the robot. Perhaps the children were simply confused about the card with the same blocks on the front but different task descriptions.

With the Q-learning, the children did not have any particular difficulties, nor did they seek extensive support from the author, as indicated in Section 7.3. However, the children sometimes focused on less critical issues, which could be interpreted as meaning that their cognitive load was high, and they were overwhelmed by what was happening on the Q-learning Playground.

Similar to Kandlhofer et al. (2016), it is unclear whether the children would be able to transfer their knowledge of the machine learning concepts they explored to similar problems since they were occupied with a pre-selected set of algorithms and machine learning problems. The question also remains of whether they could retain their knowledge over time.

8.2 Reflections on Extensions and Teaching Approaches

Based on the results and the analysis of the user study, this section discusses the extensions that were developed – the Neural Network Playground and Q-learning Playground – and the teaching approaches that were selected, with a focus on the learning activities and materials. Thus, the section reflects on the possibilities identified by the author and the process of establishing and implementing the approaches in Open Roberta Lab, as posed in the first and second research questions.

8.2.1 Implementation of Extensions and Development Process

Technical implementations were reflected in detail in Section 5.5. In this section, the author adds general considerations to the extensions and the development process.

The extensions developed by the author show a significant difference to the applications that currently exist for teaching machine learning. While previous approaches used blocks as high-end APIs to access AI cloud services (Druga, 2018; Kahn & Winters, 2017; Lane, 2020) or to reproduce the machine learning algorithms (Kahn, Lu, Zhang, Winde, et al., 2020; Kahn, Lu, Zhang, Winters, et al., 2020; Kahn et al., 2018), the author developed a completely different approach. The extensions developed by the author (1) break the technology down to the essentials while maintaining the technical correctness of the underlying principles and their accuracy, (2) provide a direct interface to the algorithms of machine learning via blocks, (3) reduce the complexity of the algorithms by adapting the technical vocabulary and by visualising underlying principles graphically and (4) open the black box problematised in Section 2.4 by offering the possibility of experimenting with underlying technologies of machine learning in Playgrounds with robots. The ideas for future implementations are proposed in Section 8.3.4.

Overall, the development process was intensive. By using tools and methods presented in Section 4.4, the ideas elaborated in Sections 3.1.2 and 3.2.2 could be successfully implemented. The author's proposals concerned the extensions of the major processing steps in Open Roberta Lab – including the definition of blocks, the integration of the blocks' functionalities in the back-end, the implementation of the Q-learning algorithm and simple neural networks, the design of the Playgrounds both server- and client-side, and graphical adjustments of the simulated robot's behaviour. For this purpose, the author intensively dealt with the central processes and workflows of Open Roberta Lab, a complicated undertaking, since Open Roberta Lab is poorly documented.

Nevertheless, the development process was successful, and extensions work smoothly. There were no particular technical difficulties during the tests. As mentioned in Section 7.4.4, one of the students of grades 7–9 criticised the implementation of neural networks – the student opposed the concept of weighting and solving neural networks with if-then queries. Although the author finds the criticism partly justified and it would be possible to represent simple networks with if-then statements,[22] it would not be possible to represent completely the graph connections realised in more complex neural networks.

[22] If-then statements can be used for Boolean weight values, not for non-Boolean values, e.g., for ranges between 0 and 1.

8.2.2 Using Simulated Robots

The user study suggests that the students could gain insights into how the simulated robot perceives the environment and how it learns. By teaching the robot and experiencing the environment from the agent's perspective, the students could also deepen their mental models of the capabilities and limitations of different machine learning approaches.

The experience with the "Rally" card described in Section 7.3 indicates that the children were engaged in investigating why the robot did not behave correctly. With the Q-learning algorithm, the robot's failure to learn how to find the best way out of the labyrinth made the students curious to find out why the robot did not learn properly and why it could not find its way out of the labyrinth. These observations are consistent with the findings of Lin et al. (2020); Williams, Park, and Breazeal (2019), who found that children were particularly curious when the robot did not behave as expected. The children's determination to correct the robot can be used by instructors to convey how agents can be trained and that they are not perfect. Letting the robot learn and allowing it to make mistakes might be a successful strategy that could be used in the future to teach machine learning topics.

8.2.3 Using a Visual Programming Language

Although the children could successfully use blocks to write machine learning applications, the children noticed the complicated workflow and usability difficulties associated with blocks, which were problematised in Section 7.3.

These remarks agree with the findings, which the author determined in the course of the developments. Blockly blocks are bulky, and their possibilities are limited. First, they are barely suitable for displaying graphs, as with a neural network. Second, Blockly also offers hardly any design possibilities for the blocks. In the case of the Q-learning Playground, it was barely possible to adapt the configuration blocks for the algorithm so that the children could design the Q-learning map interactively. The only way to create a unique environment was to prepare a long list of obstacles, which allowed the manipulation of the maps on the Q-learning Playground. However, the lists were so extensive that the children quickly lost track of where they had set an obstacle and where they had not.

In the future, frameworks such as the Neural Network Playground (Smilkov & Carter, 2020) can be explored to help children make the transition from blocks to Playgrounds.

8.2.4 Plugged vs. Unplugged Activities

Feedback from students on the unplugged activity that introduced the unsupervised learning indicates declining interest. Students at all school grades found the k-means algorithm less interesting than the two plugged activities with direct supervision and Q-learning, whereby the questionnaire results indicate that the difference was not significant for primary and middle school students. These results are consistent with the findings of Erümit and Sahin (2020), which found students to be enthusiastic about both plugged and unplugged activities. Following the approaches of Jatzlau et al. (2019); Michaeli et al. (2020), it would be interesting in future studies to experiment with introducing the machine learning topics first with the unplugged activity, followed by the plugged activity, and to measure whether children's understanding varies depending on whether the topic was introduced plugged or unplugged.

8.2.5 User Experience in Playgrounds and Materials

In contrast to Kahn, Lu, Zhang, Winters, et al. (2020); Kahn et al. (2018); Kahn and Winters (2017), the materials and Playgrounds were designed with the young students in mind. Inspired by children's books covering machine learning topics, the themes were adapted by reformulating the technical descriptions in story-like narratives and revising the technical terminology. In visual communication, a comic style, hand drawings, and colourful illustrations were used. Primary, middle, and high school students found the design of the extensions and the material appealing. This means of presenting complicated content can be used in the future.

In observing how the children used the learning cards, the author noticed that the children sometimes did not read the task descriptions thoroughly despite short, child-friendly texts. As noted in Section 7.3, the children were impatient and wanted to turn the card over and look for the solution as quickly as possible. The possible solution for future studies could be to use the learning cards only as a medium for the students to get started. At the same time, there may be rules that allow the card to be turned over, or the solution may only be partially presented on the backside of the card.

8.3 Limitations and Recommendations

The following chapter summarises the limitations of the study and proposes recommendations for future research and approach design.

8.3.1 Peer Learning

As mentioned in Section 4.2, it was not possible to apply the methods of peer learning when evaluating the extensions and materials. Although it could be observed that the children of all school grades were curious about how their classmates solved the tasks

and wanted to help each other if possible, they were not allowed to work in tandem or in groups on shared projects. However, working on projects with peers is a promising methodology (Resnick & Robinson, 2017) that is well-founded (Büttner et al., 2012; Hattie, 2008; Lebedynska, 2017; Zeneli & Tymms, 2015) and should be included in future research if possible.

8.3.2 Playfulness in Extensions and Materials for Machine Learning

Although one of the objectives proposed in Section 4.1.3 was to give children more room for experimentation, this was only partly achieved. Most students experimented with the underlying processes and algorithms based on the material provided. Only some students who were faster than others and had time continued to work on their projects, creating more complicated learning environments or more complex neural network architectures, as shown in Section 7.3. This casts doubt on the extent to which the children were allowed to live out their creativity and playful tinkering as a critical principle of constructivism and constructionism and the Four P's framework of Creative Learning.

The reason that the project work was neglected was due partly to a tight schedule, but also to the COVID-19 measures, which prohibited teamwork. As explained in Chapter 6, the learning activities and materials were designed for children to get started and work individually. In the future, tinkering with the projects should be further enabled and emphasised to a much greater degree.

8.3.3 Questionnaire Limitations

The questionnaire used in this study imposes several limitations. By problematising how interesting something is or how difficult, the questionnaire did not measure the understanding of children, the increase in their knowledge or the effectiveness of the approaches and materials developed. Nor was it intended to measure the long-term impact on how much the children retain from the sessions.

In the future studies, the problem of a deeper understanding of machine learning concepts can be addressed by asking children to describe the process of how they understand the learning process of the robot, for instance, in semi-structured interviews. The effectiveness of the approaches and the increase in knowledge can be examined by asking what the students think, how the robot arrives at the solution. In the future, the use of more age-specific formats for the design of questionnaire might be considered. The long-term effects on how much of the content the children retain after a certain period could be measured by post-assessing the children's knowledge in a given period.

8.3.4 Recommendations for Future Research

This study focused on using simulated robots as teachable agents to introduce a novice to the technical aspect of machine learning. The results and the discussion of the findings suggest that this approach could arouse the interest of the children and at the same time limit the difficulties that they experience. Future research may take into account that the use of simulated robots for introduction to machine learning is promising, especially if the robot is used to teach how agents are trainable and that they are not perfect. Transferring the approaches investigated here to real robots and measuring whether there are differences in the introduction of machine learning with simulated versus real robots could also be considered.

Future research may investigate how the Playgrounds can be improved and extended to help young students experiment and open the black box of machine learning algorithms. For example, the Q-learning Playground could be extended through the addition of a Q-learning table, as suggested by Jatzlau et al. (2019). The Neural Network Playground could be enriched graphically through the addition of animating tensors that visualise the data flow. Future research could also investigate how playful visualisation used in this study and by Lin et al. (2020) to communicate the underlying system models could be extended to other machine learning algorithms. Furthermore, it can be examined how the black box of complex machine learning algorithms and processes can be opened with other block-based programming languages.

This study proved that the materials and extensions designed with young learners in mind were also positively perceived by older students. As mentioned in Section 2.3.1, there are children's book authors who follow this approach, designing books with children in mind but that also address adults. Therefore, in order to promote understanding of machine learning among young students, future research may consider designing applications and materials that focus on the youngest students and pick up the older children at the same time. It can also be questioned whether children from high school are less attracted to materials oriented towards primary school students due to the different environments that surround the students.

The applications designed to explore machine learning paradigms should be open for experimentation, creativity, and play. As Resnick and Robinson (2017) remarked, learning environments should be more like Playgrounds, providing room to experiment, move, and collaborate. This work was a step in this direction, although this approach may often be incompatible with institutionalised forms of learning, especially in schools. After problematising the shortcomings of this work in terms of playfulness in Section 8.3.2, future research should focus on exploring more deeply how applications for introducing

machine learning to young students can be designed to resemble Playgrounds that foster creativity rather than Playpens that restrict and limit opportunities.

9 Conclusion

In this work, the author presented a new approach to introducing machine learning using robots, playful learning, and child-oriented design. In the following, the author's main contributions are summarised as answers to the three research questions posed in the introduction. The chapter reflects the entire research study and its role for future research.

In order to meet the requirements and to anchor the topic in the current context of educational research, the author analysed the specific curricular needs concerning the topic of machine learning in schools. It was found that there are high expectations for what children should know about AI, which are summarised under the term AI literacy (Long & Magerko, 2020). Special attention is devoted to machine learning, which is presented as one of the five Big Ideas of AI in the proposal for international guidelines for the development of AI curricula in schools (Touretzky et al., 2019). To demonstrate its relevance, the author also collected several case studies that show that students are expected to be able to cope with the different areas of machine learning – supervised, unsupervised, and reinforcement learning.

After outlining curricular needs, the author analysed the current efforts and approaches to introducing machine learning in schools. It was found that approaches are sparse and do not reflect the complexity and breadth of the field. Supervised learning is the topic for which the most approaches are currently available; however, they all share a common trait: The principles underlying the applications remain hidden from the user. Even if some approaches open the black box using VPLs, they often only provide an interface to a powerful high-end API. It remains unclear for the students how the models are trained and why they make the concrete decision. The topics of reinforcement and unsupervised learning are underrepresented. Although some approaches use a block-based programming language, they are only suitable for students at high schools due to high technical complexity and numerous details.

While summarising the didactic methods of current approaches, it was found that the use of a VPL to teach the topics of AI and machine learning is common and promising. The available teaching material is sparse, focuses on older children, and is not suitable for children to learn on their own. The use of robots and robot simulators in the classroom is overall effective. However, only a few studies have focused on the use of robots to teach children machine learning. All these studies found that the efforts were successful, and even kindergarten students could cope with machine learning topics using robots.

Based on these findings and using the benefits of visual block-based programming languages and educational robots, the author established the theoretical framework on supervised, unsupervised, and reinforcement learning. She elaborated three approaches that could meet the requirements and close the identified gaps: introducing direct supervision with neural networks, Q-learning, and k-means algorithms in Open Roberta Lab.

The answer to the first research question was thus complete: The author analysed specific needs with regards to the topics of machine learning in schools, examined the available possibilities for introducing machine learning, identified the limits of current concepts, and worked out three approaches to meet the identified requirements.

The author then turned to the second research question, which how previously defined proposals can be pedagogically anchored and concretely implemented in Open Roberta Lab in order to promote the transparency of the underlying machine learning algorithms and make them accessible to all interested parties.

In the course of working to answer this question, the author developed two machine learning extensions and elaborated a series of learning materials and activities. The following overview summarises the contributions:

(1) **New blocks for neural networks** allow the user to program applications with neural networks. The **Neural Network Playground** allows the user to experiment with simple neural networks in Open Roberta Lab. The student can program simple neural networks with blocks and then train them by modifying the weights and directly observing the effects on the simulated robot, grasping the concept of “direct supervision”.

(2) **New blocks for Q-learning** allow the user to program applications based on the Q-learning algorithm. On the **Q-learning Playground**, the student can tinker with the Q-learning algorithm by creating unique learning environments for the robot and playing with the parameters of the algorithm in Open Roberta Lab. Step by step, the student can debug the algorithm and explore how the robot is learning from the agent’s perspective.

(3) An **image for the local installation of Open Roberta Lab with extensions for machine learning** on a Raspberry Pi enables any interested party to easily install and use Open Roberta with machine learning features.

(4) The learning activities that the author has developed to introduce students to supervised, reinforcement, and unsupervised learning are summarised in a **Machine Learning Curriculum**, which educators may use as a guide. The

curriculum suggests learning activities for approximately six school hours of 45 minutes each. Every educator may decide whether individual activities should take more or less time.

(5) A set of nine **Neural Network Cards** helps the beginners get started with neural networks and supervised learning in Open Roberta Lab. With simple tasks, intuitive descriptions and appealing illustrations of expected robot behaviour, students can explore the essential elements of neural networks and learn how the neural networks are programmed and trained.

(6) A set of four **Q-learning Observation Cards** helps the novice start with reinforcement learning in Open Roberta Lab. The cards focus on the essentials of the Q-learning algorithm needed to conduct the experiments and provide a space for the documentation of experiments and reflections.

(7) A set of auxiliary learning materials to help the beginner get started with Q-learning on their own was created:

   a. The **Q-learning map** explains the navigation bar and central features of the Q-learning Playground.

   b. The **flow diagram** adapted for children, which introduces the Q-learning algorithm step by step.

   c. The **programme code**, which explains how to program applications based on Q-learning with blocks.

   d. The **Q&A: Reinforcement Learning**, which summarises possible questions and answers from children on Q-learning.

(8) An **unplugged activity that introduces unsupervised learning** with the k-means algorithm as an experiment that can be conducted by a teacher or as a game that can be played in tandem by two or more students.

With these contributions, the author has anchored her proposed approaches pedagogically and implemented them practically in Open Roberta Lab, thereby answering the second research question.

In order to evaluate how beginners perceive the approaches and thus answer the third research question, the author conducted a user study with 24 children from primary, middle, and high school. In total, the author led three sessions, with each session lasting six school hours.

The results of the questionnaire, the oral feedback from the students, and the comments of the observer indicate that the vast majority of the children in all three age groups perceived the topics as exciting and easy to follow and expressed the intention to learn more about AI and machine learning in the future. The children reported overall positive experiences with the machine learning extensions and repeatedly emphasised the accompanying learning materials as appealing. The feedback from the students on the unplugged activity introducing unsupervised learning with the k-means algorithm showed a slight decrease in interest among all tested groups of children. The author briefly assessed what the children knew about AI and machine learning at the beginning and the end of the session. The result indicated that children's answers were more differentiated at the end compared to the beginning, and many more children associated AI with learning.

Programming simulated robots with the visual block-based programming language NEPO and experimenting with them on Playgrounds proved to be a successful approach that excited the students in all sessions. In particular, when the robot did not behave as expected, the students were curious to find out why. They tried to retrain the neural network or reconfigure the Q-learning algorithm, and if the efforts were successful, the students were excited. By putting themselves in the robot's shoes, the students experienced hands-on the underlying principles of machine learning. They understood how agents can be taught and that they are not perfect. Overall, the practical robot simulation made learning more playful but did not require compromising content or reducing the technical scope. In the future, training an AI robot could be extended by developing challenges for students such as building the fastest robot.

Despite the primary school children who needed help with supervised learning and training of the neural networks, the students did not require any exceptional help. The author's support was moderate. If the students were able to work together with peers, the support provided by the author would be probably much less.

The author also identified further limitations to the machine learning extensions and the pedagogical activities that were developed: The aspect of playfulness should be promoted much more in future sessions and in the development of the extensions. The questionnaire can be extended or redesigned to measure the increase in knowledge of machine learning concepts and the children's in-depth understanding. Future research can also consider integrating clustering into the robot simulation environment of Open Roberta Lab and mirroring all extensions from simulated to real robots.

With the successful user study, the feedback of students, and the reflections in the discussion, the author answered the third research question and showed the significant potential of the developed extensions and materials for future research.

The author’s contributions include a solid basis for the introduction of machine learning among novice learners and close the gap identified at the beginning of this research study. The developed approaches reflect the breadth of the field of machine learning and offer a contrasting alternative to the black box approaches currently available in the educational landscape. The successful evaluation study with students from different school grades underlines the practical feasibility of the concept.

The author regards the overall course of this research project as a very intensive but enriching process. The development of machine learning extensions was challenging, as Open Roberta Lab is a large project and is poorly documented. Although the user study was considered optional due to the closure of schools because of COVID-19, the opportunity arose to test the extensions that were developed, and the author succeeded in conducting the evaluation with school children in live sessions. From the author's perspective, this was the most enriching part of the project. If the author were to lead the sessions again, she would allow much more time for the individual topics, so that the children have time to live out their creativity. A hackathon is an exciting possibility. Overall, the author intends to continue working on this topic and hopes that this research study will inspire other people to make machine learning tangible for young students and at the same time appeal to beginners of all ages.

10 Bibliography

Adobe. (2020a). Adobe Fresco Website. Retrieved from https://www.adobe.com/products/fresco.html#. Accessed: 06.10.2020.

Adobe. (2020b). Adobe Illustrator Website. Retrieved from https://adobe.ly/3d4T9As. Accessed: 06.10.2020.

Adobe. (2020c). Adobe InDesign Website. Retrieved from https://adobe.ly/2Gyy4T4. Accessed: 06.10.2020.

Aggarwal, C. C., & Reddy, C. K. (2013). *Data Clustering : Algorithms and Applications*. Philadelphia, PA, UNITED STATES: CRC Press LLC.

Balzert, H. (2011). *Lehrbuch der Objektmodellierung: Analyse und Entwurf mit der UML 2*: Spektrum Akademischer Verlag.

Barker, B. S., Nugent, G., Grandgenett, N., & Adamchuk, V. I. (2012). *Robots in K-12 Education: A New Technology for Learning*.

Blakeley, H. P., & Breazeal, C. (2019). *An Ethics of Artificial Intelligence: Curriculum for Middle School Students*: MIT Media Lab.

Brabazon, A., O'Neill, M., & McGarraghy, S. (2015). Neural Networks for Supervised Learning. In *Natural Computing Algorithms* (pp. 221-259). Berlin, Heidelberg: Springer Berlin Heidelberg.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*: MIT Press.

Burgsteiner, H., Kandlhfer, M., & Steinbauer, G. (2016). IRobot: Teaching the Basics of Artificial Intelligence in High Schools. *Proceedings of the Sixth Symposium on Edcuational Advances in Artificial Intelligence (EAAI-16)*.

Büttner, G., Warwas, J., & Adl-Amini, K. (2012). Kooperatives Lernen und Peer Tutoring im inklusiven Unterricht. *Zeitschrift für Inklusion*(1-2), 14. doi:URN: urn:nbn:de:0111-opus-58778.

Castella, K. (2018). *Designing for Kids: Creating for Playing, Learning, and Growing*: Taylor & Francis.

Chin, K., Hong, Z.-W., & Chen, Y. (2014). Impact of Using an Educational Robot-Based Learning System on Studentì Motivation in Elementary Education. *IEEE Transactions on Learning Technologies, 7*, 333-345.

Clarke, B. (2019). *Artificial Intelligence - Alternate Curriculum Unit*. Accessed.

code.org. (2020). AI for Oceans – Learn how AI and machine learning can be used to address world problems. Retrieved from https://studio.code.org/s/oceans. Accessed: 12.10.2020.

Codingschule gGmbH. (2020). Codingschule junior. Retrieved from https://www.codingschule-junior.de. Accessed: 03.04.2020.

Cooper, M., Keating, D., Harwin, W., & Dautenhahn, K. (1999). Robots in the classroom-tools for accessible education. *Assistive technology on the threshold of the new millennium, 6*, 448.

D4CR Association. (2020). DESIGNING for CHILDREN'S RIGHTS GUIDE. Retrieved from https://childrensdesignguide.org/. Accessed: 05.10.2020.

Das, S., & Cakmak, U. M. (2018). *Hands-On Automated Machine Learning: A beginner's guide to building automated machine learning systems using AutoML and Python*: Packt Publishing.

Dhoot, D. (2019a). *ABCs of Machine Learning*: Tinker Toddlers.

Dhoot, D. (2019b). *Machine Learning for Kids*: Tinker Toddlers.

Dhoot, D. (2019c). *Neural Networks for Kids*: Tinker Toddlers.

Dodds, Z., Greenwald, L. G., Howard, A., Tejada, S., & Weinberg, J. B. (2006). Components, Curriculum, and Community: Robots and Robotics in Undergraduate AI Education. *AI Magazine, 27*, 11-22.

Druga, S. (2018). *Growing Up with AI: Cognimates : from Coding to Teaching Machines*: Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences.

Druga, S., Qiu, T., T.VU, S., Likhith, E., & Dale, S. (2020). An AI education platform for building games, programming robots & training AI models. Retrieved from http://cognimates.me/home/. Accessed: 12.10.2020.

Druga, S., T.Vu, S., Likhith, E., & Qiu, T. (2019). Inclusive AI literacy for kids around the world. *Proceedings of ACM Fablearn conference (Fablearn' 19)*. doi:https://doi.org/10.475/123_4.

Druga, S., Williams, R., Park, H. W., & Breazeal, C. (2018). How smart are the smart toys?: children and parents' agent interaction and intelligence attribution. *Proceedings of the 17th ACM Conference on Interaction Design and Children*. doi:doi.org/10.1145/3202185.3202741.

Ertel, W., & Black, N. T. (2018). *Introduction to Artificial Intelligence*: Springer International Publishing.

