Bielefeld University

Faculty of Technology
Neuroinformatics Group

Dissertation

Scaffolding for learning from reinforcement: Improving interaction learning

Sascha Fleer

Supervisor Prof. H. Ritter

2020

Gedruckt auf alterungsbeständigem Papier °° ISO 9706

Abstract

This thesis analyses the psychological concept of “scaffolding” as a candidate for being able to facilitate the learning process not only of human learners but also of artificial agents. The original concept employs a range of techniques to improve the learning process of novices by analysing their current skill level, adjusts the task’s complexity, directs their attention and provides temporary support. By transferring these methods to the field of machine learning and proposing scaffolding as a general principle for guiding the learning process of an artificial agent, this work demonstrates that the learning performance can be improved significantly and even enables the learning of new skills that would otherwise be impossible.

First, the relations of the individual key aspects of this psychological theory to historical and recent findings in the field of machine learning are discussed. Second, a suitable refined definition of “scaffolding for machine learning” is put forward by positioning scaffolding as a special form of meta-learning that is inspired by psychology. As a result, four different scaffolds for reinforcement learning agents are designed for supporting different parts of the learning process by facilitating efficient acting, perception or pre-learning of sub-skills. In the final analysis, their positive impact is demonstrated by testing these supportive approaches on selected interaction problems.

Acknowledgement

First of all, I would like to express my gratitude to Professor Helge Ritter for his supervision and support during the last years. I also wish to thank the other members of the dissertation committee. Thanks to Alexandra Moringen, Oliver Lieske, Guillaume Walck and all other members of the Neuroinformatics Group. I greatly appreciate the valuable feedback given by Lukas Twardon and Christoph Ostrau. I am grateful for the financial, academic, and administrative support that I received from the CITEC Graduate School. Last but not least, I would like to express gratitude to my family for their permanent support along the way.

Declaration

- I am familiar with the current Doctoral Regulations of the Faculty of Technology.
- I have written the thesis by myself, no parts of the text have been copied from a third party or my own assessed work without due attribution, and aids and sources used have been duly referenced and attributed in the text.
- No third party has received monetary benefits or benefits in kind, directly or indirectly, for any intermediary services or work related to the contents of the submitted thesis.
- I have not yet submitted the thesis as an assessed work for a public or other academic examination.
- I have not submitted the same paper, a similar paper, or another paper as a thesis to another university.

Bielefeld, 2020

Sascha Fleer

Contents

1 Introduction **1**
1.1 Outline 3

I The toolbox **7**

2 Reinforcement learning — a paradigm of human inspired artificial intelligence **9**
2.1 The basic description of a reinforcement learning problem 10
2.2 The Bellman equation 12
2.3 Optimal value functions 13
2.4 Q-learning using linear function approximators 15
2.4.1 Approximating the state-action space for a set of discrete actions 17
2.5 Policy gradient methods 18
2.5.1 The REINFORCE algorithm 20
2.6 Going deeper with neural networks 22
2.6.1 Deep Q-learning 25
2.6.2 Asynchronous models 25
2.7 Summary 27

3 The guiding principle of scaffolding **29**
3.1 The concept of scaffolding in educational psychology 32
3.1.1 Learning how to ride a bicycle: an example for scaffolded learning 33
3.2 Teaching devices: employing computer-based tools for scaffolding the learning process of humans 36
3.2.1 Scaffolding the learning of a foreign language with the help of a computer-based tool 37
3.3 Scaffolding artificial agents by organizing learning on a meta-level . 39
3.3.1 Recruiting and maintaining the learner's attention 40
3.3.2 Simplifying the task 43
3.3.3 Modelling and demonstration 46

3.3.4 Ongoing diagnosis and assessment 47
3.3.5 Fading support and eventual transfer of responsibility 48
3.3.6 Summary 49
3.4 Reformulating scaffolding as a principle for guiding the learning process of machines 50
3.4.1 Scaffolding in practice: inject meta-knowledge by compiling individual auxiliaries 51
3.5 A research map for scaffolding in machine learning 53
3.5.1 Four research questions for scaffolding an artificial agent . . . 54
3.6 Summary 56

II Scaffolding: a universal approach for fostering the learning process 57

4 Scaffolding attention control by exploiting "perceptive acting" 59
4.1 The concept of entropy and mutual information in the context of reinforcement learning 60
4.1.1 Exploiting mutual information as a ranking criteria for action sets 62
4.2 Applying the concept to complex environments 63
4.2.1 Estimating the probability distribution of state transitions . . 64
4.2.2 Estimating the entropy & mutual information 66
4.3 Summary 67

5 Scaffolding attention control by exploiting "active visual perception" 69
5.1 The recurrent attention asynchronous advantage actor-critic model . 70
5.1.1 Training 76
5.2 Summary 78

6 Scaffolding the learning of efficient haptic exploration using "active haptic perception" 79
6.1 From human haptic perception to robotics 80
6.2 The haptic attention model 81
6.2.1 Training 84
6.3 Summary 85

7 Scaffolding the agent's internal representation through skill transfer 87
7.1 The combination of a structured curriculum with transfer learning — 4 strategies of skill transfer 88
7.1.1 Strategy 1 & 2: simple techniques for skill transfer 90

7.1.2 Strategy 3 & 4: refining skill transfer by analysing the learners way of perception . . . 92
7.2 Summary . . . 95

III Facilitating the learning process of interaction problems: testing the proposed scaffolding approaches 97

8 A learning domain for mediated interaction 99
8.1 The general design concept of the simulation world . . . 100
8.2 Realization of a 2D simulation world with simplified physics . . . 101
8.3 Perceiving & acting: defining a suitable state and action space for multi-object interaction scenarios . . . 103
8.3.1 A general set of discrete actions . . . 103
8.3.2 Perceiving the environment . . . 104
8.4 Designing suitable learning scenarios . . . 106
8.5 Learning with a distance-related sensory input . . . 109
8.5.1 The construction of a linear Q-learner . . . 110
8.5.2 Creating a deep Q-Learner . . . 113
8.6 Summary . . . 114

9 A first scaffold for learning the "Extension-of-Reach Scenario": determining the best action set 117
9.1 Experiments . . . 118
9.2 Results . . . 121
9.3 Discussion . . . 124

10 A second scaffold for learning the "Extension-of-Reach Scenario": structuring the learning process 127
10.1 Experiments . . . 129
10.1.1 Learning and evaluation . . . 129
10.1.2 Applying the four transfer learning strategies . . . 131
10.2 Results . . . 132
10.3 Discussion . . . 141

11 Scaffolding the learning process through "active visual perception": an attention based approach 145
11.1 Experiments . . . 147
11.2 Results . . . 149
11.3 Discussion . . . 152

12 A scaffold for enabling "active haptic perception": learning efficient haptic exploration **155**
12.1 Designing the simulation world . . . 156
12.1.1 The three building blocks of the simulation world . . . 156
12.1.2 Implementing essential control primitives . . . 159
12.1.3 The classification task . . . 160
12.1.4 Creation of the dataset . . . 161
12.2 Implementation of the haptic attention model . . . 163
12.3 Experiments . . . 164
12.4 Results . . . 164
12.5 Discussion . . . 166

IV Conclusion **169**

13 Summary, conclusion & outlook **171**
13.1 Four scaffolding approaches — a summary . . . 171
13.2 Conclusion . . . 174
13.3 Recommendations for future research . . . 175

Bibliography **177**

Appendices **199**

A Pseudocode **199**

B Used learning parameters **203**
B.1 Linear Q-learning . . . 203
B.1.1 State representations . . . 203
B.1.2 Learning the "Extension-of-Reach Scenario" using different coordinate systems . . . 204
B.2 Deep Q-learning . . . 206
B.3 Recurrent attention advantage actor-critic model . . . 208
B.4 Haptic attention model . . . 209

C Floating Myrmex sensor: experimental results **211**

D Supplementary material **213**

List of Figures

1.1 The presented scaffolding approaches 5

2.1 Schematic visualization of reinforcement learning 11
2.2 Schematic visualization of an actor-critic architecture 19
2.3 Multilayer Perceptron 22
2.4 Activation Function 23
2.5 The basic structure of an asynchronous model 27

3.1 Illustration of the *zone of proximal development* 32
3.2 A simple illustration of the concept of scaffolding 33
3.3 The five key aspects of scaffolding 34
3.4 The *perception-action cycle* 41
3.5 Relating scaffolding to machine learning approaches 50
3.6 The computational skeleton for scaffolding artificial agents 51

4.1 The influence of action on the dynamics of the environment 61
4.2 Connection between world and state space 64
4.3 Illustration of a Voronoi diagram 65

5.1 An illustration of the workers of the RAA3C model 72
5.2 An illustration of the glimpse network. 74
5.3 A general illustration of the location network. 75
5.4 An illustration of the context network 76
5.5 An illustration of the actor-critic network. 77

6.1 Illustration of the HAM 83

7.1 The four strategies for skill transfer 89
7.2 Generating a structured set $\mathcal{C}$ 93

8.1 Schematic description of the learning domain 102

8.2 Interaction of the agent with the world 104
8.3 Illustrations of the sensory inputs 106
8.4 Illustrations of the different learning scenarios. 107
8.5 Illustrations of the different sampling regions for the objects 109
8.6 Illustration of two Gaussian RBFs with different mean and variance . . 112
8.7 Boyan decay 113
8.8 The designed leep Q-learner 114

9.1 Illustration of different coordinate systems 119
9.2 Course of learning for the “Extension-of-Reach Scenario” — FSR 122
9.3 Course of learning for the “Extension-of-Reach Scenario” — RBF 123

10.1 The two stages of the “Extension-of-Reach Scenario” 128
10.2 *Time to Threshold* performance metric 131
10.3 Learning Process with and without scaffolding — FSR 133
10.4 Learning Process with and without scaffolding — RBF 134
10.5 Learning Process with and without scaffolding — DQN 135
10.6 Comparison of R_{global} for the different learners 137

11.1 Full image vs. salient parts 145
11.2 Visualization of the simulation world 147
11.3 Performance evaluation for different numbers of glimpses 149
11.4 Comparison with random location policy 150
11.5 Comparing different sizes of context images 151
11.6 Initializing different LSTMs 152

12.1 Developed simulation environment with the KUKA robot arm 156
12.2 Comparison of simulated and real sensor 158
12.3 Exemplary objects employed in an identification task. 158
12.4 Experimantal setup of the object classification task 161
12.5 Visulaization of the learning performance 165
12.6 Photo of the real-world setup 168

List of Tables

9.1 Ranking R_{global} — $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ for the Fixed-Sparse-Representation . . 124
9.2 Ranking R_{global} — $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ for RBF representation 124

10.1 Evaluation of the learning process — FSR 138
10.2 Evaluation of the learning — RBF 139
10.3 Evaluation of the learning process — DQN 140

11.1 Highest average reward $\langle R \rangle$ for the training runs with $\boldsymbol{\pi}_{\mathbf{loc}}$ and $\boldsymbol{\pi}_{\mathbf{rloc}}$. 150

12.1 Classification performances — robot simulation 165

B.1 Search-ranges for parameters for the linear Q-learner 204
B.2 Learning parameters — "World System" 204
B.3 Learning parameters — "Tool-Fixpoint System" 205
B.4 Learning parameters — "Target-Tool-Goal System" 205
B.5 Learning parameters — "Tool-Centroid System" 205
B.6 Learning parameters — "Target-Tool System" 206
B.7 Learning parameters — "Goal-Tool System" 206
B.8 Search-ranges for parameters for the deep Q-learner 207
B.9 Learning parameters — deep Q-learner 207
B.10 Learning parameters — RAA3C 208
B.11 Search-ranges for parameters for the deep Q-learner 209
B.12 Learning parameters — HAM 209

C.1 Classification performance — floating Myrmex sensor 211

List of Algorithms

1 Skill transfer — *Strategy 1* 91

2 Skill transfer — *Strategy 2* & *Strategy 3* 91

3 Skill transfer — *Strategy 4* 95

4 Linear greedy Q-learning with eligibility traces 199

5 Deep Q-learning 200

6 Asynchronous advantage actor-critic algorithm 201

Introduction 1

Since ancient times, the craving for controllability appears to be one of the main catalysts of learning. The desire of humans and animals to control their environment and to make use of it for their own benefits determines their behaviour and leads to the eagerness to learn new skills in order to reach this goal. Adapted from this behaviour, one of the main issues in the field of intelligent systems is the question how to endow systems like robots with new interaction capabilities in the simplest possible way to optimize their control over the environment.

In the last years, robotic systems have become an essential part in industry, but also strive to become a more and more important aspect in our daily lives. The development of self-driving cars, kitchen gadgets, robots for automatic gardening or hoovering strongly influences the society, potentially leading to a point where robots might be accepted as native interactive tools in nearly every kind of situation. At present, the accustomed assistance of robots in daily life tasks is a challenging matter. It not only requires the fast learning of new assigned duties but also the ability to learn in ways that enables a human to easily instruct the robot. Many daily actions for example require some form of "mediated interaction" such as pouring the contents of a mug, accessing a book in a drawer or taking clothes out of a wardrobe. In order to handle these kind of scenarios, the robot can interact with the target object only after having actively prepared access via the intermediate use of some auxiliary "mediator object". Closely related to those activities is tool use, a learned capability found in humans, but also in some other species, such as some primates or birds [1]. It has been linked to higher cognition and is obviously also a desirable capability for robots to become more adapt at many daily tasks typically arising in human environments [2, 3]. Especially when the robot's morphology is resembling a human, the learning of affordances, i.e. the ability to reasonably use a tool or object within the current tasks context, is also an important aspect that is brought into focus[2, 4].

The challenge of robot learning Endowing artificial agents like robots with the ability to learn these kind of tasks in a fast and efficient manner is a challenge that has not yet been solved. Especially when the tasks require the robot to physically interact with its surroundings, many demands have to be faced. An adequate learning performance has to be reached within a short amount of time in order to reduce wear and tear, together with the likeliness that the robot harms itself or its environment because of suboptimal behaviour. The most encouraging way to achieve this goal is offered by the behaviour of natural cognitive agents themselves [5]. Stimulated by discoveries in psychology and neuroscience [6], advances in machine learning methods (deep learning [7], hierarchical reinforcement learning [8–10], efficiently solvable Markov decision problems [11]) have led to impressive improvements of what artificial systems can learn. In this context, *reinforcement learning* has been considered as an attractive method to enable robots to master various kinds of tasks. In this biologically inspired class of algorithms, the learner explores the environment via direct interaction while adapting its behaviour to the gathered experience in order to maximize a task dependent reward signal. While conceptually very attractive, a major problem arises from often prohibitively large numbers of required learning steps to achieve reasonable performance. Especially in cases when the sensory information of the environment's current state is represented using complex non-linear function approximators such as deep neural networks, the attending long training times might render learning impossible when real robots are involved in the learning process.

Scaffolding: a possible remedy? Long training times or even the absence of learning due to complex hierarchical problems and sparse rewards are popular issues that are discussed in many works within the field of machine learning [12–16]. So far, many different strategies have been introduced and proposed within the literature that all attempt to improve the learning process by inventing new algorithms or improvements for extending present methods. In this work, the learning process of reinforcement learning methods is examined under the perspective of a general principle called *scaffolding*. On the one hand, scaffolding is the term for a temporary structure outside of buildings that supports the process of constructing, repairing or cleaning. On the other hand, it is also a principle from educational psychology [17–20]. In this field, scaffolding is a theory for fostering expert-learner interaction and employs a range of techniques for guiding the learner. As a consequence, the learning process of a novice can be greatly improved. Scaffolding even has the ability to enable the learning of new skills that would else be rendered impossible.

1.1 Outline

The content of this thesis is mainly based on the following publications:

- Sascha Fleer and Helge Ritter. “Comparing Action Sets: Mutual Information as a Measure of Control”. In: *Artificial Neural Networks and Machine Learning – ICANN 2017*. Cham: Springer International Publishing, Oct. 2017, pp. 68–75
- Sascha Fleer and Helge Ritter. “Skill Transfer for Mediated Interaction Learning”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. Nov. 2018, pp. 1–8. DOI: `10.1109/HUMANOIDS.2018.8624951`
- Sascha Fleer and Helge Ritter. “Solving a Tool-Based Interaction Task Using Deep Reinforcement Learning with Visual Attention”. In: *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization*. Ed. by Alfredo Vellido et al. Cham: Springer International Publishing, 2019, pp. 231–240
- Sascha Fleer et al. “Learning efficient haptic shape exploration with a rigid tactile sensor array”. In: *PLOS ONE* 15.1 (Jan. 2020), pp. 1–22. DOI: `10.1371/journal.pone.0226880`

In summary, the principle objective of this work is to shed light on the following question:

> Can the principle of **scaffolding** be applied to the area of **reinforcement learning** as a general guiding principle for improving and accelerating the learning process?

At first, the psychological theory of scaffolding is transferred to the field of machine learning. It is then exploited for deriving four different kinds of approaches which are offering possible ways to aid the learning process for an artificial agent. The approaches are invented with a special focus on learning tasks that are connected to mediated interaction. For this reason, they are eventually examined and analysed using testbeds that are related to these kind of problems.

The thesis can be divided into four parts:

Part 1: the toolbox In the first part, the basic mechanisms of reinforcement learning are introduced, together with some common algorithms that are utilized within this work (Chapter 2).

This work proposes **scaffolding for machine learning** as a universal concept that offers ways of supporting different parts of the learning process and is thus able to tackle the problems of slow or inefficient learning. The basic idea of this theory from educational psychology is presented in Chapter 3. It is related to various approaches from the field of machine learning to find commonalities and eventually to refine the psychological concept into a general guiding concept for developing supportive approaches for artificial learners. Based on these results, general parts of the learning process — with emphasis on reinforcement learners — are identified to have a promising potential for optimization. These parts are then studied in order to create a suitable scaffold which is able to offer temporary or permanent support.

Part 2: four approaches for scaffolding the learning process In this work, two important parts of the learning process are studied that have a promising potential for optimization (see Figure 1.1). The first part is **the perception of the learning domain**. Therefore, three different scaffolds are constructed in Chapter 4 – 6 for supporting different kinds of perception: "perceptive acting", "active visual perception" and "active haptic perception". The second part, presented in Chapter 7, examines **the internal representation of the employed learning model**.

All in all, **four scaffolding approaches** are presented in this work:

- A permanent scaffold for the learning process by identifying and selecting the most suitable action set for the given learning environment [21].
- A permanent scaffold that enables the integration of efficient active perception [23].
- A scaffold for endowing an agent with the ability of active haptic perception which can be used to learn efficient haptic exploration of the domain [24].
- A temporary support for improving the learning through the pre-training of sub-skills using a combined transfer and curriculum learning approach that is refined by key characteristics of scaffolding [22].

Part 3: testing the proposed scaffolding apporoaches The next step is to test the applicability and efficiency of the four introduced ways of scaffolding. For this purpose, **two learning scenarios are designed** that are considered as a reasonable choice to be employed as testbeds. Although the goal of this work is to design scaffolds that might be able to widen the applicability of robots to

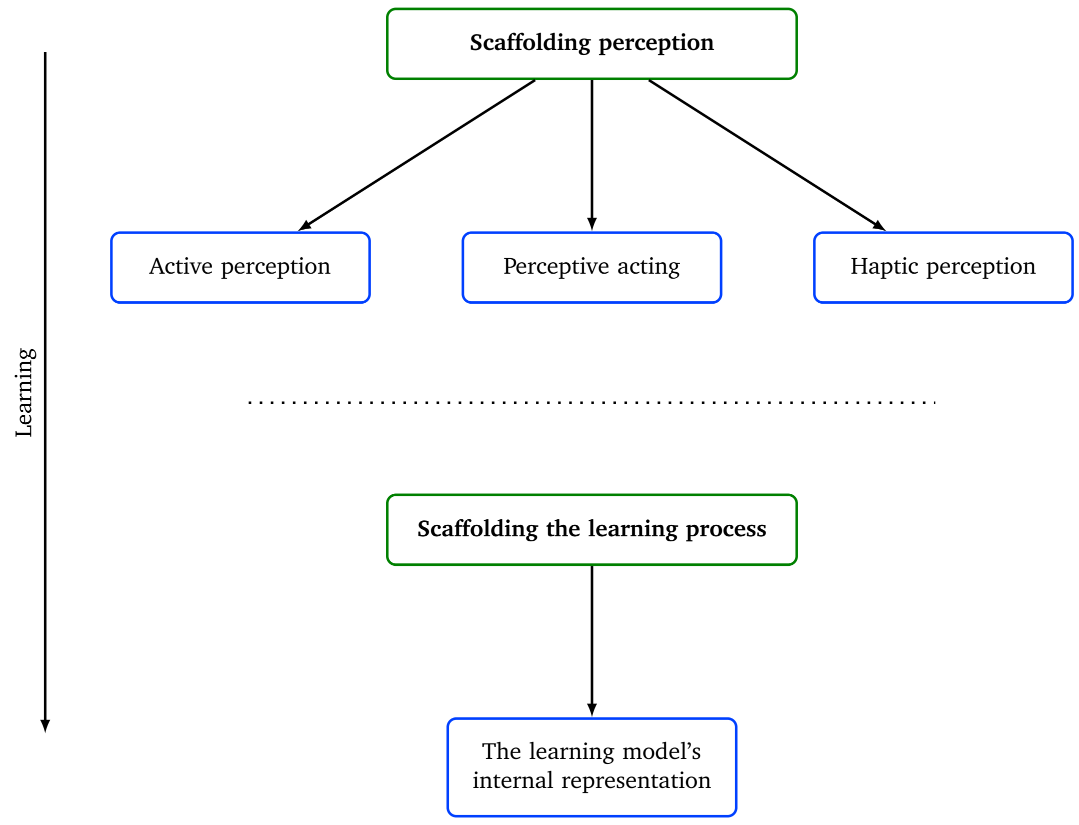


Fig. 1.1.: Illustration of the four presented scaffolding strategies. The first three target the different ways an artificial agent can perceive its surroundings. The last strategy tries to directly scaffold the internal representation of the employed learning model. The arrow on the left side indicates the chronological order of these two stages during learning as the agent first has to perceive its surroundings before it is able to learn based on the gathered information.

a new class of learning problems, simulations are the most promising way to get useful results and a clear outline of this research area. This simplification allows laying the focus on the investigation of scaffolding the learning process while circumventing hardware issues and low-level control problems that come along with a real robot.

In the first scenario, an artificial agent has to **learn to solve "mediated interaction tasks"**. For that reason, a simplified 2D world with simulated physics was designed that is described in detail in Chapter 8. As mentioned before, in these tasks the desired effects cannot be generated through direct interaction, but instead require the learner to discover how to exert suitable effects on the target object through involving a suitable "mediator object" or "tool". The need to learn first how to use the tool in a meaningful way before trying to solve the actual task indirectly imposes a hierarchy on the learning process. Within this learning domain, it is then possible to test the scaffolding based on “perceptive acting” (Chapter 9), “active visual perception” (Chapter 11) and the “model’s internal representation” (Chapter 10).

One scaffold is designed to enable efficient “active haptic perception” and thus has specific requirements to the learning domain like the existence of an artificial agent that is capable of tactile sensing. For this case, a special, simulated 3D world is constructed for learning. In this simulation, a KUKA robot has to **learn to identify different objects solely based on tactile data and the sensor’s position and orientation** (Chapter 12).

Part 4: discussion Chapter 13 gives a comprehensive summary of the whole thesis, together with a conclusion and some suggestions for future work.

Part I

The toolbox

2 Reinforcement learning — a paradigm of human inspired artificial intelligence

One of the bigger visions in the field of intelligent systems is to endow an artificial agent with the ability to solve human-level problems. To deal with such complicated tasks, the agent has to explore the learning domain autonomously by performing actions that affect the environment directly and learn from these effects. Unfortunately, most approaches in machine learning bypass this part of direct interaction between the agent and the environment. As a consequence, the effects of the agent-environment interaction during the learning process are scarcely investigated. A strong opposition is given by reinforcement learning, where the agent learns through the direct interaction with the environment [25]. Because of this human-like way of learning, it is treated as one of the best paradigms for artificial intelligence. Even though learning based on this class of methods tends to be slow, it played an important role in major breakthroughs in the field of machine learning. Examples can be found in various fields, while recently learning to play games has become one of the rising areas for test beds. Beginning with Checkers [26, 27] in the 1960s and Backgammon [28–30] in the 1990s, it is now possible to create a reinforcement learner that is capable of learning Go — currently the most complex board game [31, 32]. Some years ago, even video-games were utilized as challenging learning scenarios for newly developed reinforcement learning algorithms and are now seen as popular benchmarks. As a result, artificial agents are performing better than humans on games that were developed for the ATARI platform by relying only on the raw visual input of the environment [7, 33]. Encouraged by solving this kind of games better than a human, many other video-games were utilized as challenging learning scenarios for newly developed reinforcement learning algorithms. Examples are arcade-like fighting games like "Super Smash Bros" [34], first person shooters like "Doom" [35–37], the popular game "Minecraft" [38, 39] and also first approaches in complex strategy games such as "Starcraft II" [40]. The fruitful results in this area have not only increased the popularity of this specific class of machine learning algorithms, but also played an important role in developing learning architectures in the complex field of robotics. Some examples are learning to flip pancakes [41], to grasp objects using raw camera images [42, 43], efficient collision avoidance for real-world quadrotors, and real-world RC cars [44], stacking blocks with a robot arm [45] or — at least in simulation — complex dexterous manipulation [46, 47].

Outline In this chapter, the basic concepts of reinforcement learning that are necessary in order to understand the methods and approaches that are developed throughout this work are presented. First, the basic concepts of reinforcement learning such as the *state and state-action value* or the *Bellman-equation* are explained. In a next step, popular learning methods are presented, ending with a short introduction into the field of reinforcement learning using non-linear function approximators like (deep) neural networks.

2.1 The basic description of a reinforcement learning problem

Reinforcement learning is a class of machine learning algorithms whose underlying mechanism is inspired by behaviourist psychology, where the learner and decision maker — called *agent* — learns through direct interaction with the *environment* [25]. To explore and interact with the given learning domain, the agent executes *actions* that directly alter its surroundings. The resulting new *states* of the environment are presented through a *state or sensory signal* that encodes every information the agent is able to receive about the current situation into a *state vector*. It should thus provide the agent with all relevant information for solving the encountered problem. A second important characteristic demanded by the state signal is that it has to fulfill the *Markov property*, i.e. being independent of the history from previously visited states. The agent also receives a scalar control signal called the *reward*, leading to a feedback how much the chosen action benefits the agent in its current situation with respect to the primary learning goal.