Erümit, A. K., & Sahin, G. (2020). Plugged or Unplugged Teaching: A Case Study of Students' Preferences for the Teaching of Programming. *International Journal of Computer Science Education in Schools, 4*(1), 3 - 32. doi:10.21585/ijcses.v4i1.82.

Feinberg, E. A., & Shwartz, A. (2012). *Handbook of Markov Decision Processes: Methods and Applications*: Springer US.

Ferrie, C., & Kaiser, S. (2019). *Neural Networks for Babies*: Sourcebooks, Incorporated.

Feynman, R. P., Leighton, R. B., & Sands, M. (2011). *Six Easy Pieces: Essentials of Physics Explained by Its Most Brilliant Teacher*: Basic Books.

Fierens, W. (2020). SVG.js. Retrieved from https://svgjs.com/docs/3.0/. Accessed: 02.10.2020.

Freeman, N. K., Feeney, S., & Moravcik, E. (2011). Enjoying A Good Story: Why We Use Children's Literature When Teaching Adults. *Early Childhood Education Journal, 39*(1), 1-5. doi:10.1007/s10643-010-0439-4.

Fuchs, J., Isenberg, P., Bezerianos, A., Miller, M., Keim, D., Santos, B., & Alford, G. (2020). Teaching Clustering Algorithms With EduClust: Experience Report and Future Directions. *IEEE Computer Graphics and Applications, 40*, 98-102.

Fuste, A. (2018). *Learning computational thinking through embodied spatial programming in augmented reality.*

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2019). *Design Patterns: Elements of Reusable Object-Oriented Software*. Uttar Pradesh: Pearson Education.

Ghahramani, Z. (2004). Unsupervised Learning. In O. Bousquet, G. Raetsch, & U. von Luxburg (Eds.), *Advanced Lectures on Machine Learning*: Springer-Verlag.

Google. (2020). Teachable Machine. Retrieved from https://teachablemachine.withgoogle.com/. Accessed: 28.08.2020.

Google Developers. (2020a). Blockly – Custom Blocks. Retrieved from https://developers.google.com/blockly/guides/create-custom-blocks/overview. Accessed: 12.10.2020.

Google Developers. (2020b). Closure Library. Retrieved from https://developers.google.com/closure/library. Accessed: 05.10.2020.

Google Developers. (2020c). Google Blockly. Retrieved from https://developers.google.com/blockly. Accessed: 29.09.2020.

Hamerly, G., & Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. *Proceedings of the eleventh international conference on Information and knowledge management*, 600–607. doi:10.1145/584792.584890.

Hattie, J. (2008). *Visible learning. A synthesis of over 800 meta-analyses relating to achievement. 1st publ*. London u.a.: Routledge.

Hitron, T., Orlev, Y., Wald, I., Shamir, A., Erel, H., & Zuckerman, O. (2019). Can Children Understand Machine Learning Concepts?: The Effect of Uncovering Black Boxes. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. doi:doi.org/10.1145/3290605.3300645.

Hitron, T., Wald, I., Erel, H., & Zuckerman, O. (2018). Introducing children to machine learning concepts through hands-on experience. *Proceedings of the 17th ACM Conference on Interaction Design and Children*.

IBM. (2020). IBM Watson. Retrieved from https://www.ibm.com/watson. Accessed: 12.10.2020.

Jatzlau, S., Michaeli, T., Seegerer, S., & Romeike, R. (2019). It s not Magic After All@ Machine Learning in Snap! using Reinforcement Learning. *2019 IEEE Blocks and Beyond Workshop (B&B)*, 37-41.

JetBrains. (2020a). IntelliJ IDEA Website. Retrieved from https://www.jetbrains.com/idea/. Accessed: 06.10.2020.

JetBrains. (2020b). WebStorm Website. Retrieved from https://www.jetbrains.com/webstorm/. Accessed: 06.10.2020.

Kahn, K., Lu, Y., Zhang, J., Winde, M., & Gao, M. (2020). Programming word embeddings in Snap!

Kahn, K., Lu, Y., Zhang, J., Winters, N., & Gao, M. (2020). Deep Learning Programming by All. *University of Oxford, Beijing Normal University*.

Kahn, K., Megasari, R., Piantari, E., & Junaeti, E. (2018). *AI programming by children using Snap! block programming in a developing country*. Accessed.

Kahn, K., & Winters, N. (2017). *Child-friendly programming interfaces to AI cloud services*. Accessed.

Kandlhofer, M., Steinbauer, G., Hirschmugl-Gaisch, S., & Huber, P. (2016). Artificial intelligence and computer science in education: From kindergarten to university. *2016 IEEE Frontiers in Education Conference (FIE)*, 1-9.

Khishe, M., & Parvizi, G. R. (2020). Neural Networks: History and Applications. In A. Doug (Ed.). New York, UNITED STATES: Nova Science Publishers, Incorporated.

Klassner, F. (2002). *A case study of LEGO Mindstorms'™ suitability for artificial intelligence and robotics courses at the college level* (Vol. 34).

Klassner, F., & Anderson, S. D. (2003). LEGO MindStorms: not just for K-12 anymore. *IEEE Robotics & Automation Magazine, 10*(2), 12-18. doi:10.1109/MRA.2003.1213611.

Kleeberger, J., Prost, N., & Sternkopf, H. (2019). *Machine Learning. Intelligente Maschinen im Projekt »Medien in die Schule« – Materialien für den Unterricht –*: Medien in die Schule.

Kober, J., & Peters, J. (2014). Reinforcement Learning in Robotics: A Survey. In *Learning Motor Skills: From Algorithms to Robot Experiments* (pp. 9-67). Cham: Springer International Publishing.

Krueger, N. (2020). 3 unplugged activities for teaching about AI. Retrieved from https://www.iste.org/explore/Computer-Science/3-unplugged-activities-for-teaching-about-AI. Accessed: 31.08.2020.

Kumar, A. N. (2004). Three years of using robots in an artificial intelligence course: lessons learned. *J. Educ. Resour. Comput., 4*(3), 2–es. doi:10.1145/1083310.1083311.

Lane, D. (2020). Machine Learning for Kids. Retrieved from https://machinelearningforkids.co.uk/. Accessed: 25.09.2020.

Lebedynska, V. (2017). Peer-Tutoring – jetzt digital? *Medien + Erziehung, 61*(5), 55–61.

Leimbach, T., & Breuer, T. (2012). *Roberta-Experiment: Braitenberg-Vehikel*. Sankt Augustin: Fraunhofer-Institut Intelligente Analyse und Informationssysteme IAIS.

Li, L.-Y., Chang, C.-W., & Chen, G.-D. (2009). *Researches on Using Robots in Education*, Berlin, Heidelberg.

Lin, P., Brummelen, J. V., Lukin, G., Williams, R., & Breazeal, C. (2020). *Zhorai: Designing a Conversational Agent for Children to Explore Machine Learning Concepts.* Paper presented at the Proceedings of the AAAI Conference on Artificial Intelligence.

Lindner, A., & Seegerer, S. (2019). *AI Unplugged - Unplugging Artificial Intelligence - Activities and teaching materialon artificial intelligence* (Professorship for Computer Science Education Ed.): Friedrich-Alexander-Universität Erlangen-Nürnberg.

Lister, R. (2011). *Concrete and other neo-Piagetian forms of reasoning in the novice programmer.* Paper presented at the ACE 2011.

Liukas, L. (2019). *Hello Ruby: Wenn Roboter zur Schule gehen*: Bananenblau - Der Praxisverlag für Pädagogen.

Long, D., & Magerko, B. (2020). What is AI Literacy? Competencies and Design Considerations. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. doi:doi.org/10.1145/3313831.3376727.

MacKay, D. J. C. (2003). *Information Theory, Inference and Learning Algorithms*: Cambridge University Press.

Makeblock Co. (2020). makeBlock – a global STEAM education solution provider. Retrieved from https://www.makeblock.com/. Accessed.

Massachusetts Institute of Technology. (2020). MIT App Inventor. Retrieved from https://appinventor.mit.edu/. Accessed: 12.10.2020.

Michaeli, T., Seegerer, S., & Romeike, R. (2020). *Looking Beyond Supervised Classification and Image Recognition@ Unsupervised Learning with Snap!*

Millington, I., & Funge, J. (2018). *Artificial Intelligence for Games*: CRC Press.

Mirkes, E. M. (2011). K-means and K-medoids applet. Retrieved from http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html. Accessed: 17.09.2020.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning* (2nd. ed.): MIT Press.

Molnar, C. (2020). *Interpretable Machine Learning*: Lulu.com.

Moro, M., Arlegui, J., Pina, A., & Frangou, S. (2007). *Robotics & Constructivism in Education : the TERECoP project*.

Nguyen, Chi N., & Zeigermann, O. (2018). *Machine Learning – kurz & gut : Eine Einführung mit Python, Pandas und Scikit-Learn*. Heidelberg, GERMANY: o'Reilly.

npm. (2020). npm website. Retrieved from https://www.npmjs.com/. Accessed.

Olari, V. (2020a). blockly repository. Retrieved from https://github.com/vlebedynska/blockly/tree/feature/ai. Accessed: 20.09.2020.

Olari, V. (2020b). GitHub Account of Viktoriya Olari. Retrieved from https://github.com/vlebedynska. Accessed: 20.09.2020.

Olari, V. (2020c). Issues overview for defining machine learning blocks. Retrieved from https://github.com/vlebedynska/blockly/issues. Accessed: 30.09.2020.

Olari, V. (2020d). Issues overview for developing machine learning extensions. Retrieved from https://github.com/vlebedynska/openroberta-lab/issues. Accessed: 30.09.2020.

Olari, V. (2020e). openroberta-lab repository. Retrieved from https://github.com/vlebedynska/openroberta-lab/tree/feature/neuronalnetworks. Accessed: 20.09.2020.

Open Roberta. (2019a). Connecting blockly block with java backend. Retrieved from https://github.com/OpenRoberta/openroberta-lab/wiki/Connecting-blockly-block-with-java-backend. Accessed: 06.10.2020.

Open Roberta. (2019b). From Blockly XML to Code Generation. Retrieved from https://github.com/OpenRoberta/openroberta-lab/wiki/From-Blockly-XML-to-Code-Generation. Accessed: 06.10.2020.

Open Roberta. (2020a). blockly repository. Retrieved from https://github.com/OpenRoberta/blockly. Accessed: 20.09.2020.

Open Roberta. (2020b). Open Roberta Lab Server. Retrieved from https://www.roberta-home.de/lab/lokale-installation/. Accessed: 12.10.2020.

Open Roberta. (2020c). openroberta-lab repository. Retrieved from https://github.com/OpenRoberta/openroberta-lab. Accessed: 20.09.2020.

Open Roberta. (2020d). Overview of the Open Roberta Repositories. Retrieved from https://github.com/OpenRoberta. Accessed: 29.09.2020.

Open Roberta. (2020e). System Overview. Retrieved from https://github.com/OpenRoberta/openroberta-lab/wiki/System-Overview. Accessed: 06.10.2020.

Open Roberta Lab. (2020). The Open Roberta Lab,. Retrieved from https://lab.open-roberta.org/. Accessed: 29.09.2020.

Osmani, A. (2012). *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*: O'Reilly Media.

Papert, S. (1993a). *The children's machine: Rethinking school in the age of the computer*: ERIC.

Papert, S. (1993b). *Mindstorms: Children, Computers, And Powerful Ideas*: Basic Books.

Papert, S., & Harel, I. (1991). *Constructionism: Research Reports and Essays, 1985-1990*: Ablex Publishing Corporation.

Papert, S., & Solomon, C. (1971). *Twenty Things to Do with a Computer*: Massachusetts Institute of Technology, A. I. Laboratory.

Parsons, S., & Sklar, E. (2004). *Teaching AI using LEGO Mindstorms*.

Piaget, J., Fatke, R., & Kober, H. (2016). *Meine Theorie der geistigen Entwicklung*: Beltz.

Queiroz, R. L., Sampaio, F. b. F., Lima, C., & Lima, P. (2020). AI from concrete to abstract: demystifying artificial intelligence to the general public. *ArXiv, abs/2006.04013*.

react.js. (2020a). Introducing JSX. Retrieved from https://reactjs.org/docs/introducing-jsx.html. Accessed: 12.10.2020.

react.js. (2020b). react.js – A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/. Accessed: 05.10.2020.

require.js. (2020). require.js – JavaScript file and module loader. Retrieved from https://requirejs.org/. Accessed: 05.10.2020.

Resnick, M., Martin, F., Sargent, R., & Silverman, B. (1996). Programmable Bricks: Toys to Think With. *IBM Syst. J., 35*, 443-452.

Resnick, M., & Robinson, K. (2017). *Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play*: MIT Press.

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. 117-122. doi:10.1145/1109540.1109556.

RocketBabyClub. (2018a). *Mike's Peanuts: Machine Learning For Kids: Linear Regression*: Rocket Baby Club.

RocketBabyClub. (2018b). *Party Parrots: Machine Learning For Kids: Feature Engineering*: Rocket Baby Club.

RocketBabyClub. (2019a). *Goldfish Pond School: Machine Learning For Kids: Clustering*: Rocket Baby Club.

RocketBabyClub. (2019b). *Toby's Helpful Spirits: Machine Learning For Kids: Neural Networks*. Cambrige, MA, USA: Rocket Baby Club LLC.

RocketBabyClub. (2019c). *Toby's Video Game Puzzle: Machine Learning For Kids: Perceptron*: Rocket Baby Club.

Romero, M., Duflot-Kremer, M., & Viéville, T. (2019). Activity for learning computational thinking in pluggedand unplugged mode. Orig.: Le jeu du robot : analyse d'une activité d'informatique débranchée sous laperspective de la cognition incarnée. *Review of science, mathematics and ICT education, Laboratory of Didactics of Sciences, Mathematics and ICT, Department of EducationalSciences and Early Childhood Education - University of Patras*.

Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*: Pearson.

Sakulkueakulsuk, B., Witoon, S., Ngarmkajornwiwat, P., Pataranutapom, P., Surareungchai, W., Pataranutaporn, P., & Subsoontorn, P. (2018). Kids making AI: Integrating Machine Learning, Gamification, and Social Context in STEM Education. *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 1005-1010.

Šalamon, T. (2011). *Design of agent-based models. Developing computer simulations for a better understanding of social processes.* Řepín-Živonín: Tomáš Bruckner.

Salkind, D. N. J. J. (2006). *Encyclopedia of Measurement and Statistics*. Thousand Oaks: SAGE Publications, Inc.

Schmidt, C. (2015). *Agile Software Development Teams*: Springer International Publishing.

Scratch. (2020a). Scratch – Create stories, games, and animations, share with others around the world. Retrieved from https://scratch.mit.edu/. Accessed: 12.10.2020.

Scratch. (2020b). Scratch – Statistics. Retrieved from https://scratch.mit.edu/statistics/. Accessed: 24.08.2020.

Seegerer, S., Lindner, A., & Romeike, R. (2019). *AI Unplugged -Wir ziehen Künstlicher Intelligenz den Stecker*.

Siemens, G. (2005). Connectivism: a Learning Theory for the Digital Age. *International Journal of Instructional Technology and Distance Learning, 2*, 3–10. Retrieved from http://www.itdl.org/Journal/Jan_05/Jan_05.pdf. Accessed: 03.10.2016.

Siemens, G. (2014). Connectivism: A Learning Theory for the Digital Age. *International Journal of Instructional Technology and Distance Learning, 2(1)*, 3-10. Retrieved from http://www.elearnspace.org/Articles/connectivism.htm. Accessed: 19.06.2016.

Sjödén, B., Lind, M., & Silvervarg, A. (2017). *Can a Teachable Agent Influence How Students Respond to Competition in an Educational Game?*, Cham.

Sklar, E., Eguchi, A., & Johnson, J. (2002). *RoboCupJunior: Learning with Educational Robotics.* Paper presented at the AI Magazine.

Sloman, A. (2009). *Teaching AI and Philosophy at School ?*

Smilkov, D., & Carter, S. (2020). The Neural Network Playground. Retrieved from https://playground.tensorflow.org/. Accessed: 24.09.2020.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*: MIT Press.

TensorFlow. (2020). TensorFlow – An end-to-end open source machine learning platform. Retrieved from https://www.tensorflow.org/. Accessed: 12.10.2020.

Toivonen, T., Jormanainen, I., & Tukiainen, M. (2017). *An Open Robotics Environment Motivates Students to Learn the Key Concepts of Artificial Neural Networks and Reinforcement Learning.* Paper presented at the Robotics in Education.

Touretzky, D. (2019). Five Big Ideas in Artificial Intelligence: A Poster. Retrieved from https://raw.githubusercontent.com/touretzkyds/ai4k12/master/documents/AI4K12_Five_Big_Ideas_Poster.pdf. Accessed: 10.10.2020.

Touretzky, D. (2020, 10.08.2020). Resource Directory: ai4k12. Retrieved from https://github.com/touretzkyds/ai4k12/wiki/Resource-Directory. Accessed: 24.08.2020.

Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, **D.** (2019). Envisioning AI for K-12: What Should Every Child Know about AI? *Thirty-Third AAAI Conference on Artificial Intelligence, 33*. doi:doi.org/10.1609/aaai.v33i01.33019795.

TypeScript. (2020). TypeScript Website. Retrieved from https://www.typescriptlang.org/. Accessed: 01.10.2020.

Universität Konstanz. (2020). EduClust – A Visual Education Platform for Teaching Clustering Algorithms. Retrieved from https://educlust.dbvis.de/. Accessed: 12.10.2020.

Universität Paderborn. (2019). *MENSCH, Maschine! Wer zeigt hier wem den Weg?* Paderborn.

University of California at Berkeley. (2020). Welcome to Snap! Retrieved from https://snap.berkeley.edu/. Accessed: 12.10.2020.

University of Oxford. (2020). eCraft2Learn – Digital Fabrication and Maker Movement in Education Retrieved from https://ecraft2learn.github.io/uui/. Accessed: 12.10.2020.

van Pløn Verhagen, B. (2006). Connectivism: a new learning theory? . Retrieved from https://web.archive.org/web/20081210214143/http://www.surfspace.nl/nl/Redact

ieomgeving/Publicaties/Documents/Connectivism%20a%20new%20theory.pdf. Accessed: 18.09.2020.

Visionary Machines LLC. (2020). Calypso for Cozmo. Retrieved from https://calypso-robotics.com/. Accessed: 12.10.2020.

Vogel, L., Scholz, S., & Pfaff, F. (2020, 18.09.2020). Eclipse JDT - Abstract Syntax Tree (AST) and the Java Model. Retrieved from https://www.vogella.com/tutorials/EclipseJDT/article.html. Accessed: 29.09.2020.

w3schools.com. (2020). SVG Tutorial. Retrieved from https://www.w3schools.com/graphics/svg_intro.asp. Accessed: 02.10.2020.

Wang, W.-h. (2016). A mini experiment of offering STEM education to several age groups through the use of robots. *2016 IEEE Integrated STEM Education Conference (ISEC)*, 120-127.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*: Cambridge University.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3), 279-292. doi:10.1007/BF00992698.

Wieners, J. G. (2014). *SpoookyJS. Ein multiagentenbasiertes JavaScript-Framework zur flexiblen Implementation digitaler browserbasierter Brettspiele und spielübergreifender künstlicher Intelligenz*(Vol. PhD). Retrieved from https://kups.ub.uni-koeln.de/5971/. Accessed: 12.10.2020.

Williams, R., Park, H. W., & Breazeal, C. (2019). A is for Artificial Intelligence: The Impact of Artificial Intelligence Activities on Young Children's Perceptions of Robots. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. doi:doi.org/10.1145/3290605.3300677.

Williams, R., Park, H. W., Oh, L., & Breazeal, C. (2019). PopBots: Designing an Artificial Intelligence Curriculum for Early Childhood Education. *Personal Robots Group, MIT Media Lab*. doi:doi.org/10.1609/aaai.v33i01.33019729.

Wong, G. K. W., Ma, X., Dillenbourg, P., & Huan, J. (2020). Broadening artificial intelligence education in K-12. *ACM Inroads, 11*, 20 - 29.

Xu, K., Wu, F., & Zhao, J. (2015). Simplified online Q-learning for LEGO EV3 robot. *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 77-80.

Zeneli, M., & Tymms, P. (2015). A review of peer tutoring interventions and social interdependence characteristics. *International journal for cross-disciplinary subjects in education., Special Issue Volume 5*(2), 2504-2510. Retrieved from http://dro.dur.ac.uk/18744/. Accessed: 12.10.2020.