Using this information, a reinforcement learning problem can be formulated as a *Markov decision process* [48]. It is then defined by the tuple $(\mathcal{S}, \mathcal{A}, P^{\mathcal{A}}, \mathcal{R}, \gamma, \mathcal{S}_0)$, where $\mathcal{S}$ denotes the set of states and $\mathcal{A}$ the set of admissible actions. $P^{\mathcal{A}}$ is the set of transition matrices, one for each action $a \in \mathcal{A}$ with matrix elements $P^a_{\boldsymbol{s},\boldsymbol{s}'}$ specifying the probability to end up in state $\boldsymbol{s}' \in \mathcal{S}$ after taking action a when in state $\boldsymbol{s} \in \mathcal{S}$. The value $r \in \mathcal{R} \subset \mathbb{R}$ specifies the scalar reward which the agent receives after ending up in $\boldsymbol{s}'$. $\mathcal{S}_0 \subseteq \mathcal{S}$ defines the set of starting states. The discount factor $\gamma \in [0, 1)$ balances the weighting between present and future rewards.

The interaction of the agent with its environment is — in most cases — realized through a sequence of discrete steps as illustrated in Figure 2.1. At every discrete time step t, the agent is in a state $s_t \in \mathcal{S}$, in which it applies an action $a_t \in \mathcal{A}(s_t)$. The action induces the transition to the next state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$, were the agent receives a reward $r_t \in \mathcal{R}$.

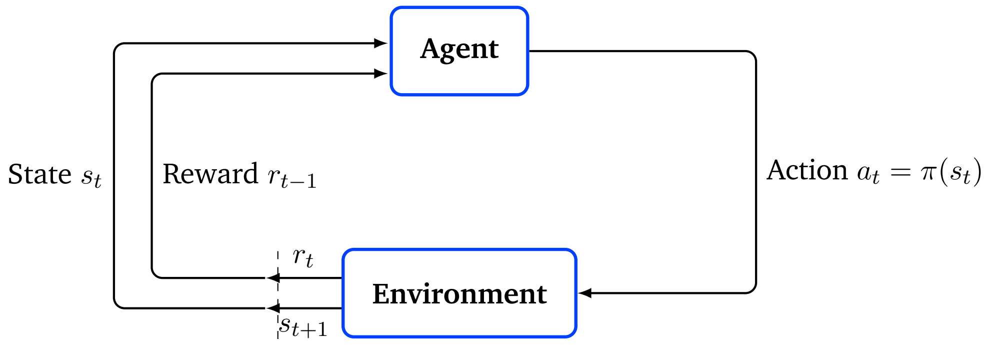


Fig. 2.1.: Schematic visualization of reinforcement learning as described in [25]

Maximizing the reward In reinforcement learning, the agent tends to maximize the reward over time. So, if the agent has to learn to solve a specific problem, it has to be formulated entirely in terms of the reward signal. This formulation of *tasks* in terms of a reward signal is one of the features which defines this class of machine learning algorithms inspired by the behaviour of humans and animals. Nevertheless the correct formulation of a task in terms of the reward is an important factor of the learning success. It is important to keep in mind that the reward signal should always define *what* the agent should achieve and not *how* it has to be achieved. An example would be to reward the agent for winning a game of chess rather than for specific intermediate configurations of the chess-pieces. As a consequence, rewards may be sparse as many actions of the agent may not be rewarded at all.

To maximize the reward over time, the agent has to adapt its strategy for interacting with the environment, i.e. the decisions which actions to choose in which situations. This is realized through the agent's *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. The policy $\pi(a_t|s_t)$ is a function which maps a state s_t at time-step t to the probability of selecting a specific action a_t. Reinforcement learning algorithms are now aiming for the optimal policy in order to maximize the accumulated discounted future reward

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k}. \tag{2.1}$$

Here, T is set to infinity for continuous tasks. When the task is episodic and ends after a certain terminal step, T is finite.

2.2 The Bellman equation

The reward signal r_t indicates how good it is to be in the current state s_t. As the reward rates the state s_t in the current situation, it does not indicate the advantage of visiting the current state in the long run. In order to quantify this measurement, the *state value* or *value function* $V^\pi(s_t)$ is introduced. The state value measures the expected return R_t the agent is able to achieve by starting in the current state s_t and following π afterwards. It thus depends on the current policy and can be learned through the experience the agent gathers during its exploration of the environment. For Markov decision processes, V^π can be formulated as

$$V^\pi(s) \equiv \mathbb{E}^\pi \left[R_t \mid s_t = s, \pi \right]. \tag{2.2}$$

Here, $\mathbb{E}^\pi$ denotes the expected value of a random variable under the condition that the agent follows the policy π.

From (2.2) an important recursive relation can be derived. After explicitly writing the expected reward and taking the first term out of the sum, the first time-step s_t can be written in terms of probabilities.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}^\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+k} \,\middle|\, s_t = s, \pi \right] \\ &= \mathbb{E}^\pi \left[r_t + \gamma \sum_{k=0}^{T} \gamma^k r_{t+1+k} \,\middle|\, s_t = s, \pi \right] \\ &= \sum_{a \in \mathcal{A}_s} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma \underbrace{\mathbb{E}^\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+1+k} \,\middle|\, s_{t+1} = s', \pi \right]}_{V^\pi(s')} \right] \\ &= \sum_{a \in \mathcal{A}_s} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma V^\pi(s') \right] \end{aligned} \tag{2.3}$$

In (2.3) the value function for the next step $V_\pi(s')$ can be inserted. This formula, called the *Bellman equation* [49], is able to look ahead to the next successor states by weighting the value function for the next step with the probabilities of the triple (a, r, s'). If (2.3) is rewritten in the form

$$\begin{aligned} V^\pi(s) &= \sum_{a \in \mathcal{A}_s} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma V^\pi(s') \right] \\ &= \sum_{a \in \mathcal{A}_s} \pi(a|s) \left[\sum_r r \cdot p(r|s,a) + \gamma \sum_{s'} p(s'|s,a) V^\pi(s') \right], \end{aligned} \tag{2.4}$$

it can be seen that the value function is the sum over the expected reward r for ending up in the next state s' plus the reward which is expected in the future.

The action-state value Similar to the value function V^π one can define the *state-action value*[1] $Q^\pi(s,a)$. It has the interpretation of the discounted future reward, expected from following the current policy π after taking a single freely choosable (and possibly suboptimal) action a_t from state s_t at time-step t. The action-value function is connected to the state-value function through

$$V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s,a)$$

and can be seen as a more informative form of V^π that measures the advantage of choosing action a while being in state s. It is also possible to again formulate a Bellman equation for $Q^\pi(s,a)$ which is given by

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}^\pi\left[\sum_{0=k}^{T} \gamma^k r_{t+k} \,\middle|\, s_t = s, a_t = a, \pi\right] \\ &= \sum_{s',r'} p(s',r'|s,a)\left[r' + \gamma V^\pi(s')\right] \\ &= \sum_{s',r'} p(s',r'|s,a)\left[r' + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s',a')\right]. \end{aligned} \tag{2.5}$$

2.3 Optimal value functions

The goal in reinforcement learning problems is to find a policy that leads to the highest possible accumulated reward. This policy is called the *optimal policy* $\pi^\star$. As seen in Section 2.2, there exists a relation between the used policy π and the corresponding state value V^π. For finite Markov decision processes it is possible to derive some useful statements[2] so that the optimal policy can (in principle) be calculated analytically.

[1]Alternative terms for the state-action value are *action-value* or *Q-value*.
[2]As (2.5) shows, is Q^π implicitly included in V^π. Thus, all statements that are true for V^π are also true for Q^π.

The optimal action-state value function If the target policy π is deterministic, one can replace $\sum\limits_{a'} \pi(a'|s')$ in (2.5) by a deterministic target policy $\eta : \mathcal{S} \to \mathcal{A}$ to avoid inner expectation:

$$Q^{\eta}(s) = \sum_{s',r'} p(s', r'|s, a) \left[r' + \gamma Q^{\eta}(s', \mu(s'))\right]$$

As the expectation of the next action is depending only on the environment, it is possible to learn Q^{η} off-policy by using state transitions that are generated from a different stochastic behaviour policy than π. A commonly used choice for η is the greedy policy

$$\eta(s) = \operatorname*{argmax}_{a \in \mathcal{A}(\mathcal{S})} Q(s, a), \tag{2.6}$$

that finally specifies the Bellman optimality equation

$$Q^{\star}(s) = \sum_{s',r'} p(s', r'|s, a) \left[r' + \gamma \max_{a' \in \mathcal{A}(\mathcal{S})} Q^{\star}(s', a')\right] \tag{2.7}$$

as presented in [50].

The optimal state value function As the state value can be computed by using the Bellman-Equation (2.3), the optimal state value can be computed through the same iterative principle with the help of the *Bellman optimality equation* [49].

The optimal state value represents the expected return for the best choice of actions. It leads to the relation

$$V^{\star}(s) = \max_{a \in \mathcal{A}(\mathcal{S})} Q^{\star}(s, a).$$

With the help of (2.5) this can be rewritten into the Bellman optimality equation

$$V^{\star}(s) = \quad \max_{a \in \mathcal{A}(\mathcal{S})} \sum_{s',r'} p(s', r'|s, a) \left[r' + \gamma V^{\star}(s')\right].$$

Finding the optimal policy The computation of V^{π} leads to a partial ordering of the policies π. In this ordering one policy π' is only equal or better than the policy π if their respective value functions are behaving in the same manner, i.e.

$$\pi' \geq \pi \;\Leftrightarrow\; V^{\pi'}(s) \geq V^{\pi}(s) \quad \forall s \in \mathcal{S}.$$

Although there can be more than one policy $\pi^\star$ that is optimal as there always exists a $\pi^\star$ which is equal or better than the actual optimal policy, all these policies share the same optimal state-value

$$V^\star(s) = \max_\pi V^\pi(s) \quad \forall s \in \mathcal{S}$$

and hence the same optimal action-value

$$Q^\star(s,a) = \max_\pi Q^\pi(s,a) \quad \forall s \in \mathcal{S},\ a \in \mathcal{A}(\mathcal{S}).$$

Consequently, if $V^\star(s)$ is known, an optimal policy is given by choosing the action a which induces the transition into the state s' with the highest possible state-value function $V^\star(s')$. If $Q^\star(s,a)$ is known, an optimal policy is given by choosing the action a according to the greedy policy (2.6).

2.4 Q-learning using linear function approximators

For finite Markov decision processes, the Bellmann optimality equation has a unique solution which is also policy independent and leads to a set of non-linear equations with one equation for each state s. If the Markov property is given for the system and all dynamics of the environment are known, the optimal value function can be computed. In order to compute the solution, one has to do a forward search which includes all possible events with their probabilities of occurrence plus their expected reward. However, the expected reward is often precluded by the lack of sufficient computational power for environments with lots of states. To overcome this difficulty, most of the reinforcement learning algorithms approximate the Bellman optimality equation which, nevertheless, leads to remarkable results.

One important breakthrough in reinforcement learning was the development of the *Q-learning* algorithm by Watkins in his dissertation nearly thirty years ago [51]. As the name Q-learning already emphasizes, it is an algorithm that adapts to approximate the action-state value in a fully incremental fashion. Therefore, at every time-step t, the agent uses the tuple (s_t, a_t, r_t, s_{t+1}) in order to update the Q-values *on-line*[3]. Additionally, Q-learning is an *off policy* method[4], directly approximating the optimal action-value function $Q^\star$ independently of the current active policy.

[3]If the learning step is expected only at a certain number of time-steps, for example at the end of a learning episode, the algorithm is called an *off-line* learning method.

[4]This is caused by the fact, that the greedy policy η, defined in (2.6), is used as the policy for selecting the next action within the Bellman optimality equation (2.7).

If the dimension of the state space is huge, which is the case in most complicated learning scenarios, a common approach is to approximate $Q(s,a)$ by using a linear function

$$Q(s,a,\boldsymbol{w}) = \boldsymbol{w}^\intercal \cdot \boldsymbol{\Phi}(s,a) \tag{2.8}$$

with a weight vector $\boldsymbol{w} \in \mathbb{R}^n$ and a function $\boldsymbol{\Phi}(s,a) \in \mathbb{R}^n$ of lower dimensional features that represents the state-action pair (s,a). The optimal state-action value can then be approximated by iteratively minimizing the mean-squared Bellman error

$$\mathcal{L}_t(\boldsymbol{w}_t) = \mathbb{E}\left[(r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \boldsymbol{w}_t) - Q(s_t, a_t; \boldsymbol{w}_t))^2\right]. \tag{2.9}$$

By differentiating the Bellman error with respect to the weights, the resulting gradient is given by

$$\begin{aligned}\nabla_{\boldsymbol{w}_t}\mathcal{L}_t(\boldsymbol{w}_t) \quad &= \quad \mathbb{E}\left[r_t + \gamma \cdot \max_{a\in\mathcal{A}(s_{t+1})} Q(s_{t+1}, a; \boldsymbol{w}_t) - Q(s_t, a_t; \boldsymbol{w}_t)\right] \cdot \\ &\quad \nabla_{\boldsymbol{w}_t} Q(s_t, a_t; \boldsymbol{w}_t). \end{aligned} \tag{2.10}$$

If the state is approximated using a linear function approximator as shown in (2.8), the derivative $\nabla_{\boldsymbol{w}_t} Q(s_t, a_t; \boldsymbol{w}_t)$ is given as

$$\nabla_{\boldsymbol{w}_t} Q(s_t, a_t; \boldsymbol{w}_t) = \boldsymbol{\Phi}(s_t, a_t). \tag{2.11}$$

By inserting (2.11) in (2.10), a general gradient descent update for the weights $\boldsymbol{w}$ at time step $t+1$ can be derived:

$$\boldsymbol{w}_t = \boldsymbol{w}_t + \alpha_t \left[r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \boldsymbol{w}_t) - Q(s_t, a_t; \boldsymbol{w}_t)\right] \cdot \boldsymbol{\Phi}(s_t, a_t)$$

The state-action value $Q(s,a)$ can now be updated iteratively, while the update size is determined by a (typically decreasing) learning rate $\alpha_t \in [0,1]$.

ϵ-greedy Q-learning: exploration vs. exploitation In order to ensure the collection of the maximal reward, the agent should follow the greedy policy (2.6) as it chooses the action at each time step which maximizes the current state-action value. While this deterministic policy is a good choice in the late stages of learning, it can undermine the learning process at the beginning as many potentially good states might not be visited due to the initialization of the Q-values. In order to nevertheless provide an overarching exploration that is potentially suboptimal according to the greedy policy η introduced in (2.6), a scalar exploration factor $\epsilon \in [0,1]$ is introduced. During learning, it forces the agent to select a random action with a probability ϵ.

Eligibility traces for Q-learning One problem with the presented version of Q-learning is that the Q-value is updated only based on the information of the current step. Thus, learning could be improved by also taking the information of previously visited states into account which can be achieved by employing *eligibility traces* [50]. They can be seen as an implementation of a short-term memory into the learning algorithm that fades away according to a decay-factor $\lambda \in \mathbb{R}$ within the range $\lambda \in [0,1]$. During learning, an eligibility trace $\boldsymbol{e}_t \in \mathbb{R}^n$ is generated by accumulating the decayed variation of the action-state value with respect to the weights from previous steps, until a random action is taken or the episode terminates. This variation is given by the derivative $\nabla_{\boldsymbol{w}} Q(s_t, a_t; \boldsymbol{w})$. For Q-Learning with linear function approximation it thus directly corresponds to the feature vector $\boldsymbol{\Phi}(s_t, a_t)$ according to (2.11).

A pseudocode for “linear greedy Q-learning with eligibility traces” can be found in Algorithm 4 in Appendix A.

2.4.1 Approximating the state-action space for a set of discrete actions

To approximate the state-action value $Q(\boldsymbol{s}, a)$, the used feature vector must be able to represent not only the state s, but a state-action pair (s, a). For a set of discrete actions $\mathcal{A}$, one option is to expand the utilized state representation $\phi(\boldsymbol{s})$ to

$$\boldsymbol{\Phi}(\mathrm{s}, a) = \left[\boldsymbol{\phi}^1(\mathrm{s}), \ldots, \boldsymbol{\phi}^a(\mathrm{s}), \ldots, \boldsymbol{\phi}^{|\mathcal{A}|}(\mathrm{s})\right]^{\mathsf{T}} \tag{2.12}$$

with $\dim(\boldsymbol{\Phi}(\mathrm{s}, a)) = |\mathcal{A}| \cdot \dim(\boldsymbol{\phi}(\mathrm{s}))$ and $|\mathcal{A}|$, the total number of actions in $\mathcal{A}$.

This approach, proposed by Lagoudakis and Parr [52], creates a sequence of feature vectors according to the number of possible actions $|\mathcal{A}|$.

After choosing action $a \in \mathcal{A}(\mathrm{s})$, the entries $\boldsymbol{\phi}^\alpha(\mathrm{s})$ of $\boldsymbol{\Phi}(\mathrm{s}, a)$ are created according to

$$\boldsymbol{\phi}^\alpha(\mathrm{s}) = \begin{cases} \boldsymbol{\phi}(\mathrm{s}) & \alpha = a \\ \mathbf{0} & \text{otherwise} \end{cases}, a \in \mathcal{A}$$

by flipping all feature vectors $\Phi^\alpha(\mathbf{s})$ to 0, except the one corresponding to action a.

Therefore, equation (2.12) for state s and action a, which is taken from a set of actions $a = 1, \dots, |\mathcal{A}|$ can be written as

$$\boldsymbol{\Phi}(\mathbf{s}, a) = [\underbrace{0, \dots, 0}_{\alpha=1}, \dots, 0, \underbrace{\boldsymbol{\phi}(\mathbf{s})}_{\alpha=a}, \dots, \underbrace{0, \dots, 0}_{\alpha=|\mathcal{A}|}]^\intercal .$$

2.5 Policy gradient methods

Instead of choosing the actions according to a learned state or action-state value, it is also possible to directly learn a stochastic action policy $\pi(a|s; \boldsymbol{w})$ with its own set of weights $\boldsymbol{w}$. In order to learn an efficient policy, an appropriate performance measure has to be designed, which has then to be maximized. This performance measure can be defined for episodic problems as the "true" state value for the current policy $\pi(a|s; \boldsymbol{w})$, specified by the weights $\boldsymbol{w}$, that is given within the starting state s_0 of the current episode

$$\mathcal{L}(\boldsymbol{w}) = V^{\pi_{\boldsymbol{w}}}(s_0).$$

Now, one important question is how to appropriately change the parameters of the approximated policy in a way that improves it. This is a problem that is not easily solved, as the effects of the policy on the environment are not only depending on the chosen action, but also on the environment's generally unknown state distribution. At this point, the *policy gradient theorem* [53] offers a way out of this dilemma even when general differentiable function approximators are used. It provides a version of policy iteration that guarantees to converge to a local optimal policy. For episodic scenarios the gradient of the performance measure $\mathcal{L}$ is then given as

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \quad \propto \quad \mathbb{E}^\pi \left[\sum_{a \in \mathcal{A}} Q^\pi(s_t, a) \nabla \pi(a|s_t; \boldsymbol{w}) \right] . \tag{2.13}$$

The gradient can now be used in order to formulate an update algorithm for the weights of the policy

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t \cdot \sum_{a \in \mathcal{A}} Q(s_t, a, \boldsymbol{w}_Q) \nabla \log(\pi(a_t|s_t; \boldsymbol{w}_t)),$$

where α_t defines the learning rate. Although this update method is efficient, the state-action value $Q(s_t, a, \boldsymbol{w}_Q)$ has to be learned using its own set of weights. As an additional drawback, all possible actions a are involved in the update process.

Actor-critic models One class of methods that is based on the *policy gradient theorem* and is used in many early attempts of reinforcement learning is called *actor-critic methods*. First presented in [54] and more extensively studied in [55], an actor-critic learner, illustrated in Figure 2.2, is directly updating its *actor* through learning a stochastic action policy with its own set of weights $\pi(s, a; \boldsymbol{w})$. In addition, the executed actions are rated by an additional *critic* like the estimated V- or Q-value that are learned independently from the policy[5].

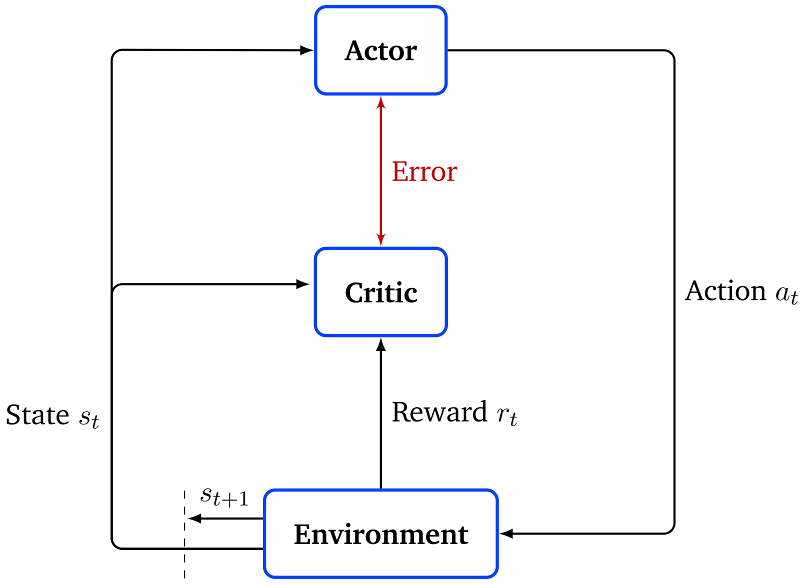


Fig. 2.2.: Schematic visualization of an actor-critic architecture as described in [25]

[5]Note that it can only be called an actor critic if the (action-)state value is used for bootstrapping, i.e. it is updated from the estimates of subsequent states.

2.5.1 The REINFORCE algorithm

While the last section has used the *policy gradient theorem* to derive a method that involves all possible actions for updating the weights of the to-be-learned stochastic action-policy $\pi(a|s;\boldsymbol{w})$, this section presents a variant that depends only on the action a_t that is taken at time-step t. Therefore, (2.13) is rewritten as

$$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}) \quad \propto \quad \mathbb{E}^{\pi}\left[\sum_{a\in\mathcal{A}} \pi(a|s_t;\boldsymbol{w})Q^{\pi}(s_t,a)\frac{\nabla\pi(a|s_t;\boldsymbol{w})}{\pi(a|s_t;\boldsymbol{w})}\right].$$

In a next step, the resulting expectation term $\sum_{a\in\mathcal{A}}\pi(a|s_t;\boldsymbol{w})$ is replaced by introducing the sampled expected action a_t, resulting in

$$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}) \quad \propto \quad \mathbb{E}^{\pi}\left[Q^{\pi}(s_t,a_t)\frac{\nabla\pi(a_t|s_t;\boldsymbol{w})}{\pi(a_t|s_t;\boldsymbol{w})}\right].$$

Now, the Bellman-equation for the Q-value (2.5) and (2.1) can be used to replace the state-action value $Q^{\pi}(s_t,a_t) = \mathbb{E}^{\pi}\left[R_t|s_t,a_t\right]$.

The resulting equation

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}) \quad &\propto \quad \mathbb{E}^{\pi}\left[Q^{\pi}(s_t,a_t)\frac{\nabla_{\boldsymbol{w}}\pi(a_t|s_t;\boldsymbol{w})}{\pi(a_t|s_t;\boldsymbol{w})}\right] \\ &= \quad \mathbb{E}^{\pi}\left[R_t\cdot\frac{\nabla_{\boldsymbol{w}}\pi(a_t|s_t;\boldsymbol{w})}{\pi(a_t|s_t;\boldsymbol{w})}\right]\end{aligned}$$

now depends only on the policy π, the action a_t that is actually taken in this time-step and the total reward R_t, leading to the *REINFORCE*[6] update rule [56, 57]

$$\boldsymbol{w}_{t+1} \quad = \quad \boldsymbol{w}_t + \alpha_t\cdot\left[R_t - b(\boldsymbol{s}_t;\boldsymbol{w}_b)\right]\frac{\nabla_{\boldsymbol{w}}\pi(a_t|s_t;\boldsymbol{w})}{\pi(a_t|s_t;\boldsymbol{w})}. \tag{2.14}$$

It increments the weights proportionally to the gained reward in the direction of the gradient of the policy when a_t is executed. The update is then divided by the policy in order to correct the oversampling of actions that are preferred by π. In order to reduce the variance, a suitable baseline b_t is additionally subtracted from the accumulated reward R_t. As the baseline tries to approximate the total accumulated reward, it is common to use the state value function, i.e. $b_t \approx V_t^{\pi}(\boldsymbol{s}_t)$. Doing this, $(R_t - b_t)$ can be interpreted as an estimate of the advantage function $A^{\pi}(\boldsymbol{s}_t,a_t) = Q^{\pi}(\boldsymbol{s}_t,a_t) - V^{\pi}(\boldsymbol{s}_t)$ which indicates how beneficial it is to take the

[6]The acronym REINFORCE is a representation for "**RE**ward **I**ncrement = **N**onnegative **F**actor × **O**ffset **R**einforcement × **C**haracteristic **E**ligibility".

action a_t for solving the given task with respect to the expected outcome. Using REINFORCE to modify the policy π, while also using an approximated state-value $\hat{V}(\boldsymbol{s}_t, \boldsymbol{w}')$ as the baseline b_t can thus be seen as an *actor-critic method*.

The REINFORCE update can not only be used for action policies but for any kind of continuous and differentiable function f. A general rule for updating the corresponding weights $\boldsymbol{w}$ of the network is given by

$$\Delta_{\boldsymbol{w}} = \alpha_t \cdot [R_t - b(\boldsymbol{s}_t; \boldsymbol{w}_b)]\, \zeta(\boldsymbol{s}_t; \boldsymbol{w}). \tag{2.15}$$

In (2.15), ζ is called the *characteristic eligibility*. It is defined as

$$\begin{aligned} \zeta(\boldsymbol{s}_t; \boldsymbol{w}) &= \frac{\nabla_{\boldsymbol{w}} f(\boldsymbol{s}_t; \boldsymbol{w})}{f(\boldsymbol{s}_t; \boldsymbol{w})} \\ &= \nabla_{\boldsymbol{w}} \log f(\boldsymbol{s}_t; \boldsymbol{w}) \end{aligned} \tag{2.16}$$

where $f(\boldsymbol{s}_t; \boldsymbol{w})$ determines the approximated kind of policy as a function of its input $\boldsymbol{s}_t$ and its weight parameters $\boldsymbol{w}$. For a differentiable parametrization of the action-policy $\pi(\boldsymbol{s}_t, \boldsymbol{w})$, ζ becomes again

$$\zeta_\pi = \nabla_{\boldsymbol{w}} \log \pi(\boldsymbol{s}_t, \boldsymbol{w}).$$

Additionally it is also possible to develop learning algorithms for an output that is determined via stochastic distributions which might depend on multiple parameters, like an adaptable Gaussian with the mean μ and the standard deviation σ. Instead of directly modeling the Gaussian by updating its corresponding weights $\boldsymbol{w}_\mu$ and $\boldsymbol{w}_\sigma$, μ and σ themselves can be treated as the adaptable parameters of the Gaussian $\mathcal{N}(x; \mu, \sigma)$.

Using this simplification, the characteristic eligibility for μ is given by

$$\zeta_\mu = \frac{\partial \log \mathcal{N}(x; \mu, \sigma)}{\partial \mu} = \frac{x - \mu}{\sigma^2}, \tag{2.17}$$

where x is the corresponding value, sampled from the Gaussian distribution $\mathcal{N}$. Analogously, the characteristic eligibility for σ is

$$\zeta_\sigma = \frac{\partial \log \mathcal{N}(x; \mu, \sigma)}{\partial \sigma} = \frac{(x - \mu)^2 - \sigma^2}{\sigma^3}. \tag{2.18}$$

2.6 Going deeper with neural networks

One drawback of using a linear function for approximating the (action-)state value is that the type of function representing the state of the environment has to be chosen very carefully to achieve a good learning performance. The difficulty of finding a reasonable representation increases with the complexity of the given state space. A way out of this dilemma is to replace the linear function with a non-linear one like a *neural network*[7]. Neural networks are built out of "artificial neurons" that are organized in *layers* as illustrated in Figure 2.3. They are modeled as continuous functions

$$f_j(\boldsymbol{x}) = \sum_i x_i w_{ji} + b_j$$

that compute the scalar product of the input vector $\boldsymbol{x}$ with their corresponding row $\boldsymbol{w}_j$ of the layer's adaptable weight matrix which is then added to a small fixed scalar-valued bias b_j. By stacking multiple layers together, arbitrary functions can be approximated.

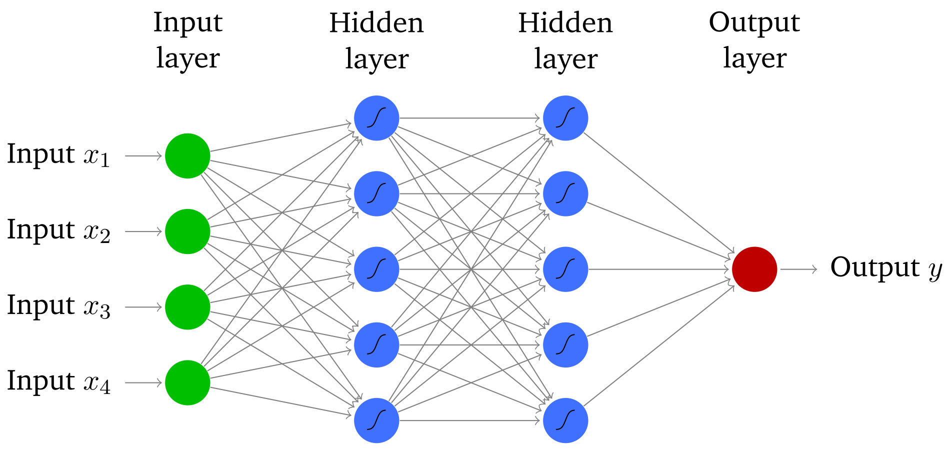


Fig. 2.3.: A simple illustration of a neural network with a four dimensional input vector $\boldsymbol{x}$, two hidden layers and a scalar output y

Activation functions The non-linearity is integrated via the additional processing of each layer through an *activation function* $\sigma(x)$, i.e. $\sigma(f(x))$. While in the early days of neural networks, S-shaped activation functions like the *sigmoid unit*

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.19}$$

[7]For a comprehensive introduction to "neural networks" and "deep learning" see for example [58].

or the *tanh*

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{2.20}$$

were the most popular ones, rectified non-linearities [59] like the *rectified linear unit* (ReLU) [60]

$$\sigma(x) = \max(0, x) \tag{2.21}$$

are nowadays present in nearly all neural network architectures (see Figure 2.4).

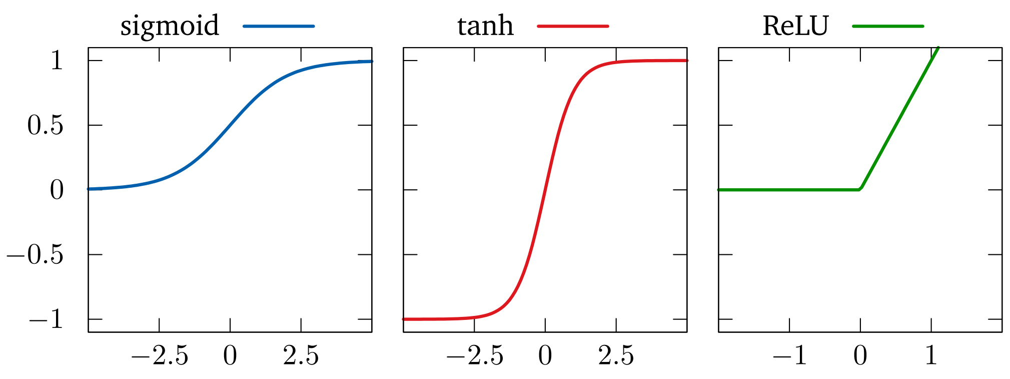


Fig. 2.4.: Illustration of the *sigmoid*, *tanh* and *ReLU* activation function.

Each neuron j of the l^{th} layer can thus be computed via

$$f_j^l(f^{l-1}) = \sigma\left(\sum_i f_i^{l-1} w_{ji}^l + b_j^l\right),$$

while f_i^{l-1} specifies the i^{th} neuron of the previous layer.

Adjusting the weights Neural networks create their output-vector $\boldsymbol{y}$ by *propagating* the information of the input vector $\boldsymbol{x}$ *forward* through the layers. To adjust the weights $\boldsymbol{w}$, the input data $\boldsymbol{x}$ and the corresponding desired network output in form of a target vector $\boldsymbol{y}_T$ are used to compute a scalar-valued error function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}_T; \boldsymbol{w})$. Optimization algorithms like *stochastic gradient descend* (with *Nesterov momentum* [61–63]) or extended algorithms with adaptive learning rates such as *AdaGrad* [64], AdaDelta [65], *RMSProp* [66] and *Adam* [67] are then utilized to perform the weight update. These methods are first generating *mini-batches* $(\boldsymbol{x}^{(b)}, \boldsymbol{y}_T^{(b)}) \in B$ by randomly sampling from the available training-data. In order to update the weights $\boldsymbol{w}$, the information of the gradients $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{x}^{(b)}, \boldsymbol{y}_T^{(b)}; \boldsymbol{w})$ are exploited which can be computed via *back-propagation* [68].

Deep neural networks While a very simple neural network[8] can be built out of an *input layer*, one *hidden layer* and one *output layer*, many more complex configurations are possible. Although it is in general possible to approximate any kind of function to an arbitrary accuracy with *shallow neural networks* using just one sufficiently large layer with sigmoid units as non-linear activation functions [70], it is much more efficient to represent complex functions by stacking multiple small layers in order to create a hierarchical composition of low-level abstractions. This leads to the concept of *deep neural networks* [71, 72], describing neural networks that are built out of more than 2 hidden layers. Additionally to the enormous increase of available computation power and size of available datasets, newly invented techniques like *dropout* [73], *batch normalization* [74] or the ReLU, combined with concepts like *convolutional neural networks* [75] have led to the undeniable success and lasting popularity of deep neural networks as a universal tool for solving various complex machine learning problems.

Recurrent neural networks Introduced in [68], *recurrent neural networks* (RNN) are specialized in the processing of sequential data like time-series or sentences. Therefore, simple RNNs are exploiting one hidden state $\boldsymbol{h}^{(t)}$ for storing and combining the relevant features of the past data of the current sequence by reusing it. The recurrent equation for computing $\boldsymbol{h}^{(t)}$ is therefore given by

$$\boldsymbol{h}^{(t)} = g\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(\boldsymbol{t})}; \boldsymbol{w}\right),$$

with some mapping function g and the input data of the sequence at position t given as $\boldsymbol{x}^{(\boldsymbol{t})}$. As simple RNNs have problems dealing with long sequences and storing long-term dependencies, more refined recurrent architectures were developed. One of them is the *long short-term memory* [76], proposing a refined controlling of the inner loop of the hidden state $\boldsymbol{h}^{(t)}$ by introducing additional hidden units that are gating the flow of information. As the size and computation time of the LSTM increased significantly compared to a simple RNN with one hidden state, *gated recurrent units* (GRU) [77, 78] were designed. GRUs change or omit pieces of the LSTM architecture in order to speed up the computation time while trying to hold the performance. Other recurrent approaches are given by *echo state networks* [79, 80] or *liquid state machines* [81] that use fixed weights for mapping from $\boldsymbol{h}^{(t-1)}$ to $\boldsymbol{h}^{(t)}$ and $\boldsymbol{x}^{(t)}$ to $\boldsymbol{h}^{(t)}$ and learn only the weights for the output.

[8]The simplest neural network would be one *without* a hidden layer as Rosenblatt's single layer perceptron [69].

2.6.1 Deep Q-learning

One popular reinforcement learning algorithm that relies on deep neural networks as a non-linear function approximator for learning the state-action value $Q(s, a)$ is the *deep Q-network* (DQN) [7, 33]. The high abstraction capacity of the deep neural network enables to use, for example the raw visual information of the learning domain as the sensory input. While the weights of the network can be updated by employing the gradient (2.10) of the mean-squared Bellman error (2.9) as for "Q-learning with linear function approximation", some additional general concepts have to be presented in order to achieve satisfactory learning results.

Experience replay To reduce unwanted temporal correlations of the gathered training data and stabilize the algorithm, the *experience replay buffer* is introduced. Following the very first idea of a replay buffer that was proposed by Lin [82], the experienced transitions (s_t, a_t, r_t, s_{t+1}) are stored. At each step, experience is sampled from the buffer which is then utilized to train the network weights via mini-batch gradient descent. Following the crucial importance of this concept for designing a functioning learning algorithm, different variants were developed over time such as "prioritize experience replay" [83, 84] or "experience replay with adaptable memory size" [85].

Target network The second important concept minimizes divergence and oscillations within the updates by generating the target Q-values using an older set of weights $\boldsymbol{w}^-$. These weights are updated only within a predefined interval and are thus delaying the adaption of the Q-values that are used for learning. The gradient of the mean-squared Bellman error is then given by

$$\begin{aligned}\nabla_{\boldsymbol{w}_t}\mathcal{L}_t(\boldsymbol{w}_t) \quad = \quad & \mathbb{E}\left[r_t + \gamma \cdot \max_{a \in \mathcal{A}(s_{t+1})} Q(s_{t+1}, a; \boldsymbol{w}_t^-) - Q(s_t, a_t; \boldsymbol{w}_t)\right] \cdot \\ & \nabla_{\boldsymbol{w}_t} Q(s_t, a_t; \boldsymbol{w}_t).\end{aligned}$$

A pseudocode for training the DQN can be found in Algorithm 5 in Appendix A.

2.6.2 Asynchronous models

It is quite safe to assume that the presentation of the deep Q-network by Mnih et al. [7, 33] was a milestone within the field of deep learning. The demonstration of a reinforcement leaner that used a deep neural network as a non-linear function approximator in order to learn to play a battery of ATARI arcade games while always relying on the same hyperparameters and only the raw visual image of the game scene as an input has inspired many subsequent contributions within the field. Despite the impressive results, the approach highlighted common and to this day unsolved problems. One problem is that millions of training-steps are required in order to achieve the presented results, while the actual computations are heavily relying on GPUs in order to make the computation time tolerable. In the following years, the framework of the DQN was continuously bolstered up using various ideas like *double Q-learning* [86, 87] or the *dueling architecture* [88]. Many of these ideas were then eventually combined into one efficient *rainbow* architecture [89]. Another approach that was studied intensively was to exploit learning models that are using parallelized architectures. This method relies on multiple distributed learners in order to solve a single-agent problem [90, 91].

Gorila One parallelized architecture, named *Gorila* (General Reinforcement Learning Architecture), was presented by Nair et al. [91]. It employs multiple DQN-processes that all have agents interacting within their own copy of the environment while sharing their gradient with a parameter server. The server updates a master network and then distributes its new weights to the processes. By using 100 processes and 30 parameter servers, the Gorila architecture outperformed the original approach on a significant level.

The asynchronous advantage actor-critic model While the Gorila model needs a massive amount of computation power, Mnih et al. presented an architecture that is also able to outperform the vanilla *Deep Q-Network* while using only a single computing machine with a standard multi-core CPU [92]. They introduced a network architecture that does not exploit the off-policy Q-Learning algorithm but relies on on-policy actor-critic methods. The key idea of this *asynchronous advantage actor-critic model* (A3C) is again to use multiple, independent (actor-critic) workers (see Figure 2.5) that all are exploring their own copy of the domain in parallel. For training, they independently compute their own gradient, which is then used to asynchronously update a master-network, for example by applying the REINFORCE update rule (2.14). This asynchronous update disentangles the gathered training data and enables a stable learning process. The workers then update themselves after a fixed number of episodes, receiving a copy of the master-network's weights.

A pseudocode for training the A3C can be found in Algorithm 6 in Appendix A.

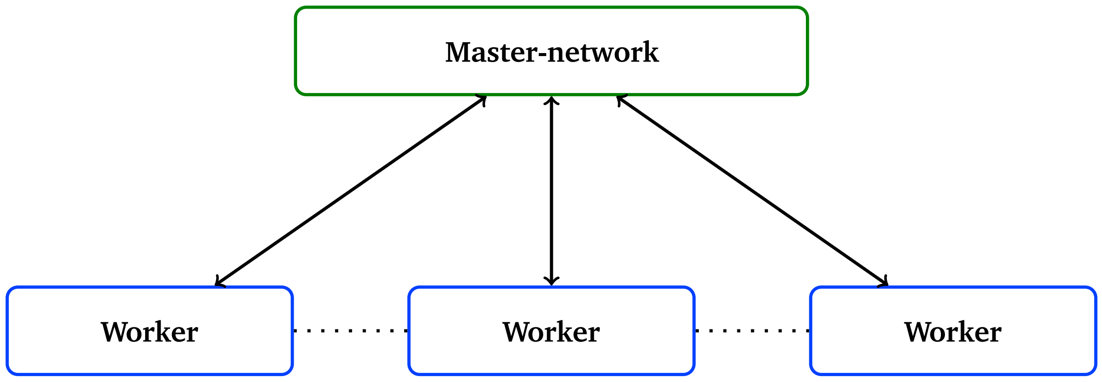


Fig. 2.5.: The basic structure of an asynchronous model. The key idea of this model is to use multiple independent workers that are all exploring their own copy of the domain in parallel. For training, they all compute their own gradient which is then used to asynchronously update a master-network.

2.7 Summary

In this section, some basic concepts of reinforcement learning were presented, together with the algorithms that are employed throughout this thesis. After giving an introduction into the general methodology of reinforcement learning, the Bellman (optimality) equation was shown for the state and action-state value. Using this information, the "Q-Learning" algorithm was derived for the case when the Q-value is approximated using a linear function for representing the sensory input which is received by the agent. As it is not only possible to create an efficient learning algorithm via learning the state or state-action value, the "policy gradient theorem" and the REINFORCE algorithm were put forward. They can be used to develop an iterative algorithm for directly learning the agents policy.

The approaches were then extended to the field of non-linear function approximators. Therefore, artificial neural networks were introduced, together with necessary methods that allow their efficient use as non-linear function approximators.

The next step This chapter has presented the necessary tools for understanding the designed learning models that are utilized throughout this thesis. The next chapter is now introducing the concept of *scaffolding in machine learning* that is exploited as the inspirational corner stone of all applications that will be described in the next part of this work.

3 The guiding principle of scaffolding

Human learning has always been an inspiration for machine learning. Only recently, machine learning has been able to reach (and sometimes surpass) learned human skills [7, 31]. However, the employed methods are in stark contrast to how humans learn and current machine learning can so far in no way compete with the flexibility of human learning. This has motivated many researchers to study how machine learning might benefit from a better understanding of how humans learn. One particularly interesting phenomenon related to human learning is their ability to teach. While the learning process of specific skills like playing an instrument or learning a foreign language might be very difficult to figure out by oneself, a skillful teacher is able to guide the learner and in doing so enables or improves the learning of a particular competence. In this regard, the modality of teaching can significantly influence the quality and speed of a novice's learning [93]. A skillful teacher is able to provide adaptable guidance in learning tasks that would usually be beyond the learner's own capabilities or to speed up and refine the learning of already partially mastered problems. This observation opened up a new field of research that can be dated back to the ancient Greek. At this time, philosophers like Aristotle and Plato conducted research in the field of education and for that matter the nature of teaching and learning. Later in the 1860s, "educational psychology" was founded by Johann Friedrich Herbart to study the art of teaching.

The art of teaching machines Studying the "art of teaching humans" by educational psychologists had a striving impact on modern human culture. Their inventions are shaping the educational system of schools and universities, improving the transfer of knowledge to the next generation in order to bring humanity forward. Over the past years, artificial intelligence has been getting more and more involved in human life, making it easier by knowing our daily routines, taste in music or enabling interaction with machines via voice or gestures. This and much more is only possible through learning. As a consequence, it is no longer only essential to explore how humans can be educated efficiently, but it is also important to study the "art of teaching machines".

The root of inspiration for many approaches within the fields of machine learning and artificial intelligence can be found in areas of research that are related to neurosciences[1] such as biology or psychology [5]. A particular example is given by the impact of the psychological *Gestalt theory* [94, 95] on the field of computer vision. The resulting laws of visual reconstruction that are explaining various perceptual phenomena were translated to the field of artificial intelligence in order to bring forward the research on perceptive systems. It did not only lead to efficient algorithms. Recent results also suggest that neural networks that are trained with natural images indeed exhibit the "law of closure" Gestalt phenomenon [96]. Other research has shown that it is possible to expose hidden computational properties of deep neural networks by exploiting tools from cognitive psychology. [97]

Organizing learning on a meta-level The above examples have shown that especially psychology has a huge potential of providing concepts that are able to be ported and to advance the field of artificial intelligence. Another concept from psychology that is not only intensively studied for humans but also for artificial agents is called *meta-learning* [98–100]. It circles around one central question:

> How can *meta-knowledge*, i.e. the knowledge about learning, be efficiently exploited in order to improve the performance of the learning process of humans?

There are many different views regarding the exact meaning of the term meta-learning in the field of machine learning [101]. Consequently, it is interpreted in different ways in literature and therefore led to various different approaches. Mostly, however, it is connected with research where artificial agents are *learning to learn* by adapting their learning processes based on acquired meta-knowledge.

Instead of imposing the task of meta-learning on the artificial agent, this work proposes to exploit the collected meta-knowledge about machine learning and embed it into **generic concepts in machine learning that are organizing learning on a meta-level** as it is done for human learners by using concepts from educational psychology. In particular, this thesis aims to analyse the potential of one popular theory developed by educational psychologists. It systematizes a number of empirical observations that are showing evidence of positively influencing the learning process when facing novel or challenging tasks. A central role is played by the concept of providing a temporary support that is tailored to the individual needs of the

[1]The term neuroscience covers all fields of study that investigate the neural processes within the brains of humans and animals. In order to achieve that goal, the knowledge and methods from various fields like biology, psychology, physics, etc. are combined.

learner. The idea of diagnosing available capabilities of the learner mixed with other concepts as simplification, demonstration and structuring are building the basis for the theory called *scaffolding* [17–20].

Outline Many of the theory's core ideas— which are explained in detail in the following sections — can already be related to approaches that were successfully integrated in the learning process of artificial agents (see the following Section 3.4). This makes scaffolding a promising candidate for not only shaping human learning but also the learning of machines in a beneficial way. The goal of this chapter is therefore to **propose scaffolding as a general guiding principle that exploits available meta-knowledge from the field of machine learning for developing approaches that are able to improve and accelerate the learning process of artificial agents**.

To reach this goal, the possibilities and limits when transitioning from the "art of teaching humans" to the "art of teaching machines" have to be analyzed. In order to study this process for the theory of scaffolding, a basic understanding of its key characteristics is essential. The next Section 3.1 therefore covers a comprehensive summary of the concept in educational psychology. As an intermediate step, the employment of tools for scaffolding human learners is discussed (see Section 3.2), finally opening up the path for elucidating the improvement of machine learning methods by artificial scaffolding. On the one hand, it is demonstrated that many approaches are already partly utilizing features that are connected to scaffolding. On the other hand, the inevitable limitations that have to be faced when trying to transfer the concept to machine learning are highlighted. The essential concepts and propositions are then taken as the point of departure for compiling a refined definition of **scaffolding as a principle for facilitating the learning of an artificial intelligence on a meta-level** that uses the emerging key-aspects as a computational skeleton for a possible research agenda. In the end, the invented principle is used for proposing **four new scaffolding approaches** that are then presented, analysed and discussed throughout the remaining part of this thesis.

Definitions This chapter deals with human learners, as well as with artificial agents that are capable of acquiring new skills. In the following, the human student that performs the process is called the *learner or novice*. The artificial learner is called the *agent*. Furthermore, when the verb *scaffolding* is used in this work, the object which is scaffolded is always *the learning process*, i.e. the learning itself. Scaffolding the learner thus means to apply scaffolding techniques to its learning process.

3.1 The concept of scaffolding in educational psychology

During the time when educational psychology was growing to a highly studied field of research, two problems arose. The first one addressed the question how to correctly assess the abilities of children. The second issue dispatches the query for a way to evaluate the efficacy of developed instructional practices. As one solution to cope with these two issues, Vygotsky introduced the *"zone of proximal development"* [102, 103], illustrated in Figure 3.1. It describes the distance between the developmental level of an unguided learner for solving a specific problem and the possible level that can be reached through the guidance of an expert or the collaboration with more versed peers. In addition, it can also be seen as the ability of a learner to unconsciously recognize the value of encountered assisting props before consciously evaluating their significance for the task [103].

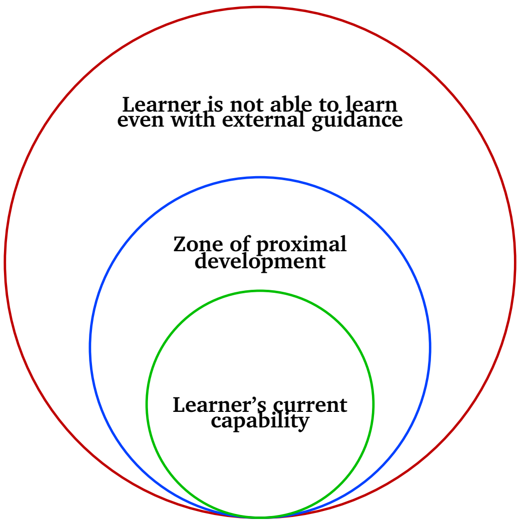

Learner is not able to learn even with external guidance
Zone of proximal development
Learner's current capability

Fig. 3.1.: Illustration of the *zone of proximal development*. The zone is placed in-between the space of the learner's current skills and the zone of skills that the learner is not even able to learn through an external teacher.

An important concept that grew out of Vygotsky's work was *scaffolding*[2] [17–20], illustrated in Figure 3.2. The support provided by scaffolding reduces the learner's cognitive load, which allows performing parts of tasks that he or she usually would not be capable of [106].

[2]The term *scaffolding* was defined by Wood et al. [104]. Later Bruner claimed that the origin of this concept lies within the "zone of proximal development" [105].

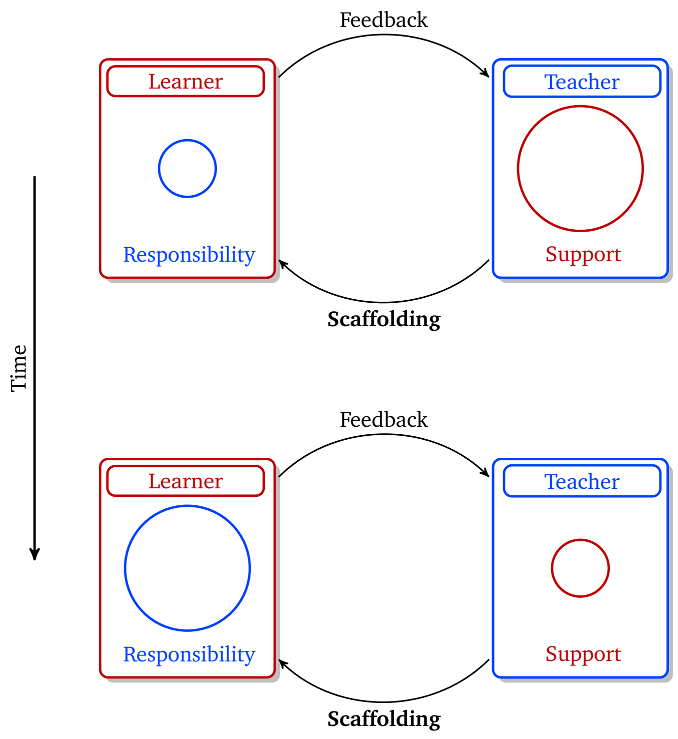


Fig. 3.2.: A simple illustration of the concept of scaffolding. Over time, the teacher's support is fading, while the responsibility of the task is also gradually transferred to the learner.