A Appendix

A.1 processNeuralNetwork function

```
public processNeuralNetwork(inputLayer, outputLayer) {

   if ($.isEmptyObject(this.neuralNetworkModule)) {
      this.neuralNetworkModule = new AiNeuralNetworkModule("#simConfigNeuralNetworkSVG",
inputLayer, outputLayer);

      this.neuralNetworkModule.player.isPlaying = true;

      let that = this;
      this.neuralNetworkModule.player.addEventListener("pause", function () {
         that.setBlocking(true);
         that.simSetPause(true);
         that.neuralNetworkModule.player.isPlaying = false;
      })
      this.neuralNetworkModule.player.addEventListener("play", function () {
         that.setBlocking(false);
         that.simSetPause(false);
         that.neuralNetworkModule.player.isPlaying = true;
      });
   }
   //set new Values in InputLayer
   if (!this.neuralNetworkModule.player.isPlaying) {
      return;
   }

   let aiNeuralNetworkInputLayer = this.neuralNetworkModule.aiNeuralNetwork.getInputLayer();
   for (let nodeID in inputLayer) {
      let node: Node = inputLayer[nodeID];
      if (aiNeuralNetworkInputLayer[nodeID].value !== node.value) {
         aiNeuralNetworkInputLayer[nodeID].value = node.value;
      }
   }

   //calculates new network nodes values
   this.neuralNetworkModule.calculateNeuralNetworkOutput();


   //set motor speed according to the new values
   this.clearDisplay();

   let value = 0;
   let textLines: Array<string> = new Array<string>();
   let textLinesPrepared: Array<string> = new Array<string>();
   let ledPrepared: string = "";
   for (let node2 of this.neuralNetworkModule.aiNeuralNetwork.getOutputLayer() ) {

      switch (node2.type) {
         case "motorport":
            if ( node2.value > 100) {
               value = 100;
            } else {
               value = node2.value;
            }
            this.setMotorSpeed("ev3", node2.port, value);
            console.log("Motorspeed" + value);
            break;
         case "text" :
            let textOutput = node2.value > 0 ? node2.name : "leer";
            textLines.push(textOutput);
            textLinesPrepared.push("<tspan x='1' dy='" + (node2.positionY*16+1) +
"'>"+textOutput + "</tspan>");
            break;
```

```
            case "sound":
               if (node2.value > 0) {
                  this.toneAction("outputNodeTon", node2.value*5, node2.duration);
               }
               break;
            case "LED":
               this.statusLightOffAction("ev3", 0);
               if (node2.value > 0) {
                  ledPrepared = node2.color;
               }
               break;
         }

      }
      if (textLinesPrepared.length > 0) {
            this.showTextActionPosition( textLinesPrepared.join(""), 0, 0, true);
      }
      if (ledPrepared != "") {
         this.lightAction("on", ledPrepared);
      }

}
```

A.2 AiNeuralNetwork.java

```
package de.fhg.iais.roberta.syntax.ai;

import java.util.ArrayList;
import java.util.List;

import de.fhg.iais.roberta.blockly.generated.Block;
import de.fhg.iais.roberta.blockly.generated.Value;
import de.fhg.iais.roberta.syntax.*;
import de.fhg.iais.roberta.syntax.lang.expr.Expr;
import de.fhg.iais.roberta.syntax.lang.expr.ListCreate;
import de.fhg.iais.roberta.syntax.lang.stmt.Stmt;
import de.fhg.iais.roberta.transformer.AbstractJaxb2Ast;
import de.fhg.iais.roberta.transformer.Ast2JaxbHelper;
import de.fhg.iais.roberta.transformer.ExprParam;
import de.fhg.iais.roberta.typecheck.BlocklyType;
import de.fhg.iais.roberta.visitor.IVisitor;
import de.fhg.iais.roberta.visitor.ai.IAiVisitor;

/**
 * This class represents <b>ai_neural_network</b> block from Blockly into the AST (abstract
syntax tree). Object from this
 * class will generate neural network including input and output layers.<br/>
 * <br>
 * To create an instance from this class use the method {@link #make(ListCreate, ListCreate,
List, BlocklyBlockProperties, BlocklyComment)}.<br>
 */
public class AiNeuralNetwork<V> extends Stmt<V> {

    private final ListCreate<V> listNNInput;
    private final ListCreate<V> listNNOutput;
    private final List<AiLink<V>> listNNLinks;

    public ListCreate<V> getListNNInput() {
        return listNNInput;
    }

    public ListCreate<V> getListNNOutput() {
        return listNNOutput;
    }

    public List<AiLink<V>> getListNNLinks() {
        return listNNLinks;
    }

    /**
     * This constructor set the kind of the object used in the AST (abstract syntax tree).
All possible kinds can be found in {@link BlockType}.
     * @param kind
     * @param listNNInput
     * @param listNNOutput
     * @param listNNLinks
     * @param property
     * @param comment
     */
    private AiNeuralNetwork(
        BlockType kind,
        ListCreate<V> listNNInput,
        ListCreate<V> listNNOutput,
        List<AiLink<V>> listNNLinks,
        BlocklyBlockProperties property,
        BlocklyComment comment) {
        super(kind, property, comment);
        this.listNNInput = listNNInput;
        this.listNNOutput = listNNOutput;
        this.listNNLinks = listNNLinks;
        setReadOnly();
    }
```

```
/**
     * creates a new {@link #AiNeuralNetwork} instance;
     * @param listNNInput
     * @param listNNOutput
     * @param listNNLinks
     * @param properties
     * @param comment
     * @param <V>
     * @return
     */
    public static <V> AiNeuralNetwork<V> make(
        ListCreate<V> listNNInput,
        ListCreate<V> listNNOutput,
        List<AiLink<V>> listNNLinks,
        BlocklyBlockProperties properties,
        BlocklyComment comment) {
        return new AiNeuralNetwork<V>(BlockTypeContainer.getByName("AI_NEURAL_NETWORK"),
listNNInput, listNNOutput, listNNLinks, properties, comment);
    }

    /**
     * implements
     * @param visitor
     * @return
     */
    @Override
    protected V acceptImpl(IVisitor<V> visitor) {
        return ((IAiVisitor<V>) visitor).visitAiNeuralNetwork(this);
    }

    @Override
    public String toString() {
        return this.getClass().getSimpleName() + " [" + " Input-Layer: " + listNNInput + "
Output-Layer: " + listNNOutput + " ]";
    }


    public static <V> Phrase<V> jaxbToAst(Block block, AbstractJaxb2Ast<V> helper) {
        List<Value> values = helper.extractValues(block, (short) 2);
        ListCreate<V> inputLayer =
            (ListCreate<V>) helper.extractValue(values, new
             ExprParam(BlocklyConstants.INPUT_LAYER, BlocklyType.STRING));
        ListCreate<V> outputLayer =
            (ListCreate<V>) helper.extractValue(values, new
             ExprParam(BlocklyConstants.OUTPUT_LAYER, BlocklyType.STRING));

        List<AiLink<V>> listNNLinks = new ArrayList<>();
        for ( Expr<V> inputNode : inputLayer.getExprList().getEl() ) {
            for ( Expr<V> outputNode : outputLayer.getExprList().getEl() ) {
                AiLink<V> oneInputOutputLink = new AiLink<V>(inputNode, outputNode, 0);
                listNNLinks.add(oneInputOutputLink);
            }
        }
        return AiNeuralNetwork.make(inputLayer, outputLayer, listNNLinks,
               helper.extractBlockProperties(block), helper.extractComment(block));
    }

    @Override
    public Block astToBlock() {
        Block jaxbDestination = new Block();
        Ast2JaxbHelper.setBasicProperties(this, jaxbDestination);
        Ast2JaxbHelper.addValue(jaxbDestination, BlocklyConstants.INPUT_LAYER,
           getListNNInput());
        Ast2JaxbHelper.addValue(jaxbDestination, BlocklyConstants.OUTPUT_LAYER,
           getListNNOutput());
        return  jaxbDestination;
    }

}
```