3.1.1 Learning how to ride a bicycle: an example for scaffolded learning

Nowadays, the metaphor of scaffolding is often used in a very broad sense [107], for example as a synonym of support [108], and is thus removed from its original theoretical context [109, 110]. The original concept of scaffolding consists of a set of specific, but not yet standardized characteristics. The key characteristics of scaffolding are stated in [19] as "contingency", "fading" and the "transfer of responsibility". The key aspect "contingency" refers to responsive and adjusted support that is tailored to the current capabilities of the learner in order to reduce the amount of frustration he or she receives during the learning process [19, 111, 112]. As this term comprises many different techniques, it is possible to fan out the feature of contingency to three sub-features that are given as "modelling and demonstration", "simplification" and "ongoing diagnosis and assessment". Hence, it is possible to associate scaffolding with five common features [20] that are summarized in Figure 3.3. By scaffolding the learning process of riding a bicycle as an example, the general concept of scaffolding and its five key aspects are now discussed in detail.

The five key aspects of scaffolding:

- Recruiting and maintaining the learner's attention towards a goal
- Simplifying the task
- Ongoing diagnosis and assessment
- Modelling and demonstration
- Fading support and eventual transfer of responsibility

Fig. 3.3.: A listing of **the five key aspects of scaffolding**.

Recruiting and maintaining the learner's attention towards a goal When a new problem is faced by a learner, an essential element of productive learning is his or her motivation and willingness to learn. For initializing such a fruitful learning process, the teacher has to recruit and then maintain the learner's attention towards a goal. A shared understanding of the learning goal has to be developed in order to raise the learner's interest and motivation to learn. The goal should be clearly formulated, while the benefits of reaching it should be explained in a way that motivates the learner to achieve them. With respect to the given example of riding a bicycle, the teacher might explain the new possibilities that come with the ability to ride a bike. This could be the option to travel faster, the sheer fun of riding a bike or also the possibility to go on bicycle tours with friends. During the whole learning process, the teacher would then occasionally remind the learner of the learning goal and why it is worthwhile to achieve it in order to maintain the direction of learning and control possible upcoming frustration during the process.

Simplifying the task For many problems, it might not be beneficial to start with the corresponding learning process immediately. According to the theory of scaffolding, it is better to learn a simpler version of the task first. This can for example either be done by reducing the main problem's degrees of freedom or by splitting it into suitable sub-problems or. Simplifying the task can also be combined with directing the attention of the learner (introduced in the previous paragraph) by focusing the learner's attention on relevant features or characteristics of the learning task [107]. This can help to reduce the degrees of freedom of a complex problem. When riding a bicycle, the novice has to maintain his or her balance while pedalling. By temporarily providing training wheels, the complexity of this process can be reduced. In addition, the learner's attention should be focused on maintaining his or her balance while pedaling at a steady pace. The learner should drive short and easy routes with the bicycle. Ideally, it should be flat with few curves.

Modelling and demonstration The teacher demonstrates the aspired skill or particular sub-skills and explains useful hints for enabling imitation. Possible ways to integrate this into the learning process would be the demonstration of riding a bicycle and the theoretical explanation how to do it, together with some useful tips regarding speed control and manoeuvrability. The teacher might also ride alongside or behind the novice to provide him with permanent verbal support. In addition, one could support the novice by exemplifying the functionality from a simplified technical perspective. In this way, the novice might be able to build an internal model of the vehicle that leads to a better understanding how to ride it.

Ongoing diagnosis and assessment During learning, the teacher has to permanently assess the learner's current level of skill. By diagnosing the learner's competence at the present time, the teacher should be able to align the difficulty of the learning sessions to the learner's current capabilities. In a next step of learning how to ride a bicycle, the routes might thus include more curves or become uneven in certain parts. In this way upcoming difficulties can be analysed and tackled, like problems to control the bicycle in certain situations.

Fading support and eventual transfer of responsibility If the skill level of the novice has reached a certain threshold, the simplification can be gradually reduced, leading to a fading of the provided support and the transfer of responsibility to the learner. In the given example, this process could be initialized by omitting the training wheels on an easy route and providing only temporary support by manually stabilizing the bike by hand when necessary. The stabilizing phase is then more and more reduced until it is completely omitted and the learner is able to maintain the balance on his or her own. The teacher might then start to pedal in front of the novice and thus reduce the amount of direct support. If the learner is able to fully control the bicycle without help, the difficulty of the courses can again be raised under ongoing diagnose and assessment of the learner's skills until he or she has fully mastered to control the vehicle in all situations without help.

Conclusion Scaffolding is an *interactive* process. At every point of learning, the teacher has to analyse the learners current level of competence or level of motivation and then determine if the current way of learning is still productive. According to these factors, the teacher has to provide tailored support and adjust the learners attention so that he or she again pursues the goal. The teacher also has to calculate the best moment to initialize the process of fading out the given support and to transfer more and more responsibility to the novice.

3.2 Teaching devices: employing computer-based tools for scaffolding the learning process of humans

When “educational psychology” was founded in the 1860’s, the world’s level of technology was far away from the one we have today. Neither was electric power publicly available, nor was the telephone patented. Things that were described in science fiction novels at that time are now everyday items. Personal computers — available in many different forms like notebooks, tablets or smartphones — combined with the internet are available as a platform for social communication and nearly any kind of information that is publicly available. This has not only changed our way of living but also of learning. When in the 1980’s the design of computers started to become more user friendly through the invention of a graphical user interface and the mouse, many researchers recognized the possible potential of computers as “teaching tools” that are used alongside or — at least temporarily — replace the human teacher. However, when switching from the human teacher to a computer program during parts or even the whole learning process, new challenges like human-computer interaction had to be faced. The software also had to be designed in a way that was learner-centered, but also user friendly [113]. One kind of tools that emerged from this educational innovation is given by “computer-based instructional programs” that might be additionally augmented with “electronic support tools” such as hyperlinks, text-boxes or calculators when dealing with mathematics [114, 115]. Another line is called *social computing* which extends the collaborative learning activities from learner-learner interaction with the option of leaner-machine interaction, improving the learning process and also fostering the collaboration between learners [116].

Even in the early days of computer-based learning tools, the idea to incorporate scaffolding into the software was presented [117]. Simultaneously, it was realized that the attempt to combine scaffolding with this new technologies was not straight forward. It was a challenge to combine the conceptual and methodological design of the traditional theory [118]. Popular issues that were discussed within literature are related to the implementation of fading [117], but also the possible violation of associated key aspects such as ongoing diagnosis or support aligned to the learners current capabilities [108]. As a consequence, an evolved notion of scaffolding was needed. The refined concept had to cope with these limitations when using different kinds of assistive technologies, but had nevertheless to keep the root idea of the theory. For artificial teachers like an instructional program, different design questions had to be addressed than for an open-ended collaborative or a web-based learning environment. As a consequence, not the whole concept was implemented but different scaffolding techniques that laid the focus on different key aspects of the traditional theory [119]. When, for example developing approaches that should foster the adaptive support of the learner, the implementation of the traditional scaffolding is now constrained to specific key aspects like on-going diagnosis, combined with some degree of fading [120, 121]. Other analyzed scaffolds are placing more emphasis on self-diagnosis without individualized support or fading [122, 123]. Despite these constrains, scaffolding nevertheless plays an important role when learning complex topics with the help of computer-based learning environments. Studies have shown that the absence of scaffolding in these kinds of learning environments has a strong negative impact on the learning performance [124–126].

3.2.1 Scaffolding the learning of a foreign language with the help of a computer-based tool

In order to get a better understanding of the difference between the scaffolding through a human teacher and a tool, a representative example is analysed within this section. The chosen class of teaching devices is given by computer-based tools like software that should support the learner while learning a foreign language. When utilizing a computer program as a teaching device, a shared understanding of the problem, focussing the learner's attention on critical features of the task and also assistance for completing the task can be easily implemented. The program might be able to motivate the learner through the presentation of intermediate goals like having short conversations or being able to read simple texts. During the different lessons, the appearance of hints can assist the learner to understand and solve the given task like for example a cloze test.

The general limitation of teaching devices At the same time, tools have problems to adjust the difficulty of the current learning session to the individual learner. The different lessons that are provided by the language program are usually built on specific vocabulary and grammatical understanding. When the learner wants to tackle this lesson, he or she thus has to fulfill exactly these requirements. In consequence, the difficulty of the lessons is not adapted to the skills of each learner on a personal scale, but just demands a specific knowledge of the language which is the same for every learner. In addition, tools are — in most cases — designed to support one specific aspect of the learning process. Most computer programs for learning a language are, for example specialized on a specific aspect, such as learning the vocabulary, the grammar or improving free writing through word processing. As a possible solution, multiple scaffolding tools could be used for providing different kinds of help on different levels of difficulty. This, however, leads to the question how the learner knows at which point he or she needs the specific non-adapting scaffold. A possible answer is to once again employ a human teacher for monitoring the learner's progress and to decide which kinds of scaffolding tools should be provided and how they have to be used [127].

The problems of fading support Although there exists very little empirical research studying the effects of "fading" in educational psychology [20], it is seen as one of the most salient and defining features of correct scaffolding [107]. Fading describes a temporary support which is gradually reduced within the learning process. It is argued that leaving out fading within the scaffolding process would reduce it to a process of "distributed intelligence" or "distributed cognition", as parts of the task are just off-loaded onto tools or persons [107, 128]. Especially in the field of technology-based scaffolding, the implementation of fading is a challenging task. On the one hand, machines do not yet have the ability to efficiently analyse the learner's learning progress at that point that is necessary to determine the starting point for the fading process. On the other hand, computer programs that are developed in order to support the learning process are intentionally designed as tools that provide constant, rather than a temporary support. If a tool is used for scaffolding, its nature might prevent the integration of fading. When, for example people use software for learning vocabulary or word processing, it is not in the interest of the learner to stop using it but to keep relying on the tool's assistance for maintaining or improving the learned word pool or the writing skills. Thus, it might be only possible to raise the challenge for the learner up to a certain point. Having the software stopping its support and transferring the whole responsibility of learning to the novice might also not be a feature that was projected by its developer.

3.3 Scaffolding artificial agents by organizing learning on a meta-level

The concepts that were introduced in the last sections are all circling around the central question how *meta-learning*, i.e the process of exploiting knowledge for understanding and improving the performance of learning, can be exploited to facilitate the performance of the learning process of humans. The collective response was given by proposing scaffolding: A theory that synergizes a battery of techniques — emerged from meta-knowledge which was gathered in many studies — that together provide a net of support that greatly enhances the learner's learning performance.

The transition from scaffolding humans to scaffolding machines In this section, this knowledge is now employed to organize not human learning but the learning of machines on a meta-level by establishing ties between the theory of scaffolding and state of the art methods for refining the learning processing of artificial agents. On that account, this thesis proposes to apply scaffolding not to a human learner but an artificial one by utilizing a refined version of the original theory as a principle for organizing learning. Again, the theory's five key-features are seen as the fundamental pillars. The difference is, however, that they now guide the learning process of an artificial agent instead of a human. This leads to the imperative of taking its five key aspects that were developed for humans and to reinterpret them for machine learners. One approach for transferring a concept from the human realm to the realm of machines is to extract the essential key ideas and techniques in order to find analogues for machines. The result should be as close as possible to the original ideas. As artificial learners are in many aspects different from human learners, this is of course not always possible. In this case, one has to search for alternative approaches that are different but share as many aspects as possible with the original one. This can, however, lead to cases where the transferred concept might have an alternative meaning and purpose.

Outline In summary, the proposed way enables the reinterpretation of scaffolding humans to a concept with the potential of enhancing the learning of machines. One drawback is, however, that one has to be careful that the required altering of specific aspects does not diverge the result too much from the original ideas. In the worst case the resulting concept might be too different to still be called scaffolding. While the last sections have introduced the concept of scaffolding the learning process of human learners not only by a human teacher but also through tools like computer programs, this section therefore now addresses the scaffolding of artificial learners. Although it is an interesting topic to use a human teacher for scaffolding the learning process — leading to the research field of social guided learning as it is, for example studied in [129–133] — this work is solely focusing on "tools" for scaffolding machines, i.e. methods or modules that can be applied without the need of a human instructor. Following this guideline, many modern approaches in the field of machine learning can be related to scaffolding. Most of them are, like the teaching devices, not implementing the full concept, but are designed based on ideas that are all encapsulated aspects of the theory. As mentioned earlier, there are also some inevitable limitations that are leading to the need to adapt the full definition of the concept. In the following text, a selection of approaches is presented that can be related to at least one of the key aspects (see again Figure 3.3) of scaffolding that were listed in Section 3.1.

3.3.1 Recruiting and maintaining the learner's attention

While one of the key aspects of scaffolding is stated as "recruiting and maintaining the learner's attention towards a goal", it might not be a reasonable quality when augmenting the learning of machines. Human learners need the motivation and the willingness to learn in order to achieve fruitful results. They are also likely to wander off from the subject and need to be pushed again into the right direction. Artificial agents neither need to be recruited nor motivated to learn to solve a problem. It is also not necessary to direct the agent's attention to a goal, as it is — at least in reinforcement learning — completely defined by the reward signal. Thus, immediately at the beginning of formulating a redefinition of the concept of scaffolding via its key characteristics, a problem arises as it is not possible to directly port the given aspect.

Adjusting key aspect's focus In Section 3.1.1 a second way of exploiting the learner's attention for augmenting the learning process was mentioned. Instead of or in addition to directing the learner's focus towards a goal, his or her awareness is guided to important features of a (sub-) task in order to simplify it by, for example reducing its degrees of freedom. In contrast to the aforementioned way of using attention, guiding the agent's attention towards the task-relevant information of the environment in order to speed up the learning is also an important aspect in machine learning. A way to achieve this goal is for example to refine the agent's perception towards better control and awareness of its surroundings. Based on these statements, an adjusted key aspect that lays the focus on supporting the agent's attention can be proposed: **recruiting and maintaining the agent's attention through guided perception**.

The perception-action cycle Perception in humans and animals is claimed to be an exploratory process [134–136]. Both kinds of biological agents are directing their interactions with the environment in a way that tries to reveal as much sensory information as possible, while moving the system closer to its current goal state. This fundamental behaviour of organisms — linking the strive for an efficient "flow of sensory information" with the simultaneous exploitation of this information for taking actions that narrow the distance of the system to the goal state — is called *perception-action cycle* [137, 138] (see Figure 3.4).

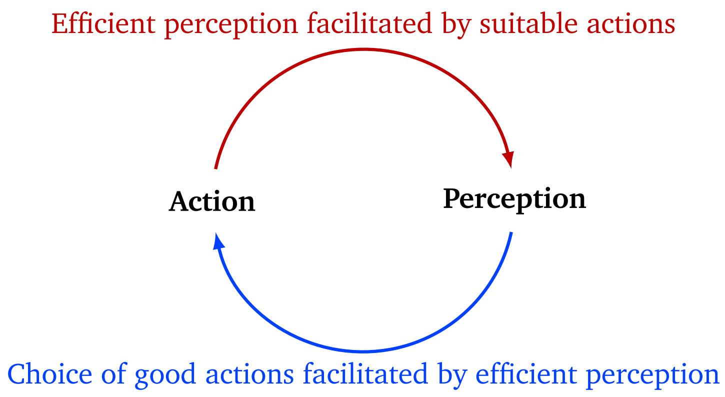

Efficient perception facilitated by suitable actions
Action
Perception
Choice of good actions facilitated by efficient perception

Fig. 3.4.: The *perception-action cycle*: It describes the flow of information that is circulating between an organism and its environment in a closed feedback loop.

Guiding perception in machine learning There is evidence that

> "The right choice of actions can augment the perception of the environment, while an efficient perception facilitates the choice of good actions in order to solve the given goal."

is a principle which is also valid for artificial agents [139]. The optimized interplay between action and perception is able to enhance the agent's *control* over the environment, i.e. the reduction of uncertainty in the process of gathering information while acting in the world. This again can greatly enhance the process of exploration. Depending on the way the agent interacts with the environment and also the type of sensory information it receives from it, it ends up in sub-categories of perception that all shape the acquisition process in different ways. In the following, three different categories are introduced, together with related approaches. These proposed methods show that the implementation or guidance of the agent's perceptual attention can have a positive influence on the agent's control over the environment and thus facilitates the learning process.

Perceptive acting This proposed sub-category summarizes all approaches that are strengthening the coupling between the agent and the world by reformulating the options or degrees of freedom that it has at its disposal in order to enhance the sampling of meaningful sensory data. These approaches exploit the knowledge of regularity in the combined space $\mathcal{S} \times \mathcal{A}$ of sensory information and actions by narrowing down the space of possible action parameters $\mathcal{A}$ to a subset of actions that leads to the highest amount of meaningful information.

One famous approach that belongs to this category is the *free-energy principle* proposed by Friston [140–142]. It is a mathematical formulation that describes how well biological agents are able to adapt to a constantly changing environment. According to the free-energy principle, the biological agent has two choices to minimize its free energy: It can either improve its perception by enhancing its internal model of the world or by optimizing the actions through enforcing a sampling of the sensory data that is consistent with its current representation. Another approach was introduced by Klyubin et al. under the name *empowerment* [143, 144]. Here, the basic idea is to choose the sequence of actions that is maximizing the achievable amount of sensory information according to the existing model of the agent. A modern application that couples the concept of perceptive acting with deep reinforcement learning is given by Jaderberg et al. [13]. Using sampled sequences of past experience, the unsupervised agent learns pixel and feature-control. For "pixel control" the visible scene is divided into cells. Now, the agent has to learn a per-cell policy for maximally changing the sensory information of the cell with its actions. The received reward is a measure of the average absolute change between the current and next frame. Additionally, feature control is introduced where agents are rewarded for learning policies that are maximizing the activation of units in specific linear layers. Another way to improve the perception in the deep learning domain by introducing a way to innervate the exploration process of the agent was proposed by Bellemare et al. [12]. The presented approach generates an additional auxiliary reward that measures the uncertainty of the agent's knowledge. This reward facilitates the encountering of novel sensory states during the exploration phase of the agent.

Active "visual" perception Bajcsy et al. [145] define an agent that is actively perceiving its environment as an *"[. . .] active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how, when and where to achieve that perception"*. Thus, the agent is not only passively observing its environment, but reasoning about how its actions might be able to maximise the flow of sensory information. Examples of designed models that implement this concept of *active perception* [139] by controlling the agent's sensory apparatus are *models of visual attention* [146–148]. In these vision systems, the agent uses actions for guiding its gaze and in this way improves the quality of gathered sensory information. While the above mentioned examples change only the sensory apparatus and are not interacting with the environment using physical contact, there are also approaches where physical interaction is used to improve the perception of the scene. An early example of this *interactive perception* [139] is given by Tsikos and Bajcsy [149], who proposed to use a robot-arm for interacting with certain objects in a scene. This setup is then exploited to simplify the scene and thus improve the effectiveness of a vision system.

Active "haptic" perception Haptic exploration is only possible by applying physical contact with the (in many cases static) environment. Additionally, it is often necessary to look into a time series of varying tactile sensory information instead of relying only on one single physical contact. This requires the development of efficient exploration policies that generate action sequences which support beneficial haptic sensing. In contrast to early studies that are heading towards an active haptic perception, see e.g. [145, 150], recent developments added machine learning techniques to their applications in order to learn, for example exploration strategies, feature extraction or to improve the estimation of different quantities. This led to approaches that are able to learn haptic object representations [151], object detection and pose estimation [152–154], texture classification or description [155–157] and reconstructing the shape of objects or the environment [158–160].

3.3.2 Simplifying the task

The "simplification" of a task can be seen as one of the most basic ways to make learning easier. In machine learning, many different ideas were developed to implement techniques into the learning process that are able to directly or indirectly simplify the task for the learner. In the following, two classes of methods are presented. While the first one provides a temporary simplification that helps to refine the internal representation of the applied learning model, the second one is based on the design of hierarchical models that are specialized in creating a beneficial structure of sub-tasks in order to simplify the learning problem.

Transfer learning The first class of methods is called *transfer learning* [161, 162]. The transfer of learning is a concept that has been known a long time in psychology [163, 164]. It has also become very popular for improving machine learning approaches. The main idea of transfer learning is to generalize knowledge across tasks. It claims that solving a problem gets easier when a task with one or more similar modalities was learned before. This can be exploited as a way to speed up the learning process by pre-training an artificial agent on simpler related challenges which are generated by varying one or more of the main problem's modalities. Examples are to train the agent on a different but related task, to utilize an alternative learning environment or to modify the agent's action set. It is also possible to vary other things like the agent's sensory information, the reward, the start state, the goal state or the entire problem space. After a "temporary restricted" training phase that can for example be defined by a fixed number of training steps or a performance threshold, the initialized model is then employed to learn to solve the main problem. The knowledge that is transferred can be the agent's policy, the (action-)state value, reward functions or something entirely different [162]. There are also other approaches like transferring knowledge through advice [165].

In the last years, researchers were able to successfully apply transfer learning on many different problems. Examples include maze navigation tasks [166], simulated robot soccer [165, 167], the DeepMind Lab [168] and also first attempts to learn tasks using a physical robot [169]. Moreover, it is one of the concepts that is not only restricted to reinforcement learning methods. Transfer learning can also be combined with supervised learning models like classifiers, for example by reusing a pre-trained model on a different or enhanced set of classes [170, 171].

Hierarchical models In addition to transfer learning, which can be seen as a temporary support of the learning process, it is also an option to use permanent approaches. The question how to endow an artificial agent with the ability to split its problem into smaller sub-tasks was extensively studied during the last decade (see e.g. [172] for a survey). One class is called *hierarchical learning models*. It comprises specified learning algorithms that have the ability to learn to tailor the main problem in a hierarchy of simpler sub-tasks and thus reduce the overall complexity. The structure of the hierarchy either has to be predefined by the programmer — like the number of sub-tasks or their individual reward functions — or has to be learned by the agent. While a temporary simplification is more in the spirit of scaffolding, a permanent support for structuring the problem might be more efficient in the long run.

Feudal architectures One of the first ideas was *feudal reinforcement learning*, presented by Dayan and Hinton [173]. They constructed a hierarchy of managers and sub-managers that are perceiving the world on their own level of resolution. The managers do not necessarily need to know the full amount of details of the world as they have to perceive it only on the level of their own tasks. Consequently, high-level managers are perceiving the world on a coarser level than low-level sub-managers. The managers have full control over their sub-managers in defining goals for them, setting rewards and giving penalties. By stacking these managers into a hierarchy, one ends up with an approach that is learning in a bottom-up manner. Inspired by this approach, Vezhnevets et al. [16] invented a hierarchical reinforcement learner that is able to utilize the concept of the feudal architecture within the deep learning domain.

Options Another way to create hierarchical agents for Q-learning was proposed by Sutton et al. [174] through the introduction of the *options framework*. The approach defines a new concept called *options* that is similar to actions but has a duration which is longer than one time-step. Each option has its own Q-learner that is coupled to a sub-problem of the complete problem. Every Q-learner has its own pseudo-reward, signaling the agent how good it is to execute a specific option in the given state. On a global scale, however, they are treated like regular actions that can be executed from a meta-Q-learner, which leads to the extension of the regular Markov decision process to *semi-Markov decision processes*. Following this principle, the regular actions are options with a duration of one time-step. In the following years this approach was improved in order to learn options at real time [175–177]. It was also ported to the setting of deep learning [178–180] and refined by the idea to define abstract sketches of task-specific policies [15].

3.3.3 Modelling and demonstration

In machine learning, many different techniques were developed that can be related to modelling and demonstration. Especially in reinforcement learning, there are many approaches that rely on a provided or learned model of the world for learning. The fact that reinforcement learning is inspired by the learning process of biological agents also gave rise to the idea to use an expert for exemplifying a good way to solve the given problem. In a following step, the expert's demonstration is then exploited for augmenting the artificial learner's process of adapting a favourable policy.

Model-based methods In reinforcement learning, there is a class of "model-based" methods, that employ a model of the world in order to predict the next step or to efficiently sample new data [25]. These kind of methods were successfully used for learning legged swimming gaits [181] or to teach a simulated biped robot to walk [182]. Although these kinds of algorithms are considered very capable, their performance is limited by the accuracy of the provided model. It is, in many cases, a troublesome task to invent an accurate model for complex domains with high-dimensional state spaces. Thus, it has become more popular to rely on model-free methods, like the (deep) Q-learner, despite the fact that they require millions of samples to learn good policies. In view of this, recent research proposed hybrid attempts, where a model-based approach is taken to initialize a model-free one [183].

Demonstration learning The idea of using demonstrations for facilitating learning has become a quite popular idea and has given rise to a battery of techniques. Even though a human expert is necessary for this kind of concept and it is thus not studied within this work, it is nevertheless worth mentioning some successful ideas for the sake of completeness. *Behaviour cloning* exploits state-action pairs that are generated by demonstration for learning a good policy. It was put to use for learning complex dexterous manipulation [46] or quadcopter navigation [184]. A second approach is based on inverse reinforcement learning that *estimates a reward function from demonstrations* [185, 186]. It was for example applied to learning autonomous helicopter aerobatics [187]. A last proposed way to improve the learning process of a reinforcement learner is given by *initializing the policies or internal models of the learner through demonstration*. It was used to accelerate the learning process of a deep Q-learner [188] or tackle the problem of sparse rewards during exploration [45].

3.3.4 Ongoing diagnosis and assessment

The diagnosis of the learner's current level of competence, also called "dynamic assessment" [107], is seen as one of the key factors that distinguishes scaffolding from other concepts that facilitate learning.

Performance metrics In machine learning, the general concept to measure a model's current quality is given by performance metrics, like the accuracy to classify data, the success rate to perform a certain task or — in case of reinforcement learning — the average reward per trial. When continuously evaluating the model, learning trends can be generated by analysing the time course of learning with respect to the model's performance. One drawback of these methods is that the model can only be evaluated "after" performing the given task.

Exploiting V & Q for dynamic assessment In reinforcement learning, the diagnosis of the agent's behaviour is a natural aspect of the learning process. As explained in Section 2.2, it is possible to compute a measurement during the learning process in form of the (action-) state value that rates the agent's current state with respect to the expected long-term behaviour for solving the current task. One kind of reinforcement learner that relies on this specific key aspect of scaffolding are *actor-critic models* (see Section 2.5). In these models the ongoing diagnosis plays a very important part, as a critic in form of the V- or Q-value is separately learned from the acting part of the model. The value function is then utilized in order to rate the actor's performance, i.e. diagnose its current skill level, which is then exploited for learning a good policy.