A.3 Changelog

```
commit: cc7bcdbc7, author: vlebedynska, date: 2020-08-05 00:22:12

  #24 - various bugfixes


  commit: 37e710841, author: vlebedynska, date: 2020-08-03 00:36:36

  #24 - various bugfixes:
  - modified cursor behaviour on rl q learning modal header, modal
    dialog and hovering over player buttons
  - changed the display of the node description in neural network
    popup if a node has no port defined
  - fixed calculation of theta angle for the correct robot rotation
    after drawing the optimal path
  - added more transparency value for path visited
  - fixed server error that caused an empty configuration toolbox to
    load


  commit: 43ccf7f68, author: vlebedynska, date: 2020-08-02 11:15:41

  #24 - preparing of the test environment for testing rl and neural
    network feature. Added libraries needed for the offline
    installation of the Open Roberta Lab.


  commit: 5d2608f2c, author: vlebedynska, date: 2020-08-02 11:09:54

  #22 - minor bugfixes for RGB channel values and color parameters of
    the color sensor block
  - minor design adjustments in the NN popup - added a background
    image, changed font, disabled dragging of the background image
  - minor changes in program toolboxes for beginner and expert

  #21 - updated RL Eisenbahn map
  - commended out unnecessary console logging
  - execution queue of rl blocks adjusted

  #23 - implemented new class hyperparameterTuning and
    qLearnerParameterOptions, ProblemParams, TestInputData,
    TestResultfor interfaces for testing of the results of q-learning
    algorithm.
  - added hyperparametherTuningTest.ts for result handling


  commit: d7fa41cf3, author: vlebedynska, date: 2020-08-01 00:58:02

  #22 - minor bugfixes for RGB channel values and color parameters of
    the color sensor block
  - minor design adjustments in the NN popup - added a background
    image, changed font, disabled dragging of the background image
```

```
#21 - added new RL map: stadt_End and updated Wald_Labyrinth and
  Eisenbahn_Design_End.svg
- implemented dynamic consideration of episodes and time entered by
  user
- minor refactorings for path finding from the svg map
- added a new class hyperparametherTuning.ts, which is a parameter
  test class for the Q-Learning-Algorithm.


commit: ad4253d4e, author: vlebedynska, date: 2020-07-29 23:56:49

#22 - extracted Color enum into class AiColorUtils
- added TODOs for future code improvements
- implemented new output node type: "led" in class AiOutput, added
  it to robotCommon.yml, new functionalities implemented in
  interpreter.robotMbedBehaviour.ts
- first draft of NNAlgorithm added to git - will be deleted in the
  next commit
- added new popup header to the neural network modal
- added duration and frequency as optional parameters to the node
  model in models.ts, implemented new properties in NodeImpl
  constructor in aiNeuralNetworkModule.ts
- added new colors to colorsMap map in aiNeuralNetworkUI.ts for
  coloring nodes in neural network


commit: 9609a8f77, author: vlebedynska, date: 2020-07-28 23:58:51

#22 - added new constants for the EV3 color sensor and added them to
  the enum Colour
- implemented the distinction between different types of colour
  sensor input: light, rgb and base colors in AiInputNodeColourSensor
- two additional output node types added and implemented :
  ai_nn_output_node_text and ai_nn_output_node_sound in AiOutput
- new blocks added to robotCommon.yml
- changed type of the ai_nn_output_node_text to OUTPUTNODE in
  blocks_compressed.js
- implemented two new input node types in
  interpreter.interpreter.ts, robotWeDo- and robotMbedbehaviour.ts
- for using EV3 as a display for displaying output results, a
  workaround for clearing display implemented, because the display
  actions are not syncronized
- bugfix for displaying text on the ev3 brick: in the simulation for
  robot.ev3.js, display.clear was moved to be the first statement
  that has to be executed


commit: 2d6eb3c47, author: vlebedynska, date: 2020-07-27 23:58:14

#10 - updated RlEnvironment class: changed the type of start and
  finishNodes to Phrase, added a new property map, changed the class
  constructor and the jaxbToAst method
- added two new properties to RlGainExperience qLearningEpisodes and
  qLearningTime, used them in the constructor and in the visitor
  method in Ev3StackMachineVisitor
```

```
- updated RlObstacle class: changed the type of start and
  finishNodes to Phrase
- added new Blockly constants - MAP, QLEARNING_EPISODES,
  QLEARNING_TIME
- added rl_obstacles_easy_list to robotCommon.yml

#21 - implemented rl map selection functionality to
  createQlearningEnvironment method in
  interpreter.robotMbedBehaviour.ts
- moved creating module from robotMbedbehaviour constructor to the
  createQLearningEnvironment method
- in interpreter.interpreter.ts
- modified collecting of finish and start nodes in
  CREATE_Q_LEARNING_ENVIRONMENT via stack
- added episodes and time properties to RUN_Q_LEARNER opc


commit: c492066c4, author: vlebedynska, date: 2020-07-26 21:52:23

#10 - added and implemented new block ai_easy_list, which allows to
  easily add neurons to the layers of the neural network. To make the
  block compatible as a block of a list type, some minor adjustments
  have been made to the ListCreate class (added default list type);
- removed function for placing threshold on stack from
  Ev3StackMachineVisitor, as it's not longer needed
- added new blocks_compressed.js file with updated and new neural
  network blocks and updated reinforcement learning blocks
- updated ev3.program.toolbox.beginner.xml &
  ev3.program.toolbox.expert.xml with new blocks
- added default threshhold value to CREATE_INPUT_NODE opc, because
  no threshold value is transferred from the backend side
- added new property to the Player class of the Neural Network
  Module: isPlaying, created getter and setter for it

#21 - new background for reinforcement learning added - forest
  labyrinth
- adjusted the behaviour of the robot after releasing the pause
  button in processNeuralNetwork function


commit: 10a3eec64, author: vlebedynska, date: 2020-07-25 00:42:02

#10 - Improvement of the module for neural networks (in progress)
- added additional information to AiInputNodeColourSensor > now it
  transfers the name of the colour sensor displayed on the neural
  network playground
- removed some class definitions from roberta.css that defined the
  styles for aiNeuralNetworkPopup because they influenced the dynamic
  creation of the neural network
- added some class definitions that define the behaviour of the
  mouse, the modal-header background and the buttons on the neural
  network playground
- in simulation.js added a new function to the constructor of
  RobotMbedBehaviour - setPause. This function is now passed to
  RobotMbedBehaviour and is used to pause program execution and robot
```

```
    activity when the pause button in the neural network modal is
    pressed.
  - new file bound to main.js - the player for the neural network
    module
  - added some test files to test the behaviour of -webkit-filter-
    properties in chrome: detected -webkit-filter- doesn't work with
    svg elements

- added a new object to the aiNeuralNetworkModule.ts - Player
    responsible for event handling when the play or pause button is
    pressed in the neural network module
- added a css class to the svg - svgViewBoxNNModule and defined the
    viewbox. Now the size of the viewbox is set dynamically depending
    on the bbox of the svg. Bbox takes the minimum square around all
    svg elements and defines x, y coordinates and width and height

- on the aiNeuralNetworkUI.ts:
  - the class aiNeuralNetworkUI.ts extended by EventTarget
  - added new colorsMap, which maps colors from the color sensor to
    colors on the UI (they are needed to map colors from the color
    sensor to the corresponding nodes in the neural network)
  - minor refactorings in the drawLayer function: UI of the nodes is
    moved to the function addNodeColor; text anchor added for the
    descriptions of the input layer nodes
  - added new function to the constructor - drawPlayer, which is
    responsible for drawing all player elements (play and pause button)
    and for event handling from the UI

- in the linkUI.ts:
  - added css classes to the slider shape depending on mousedown and
    mouseup events

- in models.ts added additional optional property color for the
    interface node
- implemented the new property color in the NodeImpl class

- created and implemented the new class Player (player.ts), which is
    responsible for event handling when the play or pause buttons are
    pressed

- adapted create_input_node_colour_sensor opc, so that it now creates
    a node directly with stmt data;
- added new function in the robotMbedBehaviour to the constructor
    simSetPause, which is transferred from simulation.js.
- bound event listeners for "play" and "pause" events to the function
    processNeuralNetwork. For the corresponding event setBlocking and
    simSetPause are set to true and false respectively. This causes
    program execution and robot activity to be (un)paused.
- implemented additional functionality: the sensor values in the
    input and output nodes are only updated if the value of the sensor
    in the input layer has changed. If not, the output layer is not
    updated.


 commit: 081435ded, author: vlebedynska, date: 2020-07-24 00:45:35
```

```
#10 - Improvement of the module for neural networks (in progress)
- Refactoring of the ai.neuralNetwork.ts module > Restructuring of
  the ai.neuralNetwork.ts > Outsourcing of all classes into different
  files, refactoring and extraction of properties into interfaces (in
  models.ts) and implementation in impl classes
- added new files in main.js, generated new js files and added them
  to OpenRobertaServer staticResources
- added a new function in the aiNeuralNetworkModule to normalize
  nodes that the Neural Network Module receives from the stack in
  robotMbedBehaviour. These nodes are transferred to the nodes that
  the Neural Network Module needs.
- added new feature to the aiNeuralNetworkUI > now the value of each
  node is displayed inside the node itself The drawLayer function
  implements an event listener - every time the value of the node is
  changed, the text inside the node is updated.
- corrected calculation of the slider value considering the start
  and end point
- improved display of the slider value above the slider shape: for
  integer values the superfluous decimal places are hidden


commit: d3e5bf4cf, author: vlebedynska, date: 2020-07-23 16:21:33

#10 - improving neural network module (work in progress)
- added new property nodeData as JSON object to AiInput.java,
  AiInputNodeColourSensor.java, AiOutput.java > now the additional
  information like the name of the input / output node is transferred
  to the Neural Network Module, so that it can be displayed as a
  description for each input and outputNode
- the function create_input_node in interpreter.interpreter.ts has
  been updated: it no longer creates an additional node object, but
  adds the properties directly the node.
- added nodeData to the input- and output nodes visitor functions in
  Ev3StackMachineVisitor
- added a new functionality in the ai.neuralNetwork.ts
  createNeuralNetwork - addNodesName.
- implemented a new function addNodesName - each node in the layer
  now gets an additional parameter node.name
- to the drawLayer function added the functionality to set node's
  name / node's description dynamically

- implemented a functionality to display a current slider value
  above the slider shape while the slider shape is moved
- for this the LinkUI class is extended by the property
  sliderValueText. When creating a new SVGSlider, this property is
  passed to the SVGSlider.
- updated sliderValue function -> added new functionality
  updateSliderValueText which updates the text field in the svg
- added functionality for positioning each sliderValueText above the
  current slider shape

- minor changes in the layout of the neural network module
- popup size changed
- modified roberta.css > class simConfigNeuralNetworkModal modal-
  dialog updated and the duplicate removed
```

```
  - added to the styles.css new classes inputNodeName, outputNodeName
    to display the names of input and output nodes; added a new class
    inputNode, which is responsible for the design of the inputNodes in
    general


  commit: a85c93bc9, author: vlebedynska, date: 2020-07-22 23:47:48

  #10 - improving neural network window (work in progress)
  - started implementing the ability to see the description for each
    node in the popup. For this reason, a new property nodeData was
    added to all nodes, which transports information such as the name
    of the input/output sensor and its port
  - extended visitAiInputNode with a new property - ultrasonic
  - implemented the function addNodesName in the ai.neuralNetwork
    module, which now extends the parameters node.name by sensor value
    name and port
  - extended the interpreter.interpreter.ts and updated an extended
    input node function with the new data parameter.
  - minor refactorings

  Neural network - slider - status update. Added correct initial
    position of the slider shape.


  commit: 580fd9e1d, author: vlebedynska, date: 2020-07-21 23:53:04

  #8 - integration of the RL popup 2.0 in Open Roberta Lab (finished)
  - completed implementation of setting the robot position to the
    beginning of the optimal path:
  - extended resetPose function in robot.js with additional optional
    parameter - pose;
  - extended updateBackground and setBackground functions by the
    optional parameter poses. If the poses array is not empty,
    resetPose function is called from robot.js.
  - extended aiReinforcementLearningModule.ts by the functionality to
    calculate the pose parameters for the robot (x-point, y-point and
    theta); These are scaled depending on the viewbox-size which passed
    to the simulation and adapted to the canvas coordinate system.
  - extended models.ts by an interface Pose
  - added functionality to drawOptimalPath to scale the scene to be
    transmitted.
  - added qValueLookup.js - this file is automatically generated from
    qValueLookup.ts when compiling.

  #10 - started to improve neural network window (work in progress)
  - implemented the consideration of startPoint and endPoint
    parameters when dragging the slider shape;


  commit: fa5fe8110, author: vlebedynska, date: 2020-07-20 00:18:03

  #8 - integration of the RL popup 2.0 in Open Roberta Lab (work in
    progress)
  - completed the implementation of the functionality that pauses the
    rl algorithm when someone hides the popup
```

```
  (addEventListenerToRLPopup, added new property to the
  QLearningAlgorithmModule constructor - jQuery popup selector),
  changed the visibility of the pause() function - now it can be
  called by closing the popup.

- Bugfix: svg with the final path could not be copied to the robot
  simulation for some reason - it got broken during the transfer
  process:
- added Eisenbahn_broken.svg for testing purposes, added correct svg
  file (svg_ok) and broken svg file for comparison purposes
- configured the test environment in index_2.js - created and
  implemented async function loadSVG() and bound all rl learning
  functionalities
- reason for this behaviour:
(1) When transferring svg via URI, the svg playground broke because
  of plaintext elements > added default text values
(2) When transferring svg via URI tspan elements could not be
  processed - this is now fixed in the drawOptimalPath function of
  the aiReinforcementLearningModule
(3) The svg transmitted via URI was also broken because the
  namespace of svgjs - xmlns:svgjs="http://svgjs.com/svgjs"- and its
  references in certain elements - like this one:
  svgjs:data="{"leading":"1.3"} - were unknown. For this reason, they
  were removed from svg by two regular expressions.
- Start of positioning the robot to the optimal path start

#21 - improved function pause: now if the state is already STOP, it
  is no longer possible to change to the PAUSE state


commit: 860f643ed, author: vlebedynska, date: 2020-07-18 23:48:21

#8 - integration of the RL popup 2.0 in Open Roberta Lab (work in
  progress)
- started implementing a functionality that pauses the rl algorithm
  when someone hides the popup (in aiReinforcementLearningModule.ts,
  in progSim.controller.js and in interpreter.robotMbedBehaviour.ts)

#21 - update svg Eisenbahn_Design_End.svg due to minor design
  changes
- new class qValueLookup created, implemented qValueLookup.ts in
  main.js and referenced it in require.js
- added new css styles for rho, rho-active, rho-text-active and for
  stars representing the current qValue: star, newStar, oldStar
- added new interface QValue to models.ts and a new property
  highestQValue to the interface QLearningStep
- in the class QValueStore added a new private local variable -
  highestQValue and a new functionality to check and update the
  current highest qValue.
- added new functionalities in the class qValueLookup
  (getOldNumberOfStars, getNewNumberOfStars,
  getAndUpdateNumberOfStars). The aim of this class is to calculate
  the current number of stars based on the current qValue and the
  current highest qValue.
- implemented class Key, which is a helper class for qValueLookup
- in the class Visualizer:
```

```
- added new functionalities to visualize the current and previous
  qValue for the current action (showCurrentQValue, bound
  qValueLookup class) and modified the presentation of rho values
- bug fix in onQLearningStep - changed newQLearnerStepData from
  global to local variable because it caused problems with styling
  elements of the current qLearner step (it was overwritten by the
  new qLearningStepDate before the animation of the previous step was
  finished)
- minor bug fixes for finding elements in the svg playground


commit: ae0ae9f38, author: vlebedynska, date: 2020-07-17 00:12:01

#8 - integration of the RL popup 2.0 in Open Roberta Lab (work in
  progress)
- added new simBackground - Eisenbahn_Design_End.svg and updated
  styles.css with classes for visualizing of rho values
- redesigned function updateBackground in simulation.js. Now it
  creates a new canvas element and converts the svg loaded from the
  popup into the png format. The reason for this change is that the
  robot created artifacts on the background when rendering.

#21 - improved workflow for sending the final svg to the simulation
  in aiReinforcementLearningModule.ts:
- changed execution of the updatebackground function to async;
- added viewbox and size property to svg that is sent > background
  is cropped and sent to the simulation without the navigation panel
- change of the svg structure led to minor refactorings for finding
  elements in visualizer.ts
- added showCurrentQValue function, which displays a q-value of each
  step in visualizer.ts
- improved display of the rho value: now not only the text is
  changed in each step, but also the colour of the background
  rectangle
- added some test parameters to index.html and index_2.js for
  testing purposes


commit: 4f15b7374, author: vlebedynska, date: 2020-07-16 00:46:51

#8 - integration of the RL popup 2.0 in Open Roberta Lab
- workflow adapted for asynchronous execution of
  setUpQLearningBehaviour(), runQLearner(), drawOptimalPath() (work
  in progress)
- refactored rl method signatures in interpreter.aRobotBehaviour.ts
  and sub-classes
- added uptdateBackground function to QLearningAlgorithmModule
  constructor

#19 - moved QlearningAlgorithmParameters, ResultState,
  OptimalPathResult, Obstacle and TakeActionResult interfaces from
  aiReinforcementLearningModule.ts to models.ts
- moved RlUtils from aiReinforcementLearningModule.ts to qLearner.ts
- minor refactorings in qLearner.ts
- in the visualizer.ts:
```

```
- added new class properties, added get method for _svg, extracted
  newQLearnerStepData as member variable fromQLearningStep function
- refactored drawOptimal method: split in separate methods -
  checkAndUpdateOptimalPath(), getCombinedPath() and
  drawFinalOptimalPath()
- minor refactorings and bug fix in getCombinedPath() method because
  the currentPathArray changed the original array and not the local
  one
- added new function drawCurrentOptimalPathOneMap, which now draws
  the current optimal path on the map from the start to the finish
  node
- added path animations to drawFinalOptimalPath() and
  drawCurrentOptimalPathOnMap()
- added new style in style.css for the current optimal path on the
  map


commit: a91e7719e, author: vlebedynska, date: 2020-07-15 23:59:26

#8 - integration of the RL popup 2.0 in Open Roberta Lab
- workflow addapted for async execution of runQLearner() and
  drawOptimalPath()
- minor changes in interpreter.interpreter.ts
- added svglookup.js to main.js


commit: 676206db2, author: vlebedynska, date: 2020-07-13 22:53:28

#20 - finished implementing caching mechanism for elements
  (elements, texts, paths) that have already been found and used:
- refactored and improved svglookup.ts (better encapsulation of
  functions)
- implemented using of new caching functionality in visualizer.ts


commit: 5d668121f, author: vlebedynska, date: 2020-07-12 23:59:08

#8 - integration of the RL popup 2.0 in Open Roberta Lab
- added style.css to staticResources, added the svg background
- added new configurations and files to tsconfig2Server
- generated files from ts to js using tsconfig2Server
- connected new files from Reinforcement Learning Module to the
  global project via require.config in main.js
- bug fixes after the implementation of the new module in Open
  Roberta Lab (in particular in interpreter.robotMbedBehaviour.ts) +
  some minor refactorings in aiReinforcementLearningModule.ts (e.g.
  changing the names of the modules for a better export from ts to
  js)
- cleaned up svgPlayground in WedoInterpreter - all files no longer
  needed deleted

#19 - started the optimization of qLearner steps visualization:
  created a new class svglookup.ts to optimize performance by caching
  svg elements
- implemented the lookup algorithm for text elements
```

```
commit: 764bd5a66, author: vlebedynska, date: 2020-07-11 00:04:58

#19 - updated the test svg -> all lines and polylines changed to
  paths
- in index_2.ts (interface to the RobotMbedBehaviour) added a call
  of the function drawOptimalPath
- minor refactorings to adapt the interface to the
  RobotMbedBehaviour: modified obstacleList from Array to Array
- in the runQLearner function extended value stored by the
  qLearningSteps array: besides the information about the
  qLearnerStep itself, the optimal path is calculated and passed to
  the instance of the PlayerImpl
- added an event listener to the drawOptimalPath function, which now
  reacts to a "stop" event, finds the optimal path and passes the
  result to the visualizer for drawing
- minor additions in the Player interface: timer added as a new
  property and the values of qLearnerSteps property changed according
  to the changes in runQLearner -> this led to minor refactorings in
  playerImpl.ts
- fixed a bug in filterOutNotAllowedActions, because the previous
  output of the function was null.
- in the visualizer, Shape was changed to Path, because there are no
  more lines or polilines on the svg
- adjusted the setMarker function, because the markers were too big
  for this map
- refactored onQLearningStep function as newQLearnerstep now comes
  as one object consisting of two parts
- added showCurrentOptimal function to show the current optimal path
  on svg playground
- two bug fixes in the function resetAllValues

-


commit: 8c008f0a1, author: vlebedynska, date: 2020-07-10 00:29:57

#19 added two new classes for line follower - inner and outer paths
- made qLearnerAlgorithm in aiReinforcementLearningModule.ts global
  to be able to access the instance in different functions
- outsourced search and creation of optimal paths in the qLearner
  and visualizer
- started to connect drawOptimalPath to the interface of
  RobotMbedBehaviour
- added second parameter in function onTimerClick:
  executionDuration. Depending on the total execution duration all
  animations are now calculated. Therefore the function
  createAndDispatchEvent was changed: it now also transfers the
  executionDuration as detail parameter.
- new button - startForOneStep - added and it's functionality
  implemented in visualizer and timer (function playOneTick).
- added a help function in qLearner to find the optimal path
- refactored the function callTick in timerImpl.ts : (1) added
  isRunning flag to prevent a tick being called by different
  threads/twice, (2) changed setInterval to setTimeout to allow
  robust control of the timer running state
```

```
- added new properties to the Visualizer to store values centrally,
  the functions that implement these values have been refactored
- extracted actions from visualizeActionOnMap into the smaller
  single functions (e.g. showCurrentRho, showCurrentStartNode,
  resetAllValues etc.)
- added a function for resetting of values after each step
- added animation sequence to the onQLearningStep function
- implemented new function delay Visualizer
- refactored function drawOptimalPath to drawPath and moved it from
  aiReinforcementLearningModule.ts to the Visualizer
- moved findPathWithIds from RLUtils to the Visualiser class
- further minor refactorings in aiReinforcementLearningModule.ts


commit: 6e3a11d0a, author: vlebedynska, date: 2020-07-08 23:59:19

#19 - new extended test svg added
- changed the qLearningParams to optimize the output of the
  algorithm
- added new class in style.css - node-not-visited, minor style
  changes
- changed the test number of episodes and the total time
- implemented a new function in utils.ts to convert seconds to the
  common time format: hh:mm:ss
- in the class Visualizer added new properties to store animation
  elements from the initial and previous state
- animation functionality implemented to each step
- added a new function setInitialValuesOnMap to initialize initial
  algorithm values setup before starting the algorithm and
  implemented it in initialize function of playerImpl.ts
- outsourced functionality to set start and finish state, set total
  time and set total number of episodes from setInitialValuesOnMap
  function
- added functions to visualize the start node of the algorithm and
  the finish node in the navi bar
- added animation to the active stroke, added parameters for easier
  understanding of the animation direction
- started developing animation timeline based on TS promises
- implemented help delay function


commit: 00d5f2e51, author: vlebedynska, date: 2020-07-07 23:55:12

#19 - added function to visualise an action on the map
- added stroke animation within one step
- help function implemented in Utils to calculate the length of the
  current shape


commit: 26cada53b, author: vlebedynska, date: 2020-07-06 23:50:59

#19 - added styles.css and defined classes for active, inactive,
  visited nodes, paths and lines
- values added for the enum Rho for its string representation in UI
- function onQLearningStep updated: now another parameter is passed
  to the visualizer: the length of the array qlearningSteps
```

```
- Visualizer class extended with the parameters nodeStartOnMap,
  nodeFinishOnMap, path and line to store UI elements from the
  previous qLearningStep
- functionalities for visualisation of QLearningStep added: view
  changes of UI components in svg map
- minor refactorings in aiReinforcementLearningModule.ts and in
  qLearner.ts: e.g. startStateQlearner renamed to startFinishState


commit: 419144c6d, author: vlebedynska, date: 2020-07-05 23:59:19

#19 - Added external react.js libraries (react.development.js &
  react-dom.development.js) to the project and configured the
  required dependencies
- returned to svgdotjs library and started implementing animations
- visualizer directly connected to the player (changes of the
  initialize function in the Player interface)
- default speed value added in the playerImpl.ts, added visualiser
  via initialize function and implemented event listeners for
  playerStarted, playerStopped and playerPaused.
- bugfixes: changed this > that for correct event handling
- removed qLearnerView - react implementation of the module, because
  of strategy changes - it's not needed for now anymore.
- in timerImpl.ts changed callTick, so it clears interval every time
  the function is called, added variable speed for dynamic speed
  changes
- implemented event logging for stop, play, pause states, tick and
  state changing
- enhanced functions for timer stop, pause and play. The function
  play is getting new parameter now - speed, which regulates the
  speed of the algorithm.
- added error handling for the case if the button is pressed more
  then once (function updateRunningState)
- extended visualizer.ts to EventTarget
- implemented addEventListeners function in visualizer.ts, which
  catches all click events the user has made on UI elements
- implemented startPlayer, stopPlayer and pausePlayer functions that
  dispatch events if the buttons are pressed
- cleaned tsconfig.json from react libraries
- renamed utils and visualiser


commit: a048afef0, author: vlebedynska, date: 2020-07-03 23:43:06

#19 - Connection of require.js and react framework to RL module (in
  progress)
- added test react project
- downloaded and connected react libraries for TypeScript


commit: 015c1c359, author: vlebedynska, date: 2020-07-02 21:35:03

#19 - Connection of require.js and react framework to RL module (in
  progress)
```

```
- playerImpl: implemented play function and onTimerTick, bound event
  handling, added qLearningSteps, timer and view properties to
  constructor
- added qLearningSteps property to Player interface
- added some test files (index.html, index.js, test.jsx,
  qLearnerview.js, map.js)
- added problem, alpha, gamma, rho and nu properties to the
  constructor of QLearningAlgorithm
- refactored runQLerarner function in
  aiReinforcementLearningModule.ts
- minor refactoring of variable names


commit: 0ca215d96, author: vlebedynska, date: 2020-07-01 19:49:26

#19 - Connection of require.js and react framework to RL module (in
  progress)
- svgdotjs library connected via require.js
- added the require.config-function to main.js
- Changes to tsconfig.json (changed attribute "paths", added new
  compiler option "sourceMap" and changed the option "target" from
  es5 to es2017)


commit: c0b32d2a1, author: vlebedynska, date: 2020-06-30 23:58:52

#19 - added some test svg backgrounds
- implemented some functions with dummy values to investigate how
  react and svg proxy selector exactly works with
  PopUPDesign_Minimal.svg
- index_1.js implemented as interface to RobotMbedBehaviour
- created and implemented functions for initializing q-learning
  environment (separated visualization of data, from data processing
  and data storage)
- moved the components responsible for the UI from
  aiReinforcementLearningModule.ts to Visualizer.ts and connected
  them ( e.g. getActions(), processNotAllowedActions) with
  aiReinforcementLearningModule.ts
- new property - time - added to interface Clock, changed properties
  of QLearningStep to readonly
- started to implement the playerView.jsx
- extracted one single step of the qlearning algorithm into function
  qLearnerStep, bound event handling every time the value of the
  qlearning step changes
- moved components of qlearning algorithm from
  aiReinforcementLearningModule.ts to qLearner.ts
- implemented the Timer class which is responsible for time counting
  and management and which will be connected to player control
  buttons in the qlearning popup
- added three new functions to Utils.ts to convert data received
  from RobotMbedBehaviour to the data required by the RLModule and a
  helper function to filter out not allowed from the allowed actions
- implemented functions for visualising of not allowed actions in
  the Visualizer.ts
```

```
commit: 6ffdd0763, author: vlebedynska, date: 2020-06-28 23:59:07

#19 - defined data models as interfaces for ProblemSource, Player,
  StateStatus, SectionState, QLearningStep, QLearnerConfiguration,
  QValueStore, Action, ProblemState, Clock in models.ts
- defined enums for Rho, Nu, RunningState
- renamed Clock to ClockImpl class and extended it by Clock
  interface
- renamed Player to PlayerImpl class and extended it by Player
  interface
- tsconfig.js updated: added new files to compile, added
  es2015.promise lib, changed module to amd from commonjs
- created class Utils and implemented file_get_contents function
  using Promise concept
- class Visualiser extends ProblemSource interface
- in the class Visualiser implemented new functions using Promise
  concept: preload, loadSvg, which now allows asynchronous Visualiser
  instance initialisation
- implemented functions getSections for getting all available
  actions for a given problem, scaleSVGtoSize for scaling the given
  svg to size needed, help function fitInNewSize
- Size interface implemented


commit: 398d36d7d, author: vlebedynska, date: 2020-06-28 00:33:13

#19 - created classes Player, Visualizer, Clock
- added test svg to the project
- created new tsconfig for compiling typescript files to JavaScript
- created index.ts with dummy data for testing and bound the
  aiReinforcementLearningModule.ts & svgdotjs library to it
- started to implement the Promise concept in the Visualizer.ts


commit: b38f991db, author: vlebedynska, date: 2020-06-27 15:05:21

#8 - added svgdojs library


commit: 598ccbc8c, author: vlebedynska, date: 2020-06-27 11:56:30

#19 - exploring the reactjs technology for optimal value changes in
  the RL popup
- imported reactjs and react-svgmt libraries for dynamic value
  changes and comfortable manipulation of svg
- created the test project and implemented the timer that changes
  the text value in the svg using reactjs
- added the svg with minimal design to be used prospectively in the
  RL module and implemented the test timer there


commit: 98831eb99, author: vlebedynska, date: 2020-06-25 18:13:30

#11 - Refactoring of the NN module (finished)
- aiNeuralNetworkModule added to main.js and tsconfig2server.json &
  tsconfig.json
```

```
- connected new neural network module to processNeuralNetwork
  function in interpreter.robotMbedBehaviour.ts
- generated new js files from ts files
- added methods from aiReinforcementLearningModule to
  interpreter.aRobotBehaviour.ts (#8)
- various bux fixes after the implementation of
  aiNeuralNetworkModule into robotMbedBehaviour
- property externalSensor changed to more abstract "value" for
  create_input_node_colour_sensor opc (#8)
- opcs for processing the blocks create_q_learning_environment,
  setup_q_learning_behaviour, run_q_learner,
  q_learning_draw_optimal_path in interpreter.interpreter.ts added
  (#8)


commit: 85de23cda, author: vlebedynska, date: 2020-06-24 19:41:12

#11 - Refactoring of the NN module (work-in-progress)
- property "externalSensor" in interpreter.interpreter.ts changed to
  more abstract "value"
- aiNeuralNetwork and AiNeuralNetworkUI connected to
  AiNeuralNetworkModule
- created two properties in AiNeuralNetwork: layers and links and
  generated getters for them
- new functionality for calculating the values of the nodes of the
  neural network implemented
- implemented new functionality for getting input and output layers
  of the neural network
- created the new class Ev3MotorOutputNode, which extends the class
  Node. The new class has all properties that current
  Ev3MotorOutputNodes need: port and type. In the future the output
  node types shall be better differentiated.
- added drawing neural network to the constructor of
  AiNeuralNetworkUI
- integration of aiNeuralNetworkModule into
  interpreter.robotMbedBehaviour.ts
- deleted old code for processing of aiNeuralNetwork in
  interpreter.robotMbedBehaviour.ts
- bound aiNeuralNetworkModule to processNeuralNetwork function in
  interpreter.robotMbedBehaviour.ts


commit: 777b37142, author: vlebedynska, date: 2020-06-23 23:28:04

#11 - Refactoring of the NN module (work-in-progress)
- new property added to the class Node: position Y
- created getters and setters for class Node
- event handling added to LinkUI class
- extracted LinkUI slider values in constants
- added functionality to create path for link from link
- added activate and deactivate link functionalities
- added functionality to identify x an y coordinates of a node
- added readonly property for all properties of SVGSlider
- created getters for sliderShape and path members in SVGSlider
- created get method and implemented set method for sliderValue
  (responsible for event handling when the slider value changes)
```

```
- extended createSlider method in SVGSlider for the event handling
- SVGUtils created and implemented: e.g. path/point calculations


commit: 095f6f5cc, author: vlebedynska, date: 2020-06-22 23:32:40

#11 - Refactoring of the NN module (work-in-progress)
- moved design/draw methods to LinkUI class.
- implemented event handling fot he class LinkUI
- extended setSliderValue
- implemented class Link


commit: 930c6cf95, author: vlebedynska, date: 2020-06-22 01:18:59

#11 - Refactoring of the NN module started
- started adding data types to class members, parameters of the
  methods
- moved functionality from RobotMbedBehaviour.ts &
  RobotMbedBehaviour.js to aiNeuralNetworkModule >
  ai.neuralNetwork.ts
- better encapsulation - new classes created and filled with
  functionalities:
- Draggable (responsible for dragging objects)
- SVGSlider (responsible for the slider to adjust the link weights)
- AINeuralNetworkUI (responsible for drawing links and nodes)
- AiNeuralNetworkModule (responsible for central data storage and
  initialization the main SVG.SVG())
- removed old test files from the aiNeuralNetworkModule


commit: cd26847f6, author: vlebedynska, date: 2020-06-14 23:45:56

#18 - extracted QLearning from the interpreter.robotMbedBehaviour
  into a module aiReinforcementLearningModule
- connected the new module with the robotMbedBehaviour
- added ai.qlearning as a module in main.js
- moved all functionalities from js to ts
- comprehensive refactoring in aiReinforcementLearningModule
  including data typing and migration of variables into interfaces
- added aiReinforcementLearningModule to tsconfig.json and
  tsconfig2server.json
- generated .js files for aiReinforcementLearningModule from .ts
- cleaned up progSim.controller.js from temporarily added QLearning
  functionalities
- #11 started to refactor aiNeuralNetworkModule


commit: d91157155, author: vlebedynska, date: 2020-06-13 15:45:57

#8 - client-side implementation of RL blocks
  (ai_q_apply_learning_rule, ai_q_barrier_mountain,
  ai_q_draw_best_path, ai_q_learner_config,
  ai_q_learning_states_and_actions_map)
- implemented the processing of new blocks in the stack machine
  (interpreter.interpreter.js)
```

```
- implemented the q-learning algorithm in robotMbedbehaviour
  (migrated RL functionalities from simConfigRLQLearningModal) and
  linked it to the RL processing functions (temporary version in js)
- created a new js class QLearningAlgorithmModule to centrally
  store, process and visualize the data required for the Qlearning
  algorithm
- updateBackground function passed from the SIM module to the
  QLearningAlgorithmModule and called in the QLearningAlgorithmModule
  in the drawOptimalPath() function
- temporarily commented out .ts files in tsconfig.json - and
  generated a new interpreter.constants.js


commit: 523a3f51c, author: vlebedynska, date: 2020-06-11 12:20:25

#14 - implemented visitor methods of Ev3StackMachineVisitor,
  Ev3SimValidatorVisitor and Ev3UsedHardwareCollectorVisitor for
  classes: RlEnvironment, RlGainExperience,
  RlObstacle(Ev3StackMachineVisitor only) and
  RlSetUpQLearningBehaviour
- created new class for RlDrawOptimalPath block
- implemented make method, JaxbToAst and astToBlock methods for
  RlDrawOptimalPath (#6)
- added new astTest for RlDrawOptimalPath block
- added test resources for RlDrawOptimalPath block
- added new constants to constantsSource.txt


commit: 383ae1f1f, author: vlebedynska, date: 2020-06-10 22:28:23

#6 implemented astToBlock methods for RlGainExperience, RlObstacle
  and RlSetUpQLearningBehaviour


commit: c8772c0df, author: vlebedynska, date: 2020-06-10 22:05:57

#6 refactoring of RlEnvironment, RlGainExperience, RlObstacle and
  RlSetUpQLearningBehaviour classes
- Utils methods moved to a new class RlUtils.java


commit: 87bd20e97, author: vlebedynska, date: 2020-06-10 02:29:19

#6 - new classes created for RlEnvironment, RlGainExperience,
  RlObstacle and RlSetUpQLearningBehaviour
- implemented jaxbToAst and "make" methods for all of them
- astToBlock method implemented for RlEnvironment
- added new Blockly type Obstacle
- new Blockly constants added
- added new blocks to robotCommon.yml
- created Ast tests for new reinforcement learning blocks
- changed test resources for RL blocks
- updated the ev3 toolbox for beginners
- updated blockly resources with changes to reinforcement learning
  blocks
```

```
commit: 3e2424e9b, author: vlebedynska, date: 2020-06-08 09:52:47

#16 - QLearning algorithm integrated into the simulation:
- transferred the algorithm from the test environment to the
  simulation
- created a new button to open the qlearning popup
- created a new popup for the visualisation of the learning process
- added style definitions for the new popup in roberta.css
- integrated event handling for hidden and displayed popup: if the
  popup is switched to the hidden mode, the content of the html is
  deleted, if the popup is switched to display, the q learning
  algorithm starts
- defined an updateBackground function in simulation.js that is
  called when the learning process has been completed and the optimal
  path has been created and drawn. It creates a new image from the
  source (string with ..-content) and calls the setBackground
  function on the onload event. -> the background is thus dynamically
  loaded as soon as the qlearning algorithm and the calculation of
  the best path and its drawing are completed.
- prepared test svg image (marsTopView.svg) with 8 nodes added to
  simBackgrounds


commit: ef22204fb, author: vlebedynska, date: 2020-06-07 16:55:37

#15 - amended drawing of the optimal path - the best single paths
  are now combined to one single path,
- improved function getBestAction - now it can select the best
  action only from the available actions and no longer randomly
- added function to hide all paths except the optimal one
- finishNode extracted as a constant
- added a predefined viewbox to the svg


commit: 74f049227, author: vlebedynska, date: 2020-06-07 12:10:56

#15 - added a function for dynamic generation of rewards for
  finishNode neighbors
- amend comment for the last commit: added an animation to the
  current action that shows the movement of the path to the target
  node (0c72bb6, 0c72bb6f4a69ab0830194b6f6de0b9afa5a2e018)


commit: 0c72bb6f4, author: vlebedynska, date: 2020-06-06 23:48:36

#15 - refactoring of the best path calculation according to the best
  q-values and its temporary storage
- added function to draw the best path
- added function to dynamically generate of statesAndActions matrix
  from the .svg file
- added function to find path by id
- added viewbox to svg2
- added new test svg file with Mars landscape and 8 nodes
```

```
commit: 2d0eafea3, author: vlebedynska, date: 2020-06-03 10:18:59

#15 - added the best path calculation according to the best q-values
  and its temporary storage


commit: 7396c3bf6, author: vlebedynska, date: 2020-06-01 23:59:59

#15 - added the svg test graph -
  Reinforcement_Learning_Playground.svg
- connected UI to the q-learning process: sync of stroke-width
  according to its q-value


commit: aa2cc491e, author: vlebedynska, date: 2020-06-01 17:12:07

#13 created AST test resources for five RL blocks:
  ai_q_apply_learning_rule, ai_q_apply_learning_rule,
  ai_q_apply_learning_rule, ai_q_learner_config,
  ai_q_learning_states_and_actions_matrix_3_x_3
- added new test class for RL blocks

#14 updated blockly_compressed.js & blocks_compressed.js
- added language support for RL blocks for English and German
- updated toolbox.beginner.xml for EV3 with new RL blocks


commit: 87046449e, author: vlebedynska, date: 2020-05-29 00:23:06

#12 implemented classes QValueStore, QLearningAlgorithm and Test;
various bug fixes in the first draft of QLearning algorithm


commit: eff38853f, author: vlebedynska, date: 2020-05-27 23:15:00

#12 implemented the ReinforcementProblem class


commit: 13888cea6, author: vlebedynska, date: 2020-05-26 23:10:05

#11 in progress: extracting ai functionality to a separate TS-Module
- created a new aiNeuralNetwork module
#12 started implementing the Q-Learning-Algorithm
- defined the basic structure


commit: 0eb7b1471, author: vlebedynska, date: 2020-05-26 00:07:18

#11 started to extract ai functionality to a separate TS-Module


commit: b1b274a05, author: vlebedynska, date: 2020-05-25 18:03:16

#10 when the slider is selected and moved, the line is visually
  highlighted
```

```
commit: 0d984e507, author: vlebedynska, date: 2020-05-24 23:02:17

#10 added slider-functionality for each link line (temp JS version):
- added functionality to create generically sliders for each link-
  line
- created generic function for svg path length measurement for
  weight calculation based on the position of the slider on the svg
  path
- implemented drag-functionality for the slider by using the
  closestPoint algorithm
- added event handlers for mousemove, mousedown and mouseup
- changes of the slider position along the svg path immediately
  update the link-line stroke width


commit: b5213aab2, author: vlebedynska, date: 2020-05-21 20:29:12

#10 adding generated JS files for link highlighting functionality


commit: 8f49a3977, author: vlebedynska, date: 2020-05-21 20:21:44

#10 link between nodes is highlighted and brought to the top when
  it's active


commit: 54d9e0093, author: vlebedynska, date: 2020-05-21 16:48:23

#9 ai colour sensor input node moved from JS to TS,
refactoring in extractColourChannelAndNormalize function


commit: 7e1b46982, author: vlebedynska, date: 2020-05-20 02:18:14

#9 jaxbToAst implementation of color sensor as ai input node
definition of visitor method for color sensor as ai input node
client-side implementation of color sensor as ai input node
#3 integration of svgdotjs library (completed)


commit: 40a5bd6ae, author: vlebedynska, date: 2020-05-18 00:33:07

Merge remote-tracking branch 'origin/feature/neuronalnetworks' into
  feature/neuronalnetworks


commit: 072a8bbb0, author: vlebedynska, date: 2020-05-18 00:28:46

#9 connected colour sensor to aiInputNode in java backend,
set up the test environment,
started with jaxbToAst implementation


commit: 5f02a8495, author: vlebedynska, date: 2020-05-18 00:28:46
```

```
#3 connected colour sensor to aiInputNode in java backend,
set up the test environment,
started with jaxbToAst implementation


commit: 41a0cfe62, author: vlebedynska, date: 2020-05-16 23:25:36

#3 bug fix: when you move the slider, the AI pop-up stays in the
  same place.


commit: 4ba249478, author: vlebedynska, date: 2020-05-16 22:41:08

#3 ai pop-up design changes: the modal is now draggable and has a
  border.


commit: 3a377febd, author: vlebedynska, date: 2020-05-16 21:21:47

#3 migration from JS to TS (in progress), integration of svgdotjs
  library (in progress)


commit: 77edbc2ee, author: vlebedynska, date: 2020-05-15 22:14:02

#3 design changes for the ai pop-up, migration from JS to TS (in
  progress)


commit: 2a7b304fe, author: vlebedynska, date: 2020-05-15 01:39:32

#3 single slider for all weights implemented. Data update in slider
  on link selection (temp JS solution);
Depending on the weight change, the stroke width changes live.
svg.js dependency added


commit: 3a0a9c2d0, author: vlebedynska, date: 2020-05-11 23:41:28

#3 function for drawing links in the neural-network-popup
  implemented (temp JS solution)


commit: b78d04cf4, author: vlebedynska, date: 2020-05-11 00:56:13

#3 implemented draw functions for drawing neural network in the
  neural-network-popup in the simulation


commit: 797e235cc, author: vlebedynska, date: 2020-05-09 21:38:31

#3 temp design for the neural networks popup


commit: 59d596bc3, author: vlebedynska, date: 2020-05-09 15:49:42
```

```
  #1 migrated neural network functionality from JS to TS.


  commit: 93de936f6, author: vlebedynska, date: 2020-05-07 20:15:38

  #3 added a pop-up for weights configuration in index.html,
implemented new function changeWeigth to create sliders for link
    weights and change the weights in this pop-up,
  updated function processNeuralNetwork - creation of a neural network
    and its connection to UI is executed only once - the first time the
    function processNeuralNetwork is called. Further calls update the
    property externalSensor of the input layer nodes in the already
    existing neural network.


  commit: 71dc95713, author: vlebedynska, date: 2020-05-05 00:49:55

  #1 changed the data type for the threshold in AiInput.java to
    Integer because null values are allowed


  commit: 35921c2a7, author: vlebedynska, date: 2020-05-05 00:46:21

  #3 added the neural network button into the simulation, added new
    font icon for the neural network pop-up window


  commit: 921ed6bcf, author: vlebedynska, date: 2020-05-04 13:00:25

  #1 implemented astToBlock method for AiInput.java and
    AiNeuralNetwork.java


  commit: 08b20d71d, author: vlebedynska, date: 2020-05-03 23:49:52

  #1 implemented astToBlock method for AiOutput.java
  started implementation of astToBlock method for AiInput.java


  commit: 35516b30f, author: vlebedynska, date: 2020-05-03 20:14:07

  #1 added function for initial creation of links for the neural
    network,
  refactored the function processNeuralNetwork


  commit: 57e12a1de, author: vlebedynska, date: 2020-05-02 23:42:17

  #1 The processNeuralNetwork function in
    interpreter.robotMbedBehaviour.js has been updated so that the
    algorithm now works for the simulation. However, a refactoring is
    required and the speed calculation for the second motor should be
    fixed as soon as possible.
  changed the visitorAiOutputNode, so that the return value is now a
    simple JSONObject.
```

```
commit: 4b1e50fcd, author: vlebedynska, date: 2020-05-02 17:06:01

#1 changed ListCreate type from <aiinput>/ to </aiinput
implemented visitAiOutputNode in AiOutput.java
added CREATE_INPUT_NODE opcode to visitAiInputNode in
  Ev3StackMachineVisitor.java
added CREATE_OUTPUT_NODE opcode to visitAiInputNode in
  Ev3StackMachineVisitor.java
added PROCESS_NEURAL_NETWORK opcode to visitAiInputNode in
  Ev3StackMachineVisitor.java
Exception handling for IAiVisitor
added new constants to C.java via constantsSource.txt
added new constants to interpreter.constants.js and
  interpreter.constants.ts
implemented processNeuralNetwork function in
  interpreter.robotMbedBehaviour.js and
  interpreter.robotMbedBehaviour.ts
updated to 3.8.8. DB version in pom.xml in Wedointerpreter


commit: 2ecda45d7, author: vlebedynska, date: 2020-04-29 18:57:43

#1 implemented JaxB to Ast generation for neural network block
added two new blockly constants for input and output layer
defined two new data types - InputNode and OutputNode
changed robotCommon.yml for ai blocks
implemented method visitAiNeuralNetwork in Ev3StackMachineVisitor
  with dummy actions for test purposes


commit: d6c799bb9, author: vlebedynska, date: 2020-04-26 23:50:30

#1 JaxB to Ast generation for AiOutput block


commit: a6cb80500, author: vlebedynska, date: 2020-04-26 21:55:58

#1 integrated AI blocks into Open Roberta Lab UI for EV3 (e.g.
  updated robotCommon.yml, added new Blockly constants, added
  language support for English and German),
updated pom.xmls with new db version 3.8.8,
refactored ai_actors and ai_sensors blocks,
updated tests ressources for AI blocks,
created AiLink class,
started impl of AiOutput block


commit: ca5e634b6, author: vlebedynska, date: 2020-04-23 01:00:34

#1 added AiInputNode functionality to visitor
  AbstractStackMachineVisitor,
threshold and sensor-AST: extracted from Jaxb block and saved as
  members in AiInputNode,
extended toString() in validateInputNode from
  NeuralNetworkComponentsTest
```

```
explored code generation for simulation


commit: 24a1a3617, author: vlebedynska, date: 2020-04-21 20:51:28

#1 added Ast-Tests


commit: 9984c0bd5, author: vlebedynska, date: 2020-04-21 20:23:57

#1 added new AI-Ast objects and registered in the robotCommon
  configuration


commit: 346fede84, author: vlebedynska, date: 2020-04-13 15:09:03

autogenerated html


commit: 327b82ca5, author: vlebedynska, date: 2020-04-13 14:46:45

db changes 3.8.7 > 3.8.8
```