Curriculum learning Another presented technique — that is also on the border to "task simplification" — is *curriculum learning* [189]. The idea of this concept is to enhance the learning process by not providing the agent with random but instead with thoughtfully chosen sequences of training data that are tailored to its current capabilities. While in early attempts of curriculum learning the training sequences were constructed manually, there are nowadays some promising approaches for autonomously generating structured training data [190–194]. By now, curriculum learning has become an attractive approach to tackling complex problems within the field of deep reinforcement learning. Examples are learning to play first-person shooter games by adapting the difficulty of the computer-controlled opponents according to the learner's performance [191] or by automatically generating a curriculum for increasing the training on rare events [193].

Curriculum learning can also be used to extend *transfer learning*, where the sequence of tasks during the pre-training is then structured in a way that is beneficial for the agent [162, 192, 194]. An example for the success of combining transfer and curriculum learning can be found in nature as well. Skinner demonstrated in a study that it is possible to facilitate the training of dogs through the creation of a task hierarchy with increasing difficulty [164].

3.3.5 Fading support and eventual transfer of responsibility

During the discussion about employing tools for scaffolding the learning process of humans, an important issue was raised by the implementation of the concept of "fading" and the connected aspect of "transfer of responsibility". It has been argued that the nature of the applied tool — as it could for example be designed for a permanent support — might prevent the integration of fading. When designing scaffolding approaches for machine learning, one can now resort to the same argumentation. For this field, however, the decision is mainly influenced by whether the "scaffold" is merged with the learner or if they are separate entities. One way is to integrate the method for enhancing the learning process into the learning model. Another way is to invent methods that are more like an auxiliary gantry which can be easily turned on and off without influencing the operational reliability of the main algorithm. Examples of approaches that are intended to be used permanently during learning are hierarchical models, but also algorithms that integrate special ways of perception as recurrent attention models. Transfer learning, on the other hand, is a good example for a class of methods that is independent of the applied learning algorithm and can be seen as temporary support. It might thus be possible to integrate a "transfer of responsibility" into the learning process by pre-training the learner on a related problem where the degrees of freedom are reduced. An example for a classifier might be a reduced number of classes that is then increased step-wise. For a reinforcement learner, the "transfer of responsibility" could be integrated by training the agent at first on a problem where only a subset of actions is needed for solving it. Other proposed ideas that can at least be identified as temporary support are the ones that use "demonstration" to initialize the learner or pre-train policies. However, there is no fading but just a hard cut between the pre-learning and the learning of the main problem. Fading might be found in methods that are related to curriculum learning, where the sequence of training data increases in complexity and eventually fades into the usual policy of generating the training data.

3.3.6 Summary

Following the proposed guideline for transferring a psychological concept from the human world to the world of machines, the last sub-sections have derived relations from different modern approaches in machine learning to specific key aspects of scaffolding, as visualized in Figure 3.5. It was possible to discover many similarities between scaffolding techniques for human learners and approaches that are supporting the learning process of an artificial agent. Only the idea to "recruit and maintain the learner's attention towards a goal" had to be adapted by shifting the focus from guiding attention towards a goal to guiding attention towards important aspects of the learning problem by "refining the learners attention". These fruitful results are strengthening the hypothesis that human scaffolding is a good prototype when developing a general guiding principle for improving and accelerating the learning process of artificial learners. The next sections are now utilizing the conclusions drawn from these findings. At first, the essential concepts and propositions are taken as the point of departure for compiling a refined definition of **scaffolding as a principle for facilitating the learning of an artificial intelligence on a meta-level** that uses the five refined key-aspects as a computational skeleton for a prospective research agenda.

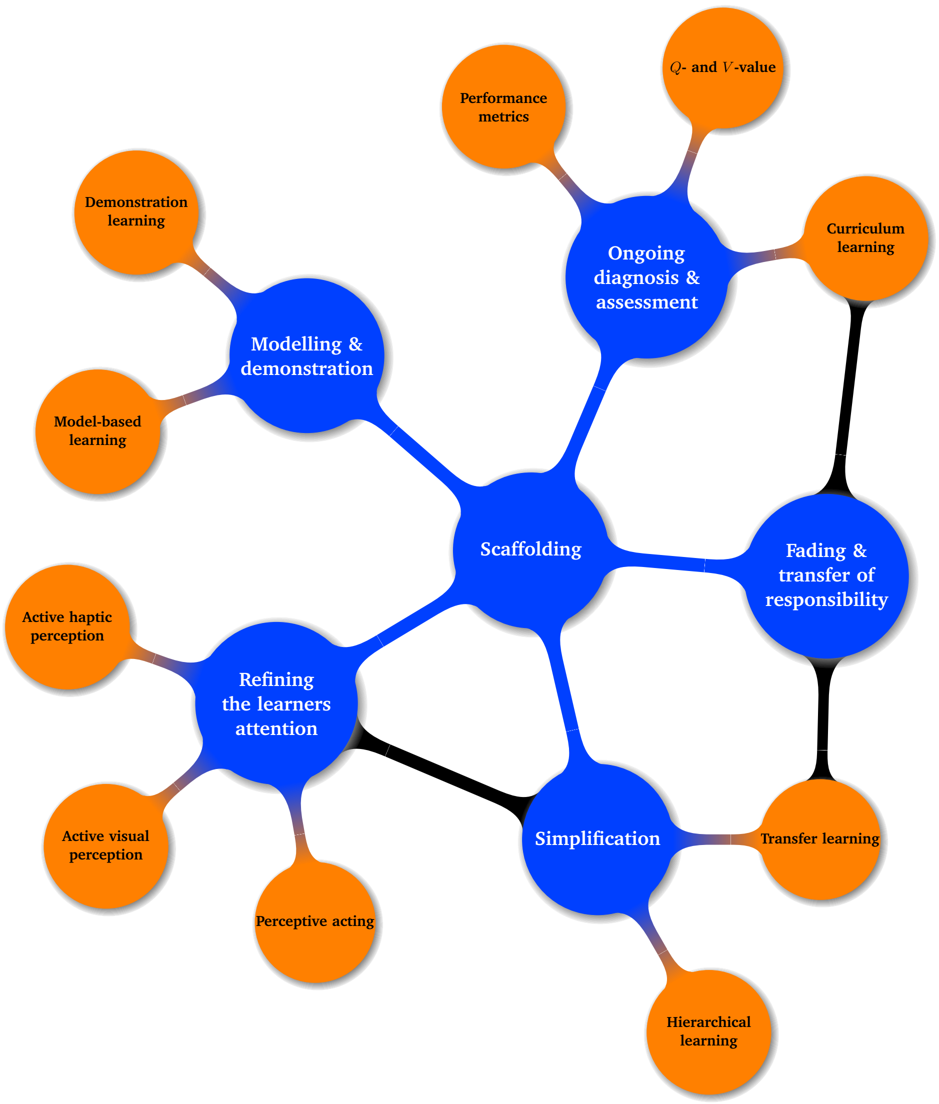


Fig. 3.5.: The figure illustrates the relation between the discussed machine learning methods and the key aspects of scaffolding.

3.4 Reformulating scaffolding as a principle for guiding the learning process of machines

The previous part of this chapter created a bridge from human scaffolding to scaffolding machines. Section 3.1 introduced the theory of human scaffolding. Section 3.2 leads it further into the field of artificial teachers by discussing the special case of scaffolding via teaching tools like computer programs. The last Section 3.3 then translated the five emerged key aspects to the realm of machine learning by connecting them with different kinds of existing approaches for improving the time course of learning. Some of them are motivated by psychology, others are based on a neurobiological, mathematical or heuristic motivation. The findings were then compressed and visualized in a mind map, given by Figure 3.5.

Figure 3.5 is now taken as the fundamental skeleton for a guideline how to design scaffolding techniques for artificial agents. While the five key aspects of scaffolding (see again Figure 3.3) were extracted from the meta-knowledge of human learning, their conceptual ideas could be connected — either directly or after necessary refinement — to efficient approaches in machine learning, building a computational skeleton for scaffolding artificial agents that are listed in Figure 3.6. This insight gives rise to the assumption that there exist notable similarities between human and machine learning on a meta-level. For that reason, the five aspects can be exploited for machine learning. Following the presented methodology, they are forming the fundamental pillars for creating scaffolds for artificial agents.

The computational skeleton for scaffolding artificial agents:

- Refining the learner's attention through guided perception
- Simplification
- Ongoing diagnosis and assessment
- Modelling & demonstration
- Fading & transfer of responsibility

Fig. 3.6.: A listing of **the computational skeleton for scaffolding artificial agents**.

3.4.1 Scaffolding in practice: inject meta-knowledge by compiling individual auxiliaries

Combining the insights of the last sections it is now possible to formulate a general definition of scaffolding in machine learning:

> The principle of scaffolding is seen as the general process of providing the artificial agent with a practical supportive construct — the scaffold — that is either integrated into or used as an extension alongside the learning framework. The function of the scaffold is either to directly bolster the learning of a task by refining the inner processes of the

utilized model or endowing it with the ability to learn a new auxiliary skill.

The scaffolds can be assigned to one or more of the five scopes of assistance, listed in Figure 3.6. They categorize the basic direction how the time course of learning is intended to be supported.

Besides these findings, some additional observations could be made in the last section. They are now seen as general rules which are exploited as guiding posts when inventing new scaffolds. The last sections gave strong evidence that the constructs and techniques which contain the resulting properties are more likely to be able to successfully improve the learning process.

Focusing on specific aspects While in the original psychological theory of scaffolding, it is desirable to design approaches that fulfill all five key aspects (see again Figure 3.3), it is not a necessary criterion for machine learning. As for teaching devices (see again Section 3.2), the concept of *scaffolding a machine learning algorithm* should be decoupled from the original one and refined in a way that is suited for the given field of research. Thus, instead of implementing all key aspects, it might be more beneficial to lay the focus only on one or two.

Scaffolds can be permanent or temporary When using (computer-based) tools for scaffolding human learning, the requirements of being a temporary support with fading have to be relaxed. It has to be decided based on the nature of the designed approach if it is more beneficial to implement a temporary or a permanent support. The same argumentation appears to hold when using machines for scaffolding machines. Many of the proposed ideas for refining the learning process of artificial agents are not designed as external tools that can be used as a temporary support. They are rather directly implemented into the learning model as a permanent auxiliary gantries which endow the learner with the technical modalities for learning new skills it would usually not be capable of. Examples are hierarchical models or the integration of active perception. On the other hand, it can also be implemented as temporary support that helps the learner to fruitfully initialize the learning process, given by transfer learning, curriculum learning or demonstration learning.

Scaffolds as self-regulating modules There are many cases where explicit guidance such as demonstration learning or feedback from either a human or artificial expert is required. The focus in this study, however, are supportive approaches that are not relying on a human expert. Thus, in this work scaffolds are seen as **self-regulating modules** that are able to operate autonomously and self contained without the guidance of a human expert.

The proposed definition of scaffolding for artificial agents at the beginning of this section, together with the three general rules for designing *good* scaffolds for machine learning is composing the basic framework that is utilized in this thesis. In the next section, it is employed to create a research map for scaffolding artificial agents by suggesting parts of the learning process in machine learning that are good candidates for further scaffolding approaches. In the end, four research questions are selected that are then employed for the rest of the thesis as practical applications for designing and testing new scaffolds.

3.5 A research map for scaffolding in machine learning

Section 3.3 has shown that many of the different ideas in machine learning which aim to facilitate the learning process could also be associated with the educational concept of scaffolding. In addition, a lot of them are apparently similar to key aspects of the psychological theory. As many approaches rely on this class of methods as a basic learning framework, there is strong evidence that especially reinforcement learning is a very good candidate for possible refinement that is inspired by psychology. The aim of this section is now to point out the stages of the learning process of an artificial agent that have the most potential to significantly improve the result in speed and quality when being efficiently scaffolded.

A research map for scaffolding One stage of the learning process that seems to get much attention in the field of machine learning is the early learning phase. Techniques like transfer learning, curriculum learning or demonstration learning are employed as pre-learning routines in order to get a good initialization of the learner. Another aspect of the learning process that seems to be intensively discussed and appears to have lots of potential left is the active perception of the environment. While the literature indicates that the process of gathering information has long been seen as something independent from the learning process, concepts like *interactive perception* [139], *empowerment* [143, 144] or the different ways of encouraging a deep learner to encounter novel states [12, 13], demonstrate that seeing perception as an interactive and adaptable process is a promising way to enhance learning. Section 3.3.1 has shown that perception has a strong influence on the learning process. Being connected to the agent's capabilities of interacting and also its way of sensing the environment, it is also very versatile. As a consequence, there exists no universal solution to augment the agents perceptional abilities. Instead, the supportive techniques have not only to be aligned to the specific learning problem, but also to the agent's individual modalities to sense and interact with the world. With this conclusion in mind, one part of this thesis lays its focus on the design of different scaffolds that intend to support the learning process through **refining the perception** of the agent with respect to its way of sensing the world and **enhance its attention to task-relevant information**.

3.5.1 Four research questions for scaffolding an artificial agent

This thesis presents two different ways of designing scaffolds. The first way is to concentrate on one specific pillar instead of trying to cover all five pillars of artificial scaffolding that are listed in Figure 3.6. In this work, the chosen pillar is given by *refining the learner's attention*. By analysing the individual characteristics of the given artificial agent in combination with the given learning problem, supportive modules can be designed that are able to bolster the learner's abilities or endow it with new ones that help to solve the given problem. For studying and demonstrating this idea, three different categories of perception are analysed and individual supporting scaffolds are designed. The second way combines the aspects of simplification, ongoing diagnosis and assessment together with fading & transfer of responsibility. This approach is not only putting the improvement of the learning process into focus but also offers a platform to study how well the different key aspects of scaffolding work together.

A scaffold based on perceptive acting While the sub-category of active perception summarizes many interesting concepts like the *free-energy principle* and *empowerment*, there seems to be no existing modular procedure that rates whole interaction strategies according to their influence on the agent's perception. Especially in reinforcement learning, where the agent learns through direct interaction with the environment, a good choice of actions is essential. By analysing their quality with respect to the given environment before learning, the problem's degrees of freedom might be significantly reduced even before starting the learning process. All things considered, one can ask the question:

> Are there general features that distinguish action sets that facilitate exploration, learning and control ("good" action sets) from action sets for which exploration, learning and control is more difficult?

A scaffold based on active visual perception There also exist some approaches that try to integrate active visual perception as an adaptable module into the learning process, they cannot yet be applied to more complex learning problems. Refining and extending one of these approaches could thus be a good starting point to exploit this kind of active perception. As a result, it would be possible to integrate an attention mechanism into the learning of these kinds of problems that helps to reduce the amount of information the agent has to process. In conclusion, one can investigate the following matter:

> Is it possible to scaffold the learning process of complex learning problems by using *visual attention* in order to implement *active perception* into the learning process of artificial agents that rely on visual input as their sensory information?

A scaffold based on active haptic perception Unlike humans, who after years of developmental process acquire goal- directed haptic exploration capabilities, it is still a great challenge even for very advanced robotic platforms to perform contact-rich interaction tasks. The integration of a scaffold into the learning model that enables the agent to learn to control its tactile sensing through active haptic perception has the potential to endow robots in future with the necessary skills. One can thus ask:

> Is it possible to scaffold the learning process of haptic problems by using *recurrent attention* in order to learn synthetical exploratory procedures for robots using optimization of motor control?

Scaffolding hierarchical problems Recent works indicate that learning temporal abstraction and hierarchical task solving are very promising. Hence, there exist many different approaches. Hierarchical reinforcement learning is — in many cases — tackled by designing new architectures that are able to learn to segment the main problem into smaller sub-tasks. It might thus be an interesting attempt to employ a variant of transfer learning combined with curriculum learning for refining the internal representation of a non-hierarchical model in order to better cope with learning tasks that have a hierarchical structure. In this way, it would be possible to better solve hierarchical tasks without the design of a specialized learning architecture but only through a scaffold in the form of a temporary support in the early stage of learning. Notwithstanding the fact that the last section listed some promising ways for learning an automatic generation of an optimal curriculum, it is an interesting attempt to design a curriculum close to the key aspects of scaffolding and study its effects on learning by studying the questions:

> Can a scaffold, based on the combined approach of transfer and curriculum learning, augment the learning process of established learning models? How is the integration of key characteristics of the refined concept of scaffolding influencing the learning process?

3.6 Summary

In this chapter, the general idea of scaffolding the learning process in the field of machine learning was discussed. At first, the original concept from educational psychology was presented. After the application and influence of the derived techniques on the learning process of humans was discussed, the idea was extended by taking the scaffolding of human learners through teaching devices like computer programs under consideration. In a further step, the transition to scaffolding machines without a human teacher was induced by relating the five key aspects of the original scaffolding theory to different kinds of popular machine learning approaches that were able to improve the training process by utilizing various kinds of methods. There are, however, good indicators that the concept of scaffolding has to be refined in order to be more adaptable to the characteristics and needs of the field. During the discussion of the different approaches, some collective features have been detected that were then used to adjust the key aspects and hence align them more to the realm of machine learning. In the end, a mind map was created that is shown in Figure 3.5. This knowledge was then used for reformulating the concept of **scaffolding as a general guiding principle that exploits available meta-knowledge from the field of machine learning** and to sketch a rough research map that highlighted regions of the learning process that seem to be good starting points for designing scaffolds.

The next step As a last step, the collected information was utilized to propose four different approaches with the intention to support the learning process of different kinds of learning problems. For the concretisation and realisation of these supportive techniques in the next part of the thesis, the reformulated concept of scaffolding (see again Section 3.4) is utilized as the main source of inspiration.

Part II

Scaffolding: a universal approach for fostering the learning process

4 Scaffolding attention control by exploiting "perceptive acting"

Actions play a crucial role in reinforcement learning as they determine how much control the agent has over the environment and thus influence the effectiveness of exploration and learning. If an agent has to navigate itself or objects through the environment, there are many different ways to design the underlying sensorimotor coordinate system that directs the way of motion. The objects can either be maneuvered in a way that is completely uncorrelated to the characteristics of the environment or by taking its features into account so that desired events are more likely to occur. By following the idea of *perceptive acting* and raising the probability of these events (e.g. object-object interaction) occurring, the learning process can be accelerated.

As the sensorimotor coordinate systems shape the agent's modality of interacting with the world, they can be used to create different action sets that define the agent's motions. This gives rise to the following question:

> Are there general features that distinguish action sets that facilitate exploration, learning and control ("good" action sets) from action sets for which exploration, learning and control is more difficult?

Obviously, criteria to recognize such action sets would be of interest for designing interactive learning algorithms that are fast and efficient. It would also be possible to **scaffold existing learning frameworks by replacing the currently used and - according to the proposed criteria - suboptimal action set with a better one**. This simple adjustment is likely to be able to improve the learning process "without" explicitly modifying the utilized learning algorithm. In order to tackle this kind of problem, the current chapter proposes an approach for evaluating the general performance of specific action sets via the computation of the mutual information (see also [21]).

4.1 The concept of entropy and mutual information in the context of reinforcement learning

This section deals with the concept of *entropy* and *mutual information* [195–197] within the framework of reinforcement learning. As mentioned in Section 2.1, state transitions within a Markov decision process can be described through a tuple (s, a, s'), while each tuple is also statistically independent. If the agent induces a state transition from state $s \in \mathcal{S}$, the final state s' is within a subset of possible states $\mathcal{S}' \subseteq \mathcal{S}$. These kind of arbitrary transitions can be described by the conditional probability $\mathcal{P}_{ss'} : \mathcal{S} \longrightarrow \mathcal{S}'$, which is completely defined by the dynamics of the environment. These state transitions can be controlled by the agent through the execution of actions $a \in \mathcal{A}$ in state s with the conditional probability $\mathcal{P}_s^a : \mathcal{S} \longrightarrow \mathcal{A}$. $\mathcal{P}_s^a = \pi(a|s)$ is also called the "policy" of the agent. Both probabilities $\mathcal{P}_{ss'}$ and $\mathcal{P}_s^a$ have to satisfy

$$\sum_{s'} \mathcal{P}_{ss'} = 1 \quad \text{and} \quad \sum_{a} \mathcal{P}_s^a = 1 \quad \forall s, s' \in \mathcal{S} \text{ and } \forall a \in \mathcal{A}. \tag{4.1}$$

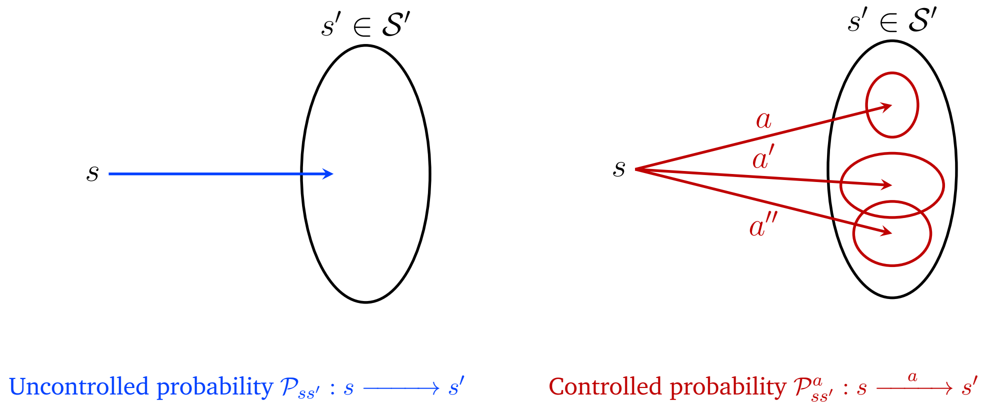
With the help of the joint probability[1] to execute an action a and then end up in a state s', given by $\mathcal{P}_s(s' \cap a)$, the transition probabilities for ending up in state s' after executing action a in a given state s can be computed as

$$\mathcal{P}_{ss'}^a = \frac{\mathcal{P}_s(s' \cap a)}{\mathcal{P}_s^a}. \tag{4.2}$$

Thus, using an action a is narrowing down the number of final states s' by increasing the probability to end up within a smaller subset $\mathcal{S}'(a) \subseteq \mathcal{S}'$, specified by the used action (see Figure 4.1). This controlled probability $\mathcal{P}_{ss'}^a : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{S}'$ is connected to the uncontrolled probability via

$$\mathcal{P}_{ss'} = \sum_{a \in \mathcal{A}} \mathcal{P}(s' \cap a) = \sum_{a \in \mathcal{A}} \mathcal{P}_s^a \mathcal{P}_{ss'}^a \quad \forall s, s' \in \mathcal{S}. \tag{4.3}$$

[1]In this work the notation of [196] is employed: If there are two events E_1 and E_2 existing, then the event $E_1 \cup E_2$ is the one in which either E_1 or E_2 or both events occur. $E_1 \cap E_2$ is defined as the event in which both events E_1 and E_2 are happening.


$s' \in \mathcal{S}'$
s
Uncontrolled probability $\mathcal{P}_{ss'} : s \longrightarrow s'$
$s' \in \mathcal{S}'$
a
a'
a''
s
Controlled probability $\mathcal{P}^a_{ss'} : s \xrightarrow{a} s'$

(a) Illustration of an uncontrolled state transition from state s

(b) Illustration of an controlled state transition from state s after executing an action a

Fig. 4.1.: Using information theory in order to describe the influence of actions on the dynamics of the environment. While the transitions from a state s into a state s' is usually entirely described by the passive dynamics of the world, given by a probability function $\mathcal{P}_{ss'}$ as illustrated in (a), the execution of actions a is able to increase the likeliness to end up in specific states, visualized in (b).

The entropy as a measure of uncertainty To measure the uncertainty about the next state, the *entropy* [195] $\mathcal{H}_s$ of the current state s

$$\mathcal{H}_s(\mathcal{S}') = -\sum_{s' \in \mathcal{S}'} \mathcal{P}_{ss'} \ln\left(\mathcal{P}_{ss'}\right) \tag{4.4}$$

can be defined. By introducing the *surprise* (or self-information) [142] for a given state s, which is defined as the negative logarithm of the probability $-\ln(\mathcal{P}_{ss'})$, the entropy can be interpreted as the average surprise over all possible states that can be reached within one transition step. By taking the possible actions $a \in \mathcal{A}$ into account, the *conditional surprise* (or conditional self-information) $-\ln(\mathcal{P}^a_{ss'})$ gives information how certain it is to end up in s' after executing a in s. Analogue to the entropy, the *conditional entropy*

$$\mathcal{H}_s(\mathcal{S}'|\mathcal{A}) \quad = \quad -\sum_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}'} \mathcal{P}_s(s' \cap a) \ln(\mathcal{P}^a_{ss'}) \tag{4.5}$$

of state s can be computed [195, 196]. It measures the average surprise of ending up in a state $s' \in \mathcal{S}'$ after starting in state $s \in \mathcal{S}$ and executing an action $a \in \mathcal{A}$. By using (4.2), it can be rewritten as

$$\begin{aligned}
\mathcal{H}_s(\mathcal{S}'|\mathcal{A}) &= -\sum_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}'} \mathcal{P}_s(s'\cap a)\ln\left(\frac{\mathcal{P}_s(s'\cap a)}{\mathcal{P}_s^a}\right) \\
&= -\sum_{a\in\mathcal{A}} \mathcal{P}_s^a \left[\sum_{s'\in\mathcal{S}'} \mathcal{P}_{ss'}^a \ln\left(\mathcal{P}_{ss'}^a\right)\right]. \qquad (4.6)
\end{aligned}$$

4.1.1 Exploiting mutual information as a ranking criteria for action sets

The rate of influence that is enforced by one set of actions $\mathcal{A}$ on the uncontrolled transitions $\mathcal{P}_{ss'}$ of a state s (by minimizing the agents prediction error) is thus given by the difference of the states entropy and the conditional entropy of the given actions in this state. The result is known as the *mutual information* [195–197]

$$\mathcal{M}_s(\mathcal{S}',\mathcal{A}) = \mathcal{H}_s(\mathcal{S}') - \mathcal{H}_s(\mathcal{S}'|\mathcal{A}). \qquad (4.7)$$

It is also possible to relate it to the *Kulbeck-Leibler Divergence* [197]

$$D(q(x)||p(x)) = \sum_x p(x)\ln\left(\frac{p(x)}{q(x)}\right),$$

that measures the "distance" between two probability densities $q(x)$ and $p(x)$. The more different these two distributions are, the higher is the information gain. By rewriting the mutual information using (4.3), (4.4) and (4.5) into the form

$$\begin{aligned}
\mathcal{M}_s(\mathcal{S}',\mathcal{A}) &= \mathcal{H}_s(\mathcal{S}') - \mathcal{H}_s(\mathcal{S}'|\mathcal{A}) \\
&= -\sum_{s'\in\mathcal{S}'} \mathcal{P}_{ss'}\ln\left(\mathcal{P}_{ss'}\right) - \left(-\sum_{a\in\mathcal{A}}\mathcal{P}_s^a\left[\sum_{s'\in\mathcal{S}'}\mathcal{P}_{ss'}^a\ln\left(\mathcal{P}_{ss'}^a\right)\right]\right) \\
&= -\sum_{s'\in\mathcal{S}'}\sum_{a\in\mathcal{A}} \mathcal{P}_s^a\mathcal{P}_{ss'}^a\ln\left(\mathcal{P}_{ss'}\right) - \left(-\sum_{a\in\mathcal{A}}\mathcal{P}_s^a\left[\sum_{s'\in\mathcal{S}'}\mathcal{P}_{ss'}^a\ln\left(\mathcal{P}_{ss'}^a\right)\right]\right) \\
&= \sum_{s'\in\mathcal{S}'}\sum_{a\in\mathcal{A}} \mathcal{P}_s^a\mathcal{P}_{ss'}^a\left[\ln\left(\mathcal{P}_{ss'}^a\right) - \ln\left(\mathcal{P}_{ss'}\right)\right] \\
&= \sum_{a\in\mathcal{A}} \mathcal{P}_s^a D\left(\mathcal{P}_{ss'}^a||\mathcal{P}_{ss'}\right) \\
&= \mathbb{E}_{a\in\mathcal{A}}\Big[D\left(\mathcal{P}_{ss'}^a||\mathcal{P}_{ss'}\right)\Big],
\end{aligned}$$

it is possible to see that the mutual information $\mathcal{M}_s(\mathcal{S}', \mathcal{A})$ measures the expected information gain when actions a from the action set $\mathcal{A}$ are used to control the state transitions from state $s \in \mathcal{S}$ instead of relying only on the passive dynamics of the environment.

Ranking action sets $\mathcal{A}$ with respect to their control over the environment As the mutual information of state s, given by $\mathcal{M}_s(\mathcal{S}', \mathcal{A})$, measures the reduction of uncertainty of the possible next states S' due to the influence of $\mathcal{A}$, it is a promising candidate for building a scaffold for reinforcement learning algorithms that tries to give support through a good selection of actions for the selected learning domain. In the presented approach, the mutual information is employed to compare different action sets $\mathcal{A}$. The ranking order is then used as a criterion for predicting the agent's learning performance while using the respective action set. The best coordinate system $\mathcal{A}^*$ should therefore be the one which induces the most control on the complete system and thus provides the learning agent with the highest gain of sensory information via *perceptive acting*. The action set $\mathcal{A}^*$ which reduces the uncertainty within the domain by the highest amount can thus be formally described as the one with the highest expected mutual information when taking all available states under consideration:

$$\mathcal{A}^* = \underset{\mathcal{A}}{\operatorname{argmax}} \ \mathbb{E}_{s \in \mathcal{S}} \left[\mathcal{M}_s(\mathcal{S}', \mathcal{A})\right] = \underset{\mathcal{A}}{\operatorname{argmax}} \left\langle \mathcal{M}(\mathcal{S}', \mathcal{A}) \right\rangle$$

If the model of the environment is fully available, it is possible to compute the mutual information directly. Unfortunately, the model is not known in most cases. Hence, the uncontrolled and controlled probability densities for the state transitions and thus the mutual information have to be estimated.

4.2 Applying the concept to complex environments

Both artificial and biological agents are perceiving the world by utilizing special sensors which are for example eyes, nose and ears for biological agents and cameras, laser scanners or pressure sensors for robots. From a mathematical perspective, these sensors can be seen as functions $F : \mathcal{W} \longrightarrow \mathcal{S}$ that map the full information of the environment $\mathcal{W}$ to the sensor space[2] $\mathcal{S}$ which is defined by the kind of sensor that is applied (see Figure 4.2).

[2]The terms sensor space and state space are used interchangeable in this work.

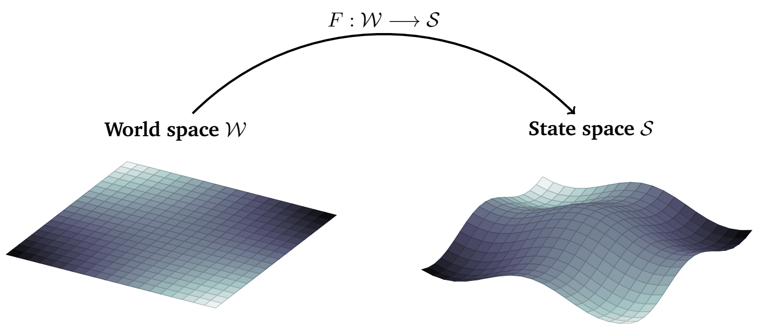

$F : \mathcal{W} \longrightarrow \mathcal{S}$
World space $\mathcal{W}$
State space $\mathcal{S}$

Fig. 4.2.: Sensors are mapping the full information of the domain onto a smaller subspace $\mathcal{S}$ called the sensor space or state space. Often, this space is of a lower dimension but also is not including the full information of the world-state. In many cases, there exists no exact inverse F^{-1}.

If now the mutual information has to be estimated in order to measure the reduction of uncertainty during the state transitions through the influence of the defined actions, the probabilities of the state transitions should not be measured within the world space $\mathcal{W}$. Instead they should be estimated within the space that contains the information that the agent is actually perceiving, i.e. within the sensor space $\mathcal{S}$.

4.2.1 Estimating the probability distribution of state transitions

When working with artificial agents that are interacting in real-world environments, the access to sensory data might be limited as it is only accessible by direct exploration of the environment. In this work, the probabilities are estimated in a two-step process:

Approximating the state space At first, a set of randomly generated sensory data is recorded. While in real-world environments this might be only possible by direct random exploration of the agent, it can be achieved easier in simulated environments, as different configurations of the environment can be easily sampled. The recorded sensory data is then used to generate K clusters c_k within the recorded sensor space, for example using the K-Means algorithm[3] [199, 200]. The set of recorded data should be large with respect to the number of to-be-generated clusters. The resulting clusters should then cover the whole visited state-space on the one hand, while additionally taking the inner structure of the sensor space into account. Consequently, they should be more concentrated in the regions which are visited more often and sparse in regions that are less frequently visited. The generated

[3] For a modern and comprehensive introduction, see e.g. [198].

cluster-points are then used as data for dividing the sensor space into cells using *Voronoi tessellation*[4] [201–203]. An illustration of a Voronoi tessellation can be found in Figure 4.3. It separates the given space into cells, where each cell is built around one cluster-point c_k. Each cell, defined by a cluster-point c_k, comprises all points whose distance to c_k is smaller than or equal to any other cluster-point c_j.

The sensor space is now segmented into cells, where the coarseness of the cells depends on the placed clusters c_k and thus how frequently the part of the sensor space is truly visited during the exploration of the agent. This approximated sensor space is defined as $\mathtt{S}$.

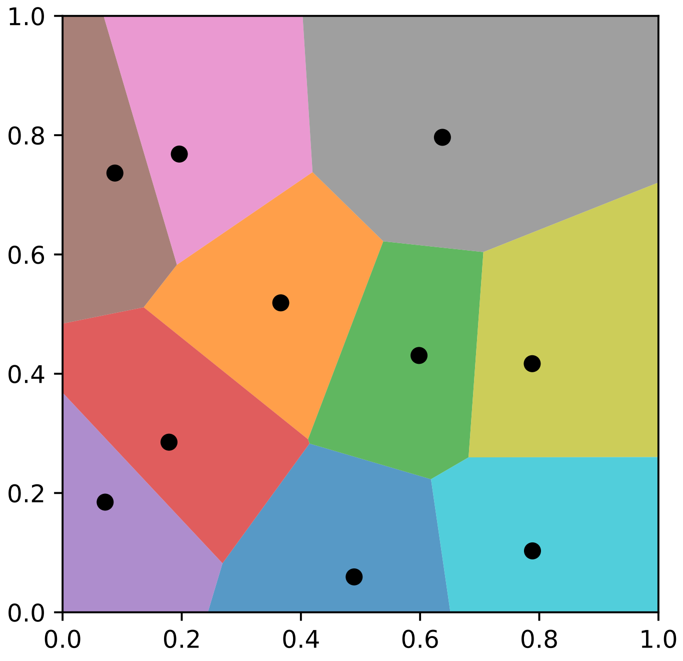


Fig. 4.3.: Illustration of a Voronoi tessellation in the euclidean space of 10 data-points $\{p_1, \dots, p_{10}\}$ that are uniformly sampled between 0 and 1. The distance of all points within in the cell, corresponding to point p_i, is smaller or equal to the distance to any other point p_j.

Estimating the transition probabilities In a second step, the transition probabilities have to be estimated while utilizing the created approximation $\mathtt{S}$ of the sensor space. For that reason, the agent has to perform a random walk within the domain while using the to-be-tested set of actions $\mathcal{A}$ in order to gather tuples (s, a, s'). These tuples are then used to count how often the cells in the discretized sensor space $\mathtt{S}$

[4]Also called *Voronoi diagram*, *Dirichlet tessellation* or *Thiessen polygons*.

are visited. Using this information, the two probability densities $\mathtt{P}_{ss'}$ and $\mathtt{P}^{a}_{s}$ can be directly estimated utilizing (4.1), while $\mathtt{P}^{a}_{ss'}$ is computed via (4.2) and (4.3).

4.2.2 Estimating the entropy & mutual information

Estimating entropies from finite samples of probability densities can be a challenging problem and has been discussed in many works, e.g [204–208]. The most basic approach is the *maximum likelihood estimator*[5] (MLE). It simply computes the entropy via (4.4), leading to

$$\mathtt{H}(\mathtt{P}_{ss'}) = -\sum_{s'\in\mathtt{S}'} \mathtt{P}_{ss'} \ln\left(\mathtt{P}_{ss'}\right).$$

To estimate the mutual information (4.7), an estimation for the conditional entropy $\mathtt{H}(\mathtt{P}(s' \cap a)$ is also needed. Like $\mathtt{H}$, the maximum likelihood estimator of the conditional entropy can be computed using (4.6) and the approximated probability densities. The estimated conditional entropy is thus

$$\mathtt{H}(\mathtt{S}'|\mathtt{A}) = -\sum_{a\in\mathcal{A}} \mathtt{P}^{a}_{s} \left[\sum_{s'\in\mathtt{S}'} \mathtt{P}^{a}_{ss'} \ln\left(\mathtt{P}^{a}_{ss'}\right)\right].$$

Finally, the estimated average mutual information of the approximate sensor space $\mathtt{S}$ for a given action set $\mathcal{A}$ is given by

$$\langle\mathcal{M}(\mathtt{S}',\mathcal{A})\rangle = \frac{1}{|\mathtt{S}|}\sum_{s\in\mathtt{S}} \left[\mathtt{H}_s(\mathtt{S}') - \mathtt{H}_s(\mathtt{S}'|\mathcal{A})\right],$$

with $|\mathtt{S}|$ the number of cells within the approximated sensor space.

Improvements of the Maximum Likelihood Estimator According to Basharin [209] and Harris [210], the entropy might be underestimated when using the simple MLE. Thus, there exist different approaches that are trying to improve the results. Examples of these kind of approaches are the *MLE with Miller-Madow correction* [204] or the *jackknifed version of the MLE* [205, 208, 211]. As this work is not

[5]Also called *naive estimator*[206] or *plug-in estimator*[207]

directly interested in a perfectly correct measure of the mutual information but just the difference of its measurements for particular action sets, it should be safe to assume that even the simple MLE is able to provide reliable results.

4.3 Summary

In this chapter, the idea of “perceptive acting” was put to use for designing a first scaffold. It is built on the concept of relying on mutual information for ranking the impact of different interaction strategies on the control of the environment. The scaffolding takes place by choosing the strategy which endows the agent with the highest control. As the agent now encounters novel and meaningful sensory states more often, the learning is faster and more efficient. As complex learning environments have huge state spaces, techniques for approximating it and extracting the probabilities for state transitions are discussed. In addition, ways to estimate the entropy and thus the mutual information from these estimated probability densities are reviewed.

The next step The next chapter deals with the integration of “active visual perception” into the learning process of a deep reinforcement learner. Therefore, the “recurrent model of visual attention” is combined with an “asynchronous advantage actor-critic model” in order to create a scaffolded reinforcement learner that is able to actively search for the salient information within a visual scenery, accumulate them and use these compressed features to solve the given problem.

5 Scaffolding attention control by exploiting "active visual perception"

In the last years, many exciting approaches were presented in the field of deep reinforcement learning (for a brief survey, see e.g. [212]). Numerous different models were proposed which endow an artificial agent with the capability to use the raw visual input of the scene in order to learn to grasp objects [42] or to play games [7, 213]. Despite these rapid developments, there are some open problems left to solve for those kinds of models that undermine and slow down the learning process. One issue is given by the fact that the whole visual scene of the environment is processed through the presented models although only a part of it may contain the relevant information that is needed to solve the given problem. A way to tackle this problem is to switch from a simple perception of static images to an *active visual perception* of the scene. While humans subjectively perceive a scene as a whole image, they are only able to recognize details of the scenery in a small central zone while the location of these fixations depends on the current task [214, 215]. Directed by image-based and task-dependent saliency cues, they are able to gather the important information about the environment [146, 216]. The information contents from these foveal "glimpses" is then combined in order to get an accumulated understanding of the visible scene.

Based on this observation from neuro- and cognitive science, some successful attempts were made to integrate this human-like way of visual perception into deep learning approaches [5]. One recently introduced example is the *recurrent model of visual attention* (RAM) [148]. The presented model is not only able to classify the MNIST training data set [217] with a success rate of more than 98% using a small number of fovea-like glimpses. It is also able to "search" for the written letters when the original images of 28×28 pixels are embedded with random location within a larger 60×60 image. It has also been shown that the same model can be exploited as a reinforcement learner that is able to solve a simplified pong task with a success rate of 85% by actively searching the scene for the ball and then adjusting the paddle to catch it. Therefore, the model has to be trained for about 20 million frames in order to achieve this performance. The fact that training takes quite long and that the paddle can be controlled using only two actions (move it left or move it right) leads to the assumption that the model needs to be improved in order to apply it to more complex tasks.

While the original RAM is a deep network that relies purely on one recurrent unit and linear layers, Ba et al. [218] presented an improved version that also implements convolutional layers [219, 220] in order to process the glimpses. They also replaced the single recurrent unit by two *long short-term memory* (LSTM) networks [76] and introduced some other advancements like the utilization of a special *context network* in order to initialize the inner states of one of these LSTM networks. This and other enhancements endowed the RAM with the ability to localize and recognize multiple objects within one image in order to transcribe house number sequences from Google Street View images.

Research question and new contribution The argumentations within the previous paragraphs lead to the generic question:

> Is it possible to scaffold the learning process of complex learning problems by using *visual attention* in order to implement *active perception* into the learning process of artificial agents that rely on visual input as their sensory information?

Switching from perception of the whole scenery to a more refined way is not only a promising way to reduce the number of weights and thus the complexity of the to-be-used learning model. Also the training time is likely to be reduced and new insights of the process of gathering and processing segments of information might be gained. For this reason, a new learning architecture is presented in the next section that integrates the RAM's abilities into a deep asynchronous actor-critic architecture in order to construct a scaffold for *active perception* around the reinforcement learner.

5.1 The recurrent attention asynchronous advantage actor-critic model

In this section, a variant of the *recurrent model of visual attention* is presented that should be able to solve tasks in learning environments that are more challenging than the classification of the MNIST dataset or the simplified pong task. An example could be an environment that provides a raw sensory input in form of an image of the current scene and a learning task that might only be solvable through the learning of physical interaction with different parts of the environment in a hierarchical manner. To enable the model to cope with such difficult kinds of tasks, the RAM is combined with the famous *asynchronous advantage actor-critic algorithm* (A3C) [92]. While the *recurrent model of visual attention* is able to scaffold the learning process by narrowing down the amount of required sensory information via *active perception*, *asynchronous models* are able to significantly reduce the computation time of complex learning problems by distributing the learning process to many independent workers.

The novel approach, presented in this section, now combines these two approaches in order generate a learning architecture that provides a permanent scaffold for controlling the process of perception while also alleviating the generation of data and the learning process by exploiting the asynchronous interplay of multiple learners. As the RAM can be trained using REINFORCE (see again Section 2.5.1), it is possible to see it as an *actor-critic* approach. This fact makes it an ideal candidate to be combined with the A3C architecture, leading to the *recurrent attention asynchronous advantage actor-critic model* (RAA3C) (also see [23]). Thus, the overall framework of the proposed architecture is an asynchronous model in which multiple workers are acting within their own domain in parallel. While the design of the individual workers is based on the RAM, there are visible changes within the architecture as illustrated in Figure 5.1.

Like the RAM, the workers are composed of five different neural network modules: the "glimpse network", the "location network", the "actor-critic network" and the "context network", that are all connected by a "memory network". If not stated otherwise, all layers use a *rectified linear unit* (ReLU) [60] as their activation function. Instead of receiving the raw visual information of the whole scene as the input, the workers are only able to process a sequence of small glimpse-like pixel patches. While the first "glimpse" of $w_g \times h_g$ pixels is random, the agent chooses a location of the scenery, using the "location network" at each consecutive time-step in order to select a new region where to look next.

As suggested in [218], a second coarser image version of the glimpse is also taken. This coarser image covers twice as many pixels as the first patch and is then also rescaled to a size of $w_g \times h_g$ and concatenated with the original glimpse in order to create the effect of a "fovea", where the image is sharp in the center and gets blurry towards the outside. The combined glimpse image is then flattened to create the glimpse vector $\boldsymbol{g}$ that is processed through the "glimpse network", together with its corresponding location $\boldsymbol{x}_g$. The extracted features are transferred through the

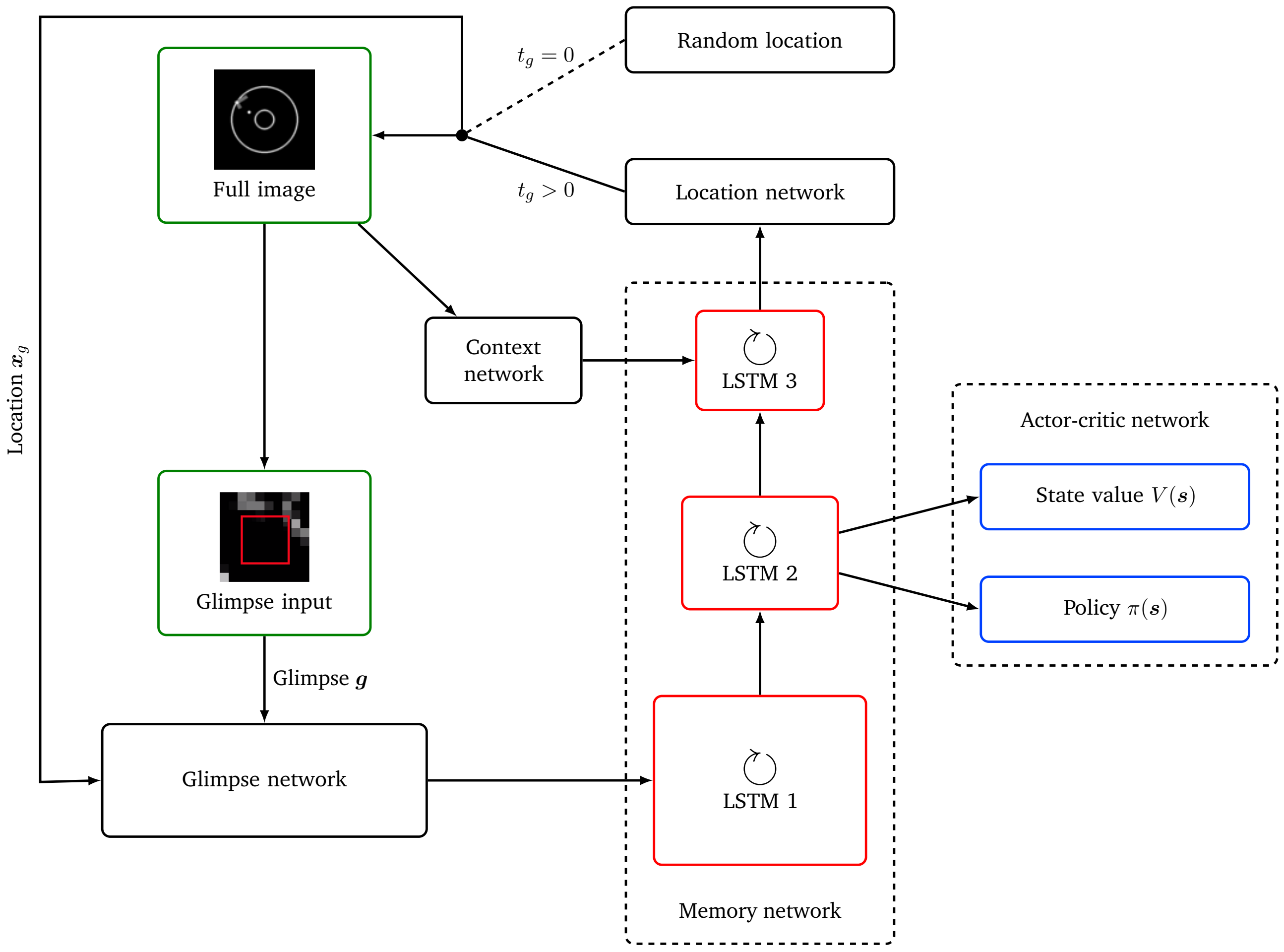


Fig. 5.1.: An illustration of the workers of the presented model. The location of the first glimpse $t_g = 0$ is chosen randomly. The location of all other glimpses $t_g > 0$ is chosen by the location network.

"memory network" in order to generate the next glimpse location $\boldsymbol{x}'_g$ with the help of the "location network". After a predefined number of these glimpses, the agent takes the accumulated information of the full glimpse sequence to choose an available action a based on its current policy using the "actor-critic network".

Glimpse network Like the input, the glimpse network receives a vector $\boldsymbol{g}$ of the dimension $\dim(\boldsymbol{g}) = 2 \cdot w_g \cdot h_g$, where h_g is the pixel-height and h_g the pixel-width of the glimpse. It encodes the flattened information of the current foveal glimpse, taken at location $\boldsymbol{x}_g$. The module then combines the current glimpse vector $\boldsymbol{g}$ with its corresponding location $\boldsymbol{x}_g \in [-1, 1]$ in order to generate suitable features that are processed through the rest of the network. Both the glimpse vector $\boldsymbol{g}$ and the location $\boldsymbol{x}_g$ are processed through one linear layer with 256 neurons each. One important factor that has to be carefully considered is how to combine the location of the glimpse with the visual information. Different options were presented in different works, such as element-wise addition [148], element-wise multiplication [218, 221] and concatenation [222]. In this work, using a concatenation, followed by two additional linear layers of 512 neurons in order to generate the input for *LSTM 1*, was the method that performed best. The architecture of the designed glimpse network is illustrated in figure 5.2.

Memory network Rather than relying on one simple recurrent neural network [148] or two connected LSTM networks [218] for storing and combining the received glimpse features, a memory network is introduced that is built out of three LSTM networks with decreasing sizes that are all connected using a ReLU activation function. While all internal states within *LSTM 1* have 512 neurons, *LSTM 2* is constructed using 128 neurons and thus reduces the features to $1/4$ of the original size. This value is again halved for *LSTM 3* which is therefore designed by using hidden states with 64 neurons. Reducing the size of the LSTM networks is not only done to reduce the overall number of trainable weights within the model but also to compress the information stored within the features. It is important to notice that the actor-critic network receives the features that are generated by *LSTM 2*, while the location network receives the features that are generated by *LSTM 3*.

Location network The location network is built out of layers with 64 neurons and illustrated in figure 5.3. It uses the accumulated features of *LSTM 2* (see Fig. 5.1) in order to generate a new 2-dimensional pair of coordinates $\boldsymbol{x}_g$ that define the center of the next glimpse $\boldsymbol{g}$. The coordinates are chosen using a stochastic policy, modeled by a two-dimensional Gaussian. The output of

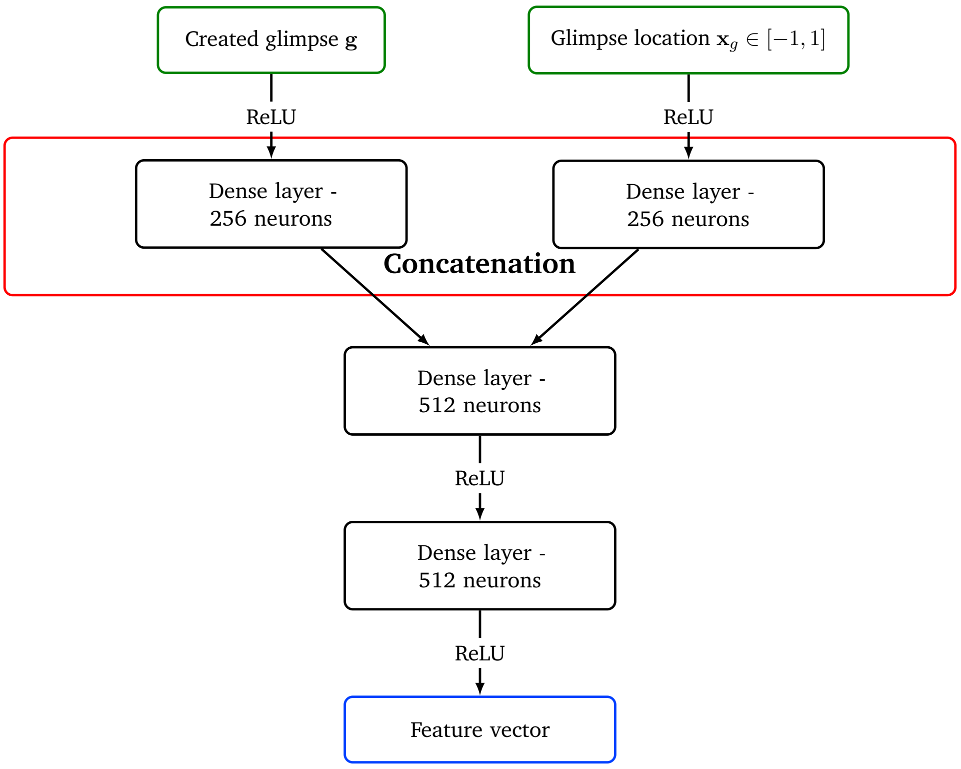

Created glimpse g
Glimpse location $\mathbf{x}_g \in [-1, 1]$
ReLU
ReLU
Dense layer - 256 neurons
Dense layer - 256 neurons
Concatenation
Dense layer - 512 neurons
ReLU
Dense layer - 512 neurons
ReLU
Feature vector

Fig. 5.2.: An illustration of the glimpse network.

the network are therefore the mean $\boldsymbol{\mu}$ and the standard deviation[1] $\boldsymbol{\sigma}$. The Gaussian is restricted to the range between -1 and 1. The used activation functions for the outputs are *tanh* for the mean $\boldsymbol{\mu} \in [-1, 1]$ and *softplus* [223] for the standard deviation $\boldsymbol{\sigma} \in [0, \infty)$.