A.4 Machine Learning Curriculum

Unterrichtsplan für 04.08.2020

„Aufwachsen mit KI: KI erlebbar machen!“

Zielgruppe: Schüler*innen in der 7-8 Klasse

Zeitlicher Aufwand: 6 Unterrichtsstunden à 45 Minuten

Ziel des Unterrichtsvorhabens: Schüler*innen untersuchen, wie die Roboter lernen, denken und fühlen. Sie gehen der Frage nach, was unter dem Begriff Künstliche Intelligenz (KI) zu verstehen ist und lernen drei Hauptbereiche der KI - überwachtes, unüberwachtes und bestärkendes Lernen - praxisnah kennen. Sie ergründen, wann die Menschen eine Maschine als intelligent bezeichnen und stärken ihr neu erworbenes Wissen, indem sie sich mit der Entwicklung und Konfiguration von einfachen anschaulichen KI-Anwendungen im Open Roberta Lab beschäftigen.

Der Unterricht am 05.08.2020 (mit den Schüler*innen der 5-6 Klassen) und 06.08.2020 (mit den Schüler*innen der 3-4 Klassen) wird ähnlich aufgebaut sein, nur dass die Arbeitsblätter, Aufgabenstellungen und Beispiele entsprechend dem Alter der Kinder aufbereitet werden.

Uhrzeit	**Plan**	**Arbeitsformen und -materialien**
Modul 1: „Wie lernt dein Roboter?“ – Einführung in Künstliche Intelligenz und Maschinelles Lernen		
09:00-09:30	**Kennenlernrunde im Plenum / im Tandem** In die Vorstellung sollen einige der folgenden Fragen einbezogen werden: - Was ist Intelligenz? Was denkst du, was macht uns zu intelligenten Wesen? - Wie lernen wir, Menschen? - Was fällt dir schwer/einfach zu lernen? - Wie lernen die Maschinen? Lernen sie überhaupt? Wenn du schon ein Programm geschrieben hast, bedeutet das, dass der Computer/dein Roboter schlauer geworden ist? - Können die Roboter fühlen? Sind die Maschinen emotional? Anschließend – Sammlung und Diskussion der Eindrücke im Plenum	Präsentation Diskussion im Tandem Plenum *Evtl.: Plakate?*
09:30-09:45	**Braitenberg Experiment mit einem Calli:bot / EV3 durchführen:** Verhaltensweise Angst ODER Verhaltensweise Freundschaft	Calli:bot oder EV3,

	Reflexion und Diskussion: - Ist die Verhaltensweise des Calli:bot intelligent? - Kennt ihr derartige Verhaltensweise aus der Natur? Evtl. EV3 Roboter-Set mitbringen und mit den Kindern die Funktionsweise von Braitenberg Experimente im Open Roberta Lab nachvollziehen.	Taschenlampe Experiment im Plenum Open Roberta Lab, wenn ein Set von EV3 Robotern vorhanden sein wird eventuell Einzelarbeit oder Tandem Beispielvideo mit der ähnlichen Verhaltensweise aus der Natur: Motte fliegt zum Licht
09:45-10:15	Bezug nehmen auf das Experiment von Braitenberg und die Diskussion davor (Was denkt ihr, können die Roboter auch intelligent sein?) und in das Thema KI einleiten. **Vortrag über KI:** was ist KI? Warum ist es wichtig, dass wir heute über KI lernen? Warum ist das für euch wichtig zu wissen, was KI ist und womit sie sich beschäftigt? Geschichte erzählen, mit Beispielen (*siehe als Vorbereitung die Vorlesung von Hod Lipson*)[23] **Ziel der UE nahebringen:** Wir lernen heute, wie die Roboter denken, lernen, fühlen und wann wir sie als intelligent bezeichnen.	Präsentation
10:15-10:30	**Pause**	
Modul 2: „Bringe deinem Roboter etwas bei" – Einführung in überwachtes Lernen und Neuronale Netze		
10:30-10:50	Mit welcher Art von Problemen haben Computer / Maschinen zu kämpfen? Mit den gleichen Problemen, wie wir Menschen! **Überblick geben**, welche Bereiche der KI wir heute kennenlernen werden: - Überwachtes Lernen - Unüberwachtes Lernen - Bestärkendes Lernen **Einführung in das Thema „Überwachtes Lernen"**	Präsentation Set von Bildern, die den Kindern bekannt und nicht bekannt sind: Tiere / Planeten / Beispiel aus dem Mathematikunterricht Arbeit im Plenum Interessant sind ebenfalls zweideutige Bilder (wie die

[23] https://youtu.be/XJP1hJ92g1Q, abgerufen am 12.07.2020.

	Experiment mit den Bildern (Eine analoge Übung, überwachtes Lernen hautnah erleben): Ein Bild soll gezeigt werden. Die Kinder sollen erkennen, was/wer auf dem Bild zu sehen ist. Was weißt du schon in deinem Alter? Was weißt du noch nicht? Dann soll die Lösung verraten werden, wer/was auf dem Bild zu sehen ist. Ein zweites, drittes, viertes Bild zeigen -> Effekt des Lernens erzeugen, indem die Kinder lernen, während der/die LehrerIn als der/die ÜberwacherIn agiert. Ein anderes Beispiel könnte man aus der Mathematik nehmen: welche Rechenaufgabe könnt ihr bereits in eurem Alter lösen? Was könnt ihr noch nicht tun? **Schlussfolgerung:** Je mehr Erfahrung wir haben, je älter wir sind – desto mehr können wir und desto schlauer werden wir. LehrerInnen / Eltern helfen uns, uns zu verbessern, sie geben uns Anweisungen, wie wir uns verhalten sollen, wie gut wir gelernt haben. So ähnlich sieht's auch bei den Computern aus!	Täuschungsbilder, Bild mit den Hunden und Keksen)
10:50-11:30	**KI erlebbar machen! Teil 1** Einführung in die Funktionsweise eines sehr einfachen neuronalen Netzes („Gehirn des Roboters")[24] und Übergang zum selbstständigen Programmieren von überwachtem Lernen im Open Roberta Lab. Viktoriya zeigt das Programmieren und Anwenden von Neuronalen Netzen vor und erklärt die Aufgabe: „Bringe dem Roboter bei, sich korrekt zu verhalten!" Übungen mit KI Lernkarten einleiten	Open Roberta Lab KI-Lernkarten *Inspiration:* *Kinderbücher zum Überwachten Lernen und Neuronalen Netzen* Einzelarbeit oder im Tandem
11:30-12:15	**Mittagspause**	
Modul 3: „Lass deinen Roboter aus Erfahrungen lernen" – Einführung in das bestärkende Lernen		
12:15-12:35	**KI erlebbar machen! Teil 2**	Präsentation

[24] Hier muss ich mir noch genauer überlegen, wie ich in das Thema einleite.

	Zunächst: analoge Einführung in das Thema „Bestärkendes Lernen“. Ziel: bestärkendes Lernen erlebbar machen. Was lernt/macht ihr gerne allein, ohne Aufforderung? Mögliche Antworten: - tanzen - singen - musizieren - Computerspiele spielen - Beispiele aus aktuellen Computer Games. Wie verbessert man sich als GamerIn mit der Zeit? - Etc. Weitere Beispiele für das bestärkende Lernen? - Hundetraining Wie ist das / was bedeutet das, wenn ein Roboter durch das bestärkende Lernen lernt? Eigentlich geht es ihm dann ähnlich wie uns! Beispiele für bestärkendes Lernen im Computer/Robotik-Welt zeigen: - Go/Schach-Spiele - Tischtennis-Spiel[25] - Andere Beispiele aus Computerspielen – Agent lernt, die Belohnungen zu sammeln, anstatt in der Race zu gewinnen. - Das Auto konstruiert sich von allein durch „Trial and Error“. Oft fragen wir uns – was ist eigentlich gut? Können wir Menschen, Maschinen / Robotern / Computern beibringen, gut zu sein? Und was heißt eigentlich gut im Sinne eines Computerprograms? - Beispiel einer Interaktionskette – Ball treffen ist gut, aber was heißt eigentlich, einen Ball zu treffen? Sich zwei Mal nach oben, einmal nach unten zu bewegen, bringt den Agenten vielleicht einmal zum Erfolg, ein anderes Mal jedoch zum Misserfolg. Beim bestärkenden Lernen geht es nicht darum, Computer etwas beizubringen, sondern, es geht	Beispiele aus gym.openai.com oder ähnlichen Plattformen *Evtl. Beispielvideo Hundetraining* Diskussion im Plenum

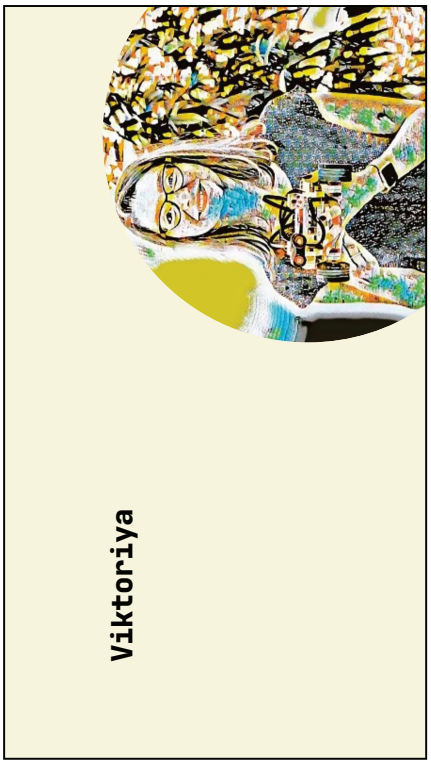
[25] https://gym.openai.com/envs/Pong-ram-v0/, abgerufen am 12.07.2020.

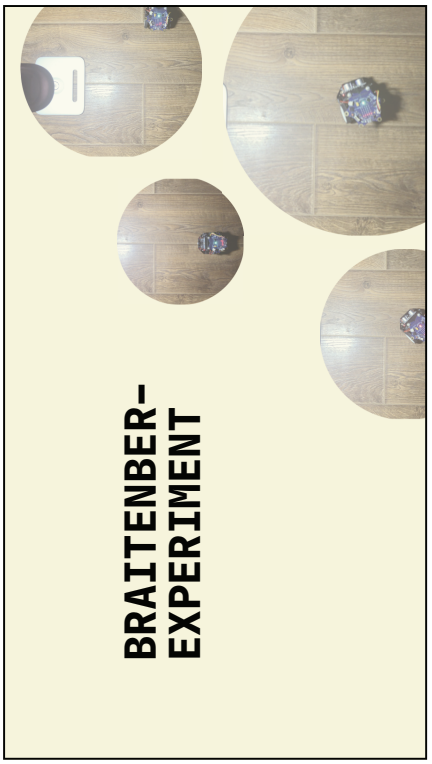
	darum, Computer lernfähig zu machen und Lernprozess anzustoßen.	
12:35-13:30	**Machen wir nun unseren Roboter lernfähig!** Einführung in die Arbeit mit dem bestärkenden Lernen im Open Roberta Lab. Diskussion: - Was bedeutet „gut“ für diese Problemstellung? - Was ist ein optimaler Weg für uns, Menschen? Beispiel: ein optimaler Weg für uns ist, wenn es an einem Eiscafé / Spielplatz vorbeiführt. - Was ist ein optimaler Weg für einen Straßenstaubsaugroboter? Spielregeln erklären und in die Blöcke im Open Roberta Lab einführen. Arbeitsblätter verteilen. In der Zwischenzeit: Diskussion starten – was merkt ihr? Wie wirken sich unterschiedliche Einstellungen der Parameter auf die Verhaltensweise des Roboters aus? Experiment: Wer ist schneller / besser? Der Roboter oder wir, Menschen? Wer von uns beiden findet den schnelleren/besseren Pfad?	Open Roberta Lab Spielregeln Hintergründe (Eisenbahn, Labyrinth, Stadtkarte) Aufgaben Erklärung zu den Blöcken Einzelarbeit oder im Tandem, anschließend Diskussion im Plenum
13:30-13:45	**Pause**	
Modul 4 „Können Roboter selbstständig lernen?“ – Einführung in unbeaufsichtigtes Lernen		
13:45-14:30	**KI erlebbar machen! Teil 3** **Einführung in das Thema „Clustering“ und unüberwachtes Lernen.** Dieses Thema wird voraussichtlich nur analog behandelt. Clustering erlebbar machen – Experiment mit verschiedenen Gefäßen. Clustering: was ist das? Es ist die Kunst von Gruppenbildungen. Beispiele für verschiedene Gruppenbildungsprobleme / Zweideutigkeiten mitbringen und mit Kindern diskutieren. - Wie würdest du diese Gegenstände zusammen gruppieren?	Präsentation – Clustering analog erklären Experiment mit verschiedenen Gefäßen: verschiedene Flaschen, Trinkgläser Arbeit in Plenum

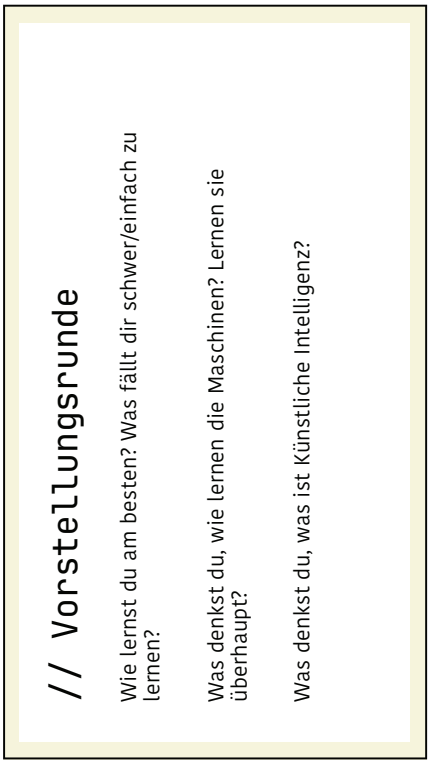
	- Wo kommen deine Kriterien her? Warum hast du dich für diese Gruppierung entschieden? Überleiten zum Thema: wie gruppiert ein Roboter die Gegenstände?	
Puffer und Überleitung zur Abschlussdiskussion	Was haben wir also heute alles gelernt? Zusammenfassung der drei Arten, wie die Roboter lernen können + Zusammenfassung zentraler Erkenntnisse. Damit die Kinder vor der abschließenden Diskussion noch mal in sich gehen und Gedanken dazu machen können, wird ein Arbeitsblatt verteilt: Wie lernst du? Wie lernen Maschinen? - Überleitung zur abschließenden Diskussion und Reflexion. - Evtl. Position beziehen – Gedanken zu ethischen Fragen machen - Oder ein weiteres interessantes Thema: Jobs, die heutzutage von Menschen erledigt werden, Jobs, die zunächst nur von Maschinen erledigt werden.	Arbeitsblatt: Wie lernst du? Wie lernen Maschinen?[26] Arbeitsblatt: Position beziehen zu ML-Anwendungen[27] Oder: Arbeitsblatt für die zukünftigen Jobs[28] Einzelarbeit
14:40-15:00 Uhr	**Diskussion/Reflexion – was hast du heute gelernt, wie die Roboter lernen?** - Wie hat es dir gefallen? - Welcher Teil hat es am meisten Spaß gemacht? - Was nimmst du aus diesem Tag mit?	Diskussion im Plenum *Evaluationsbogen?* *Evaluationsplakat?* Einzelarbeit und dann Diskussion im Plenum

[26] https://www.medien-in-die-schule.de/wp-content/uploads/Arbeitsblatt_MachineLearning_15.pdf, abgerufen am 12.07.2020.
[27] https://www.medien-in-die-schule.de/wp-content/uploads/Materialblatt_MachineLearning_22.pdf, abgerufen am 12.07.2020.
[28] https://www.medien-in-die-schule.de/wp-content/uploads/Materialblatt_MachineLearning_25.pdf, abgerufen am 12.07.2020.

A.5 Presentation

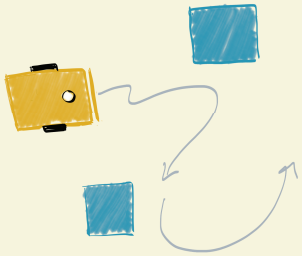






Was habt ihr beobachtet?

Ist die Verhaltensweise des Roboters intelligent?



5

Wie „denken“ die Roboter?

Wie lernen sie?