While the glimpse network is using the raw sampled coordinates $\boldsymbol{x}_g \in [-1, 1]$, they have to be transformed from the given coordinate system to the corresponding pixels of the input image (I_x, I_y) for the creation of a new glimpse. The transformation to the coordinate system of the image with the origin in the bottom left corner can then be done using

$$x_i = \frac{1 + x_g \cdot \eta}{2} \cdot I_x$$

and for y_g respectively. Additionally, a factor η is introduced that defines the pixel range that the location policy is able to cover and suppresses the output of locations at the image border.

Context network The context network, illustrated in Figure 5.4, is introduced in [218, 221] to help the model to decide on the best location of the first manually

[1]In [148] and [218] only the mean is learned, while the standard deviation is set to a fixed value.

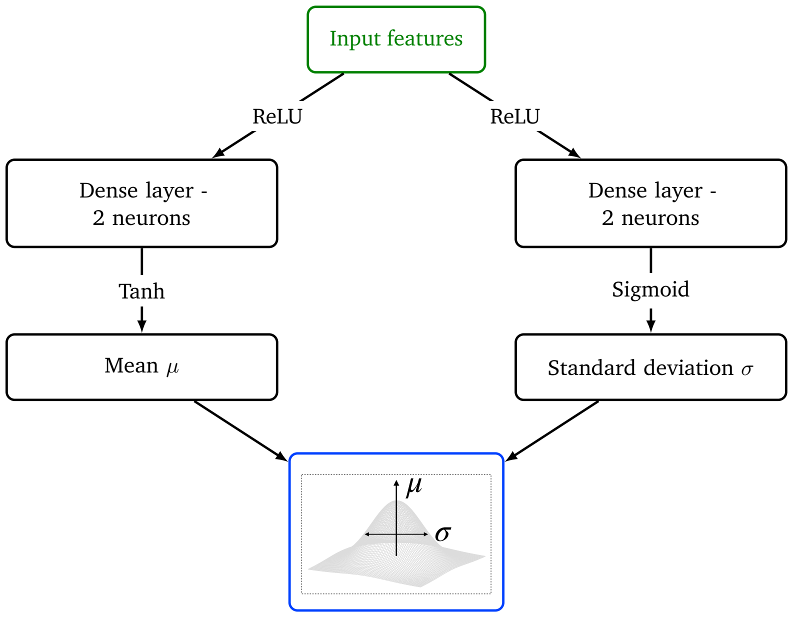


Fig. 5.3.: A general illustration of the location network.

created glimpse[2]. Its input is a low resolution image of the whole scenery that, in this proposed approach, is propagated through a linear layer with 64 neurons in order to generate suitable features. These features are then taken as the initial states of *LSTM 3*, whose output is only used to generate the new glimpse locations through the location network. The other LSTMs are always initialized with zeros.

Actor-critic network The actor-critic network, illustrated in Figure 5.5, is built out of a state value network that estimates the state value for the current state and a policy network generating the actual action policy. The policy network is connected to *LSTM 2* as illustrated in Figure 5.1. The output of this LSTM network is the input of a linear layer, which then outputs the probabilistic action policy $\pi(\boldsymbol{s}_t, \Theta)$ using the *softmax* activation function. The number of neurons is defined by the number of possible actions $|\mathcal{A}|$ the agent has as its disposal. The output of the same LSTM network is also used to approximate the state-value function $\hat{V}(\boldsymbol{s}_t, \Theta)$ which plays an important role during the update process of the designed model.

[2]The very first glimpse of each step is always random.

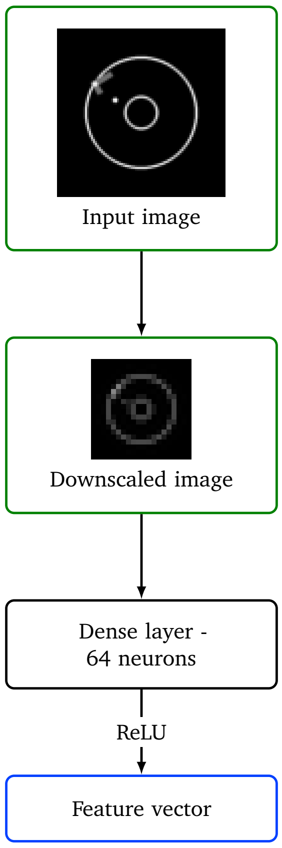


Fig. 5.4.: An illustration of the context network that is used to initialize the internal states of *LSTM 3*.

5.1.1 Training

Both the action and the location policy can be trained using the REINFORCE algorithm (2.15). For the action policy, the characteristic eligibility from (2.17) can be used. During the current episode, the transition tuples $(\boldsymbol{s}_t, a_t, r_t, V_t)$ are saved into a worker dependent experience buffer $\mathcal{E}$ of size N_{EB}. Here, $\boldsymbol{s}_t$ corresponds to the full image of the environment. After the end of an episode or if the episode buffer is full, the corresponding worker computes the weight update using the sequence of stored tuples and clears the buffer thereafter. To compute the gradient for one transition tuple, the network receives the recorded image $\boldsymbol{s}_t$ as its input, which is then used for a full forward-pass through the network, including the creation of the glimpses $\boldsymbol{g}$ and its corresponding locations $\boldsymbol{x}_g$, in order to get the action policy $\pi(\boldsymbol{s}_t, \Theta)$ and the approximated state value $\hat{V}(\boldsymbol{s}_t, \Theta)$.

For the training of the stochastic location policy, the output layer for the mean $\boldsymbol{\mu}$ is trained using the REINFORCE algorithm with the characteristic eligibility ζ_μ defined in (2.17), while the standard deviation $\boldsymbol{\sigma}$ is trained using the characteristic

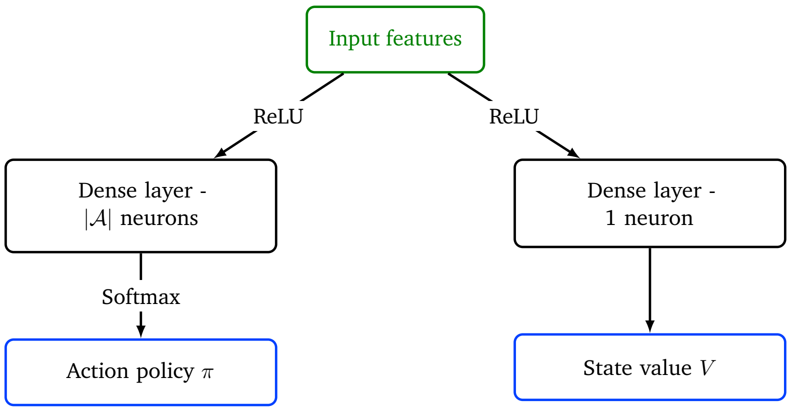


Fig. 5.5.: An illustration of the actor-critic network.

eligibility ζ_σ that is given in (2.18). It is worth mentioning that the gradient which is propagating through the location network is stopped after *LSTM 3* during the update process. A sequence of S glimpses

$$\boldsymbol{g}_1 \rightarrow \boldsymbol{g}_{1:2} \rightarrow \cdots \rightarrow \boldsymbol{g}_{1:S-1} \rightarrow \boldsymbol{g}_{1:S}$$

is constructed by starting with the first glimpse $\boldsymbol{g}_1$ and then taking the next glimpse based on the information of the previous ones. Instead of training the network only on the locations of the last glimpse $\boldsymbol{g}_{1:S}$ that was generated in a sequence of length S, the training can be improved by also using all included sub-sequences $\boldsymbol{g}_{1:s}$ with $s \leq S$ [221]. Thus, the characteristic eligibilities $\zeta_{\mu_{1:s}}$ are computed for all sub-sequences, while the sum is used for updating the weights as

$$\zeta_\mu = \sum_{s=1}^{S} \zeta_{\mu_{1:s}} = \sum_{s=1}^{S} \frac{(x_s - \mu_s)}{\sigma^2}.$$

for the mean and for the standard deviation respectively.

For training, a hybrid-loss [218, 221] is used that includes the objective functions for the action and location policy, plus the mean-squared error between $\hat{V}_t$ and the current estimation of R_t, based on the tuples in the experience buffer $\mathcal{E}$. The full objective function then becomes

$$\mathcal{L} = \alpha \cdot \Bigg[A_t(\mathcal{E}, \hat{V}(\boldsymbol{s}_{t+n}; \Theta)) \cdot [\zeta_\pi(\boldsymbol{s}_t; \Theta) + \beta \cdot \zeta_\mu(\boldsymbol{s}_t; \Theta) + \beta \cdot \zeta_\sigma(\boldsymbol{s}_t; \Theta)] + \left(\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \hat{V}(\boldsymbol{s}_{t+n}; \Theta) - \hat{V}(\boldsymbol{s}_t, \Theta) \right)^2 \Bigg], \quad (5.1)$$

where $n-1$ is the size of the current experience buffer. If the episode is terminated, the total reward R_t can be computed for each visited state. For the case of a full episode buffer, the bootstrapped state-value $\hat{V}(\boldsymbol{s}_{t+n};\Theta)$ is used to estimate the total reward R_t, as displayed in (5.1). The advantage function A_t is approximated via *generalized advantage estimation* [224] using the rewards and state-values from the experience buffer $\mathcal{E}$.

The scalar β is weighting the contribution of the computed errors derived from the location and action policy to the update. While for $\beta = 1$ both errors contribute equally to the weight update, a smaller factor of $\beta < 1$ assigns more resources to the action policy part, for $\beta = 0$ the location policy part is completely omitted [221].

For a practical implementation, the presented pseudocode for the A3C model, listed in Algorithm 6 in Appendix A, can mostly be reused. The only changes that have to be made are to use an update rule for the network's weights that is based on the new objective function (5.1) and to implement the computation of the advantage function A_t.

5.2 Summary

In this part of the thesis, a scaffold was developed that intends to support the learning by integrating an adaptable attention guided visual input. The designed *recurrent attention asynchronous advantage actor-critic model* combines the *asynchronous advantage actor-critic model* with the *recurrent model of visual attention*. The designed framework has the ability to learn an active generation of fovea-like glimpse-sequences as an input instead of the high-dimensional image of the whole state of the environment. In addition, the asynchronous actor-critic part, together with the reduced size of the input, has the capability to rapidly learn to solve complex problems.

The next step Compared to vision, the tactile capabilities of robots are currently hardly developed, which makes it difficult to generate learning models where an artificial agent has to solve a task, based on tactile data as its only sensory input. The next section presents a scaffolding approach that incorporates "active haptic perception" into the learning process. Similar to the current chapter, the idea is to exploit a recurrent attention model in order to endow the agent with the ability to actively learn to generate an attention guided input based on tactile sensory data.

6 Scaffolding the learning of efficient haptic exploration using "active haptic perception"

Although the sense of touch is existential to human life, tactile capabilities of robots are currently hardly developed. This stark contrast becomes even more apparent if one compares touch and vision: while good camera sensors have become affordable and ubiquitous items and huge image and video databases together with deep learning have brought computer vision close (some would argue on par) to human vision [43, 225, 226], comparable advances in robot touch are widely lacking [227, 228]. One reason is the very limited maturity of tactile sensors if compared to human skin. A second and more significant reason is that touch differs from vision in an important way: while looking at an object leaves its state unaffected, touch requires physical contact, strongly coupling the sensor and the object in potentially complex ways that usually also change the position, orientation or even the shape of the object. Human haptics makes active and sophisticated use of this strong coupling to lend us skills such as haptic exploration, discrimination, manipulation and more. Large parts of these tasks are hard or impossible to model sufficiently accurately to replicate them on robots, thereby calling again for machine learning approaches similar to those that were highly successful in vision. However, the highly interactive nature of touch makes not only the learning problem itself much more difficult but also creates a problem for the availability of meaningful training data, since information about interactive haptics is much harder to capture in databases of static tactile patterns. As a consequence, learning approaches for the modality of interactive touch are still largely in their infancy and tactile skills enabling robots to establish and control rich and safe contact with objects or even humans are still a largely unsolved challenge which severely limits the use of robots in both domestic and industrial applications. Recent related work [229] presenting an approach for learning of contact-rich tasks with a robot given a goal specification in the space of tactile measurements, highlights the desirability of tactile sensing for robotic control during manipulation and also emphasizes the difficulties connected with the absence of physics simulation for modeling of contact.

6.1 From human haptic perception to robotics

In human perception, the visual information is not acquired at once, but is accumulated from details that are recognized in small zones via the central fovea. In order to gather these pieces, the observer has to learn to pay attention to the important salient points of the scene and then gain the information via an active process of looking. As **visual perception is a touch-like process** [136], recent research indicates the presence of similar limitations in the tactile modality [230]. In this work, the integration of attention mechanisms is seen as an efficient way to scaffold the perceptual process within a learning task. As a result, the idea is extended from the visual to the haptic domain. The presented approach (see also [24]) is again based on the *recurrent model of visual attention*. By taking inspiration from insights about the organization of haptic exploration in humans, it is used to create a potentially interesting new bridge between a computational understanding of interactive touch in robotics and in human haptics. The focus of this section is thus to **design a learning scaffold for the synthesis of one central and important haptic skill**: the discrimination of unknown object shapes through a sequence of actively controlled haptic contacts between a sensor and the object surface.

Exploratory procedures & haptic glances In humans, haptic capabilities are available from birth, for example those that are necessary for a neonate to suckle. During early development, children acquire motor control and the ability to focus their attention which increasingly develops sophisticated haptic exploration. By pre-school age, children demonstrate adult-like patterns of exploration [231] that they gate according to contextual demands [232]. This developmental process results in a small set of optimized action patterns, widely known under the term *exploratory procedures* (EPs) [233]. Robots, like humans, benefit from haptic sensors in order to find, identify, and manipulate objects.

Humans use a small set of exploratory procedures to extract properties such as texture, hardness, weight, or volume. Under some circumstances, the level of complexity in haptic exploration can be effectively reduced to what was termed the *haptic glance* by [234]. Specifically, Klatzky and Ledermann define the haptic glance as brief, spatially constrained contact that involves little or no movement of the fingers. In the same work they pose the question how the information from a haptic glance is translated into effective manipulation.

Haptic glances in robotics In order to utilize the concept of haptic glances in the field of robotics, they are proposed as atomic, primitive exploratory entities within this work. Exploratory procedures can be thus viewed as a sequence of such primitives. For computational purposes the following assumptions were made:

- A haptic glance - being the simplest haptically directed action - is a foundation for any more complex haptic behaviour, including haptic exploratory procedures of any type.
- It is assumed that a haptic glance is defined by a tuple consisting of a pressure profile yielded by the tactile sensor at contact and the associated sensor pose.

Of course, this is not the only way to define haptic glances. In [235], haptic glances are interpreted as static images — also termed tactile imprints — and are calculated as normals to the surface at heuristically estimated points of interests. They can also be parameterized by a target value lying in the tactile space, similar to [229], or specified by the magnitude of applied force, in case the contact has been previously established.

6.2 The haptic attention model

Using the definitions of exploratory procedures and haptic glances, the idea of the current approach can be summarized in an intermediate research question:

> Is it possible to scaffold the learning process of haptic problems by using *recurrent attention* in order to learn synthetical exploratory procedures for robots using optimization of motor control?

Therefore, the *recurrent model of "visual" attention* [148, 218] that was previously discussed in Chapter 5 is transferred to the tactile domain as illustrated in Figure 6.1. The design of the resulting *haptic attention model* (HAM) is — like the original RAM — designed as a classifier. But instead of classifying images, the HAM has to discriminate different objects using a sequence of tactile sensory data. While in the visual domain an image has to be classified using the accumulated information of visual glimpses, the image is replaced by a to-be-classified object and the glimpses by *haptic glances*. These glances can be presented by a vector $\boldsymbol{s} = (\boldsymbol{l}, \boldsymbol{p})^\top$ consisting of the pose $\boldsymbol{l}$ of the used tactile sensor and the associated pressure profile $\boldsymbol{p}$ that is measured by the tactile sensor. The full target pose of a sensor can be described by a vector $\boldsymbol{l} = (x_g, y_g, z_g, e_1, e_2, e_3)$ that is constructed out of three variables x_g, y_g and z_g that are defining its position in the global coordinate frame and three Euler angles (e_1, e_2, e_3), defining its orientation. Depending on the experimental setup and the designed learning task, the HAM can be designed to either control all six variables for generating a new haptic glance or only a subset. The HAM that is presented in the following text is designed to modify only two of the six parameters: the position along the x-axis (x_g) and the angle around the y-axis (e_2). In this configuration, the remaining variables y_g, e_1 and e_3 are now either staying constant or are controlled externally. For the sake of readability, the alterable position x_g is called x and the angle e_2 is called φ in the following text. Thus, each haptic glance can be represented by a pose (x, φ) of the tactile sensor. Like an image, which is built out of a 2-dimensional pixel-matrix, many sensors are also able to measure the pressure profile of a 2-dimensional area. This $p_x \times p_y$ pressure matrix is then flattened to a normalized pressure vector $\boldsymbol{p}$.

Tactile network First, the input is processed through a *tactile network* which is arranged like the *glimpse network* of Section 5.1 but built out of linear layers with just 64 neurons each. Instead of processing glimpses and their respective locations, it combines the recorded pressure profile $\boldsymbol{p}$ with its associated pose (x, φ) into one single feature vector.

Memory network The features are then propagated through the *memory network*. It consists of one single LSTM network with hidden state of 256 neurons. The LSTM provides features to the *object classifier* and to the *location network* that in turn provides a new pose.

Location network The *location network* uses the feature vector that is computed by the memory network for generating a new location-orientation pair that is then used for the next haptic glance. Except smaller linear layers with only 64 neurons, it has the same structure as the one presented within Section 5.1. Therefore, Gaussian distributions are used for sampling the position x and orientation φ respectively by adapting the mean μ and standard deviation σ. For computing the four necessary variables μ_x, μ_φ, σ_x and σ_φ, the features of the LSTM are propagated through an independent linear layer with 64 neurons for each of them. The means are sampled within the range $\mu \in [-1, 1]$ using *tanh* as the activation function. For the standard deviations $\sigma \in [0, 1]$ the *sigmoid* function is employed. To ensure that the location and position of the sensor is valid, the sampled values of the Gaussians are restricted to the

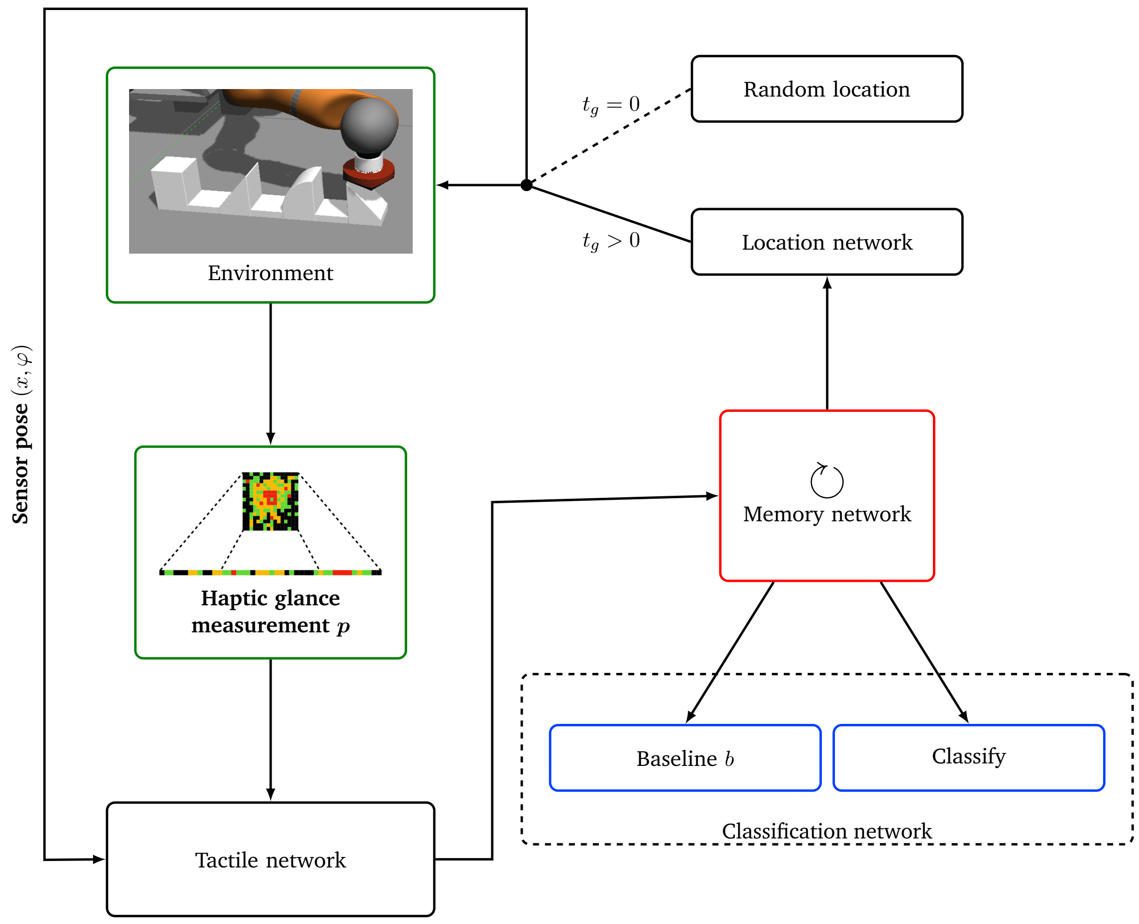


Fig. 6.1.: Illustration of the overall network architecture for the *haptic attention model*. The figure illustrates the overall design of the multi-module meta-controller model and its interaction with a simulation environment.

range $[-1, 1]$. Thus, if the sampled x or φ is sampled outside this range, it is resampled using the same mean and standard deviation.

Classification network In order to classify a given object, the generated feature vector of the LSTM is not only transferred to the location network, but also propagated through a different linear layer that is then used for classification. Therefore, the output is processed through a *softmax* function in order to encode the predicted class-affiliation of the current object in a probability density $\pi(o'|\boldsymbol{g}_{1:s}; \theta_t)$, representing the current policy of the reinforcement learning agent. Here, $\boldsymbol{g}_{1:S}(\theta_t)$ encodes the accumulated LSTM feature vector after S glimpses, using the current set of weights θ_t at training step t. For classification, the class o with the highest probability

$$o = \underset{o'}{\operatorname{argmax}}\, \pi(o' | \boldsymbol{g}_{1:S}; \theta_t)$$

is taken as the prediction.

6.2.1 Training

One training step is given through the sequence of a pre-defined number of haptic glances, followed by a classification attempt. The designed model can be seen as a reinforcement learner which has to choose the right action in order to classify the given object. For classifying the object correctly, it receives either a reward of $r = 1$ or $r = 0$. The predicted probability of correctly identifying the target object o after S glances is then given as $\pi(o|\boldsymbol{\tau}_{1:S}; \theta)$. To this end, the *categorical cross-entropy* can be used to compute the loss. The classifier and the location network are then trained together using an update rule (6.1) derived from a hybrid loss that combines eligibility-weighted updates from the *REINFORCE algorithm* (2.15) with a *cross-entropy* misclassification penalty. The mean μ, and standard deviation σ of the sampled position can directly be seen as the adaptable variables of the location network instead of its weights θ. Thus, for learning the mean μ of the location component of the policy, the characteristic eligibility outlined in (2.17) is used. The standard deviation σ is learned by applying (2.18).

Including a cross-entropy based penalty of misclassification then leads to

$$\Delta_\theta = -\alpha \cdot \left[\beta \cdot (r_t - b_t) \cdot (\zeta_\mu + \zeta_\sigma) + \sum_{o=0}^{O} \log(\pi_c(o)) \cdot y_o \right]. \tag{6.1}$$

The function $\pi_c(o)$ gives the computed classification probability that the target object is object o, while y_o is 1 if o corresponds to the correct object and 0 otherwise. ζ_μ and ζ_σ are the characteristic eligibilities, rating the generated mean and standard deviations. The value β is weighting the contribution of the location policy within the weight update as it was explained for the objective function of the RAA3C model in Section 5.1.1.

The baseline layer is updated separately using the mean-squared error. Instead of training the baseline b only on the accumulated tactile information of the last glance

$\boldsymbol{\tau}_{1:S}$, the training can be improved by also using all included sub-sequences $\boldsymbol{\tau}_{1:s}$ with $s \leq S$ [221]. This leads to the update rule

$$\Delta_{\theta_b} = \sum_{s=1}^{S} \left[r_t - b(\boldsymbol{\tau}_{1:s}; \theta_b) \right]^2 . \tag{6.2}$$

6.3 Summary

The presented approach extends the *recurrent model of visual attention* to the haptic domain for enabling artificial agents, like robots, to learn to identify different objects not by vision but touch. Inspired by the fact that humans are haptically exploring their environment by exploiting various kinds of "exploratory procedures", the *recurrent model of "haptic" attention* is designed as a **scaffold to endow the agent with the ability to learn its own exploratory procedures** using a pre-defined number of atomic interaction primitives, called *haptic glances*.