6

Spielregeln

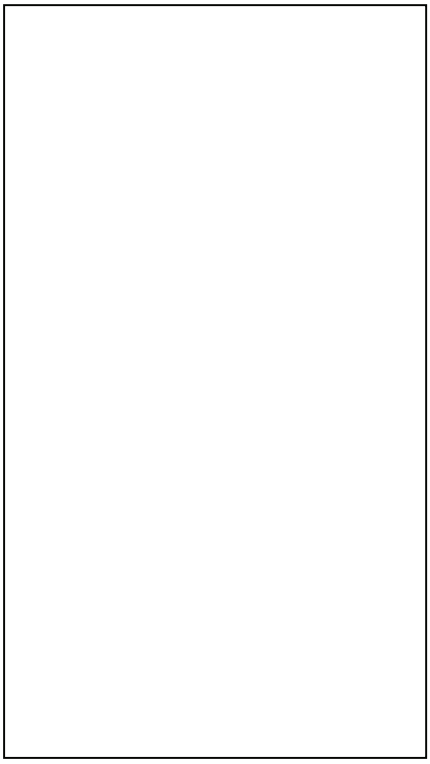
7

// Plan für heute

Was ist KI?	Bestärkendes Lernen	Überwachtes Lernen	Unüberwachtes Lernen
9:00–9:50	10:00–12:00	13:00–14:15	14:15–15:00
	Wie sammelt ein Roboter die Erfahrung? Was wäre, wenn der Roboter in der Schule Noten bekommen würde?	Was wäre, wenn der Roboter zur Schule gehen würde? Oder braucht ein Roboter eineN LehrerIn, um zu lernen?	Wie lernt ein Roboter selbständig? Gruppen bilden und was das mit den Flaschen zu tun hat

8

10



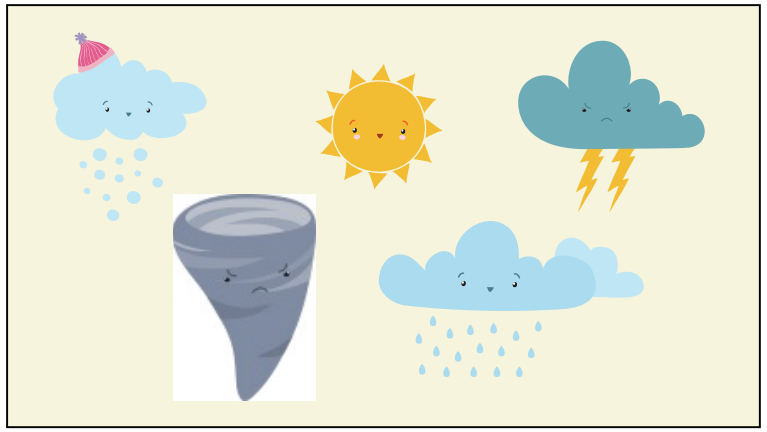
12

9

11

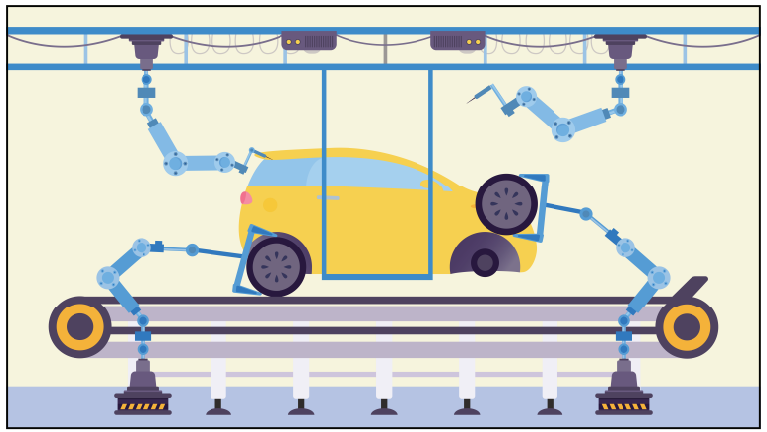
KI passiert im Hintergrund

13



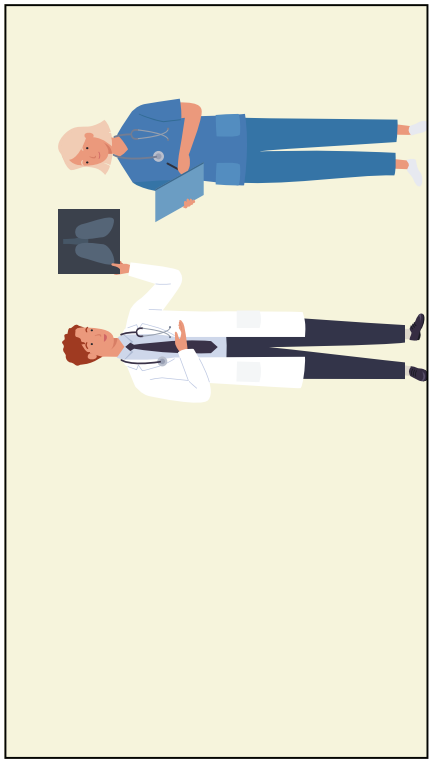
14

15

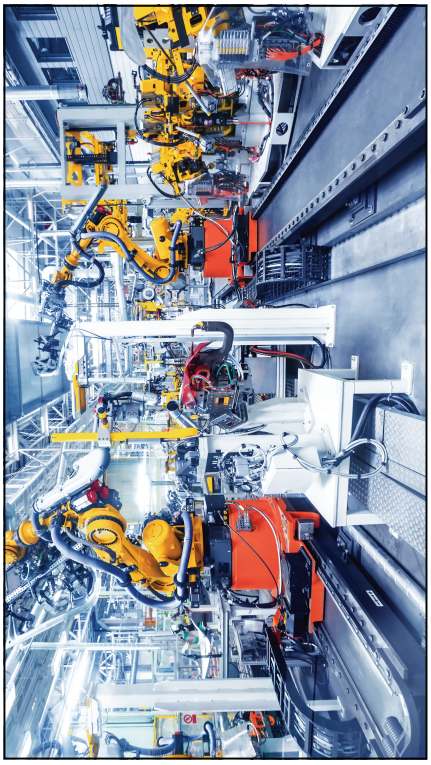


16

18



20



17

19

KI – Frage der Perspektive

Menschliches Denken	Rationales Denken
„Das spannende, neuartige Unterfangen, Computern das Denken beizubringen, … Maschinen mit Verstand im wahrsten Sinne des Wortes." (Haugeland, 1985) „[Die Automatisierung von] Aktivitäten, die wir dem menschlichen Denken zuordnen, Aktivitäten wie beispielsweise Entscheidungsfindung, Problemlösung, Lernen …" (Bellman, 1978)	„Die Studie mentaler Fähigkeiten durch die Nutzung programmiertechnischer Modelle." (Charniak und McDermott, 1985) „Das Studium derjenigen mathematischen Formalismen, die es ermöglichen, wahrzunehmen, logisch zu schließen und zu agieren." (Winston, 1992)
Menschliches Handeln	**Rationales Handeln**
„Die Kunst, Maschinen zu schaffen, die Funktionen erfüllen, die, werden sie von Menschen ausgeführt, der Intelligenz bedürfen." (Kurzweil, 1990) „Das Studium des Problems, Computer dazu zu bringen, Dinge zu tun, bei denen ihnen momentan der Mensch noch überlegen ist." (Rich und Knight, 1991)	„Computerintelligenz ist die Studie des Entwurfs intelligenter Agenten." (Poole et al., 1998) „KI … beschäftigt sich mit intelligentem Verhalten in künstlichen Maschinen." (Nilsson, 1998)

21

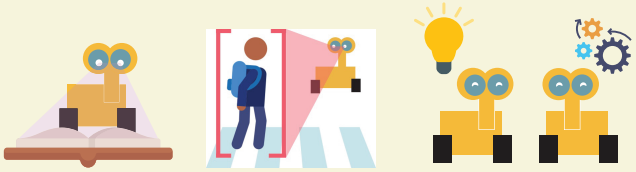
KI – Frage der Perspektive

…, Computer dazu zu bringen, Dinge zu tun, bei denen ihnen momentan der Mensch noch überlegen ist. (Rich und Knight, 1991)

KI, … beschäftigt sich mit intelligentem Verhalten in künstlichen Maschinen. (Nilsson, 1998)

22

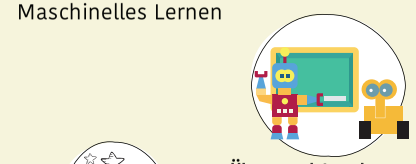
Wann ist eine Maschine intelligent?



23

Heute geht's ums Lernen

Maschinelles Lernen



Überwachtes Lernen

Bestärkendes Lernen

Unüberwachtes Lernen

24

Bestärkendes Lernen

wird eingesetzt, wenn der Roboter aus Erfahrungen lernen soll.

Zeit

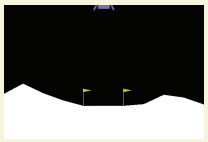
Ziel

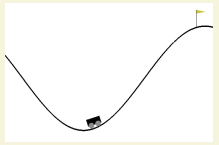
Lernverhalten

Feedback

25

Lernen aus Erfahrungen





Quelle: https://gym.openai.com/

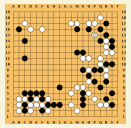
26

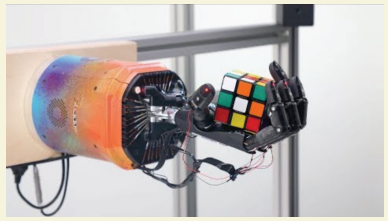
Hubschrauber lernt zu fliegen

27

Spiele spielen

- Schach
- Go
- Rubik's Cube



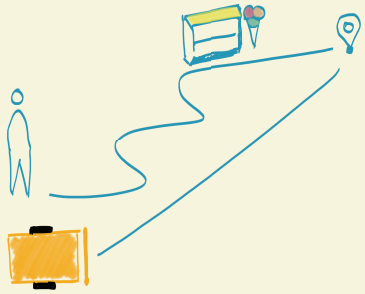


28

Q-Lernen: finde den besten Weg

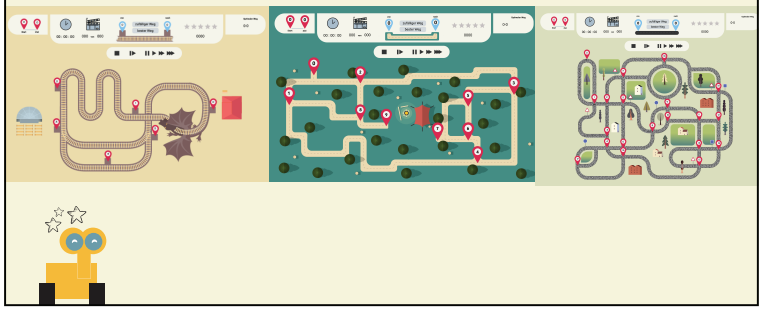
Was ist ein bester Weg für uns Menschen?

Was ist ein bester Weg für den Roboter?

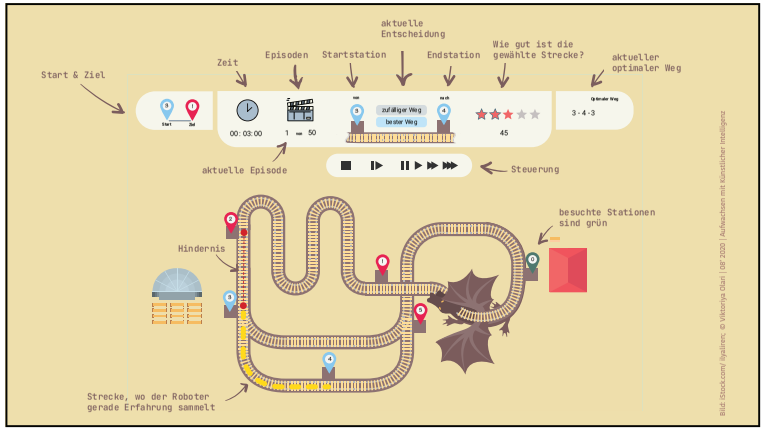


29

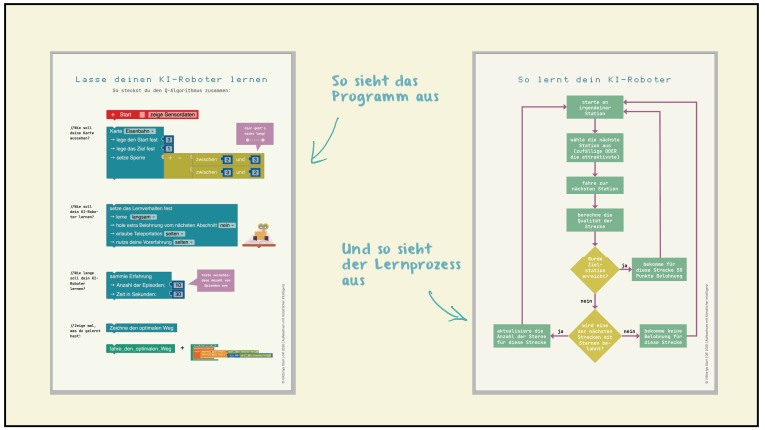
Machen wir nun unseren Roboter lernfähig!



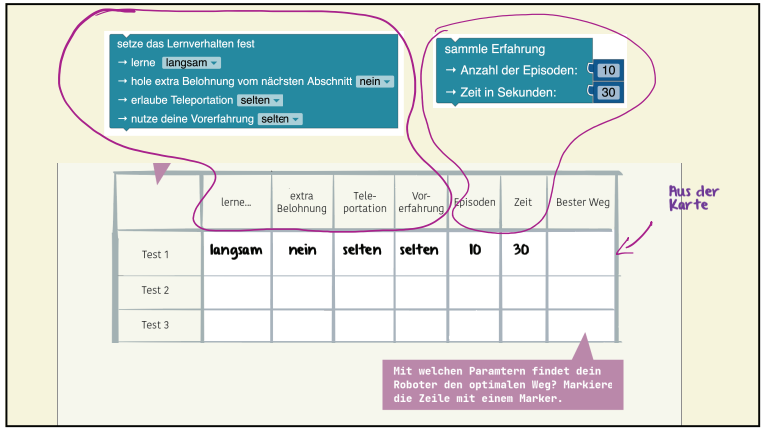
30



31



32



33

Spielregeln

- Der Roboter bewegt sich immer zwischen zwei Stationen.
- Die Abschnitte, die zur Endstation (zum Ziel) führen, geben die größte Belohnung.
- Je nachdem, wie du deinen Algorithmus konfiguriert hast, startet der Roboter entweder an einer zufälligen Stelle oder da, wo er aufgehört hat.
- Nach jedem Zug berechnet der Roboter die Qualität der Staße aufs Neue: ist sie jetzt basierend auf seinem Wissen attraktiver geworden als vorher oder nicht?

34

Los geht's ins Lab!

Testet verschiedene Werte für die erste Karte

35

Los geht's ins Lab!

Testet verschiedene Werte für die letzten zwei Karten.

36

Was habt ihr beobachtet?

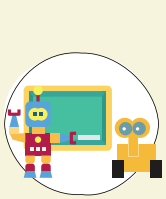
Wann lernt der Roboter am schnellsten?
Wie viel Zeit braucht er?
Welche Einstellungen sind die besten?

37

MITTAGSPAUSE

38

Aufwachsen mit KI



Überwachtes Lernen

Bestärkendes Lernen

Unüberwachtes Lernen

39

Unüberwachtes Lernen

Wird eingesetzt, wenn die gewünschten Ergebnisse nicht im Voraus bekannt sind, der Computer aber nach Mustern in den Daten sucht.

40

Clustering

Wie würdet ihr diese Gegenstände gruppieren?

41

Clustering

Wie würdet ihr diese Gegenstände gruppieren?

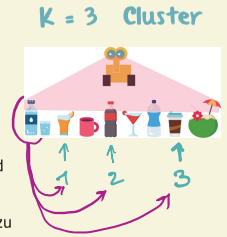
Wo kommen eure Kriterien her? Warum habt ihr euch für diese Gruppierung entschieden?

42

Wie gruppiert eine Maschine die Gegenstände?

43

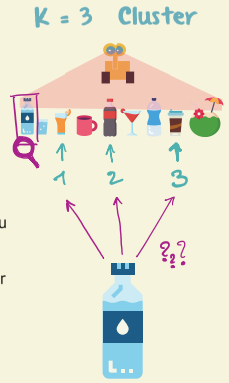
k-Means-Algorithmus



1. Wähle die Anzahl der Gruppen aus, die du identifizieren möchtest.
2. Wähle zufällig drei verschiedene Gegenstände aus. Sie sind unsere Cluster.
3. Messe den Unterschied zwischen dem ersten Gegenstand zu den drei Clustern.
4. Weise den ersten Gegenstand dem nächstgelegenen Cluster zu.
5. Wiederhole die letzten zwei Schritte für alle Gegenstände.

44

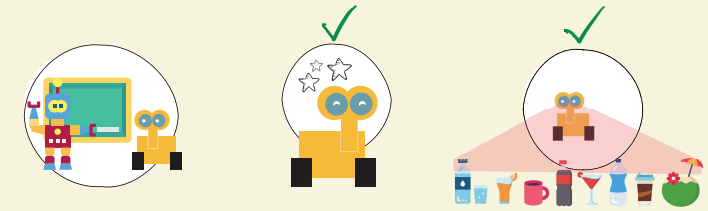
k-Means-Algorithmus



1. Wähle die Anzahl der Gruppen aus, die du identifizieren möchtest.
2. Wähle zufällig drei verschiedene Gegenstände aus. Sie sind unsere Cluster.
3. Messe den Unterschied zwischen dem ersten Gegenstand zu den drei Clustern.
4. Weise den ersten Gegenstand dem nächstgelegenen Cluster zu.
5. Wiederhole die letzten zwei Schritte für alle Gegenstände.

45

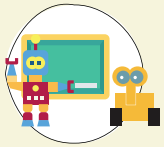
Aufwachsen mit KI



Überwachtes Lernen | Bestärkendes Lernen | Unüberwachtes Lernen

46

Überwachtes Lernen



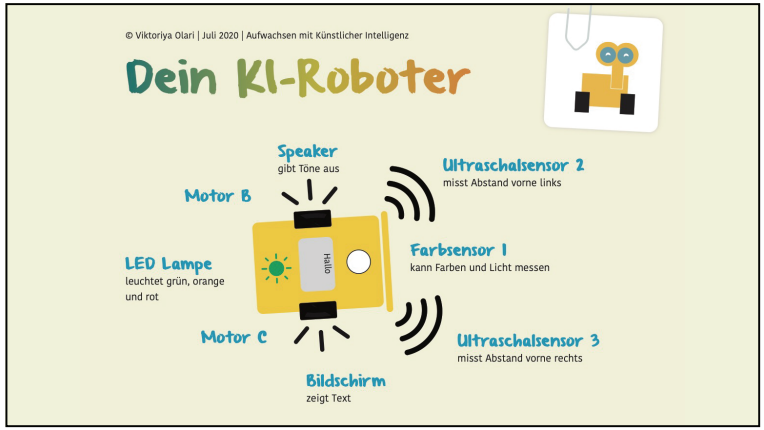
Wird eingesetzt, wenn das gewünschte Ergebnis im Voraus bekannt ist.

47

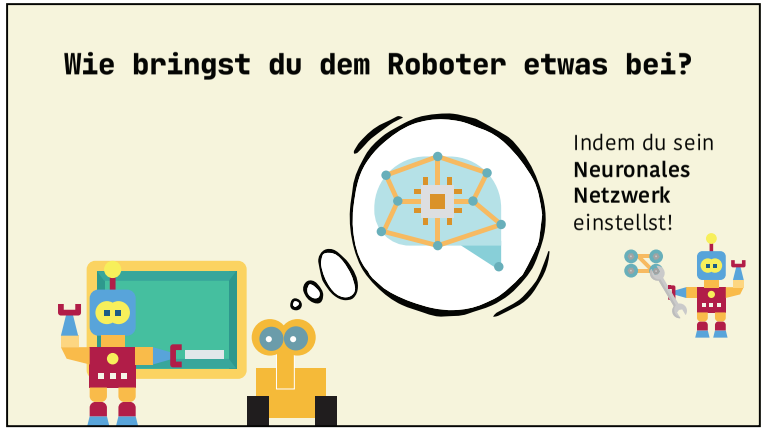
Bringt dem Roboter die Sinne bei

Und wie er sich in bestimmten Situation korrekt verhalten soll

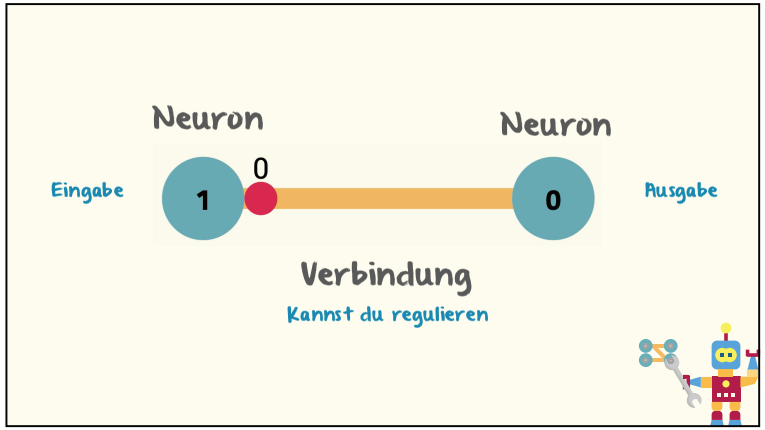
48


© Viktoriya Olari | Juli 2020 | Aufwachsen mit Künstlicher Intelligenz
Dein KI-Roboter
Speaker
gibt Töne aus
Ultraschallsensor 2
misst Abstand vorne links
Motor B
Farbsensor 1
kann Farben und Licht messen
LED Lampe
leuchtet grün, orange und rot
Hallo
Motor C
Ultraschallsensor 3
misst Abstand vorne rechts
Bildschirm
zeigt Text

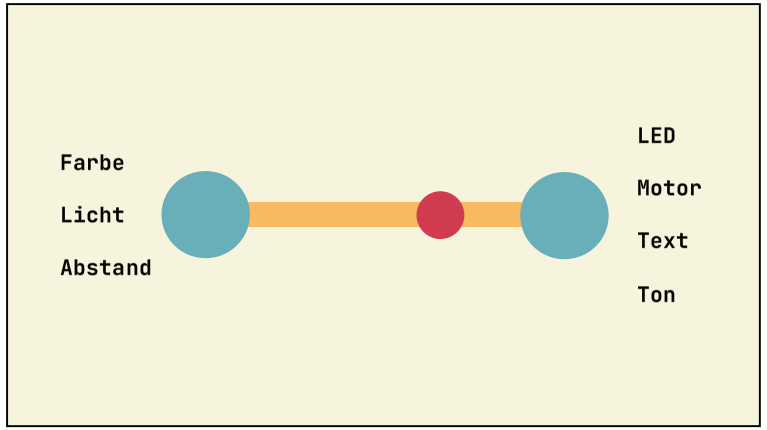
49


Wie bringst du dem Roboter etwas bei?
Indem du sein
Neuronales
Netzwerk
einstellst!

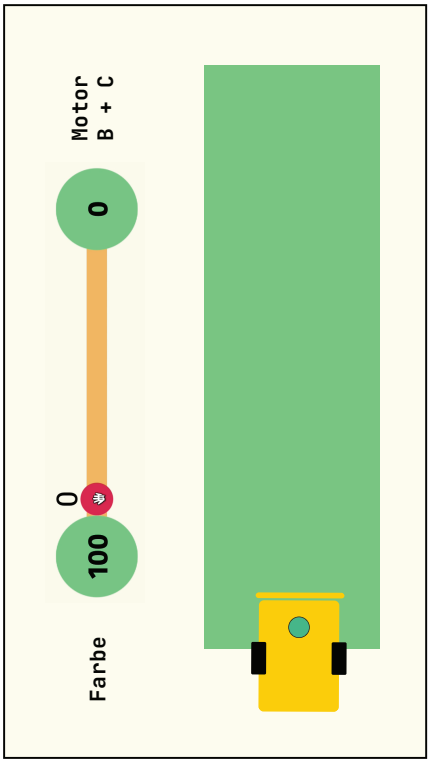
50


Neuron
Neuron
Eingabe
1
0
0
Ausgabe
Verbindung
Kannst du regulieren

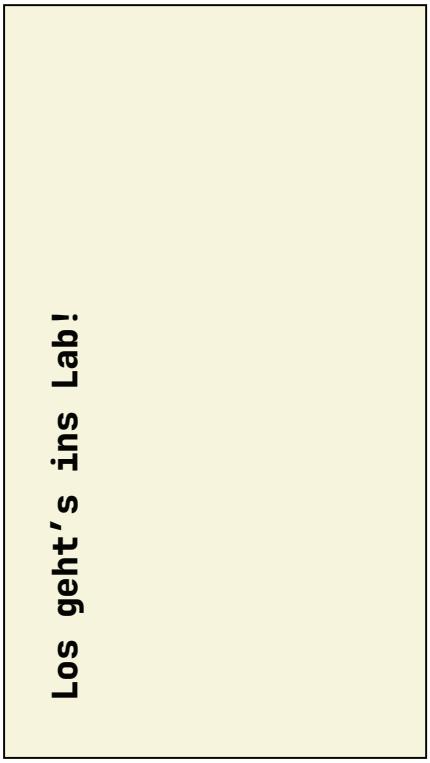
51


Farbe
Licht
Abstand
LED
Motor
Text
Ton

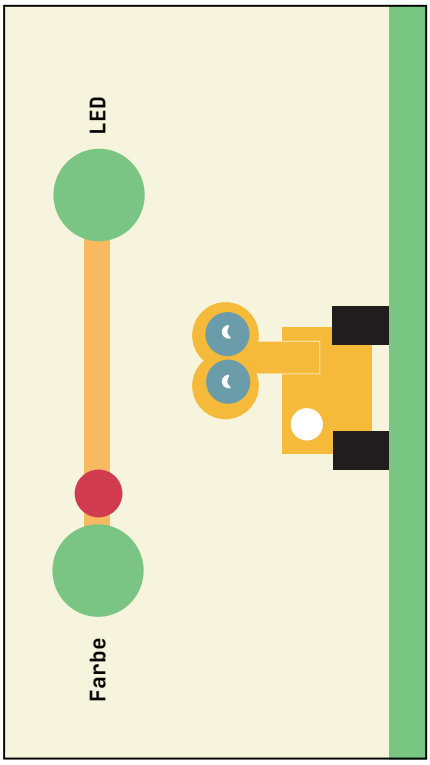
52


Motor
B + C
0
0
100
Farbe

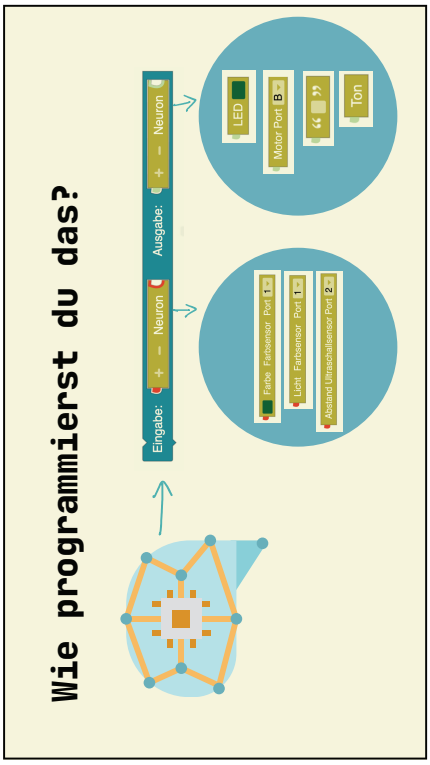
54


Los geht's ins Lab!

56


LED
Farbe

53


Wie programmierst du das?
Eingabe:
Neuron
Ausgabe:
Neuron
LED
Motor Port B
Ton
Farbe Farbsensor Port 1
Licht Farbsensor Port 1
Abstand Ultraschallsensor Port 2

55

Ablauf

Testen

Netz konfigurieren

Programm starten

Netzwerkfenster öffnen

SIM Simulation öffnen

Programm zusammenstecken

57

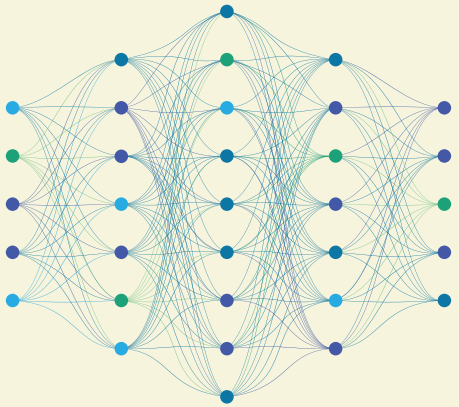


Bild: iStock.com/Andrii Shyp

58

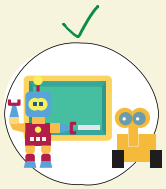
KI-Karten

59

PAUSE

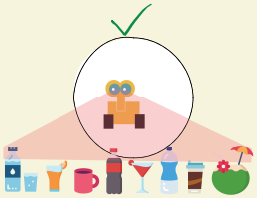
60

Aufwachsen mit KI



Überwachtes Lernen

Bestärkendes Lernen



Unüberwachtes Lernen

61

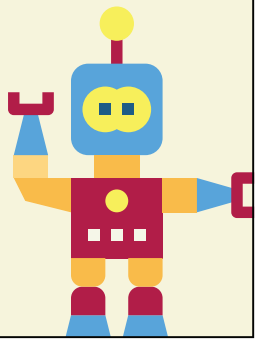
Feedback

- Wie hat es dir gefallen?
- Welcher Teil hat es am meisten Spaß gemacht?
- Was nimmst du aus diesem Tag mit?
- Was nimmst du aus diesem Tag mit, was KI ist?
- Was denkst du, was KI ist?

https://forms.gle/vjQXhtjP4VJqKRSw6

62

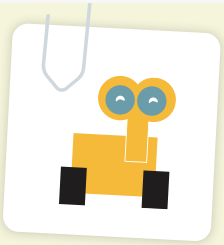
Bis zum nächsten Mal!

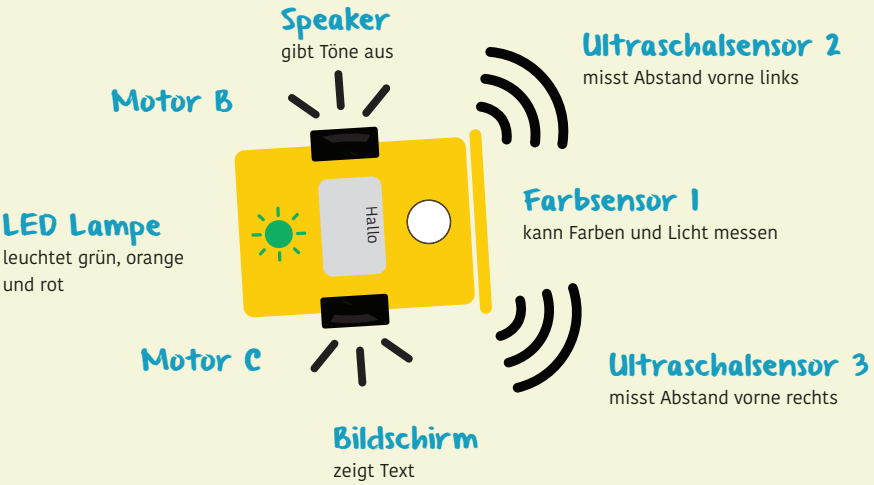


63

A.6 Neural Network Cards

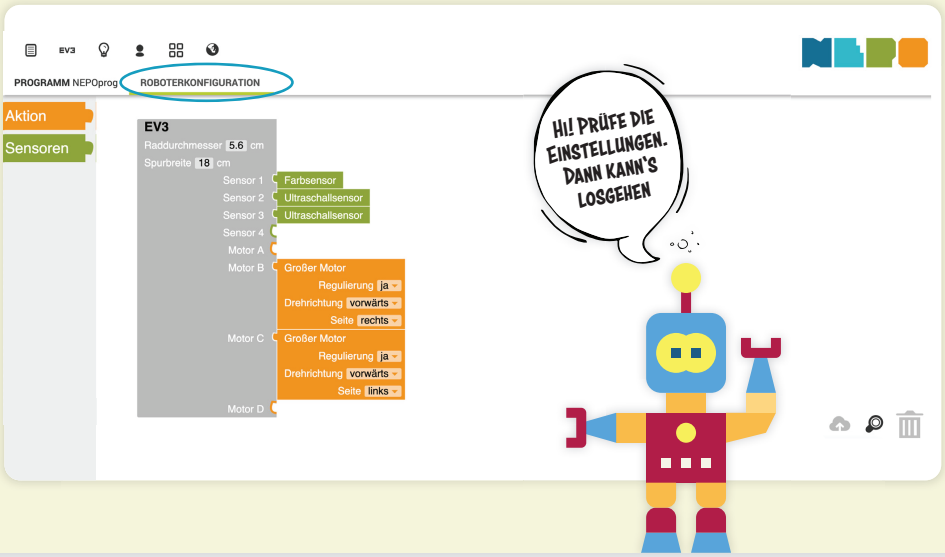
Dein KI-Roboter



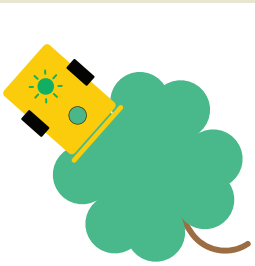


So stellst du deinen KI-Roboter ein

Überprüfe, ob dein KI-Roboter richtig konfiguriert ist.



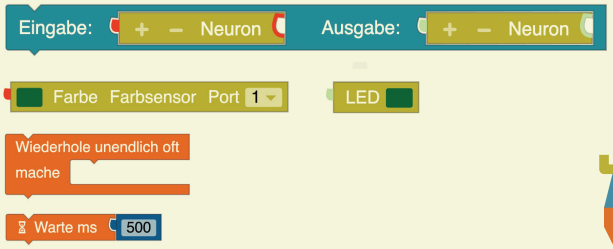
Chameleon

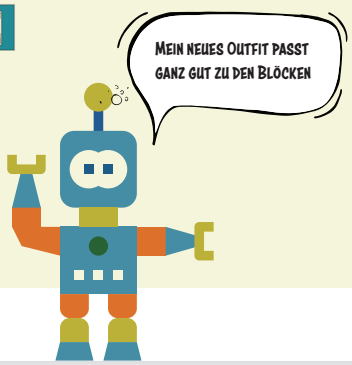


Auch Roboter können lernen, sich der Umgebung anzupassen

Bringe das deinem Roboter bei! Schiebst du deinen Roboter auf das grüne Blatt, soll seine LED-Lampe grün aufleuchten.

Du brauchst:

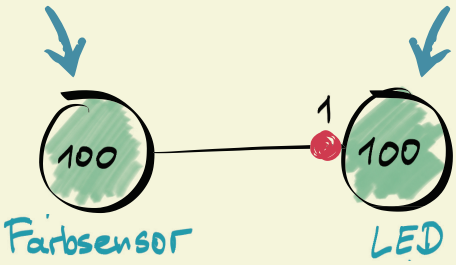




Lösung

So steckst du dein Programm zusammen:

Damit dein Roboter seine LED an die Farbe des Blattes anpasst, sollst du dein neuronales Netz wie folgt einstellen:



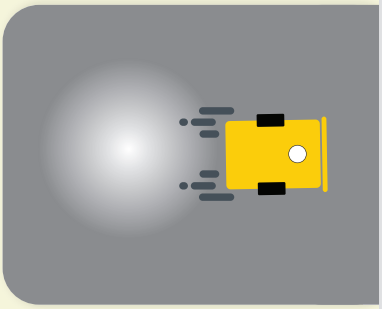
Aufgabe

Bringe deinem Roboter bei, zu jedem Blatt eine andere Farbe anzuzeigen.

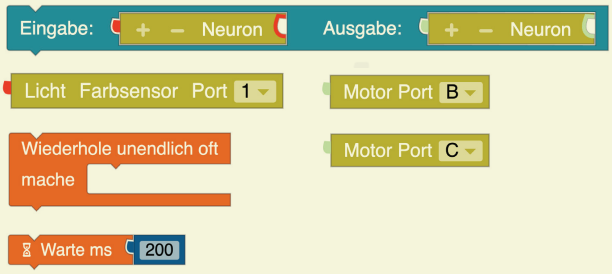
Inkognito

Lass deinen Roboter nicht erwischen!

Bringe dem Roboter bei, helle Orte zu meiden. Je heller die Umgebung, desto schneller muss dein Roboter fahren.



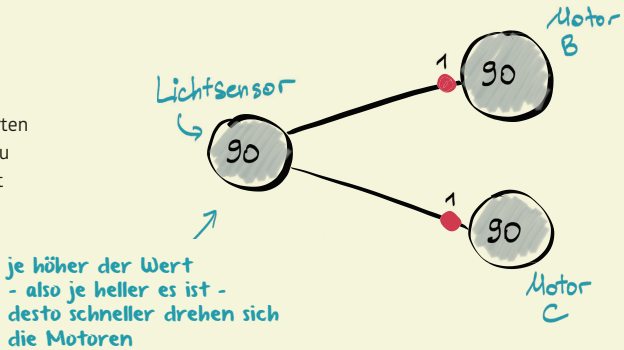
Du brauchst:



Lösung

So steckst du dein Programm zusammen:

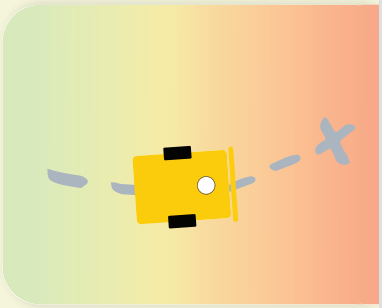
Damit dein Roboter die hellen Orte meidet – d.h. an hellen Orten schneller vorbeifährt – sollst du dein neuronales Netz wie folgt konfigurieren:



Vorsichtigkeit

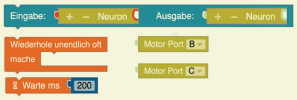
Bringe deinem Roboter die Verkehrsregeln bei 🤖

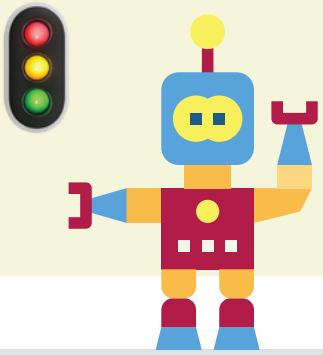
Auf “Grün” darf der Roboter mit voller Geschwindigkeit fahren, auf “Gelb” soll er langsamer werden und auf “Rot” muss er stehen bleiben.



Neue Blöcke, die du brauchst:

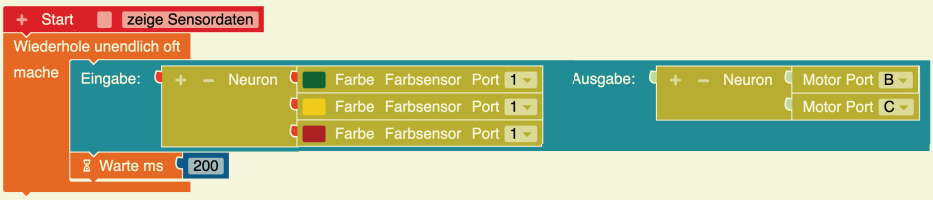
+ die Blöcke, die du schon kennst:



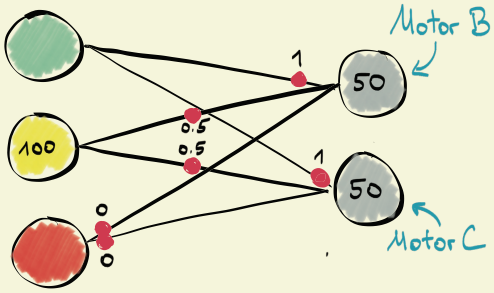


Lösung

So steckst du dein Programm zusammen:

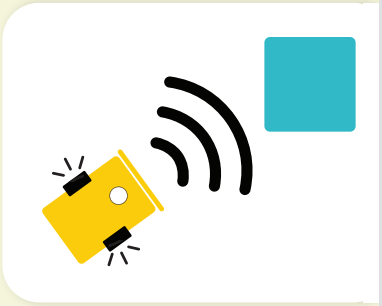


Damit dein Roboter die Verkehrsregeln beherrscht, sollst du dein neuronales Netz wie folgt einstellen:



fährt dein Roboter auf dem gelben Grund, so drehen sich seine Motoren nur mit halber Geschwindigkeit.

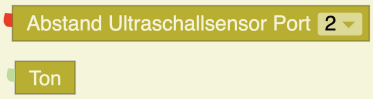
Lauter Abstand



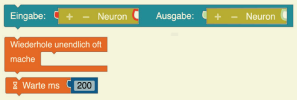
Lass den Roboter den Abstand zum Hindernis laut messen

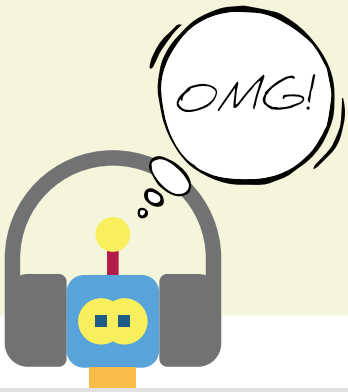
Bringe dem dem Roboter bei, dass je näher er zu einem Hindernis ist, desto tiefer soll der Ton sein, den er ausgibt.

Neue Blöcke, die du brauchst:



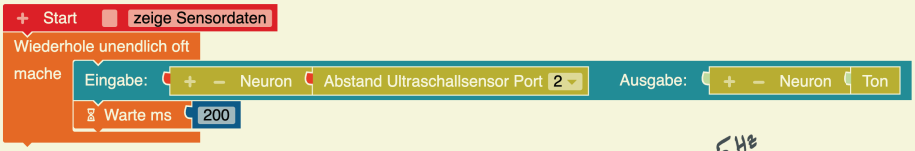
+ die Blöcke, die du schon kennst:



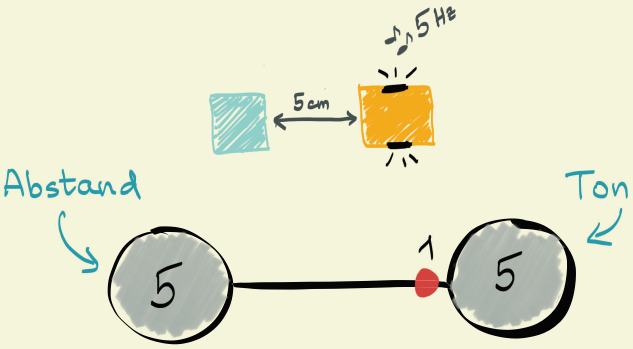


Lösung

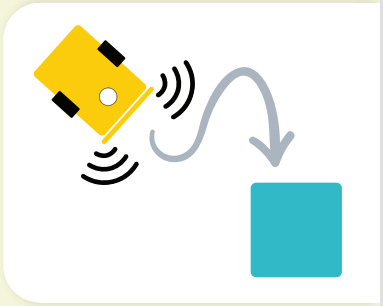
So steckst du dein Programm zusammen:



Damit der Roboter einen tiefen Ton ausgibt, je näher er zum Hindernis ist, sollst du dein neuronales Net wie folgt einstellen:



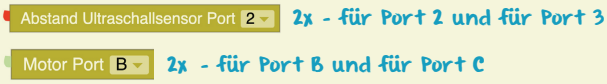
Freundschaft



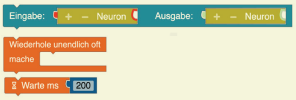
Auch Roboter können freundlich sein 🤖

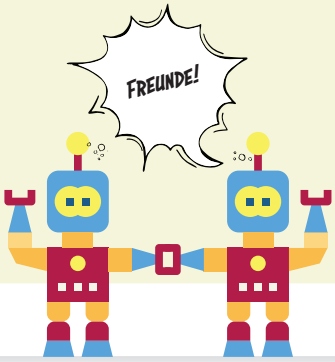
Bringe deinem Roboter bei, sich freundlich zu verhalten!
Sobald der Roboter ein Objekt in seiner Umgebung erkennt, soll er dem Objekt näherkommen.

Du brauchst:



+ die Blöcke, die du schon kennst:

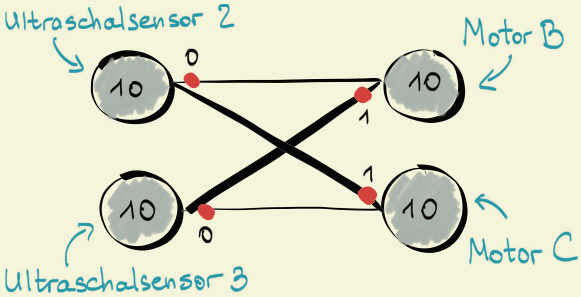




Lösung

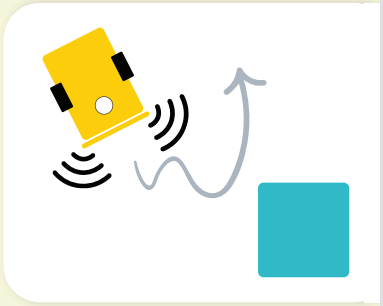
So steckst du dein Programm zusammen:

Damit dein Roboter sich freundlich verhält, sollst du dein neuronales Netz wie folgt konfigurieren:



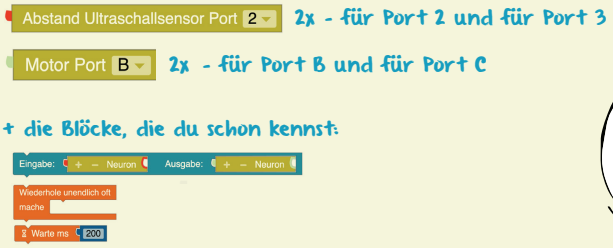
Angst

Lass deinen Roboter «Angst» vor Hindernissen haben



Bringe deinem Roboter bei, den Hindernissen auszuweichen. Sobald der Roboter ein Objekt in seiner Umgebung erkennt, soll er von diesem Objekt fliehen.

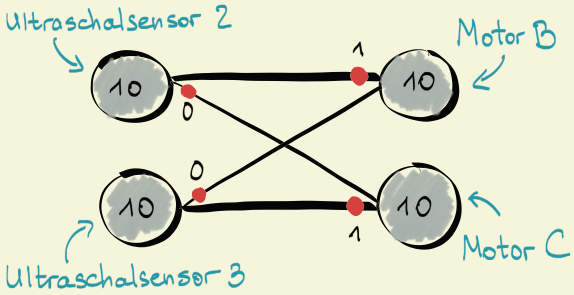
Du brauchst:



Lösung

So steckst du dein Programm zusammen:

Damit dein Roboter die Gegenstände meidet, sollst du dein neuronales Netz wie folgt konfigurieren:



Worin unterscheidet sich das Program auf dieser Lernkarte von dem Programm auf der Karte "Freundschaft" ?

Interesse

Lass deinen Roboter die Landschaft explorieren!

Bringe deinem Roboter bei, korrekte Begriffe zu den Gegenständen auf dem Bildschirm zu zeigen. Fährt der Roboter auf “Grün” - sollt der Text “Wiese” angezeigt werden. Fährt er auf die Farbe “Gelb” - soll er das Wort “Sand” anzeigen. Bei der Farbe “Blau” - soll er den Text “Wasser” ausgeben.

hier wird das Ergebniss ausgegeben

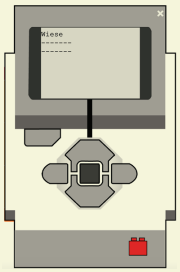
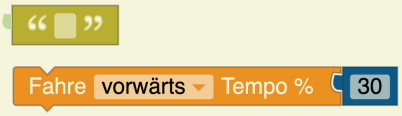


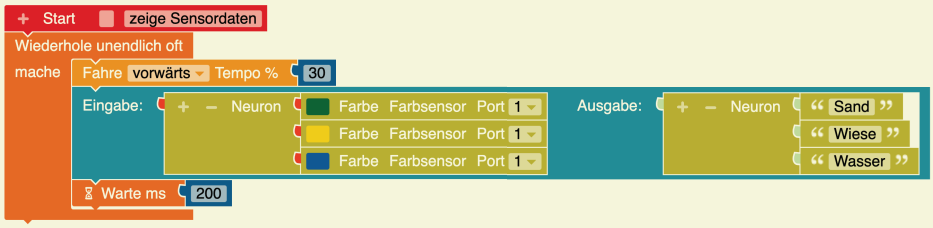
Bild: iStock.com/ Ilyaliren

Neue Blöcke, die du brauchst:

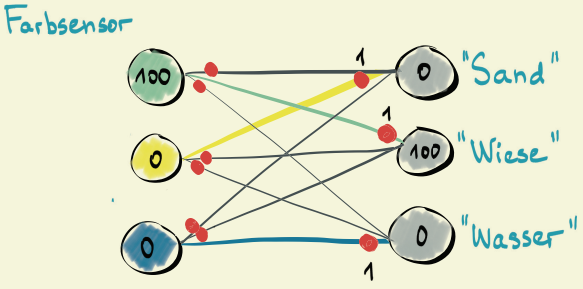


Lösung

So steckst du dein Programm zusammen:

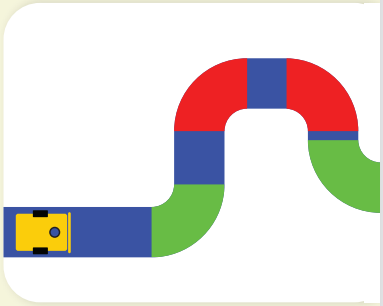


Damit dein Roboter die Landschaft exploriert und den richtigen Begriff zum richtigen Untergrund anzeigt, sollst du dein neuronales Netz wie folgt konfigurieren:



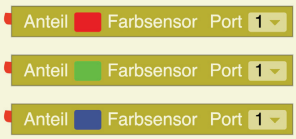
Rally

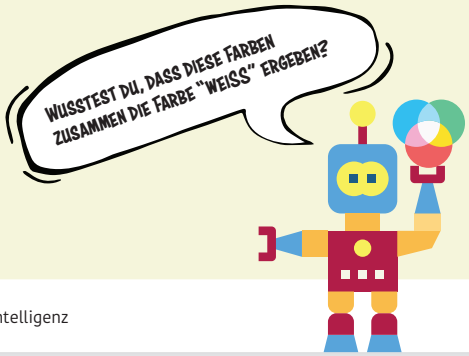
Lass deinen Roboter selbständig die farbigen Kurven meistern!



Auf Grün dreht sich der linke Motor langsamer als der rechte,
bei Rot dreht sich der rechte Motor langsamer als der linke,
bei Blau drehen sich die Motoren gleich schnell.

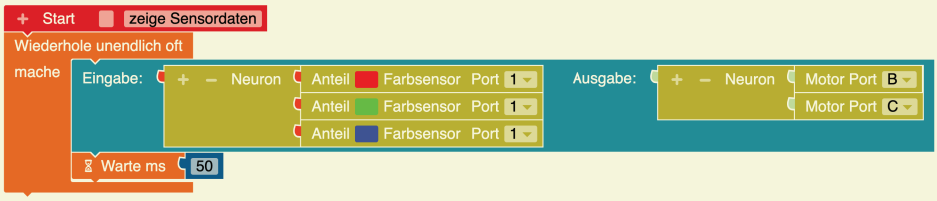
Neue Blöcke, die du brauchst:



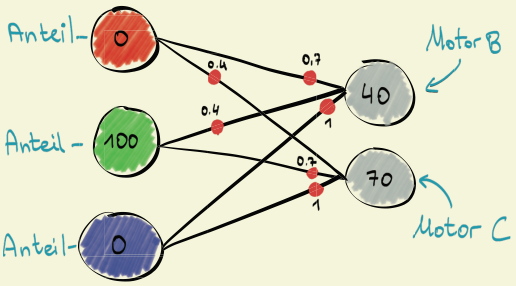


Lösung

So steckst du dein Programm zusammen:



Damit dein Roboter die farbige Kurve kriegt, sollst du dein neuronales Netz wie folgt konfigurieren:



Wie verhält sich der Roboter, wenn er über den weißen Hintegrund fährt? Warum verhält er sich so?

A.7 Q-learning Materials

A.7.1 Q-learning Cards

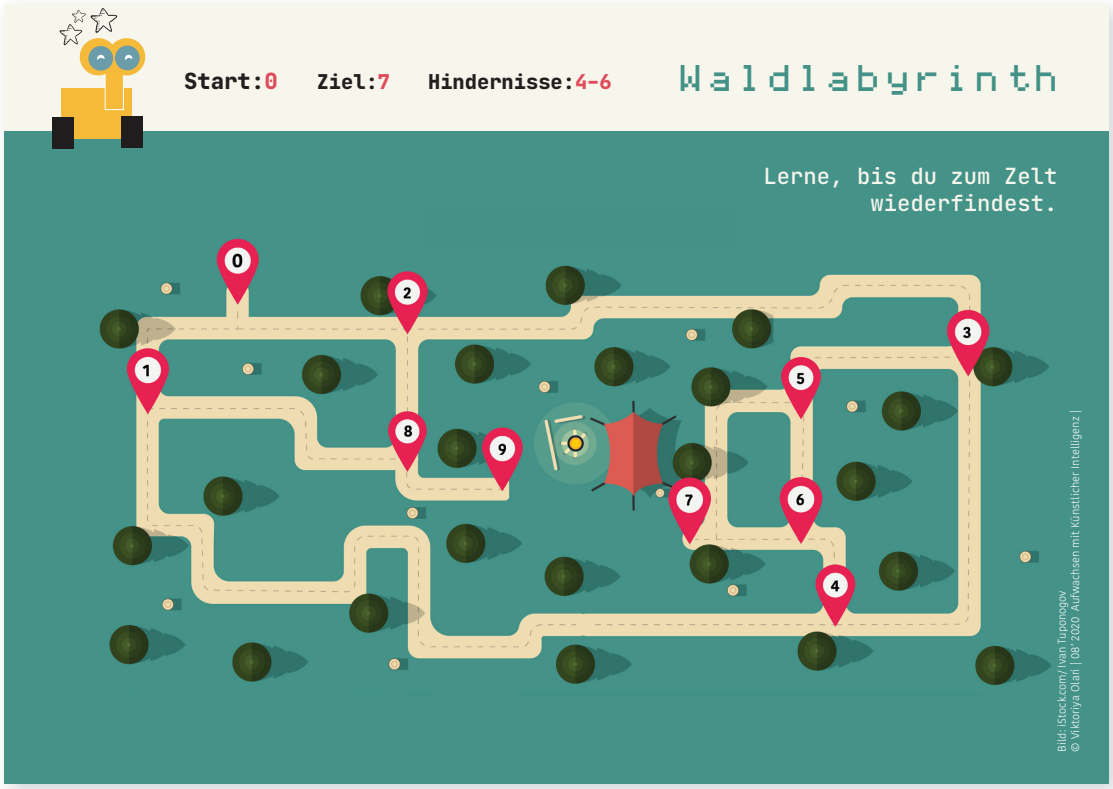
Start:3 Ziel:1 Hindernisse:3-2, 2-3

Was denkst du, welcher Weg ist der schnellste?

Probiere verschiedene Einstellungen aus! Spiele mit Episoden: 10, 100 oder 1000? Gibt's da einen Unterschied?

	lerne...	extra Belohnung	Tele-portation	Vor-erfahrung	Episoden	Zeit	Bester Weg
Test 1							
Test 2							
Test 3							

Mit welchen Paramtern findet dein Roboter den optimalen Weg? Markiere die Zeile mit einem Marker.



Start:0 Ziel:7 Hindernisse:4-6

Was denkst du, welcher Weg ist der schnellste?

	lerne...	extra Belohnung	Tele-portation	Vor-erfahrung	Episoden	Zeit	Bester Weg
Test 1							
Test 2							
Test 3							
Test 4							
Test 5							

Start:0 **Ziel:15** **Hindernisse:16-15**

In der Stadt

Lerne, bis die Route frei ist.

Start:0 **Ziel:15** **Hindernisse:16-15**

Was denkst du, welcher Weg ist der schnellste?

	lerne...	extra Belohnung	Tele-portation	Vor-erfahrung	Episoden	Zeit	Bester Weg
Test 1							
Test 2							
Test 3							
Test 4							
Test 5							

A.7.2 Q&A: Reinforcement Learning

Q & A : Bestärkendes Lernen

so steckst du den Q-Algorithmus zusammen

Wie lernt dein KI-Roboter mit dem Q-Learning Algorithmus?

Schaue dir dazu das Bild auf der Rückseite des Blattes an.

Was ist eine Episode?

Es ist ein Durchlauf des Algorithmus von einer Station zu der anderen.

Kann ich den Start oder das Ziel ändern?

Klar! Probier aus, wie schnell dein Algorithmus die optimale Strecke findet.

Was bedeuten die Sterne?

Die Sterne zeigen, wie attraktiv die Strecke für deinen Roboter ist.

sehr attraktiv ★★★★★ nicht attraktiv ★★★★★

Was hat diese gelbe Linie zwischen zwischen zwei Pins zu bedeuten?

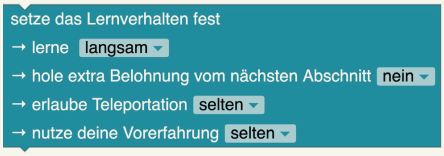
Es ist die Strecke, die dein Roboter aktuell durchläuft.

Warum beginnt mein Roboter immer an einer anderen Stelle?

Weil du wahrscheinlich die Teleportation erlaubt hast.

Wie berechnet der Roboter, ob die Strecke attraktiv ist?

Der Roboter berechnet aufgrund der Einstellungen, die du in diesem Block ausgewählt hast, ob die Strecke attraktiv ist oder nicht.



Ist lerne „langsam" eingestellt,

dann lernt der Roboter langsamer, dafür merkt er besser, was er gelernt hat. Schnelles Lerntempo ist nicht immer gut, denn dann vergisst dein Roboter auch schneller.

Ist extra Belohnung auf „ja" eingestellt,

dann schaut der Roboter zwei Schitte im Voraus, ob die Strecke danach auch eine attraktive ist. Extra Belohnung ist auch nicht immer gut, denn es kann den Roboter in die Irre führen. Die Strecke im übernächsten Schritt ist vielleicht gar nicht so gut.

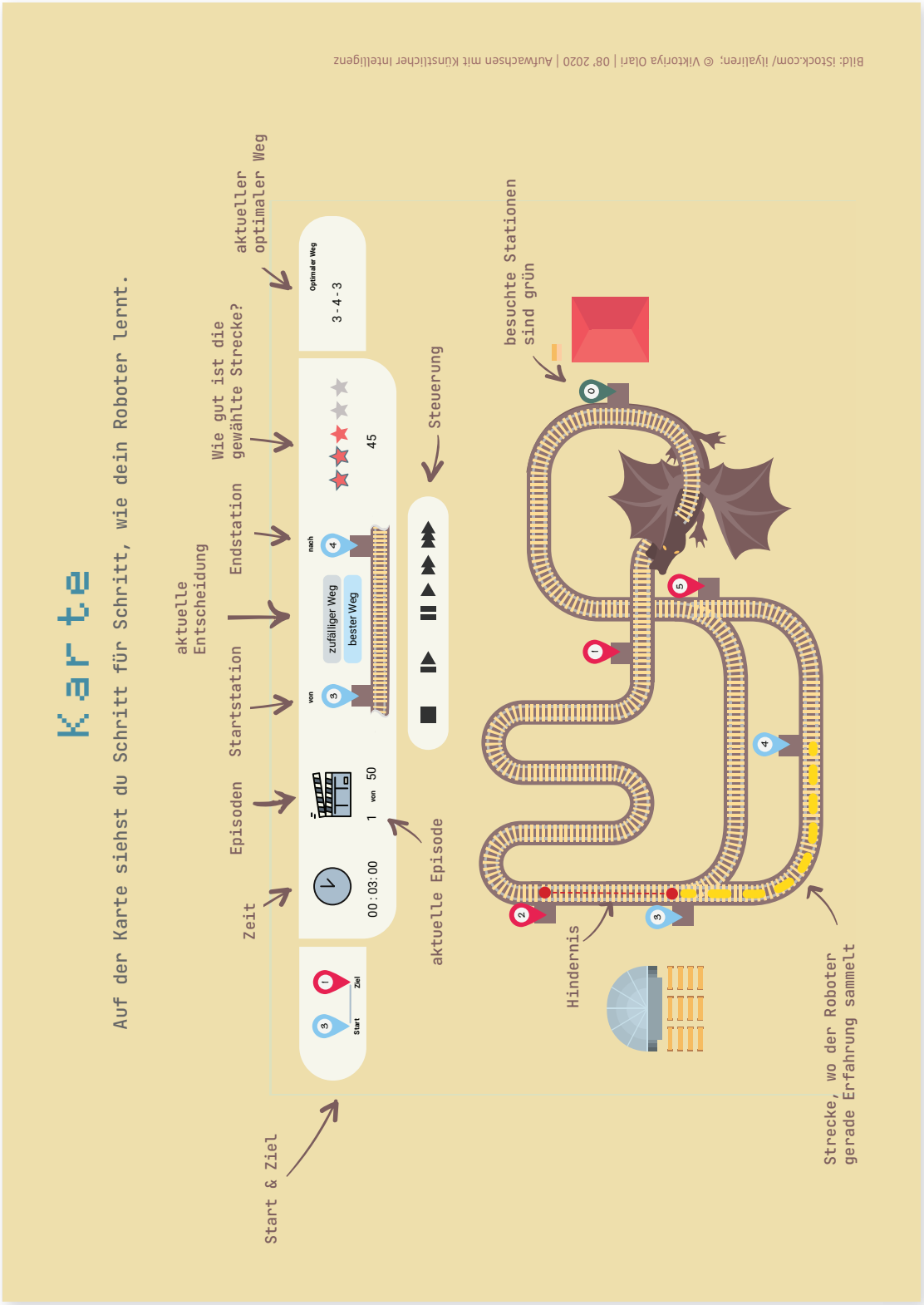
Ist Teleportation erlaubt,

dann springt der Roboter beim nächsten Schritt zu einer beliebigen Station und startet von da aus.

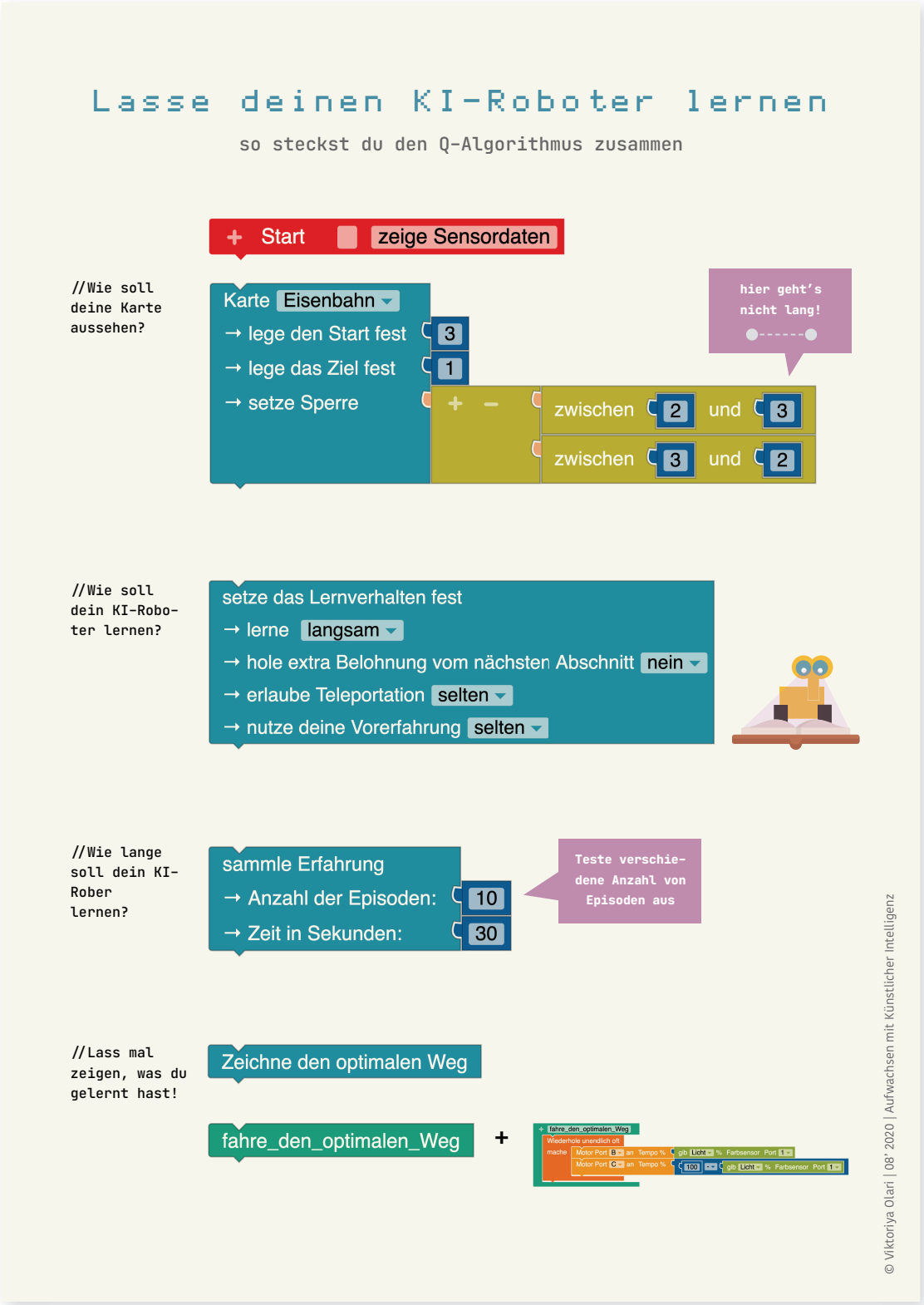
Darf der Roboter seine Vorerfahrung nutzen,

dann greift er bei jedem Schritt auf sein vorheriges Wissen zurück.

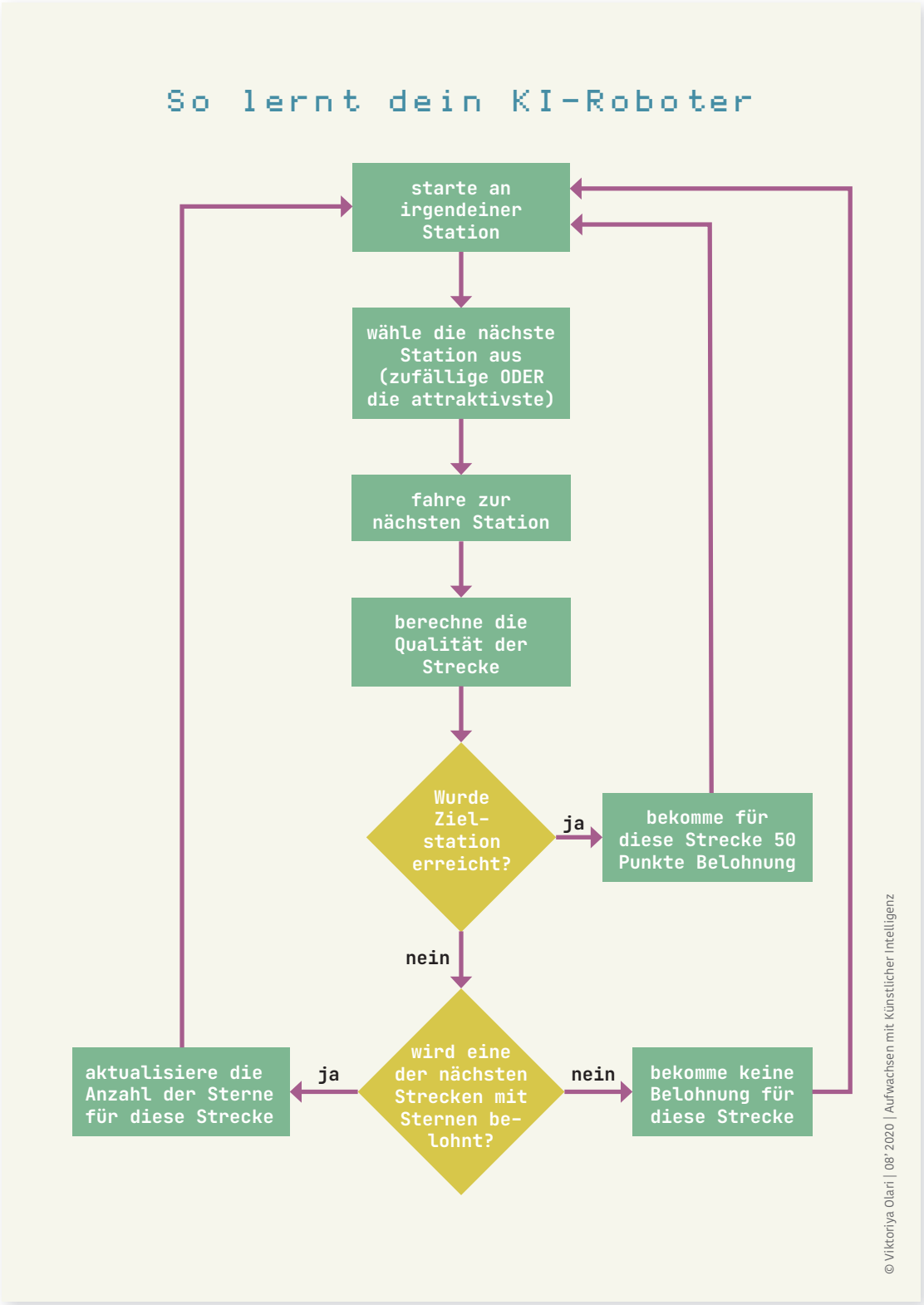
A.7.3 Q-learning Map



A.7.4 Q-learning Program



A.7.5 Q-Learning Algorithm Flow Diagram



A.7.6 Q-learning Observation Card

Beobachtungskarte

	Eisenbahn	Wald-labyrinth	In der Stadt
Minimale Zeit für den optimalen Weg			
Maximale Zeit für den optimalen Weg			
Maximale Episode-nanzahl, um zum Ziel zu kommen			
Minimale Episode-nanzahl, um zum Ziel zu kommen			

Was denkst du, was ist die beste Einstellung für deinen KI-Roboter?

Lerne	Extra Belohnung	Teleportation	Vorerfahrung