The next step In the last three chapters, the idea of the *perception-action cycle* was utilized for creating different approaches that use a permanent scaffolding of the perception process for two families of reinforcement learning problems, where the agent either interacts with the environment in order to solve a specific problem or has to classify objects while relying only on tactile information. After designing scaffolds with the aim to improve its perceptual skills, the next chapter puts the learning process itself into focus. By carefully shaping the agent's learning behaviour through the generation of a suitable sequence of sub-tasks, a temporal scaffold can be constructed that has the ability to not only speed up the learning process but also to significantly improve the learning performance.

7 Scaffolding the agent's internal representation through skill transfer

Transfer learning has been shown to be very effective to initialize a reinforcement learner from the results of previous learning of simpler *source tasks* which share structure with or are suited as building blocks for the original *target task* of an artificial learner [161, 162, 168]. Figure 3.5 of Section 3.3.6 displays a mind map that relates machine learning methods and the key aspects of scaffolding to transfer learning and curriculum learning. Both methods also had a special position within the presented approaches. They could not only be affiliated with *fading & transfer of responsibility* but also to a second characteristic. While *transfer learning* could also be related to *simplification*, it was possible to connect curriculum learning to *ongoing diagnosis & assessment*. As curriculum learning was also utilized to extend transfer learning [162, 192, 194], it is a promising combination for designing a scaffold for facilitating the learning process of already established learning models — like existing (deep) reinforcement architectures — for mastering complex learning tasks. One special aspect that distinguished both approaches from many others is that the basic principle of both transfer and curriculum learning is the efficient manipulation of training data and exploitation of existing experience of the artificial learner. They can thus be combined with existing methods without the need to modify them.

Linear and also deep reinforcement learning architectures, for example lack the ability to efficiently learn these kinds of hierarchical scenarios to a satisfying level. Therefore, many approaches propose special learning models that have the ability to learn "temporal abstraction" for splitting problems into a hierarchy of simpler sub-tasks (see again Section 3.3.2). Alternatively, in this chapter the hypothesis is put forward that the pre-learning of a carefully chosen selection of simpler sub-goals before starting to learn the main problem can be used for improving non-hierarchical learning models and thus lead to better and faster learning. The resulting approach can thus be seen as a **self-regulating module that supports existing learning architectures by improving its internal representation**. An important aspect in this connection might be the native "gluing" of *simplification* and *ongoing diagnosis*

& assessment to *fading & transfer of responsibility* by these two techniques. It is therefore worth focussing especially on their integration into the scaffolding and also the studying of their combined effect on learning, i.e.

> Can a scaffold, based on the combined approach of transfer and curriculum learning, augment the learning process of established learning models? How is the integration of key characteristics of the refined concept of scaffolding influencing the learning process?

Outline At first, the main idea of the designed scaffold is explained. In the next part, the core ideas of the strategies are presented, together with some necessary terminology and methods for representing and generating learning tasks.

7.1 The combination of a structured curriculum with transfer learning — 4 strategies of skill transfer

While most transfer learning schemes don't pay much attention to the fashion in which the source tasks are provided, they can be combined with *curriculum learning* [162, 189, 192]. Although curriculum learning is a promising attempt, an important question in this regard is how to choose and how to structure the training data for enabling an improved learning process. In the last years, various techniques for automating this kind of process were proposed [190–194]. They vary from formulating curriculum sequencing as a Markov Decision Process [192] and the learning of a curriculum policy [194] to adapting a curriculum by taking the rarity of the occurring events into account [193].

Main idea In this context, the current chapter also presents a newly invented method for the automatic generation of a curriculum. This one, however, is not learned but is adapting by utilizing the concept of *scaffolding* as it is described in Section 3.4. Through the modeling of a **temporary supporting** strategy of choosing source tasks such that the **level of difficulty of each new source task increases with the proficiency of the learner**, it always stays within what psychologists denote as the learner's *"zone of proximal development"* [102, 103] (see also again Figure 3.1 in Section 3.1) and is thus able to speed up the learning process. To this end, a sequence of four strategies is presented, illustrated in Figure 7.1, that should facilitate the learning and the transfer of sub-skills. Beginning with the key characteristic of *simplification*, every following strategy is gradually integrating a new aspect for choosing suitable to-be-learned source tasks that are related to the sub-goals. These additional aspects are *fading*, *transfer of responsibility* and *ongoing diagnosis & assessment*. By applying all strategies to the same learning setup, it is possible to get insights into the individual contribution of the different characteristics. Although, one usually aims for the most efficient approach, this one is intentionally split into four strategies. Doing that allows analysing how important the different aspects are for facilitating the learning of an artificial agent and to get hints for questions like: Is fading a necessary aspect for this kind of scaffold? How important is the ongoing diagnosis of the agent's current skill level?

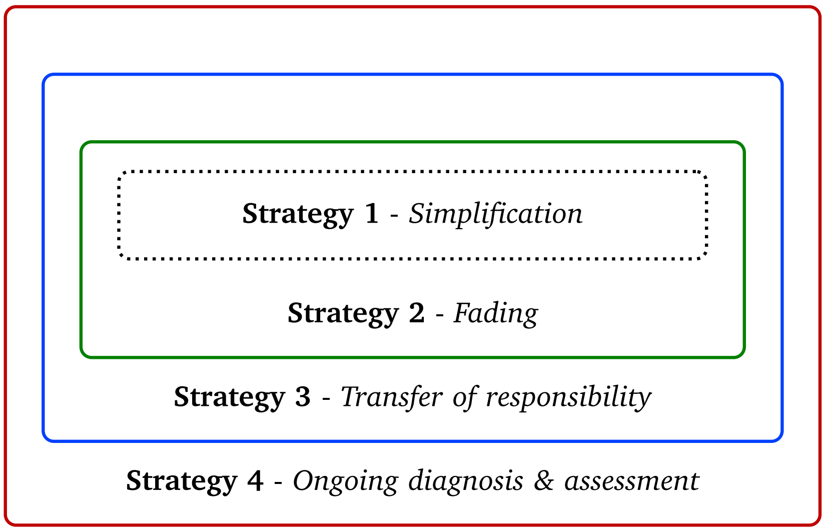

Strategy 1 - *Simplification*
Strategy 2 - *Fading*
Strategy 3 - *Transfer of responsibility*
Strategy 4 - *Ongoing diagnosis & assessment*

Fig. 7.1.: The four different strategies are built upon another while gradually adding more features of the "psychological concept of scaffolding" into the process of generating and assigning source-tasks.

All strategies assume a simple and generic structure of the learning domain:

1. Learning should occur in a sequence of episodes.

   a) Each episode is focused on a single task instance, attempting to solve it in a number of consecutive time-steps (number of actions) and terminating after success or when a step limit is exceeded.

   b) After termination, another episode starts.

2. It is assumed that source and target tasks are from the same domain, so that both can be handled by the same learning process.

The strategies are also motivated from a typical characteristic that can be found in many different tasks: reaching the goal requires to suitably concatenate a number of sub-skills. One particular example is given by mediated control tasks, i.e. tasks that involve some kinds of tools (also called mediator objects) in order to control a second target object. In this tasks, each sub-skill refers to the target object or to one of the (single or more) mediator objects. To solve such kinds of tasks, the agent has to learn these sub-skills and bring them together in a suitable sequence. For complex tasks, each sub-skill may itself require a decomposition, requiring the agent to organize an entire hierarchy of sub-skills. This existence of sub-skills can be exploited by pre-learning the most essential low-level ones, for example the identification and manipulation of the target object as well as the recognition of the goal conditions. Then, instead of learning source tasks up to perfection, the agent only learns them a small number of times and uses the imperfectly learned sub-skills for a "lightweight initialization" of the learning process of the target task. This also prevents specialization on the source tasks to interfere with the learning of the target task. In the target task the agent should learn, based on the knowledge it has achieved in the source task, more intermediate skills like how to use objects as tools or object-object interaction.

7.1.1 Strategy 1 & 2: simple techniques for skill transfer

Strategy 1 Formalized in Algorithm 1, *Strategy 1* builds the core concept of the presented approach by using transfer learning as a temporary support in order to facilitate the learning process of a target task using random generated source tasks that have to be learned within the same domain before the main task. It uses only skill transfer based on prior learning of a number of randomly selected, easier source tasks before switching to learning instances of the target task and is thus a *simplification approach*. It has the number of to-be-solved source tasks N_{source} as its only tunable input parameter. During the learning process, a random source task is generated at the beginning of each episode. After a given set of to-be-learned source task instances has been solved, the learning is switched from source task to target task learning without stopping the learning process.

Strategy 2 In the first strategy, a new source-task is always generated if the agent has failed to solve the current one. As the source tasks are generated at random, they might not be presented in a sequence that fosters the learning of the agent. It could for example lead to situations where the learner has to solve difficult tasks too early or easy tasks within the late phase of the learning process. The first stage to suppress this kind of sequences is given by *Strategy 2*. The solvable source tasks

Algorithm 1: Skill transfer — *Strategy 1*

```
Data: $N_{\text{Source}}$ - Number of uniformly sampled to-be-learned source task
      configurations
for learning episode = 1, M do
    if $N_{Source} \neq \emptyset$ then
        Random sample task instance from $C$
        start learning episode
        if Agent solves task then
            $N_{\text{Source}} = N_{\text{Source}} - 1$
        end
    else
        start episode and learn the target task
    end
end
```

c can be merged to a set $\mathcal{C}$ that is provided for the source learning stage. The number of to-be-solved source tasks, i.e. the number of elements in $\mathcal{C}$ is given by N_{source}. Additionally $\mathcal{C}_{solved} \subseteq \mathcal{C}$ can be defined. $\mathcal{C}_{solved}$ is a subset of $\mathcal{C}$, containing the source tasks that already have been solved at a particular time step. Using the set $\mathcal{C}$ of source tasks, *Strategy 1* can be refined by generating a fixed set of randomly generated source tasks. Now a source task is picked randomly from $\mathcal{C}$ at the beginning of the episode during the first stage of the learning process. If the agent was able to solve the given task, it is removed from $\mathcal{C}$. When all source tasks are solved, i.e. $\mathcal{C} = \emptyset$, the agent starts to learn to solve the target tasks as in *Strategy 1*. By generating a set using a fixed number of pre-sampled source tasks, a mechanism that loosely resembles a *fading support* is embedded into the approach. At first the agent is likely to solve the easy source tasks that are eventually removed from $\mathcal{C}$, leaving only the more difficult ones that the agent is now forced to learn.

Algorithm 2: Skill transfer — *Strategy 2* & *Strategy 3*

```
Data: Set $\mathcal{C}$ of $N_{\text{source}}$ source tasks
for learning episode = 1, M do
    if $\mathcal{C} \neq \emptyset$ then
        Random sample task instance from $\mathcal{C}$
        start learning episode
        if Agent solves task then
            remove task from $\mathcal{C}$
        end
    else
        start episode and learn the target task
    end
end
```

7.1.2 Strategy 3 & 4: refining skill transfer by analysing the learners way of perception

The importance of the perception of the artificial learner for the learning process was elaborately discussed in Section 3.3.1 and has also inspired the previously presented scaffolding approaches (see Chapter 4, 5 and 6). In this approach, the agent's way of perception is again incorporated. This time, however, it is not directly influenced during the learning process. Instead, it is used to create the to-be-learned source-tasks of the agent not with respect to their physical characteristics but the agent's sensory information. In this way, the agent might be able to explore many more new sensory states instead of perceiving only agglomerated sensory states within certain regions. It was already mentioned in Section 4.2 that artificial and biological agents perceive the world by utilizing special sensors which are for example eyes, nose or ears for biological agents and cameras, laser scanner or pressure sensors for robots. The sensors can then be seen as a function $F : \mathcal{W} \longrightarrow \mathcal{S}$ that maps the full information of the environment $\mathcal{W}$ (the world space) to the sensor space $\mathcal{S}$ that is defined by the kind of used sensor (see again Figure 4.2). In addition to the sensor space, a *task space* $\mathcal{T} \subseteq \mathcal{W}$ can be defined as the space of all possible configurations of to-be-solved tasks $\boldsymbol{t} \in \mathcal{T}$ for the given learning scenario. Yet, the task space differs in most cases from the sensor space which is used as an input for the learning algorithm. In the worst case, the mapping from the task space to the sensor space is also not invertible. This possible decorrelation of task and sensor space can have a huge impact on the learning process as small deviations within the task space can have completely different effects within the sensor space. Consequently, the agent's modalities of perception, encoded in the sensor space, have to be taken into consideration during the structuring of the learning process.

Under these assumptions, each task (irrespective of being a source or target task) can be represented by a pair $\boldsymbol{c} = (\boldsymbol{s}, \boldsymbol{t})$, where $\boldsymbol{s}$ represents the task in the sensor space of the agent, and $\boldsymbol{t}$ represents the task in some task space coordinates (which usually differ from the sensor coordinates).

Modelling the sensor space The aim is now to minimize the number of required source tasks N_{source} while maintaining a uniform distribution over the relevant part of the sensor space. One way to realize this is through a Voronoi tesselation of the sensor space (as it was done in Section 4.2). The difference to the earlier approach is that now a large number of random source tasks $\boldsymbol{c}_r = (\boldsymbol{s}, \boldsymbol{t})$ is generated within the task space. The next step is to cluster these tasks $\boldsymbol{c}_r$ within the sensor space using $\boldsymbol{s} \in \mathcal{S}$. This can again be achieved with for example the K-Means algorithm

[198–200] and leads to a desired number[1] of clusters. For the prototype within each cluster, the *task configuration* $\boldsymbol{c}_r$ *with the most similar representation* $\boldsymbol{s}$ *in the sensor space* is chosen to be in the task set $\mathcal{C}$. These N_{source} source-task configurations that are now defined as $\boldsymbol{c}(\boldsymbol{t}, \boldsymbol{s})$, are the ones that are actually used within the learning process. The whole process of generating the set $\mathcal{C}$ is illustrated in Figure 7.2.

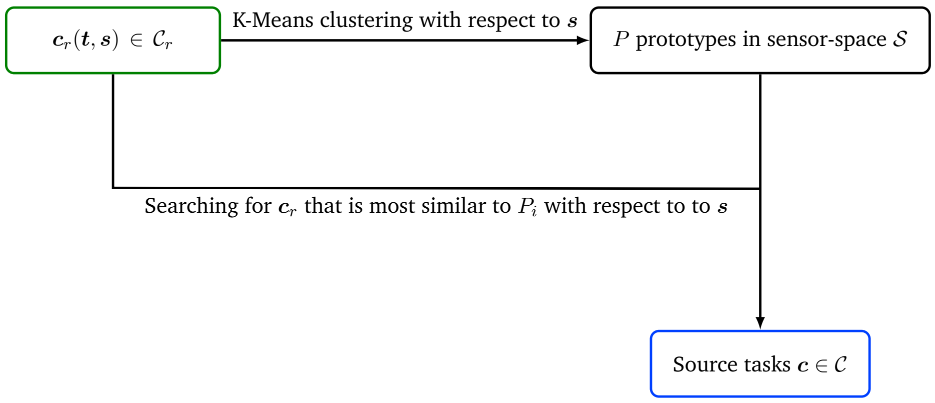

$\boldsymbol{c}_r(\boldsymbol{t}, \boldsymbol{s}) \in \mathcal{C}_r$
K-Means clustering with respect to $\boldsymbol{s}$
P prototypes in sensor-space $\mathcal{S}$
Searching for $\boldsymbol{c}_r$ that is most similar to P_i with respect to to $\boldsymbol{s}$
Source tasks $\boldsymbol{c} \in \mathcal{C}$

Fig. 7.2.: Illustration of process for generating a structured set of source-task configurations $\mathcal{C}$.

Similarity metrics A good measure of similarity is a key ingredient for scaffolding strategies. To efficiently structure the learning process, the pending tasks have to be conveniently compared to find the one which corresponds best with the agent's skills at that time. One reasonable choice is to measure the correlation between different tasks $\boldsymbol{s}$ in the sensor space by utilizing the cosine similarity

$$\text{CosSim}(u, v) = \frac{(\boldsymbol{u} \cdot \boldsymbol{v})}{||\boldsymbol{u}||_2 ||\boldsymbol{v}||_2}. \tag{7.1}$$

The best way to determine the correlation of the tasks heavily depends on the structure of their embedded space and the processing afterwards. Thus the similarity of tasks is not only influenced by their own composition but also by the characteristics of the used representation and learning algorithm.

For detecting the kind of measurement that is most beneficial for the given task, the literature states numerous different approaches. They range from taking the negative squared euclidean distance of two vectors [236] to biologically inspired models [237]. So, it is not always a good choice to use equation (7.1) for measuring the similarity between the different task configurations within the sensor space.

[1]For N_{source} source-tasks, the same number of clusters has to be generated.

A notable alternative to the cosine similarity is based on the "Pearson correlation coefficient" [238] and defined as

$$\mathrm{Corr}(\boldsymbol{u}, \boldsymbol{v}) = \frac{(\boldsymbol{u} - \bar{u}) \cdot (\boldsymbol{v} - \bar{v})}{||(\boldsymbol{u} - \bar{u})||_2 ||(\boldsymbol{v} - \bar{v})||_2}. \tag{7.2}$$

This coefficient lies in between -1 and 1 and measures, like (7.1), the linear correlation between two data points, but is also invariant to changes in location and scale of the two vectors.

Strategy 3 The idea to create a set $\mathcal{C}$ of source tasks, where the elements are not generated randomly but in a way that tries to uniformly cover the whole space of possible source tasks is realised in *Strategy 3*. It provides a much more thoughtful distribution of source-tasks than it was used in *Strategy 1* and *2* by using prototypes that should be distributed more uniformly over the sensor-space. As a consequence, it adds the feature of "smoothing out" the transition process from easy source-tasks to hard source-tasks. It can be seen as a *transfer of responsibility* by homogeneously selecting the to-be-learned source-task over the full range of states within the learning world that the agent is able to perceive.

Strategy 4 Scaffolding claims that the time-course of learning is accelerated when new learning instances are not selected at random, but instead are matched to the learner's proficiency at that time. This requires a new example to be neither too easy nor too difficult for the learner. This can be achieved through a *"similarity-similarity based heuristic"*: choosing each new learning example to be similar to one of the instances that the learner has solved previously. This process of *ongoing diagnosis & assessment* is implemented within the last *Strategy 4* in order to help the agent to stay in its "optimal zone of proximal development". Formalized in Algorithm 3, it complements *Strategy 3* with an approach that also replaces the random selection of the source tasks during the learning process with a structured selection that restricts the choice of new source tasks to a vicinity of the set of already solved source tasks. This vicinity is defined with the help of a suitable similarity metric (as presented in the last subsection) in the source domain. Therefore the next to-be-solved source task is selected by minimizing the distance to a solved source task, according to the chosen similarity metric. Similar to *Strategy 3*, its main input parameter is again a set C of to-be-solved source tasks before switching to learning the target task. Once more it is important to mention that the similarity of two source tasks

$\boldsymbol{c}_i(\boldsymbol{t}, \boldsymbol{s})$ and $\boldsymbol{c}_j(\boldsymbol{t}, \boldsymbol{s})$ is not measured using their configuration in tasks space, but in sensor-space.

```
Algorithm 3: Skill transfer — Strategy 4
Data: Set $\mathcal{C}$ of $N_{\text{source}}$ clustered source tasks
for learning episode = 1, M do
    if $\mathcal{C} \neq \emptyset$ then
        if $\mathcal{C}_{solved} = \emptyset$ then
            Sample task $\boldsymbol{c}$ from $\mathcal{C}$
        else
            Sample random task $\boldsymbol{k} \in \mathcal{C}_{\text{solved}}$
            Among unsolved source tasks $\mathcal{C}$:
            find task $\boldsymbol{c} \in \mathcal{C}$ that is similar to $\boldsymbol{k}$
        end
    else
        Sample random task $\boldsymbol{c}$ for the target task
    end
    start episode for learning $\boldsymbol{c}$
    if Agent solves task $\boldsymbol{c}$ and $\mathcal{C} \neq \emptyset$ then
        add $\boldsymbol{c}$ to $\mathcal{C}_{\text{solved}}$
        remove $\boldsymbol{c}$ from $\mathcal{C}$
    end
end
```

7.2 Summary

This chapter has presented an approach for speeding up the learning process of an artificial agent by combining transfer and curriculum learning on the one hand and the integration of key aspects of scaffolding like “fading”, “transfer of responsibility” and “ongoing diagnosis & assessment” into the learning process on the other hand. The proposed method aims to improve the internal representation of non-hierarchical learning models by shaping it through a sequence of pre-learned source tasks whose difficulty is aligned to the capabilities of the artificial learner at that time. Two special aspects of this method are that the learning of source tasks is performed together with the learning of the target task within one single learning run while the curriculum for learning the source tasks is inspired by some of the developed features of *scaffolding artificial agents* (see again Figure 3.5). To study the influence of the individual characteristics, four different learning strategies were designed. The strategies are all using the same main concept, while gradually integrating more and more ideas of scaffolding. Testing all four strategies on a benchmark problem should thus give some insight into how strong the individual aspects are influencing and improving the learning process.

The next step In the current part of the thesis, different strategies for scaffolding the learning process of an artificial agent for specific tasks were presented and discussed. In the next part, these scaffolds are applied on benchmark problems in order to analyse them and to test their efficiency.

Part III

Facilitating the learning process of interaction problems: testing the proposed scaffolding approaches

8 A learning domain for mediated interaction

The interaction of humanoid robots and humans is a challenging task which not only requires the fast learning of new assigned duties but also the ability to learn in ways that enable the human to easily instruct the robot. Many daily actions, such as pushing an object with a stick, pouring the contents of a mug, or accessing a book in a drawer, require some special class of object-object interaction, called *mediated interaction*. In this class of interactions, the agent can interact with the target object only after actively preparing access via the intermediate use of some auxiliary *mediator object*. Mediated interaction requires at least two objects. One goal-related object — the *target object* — and a second object — the *mediator object* or *tool* — that enables the indirect interaction with the target object. If the robot's morphology is resembling a human, tool affordance [134] — for example their use as a mediator object within a tasks context — is an important aspect that has to be taken into account [2, 4]. Learning to exploit objects as tools for different tasks is a capability found in humans, but also in some other species, such as primates or birds [1, 239]. It has been linked to higher cognition and is also a desirable capability for robots to become more adapt at many daily tasks typically arising in human environments [2, 3]. While infants are able to learn to identify the goals of novel tool-use events and also to master out-of-reach tasks [240], it is a challenging task to learn for robots and artificial agents in general.

Outline The aforementioned statement motivates not only the need for learning mediated interaction tasks but also emphasizes their difficulty. It is thus a reasonable choice to use suitable mediated interaction scenarios in order to test if the scaffolds that were presented in the last part of the thesis are able to positively influence the learning process. This chapter presents a constructed simulation world for mediated interaction, together with designed learning scenarios. At first, the overall concept is described, together with the underlying goals. After the discussion of the possible obstacles and how they were resolved, the final realization is discussed. The "Simulation Framework for Mediated Interaction Tasks" is designed as a lightweight and adaptable learning domain that can be used for testing the approaches dealing with *perceptive acting* (Chapter 4), *active visual perception* (Chapter 5) and also the one that is designed in order to improve the learning through the *refinement of the internal representation of the model* (Chapter 7). As the proposed method for endowing an artificial agent with the ability to learn efficient *active haptic perception* (Chapter 6) requires special demands to the learning domain and also needs a properly designed task for testing, it is examined in a different test bed.

8.1 The general design concept of the simulation world

The design of the simulation environment was driven by two major goals. First of all, it should offer a flexibly modifiable framework to study a wide range of interaction scenarios that are sufficiently close to real physical situations to be useful for a subsequent porting step to a real robot. At the same time, the environment should be abstract enough to avoid burdening the experiments with details that may be peripheral to obtain insights into the properties of different learning strategies. In order to bring the interaction with objects into the scenario's focus, the agent should not learn how to reach the available items. Instead the agent should be able to pilot to pre-defined points on the object's surface, where it can manipulate their position and orientation. These multiple pre-defined points simulate different ways to grasp an object.

This thesis puts its focus on scaffolding learning processes that are relying on reinforcement learning approaches for adapting the behaviour of an artificial agent. In this class of machine learning algorithms a discretized world is required, where a specific action can be assigned to every time step. For that reason, the kinematics in the scenario are built out of distinct movements that either translate or rotate the items by a fixed value. By focusing on "mediated interaction", this work goes beyond the usually considered learning scenarios where an agent directly has to reach a goal or act on a goal object (e.g. pushing something to a goal position). Instead, learning strategies are explored where desired effects cannot be created through direct interaction, but instead only by learning to shape a suitable "hierarchical interaction structure" between the agent, a mediator object and a goal object. The interaction of objects also directs to the need of a suitable physics engine that has to be capable of handling the interaction between multiple objects.

8.2 Realization of a 2D simulation world with simplified physics

The resulting designed simulated environment, shown in Figure 8.1, was constructed by taking the abovementioned conceptual ideas under consideration. In order to keep the computation cost low, a 2-dimensional world is employed. The whole learning domain has a size of 30×30 units. It contains a *target object* and a *mediator object*, also called the *tool*. The target object is designed as a green disc with a radius of 0.5 units. The circular shape gives the agent the same contact surface in every direction, which simplifies the learning of object-object interaction patterns. The difficulty of precise pushing and pulling, however, is increased as a round object is more likely to slip away. The default shape of the used tool (colored in orange) for all scenarios is that of an "L". This tool is complex enough for generating challenging learning tasks, which are nevertheless solvable for a reinforcement learning agent. Furthermore, it is designed for learning scenarios where a target object has to be grasped and pulled into a specific direction, or has to be pushed away. As the form of these mediator object allows situations where the target object can slip out of the initialized grasp by sliding along the edges, the agent has to learn to find stable grasping poses within the experiments. The physics within the domain is simulated with the open source `Box2D` physics engine[1]. By default, the friction of the objects is set to a high value. Thus, if the agent interacts with the target-object via the tool, the resulting sliding movement is limited. For object-placement tasks[2], a part of the environment is marked as the *goal-area*. The goal area, visualized as a grey circle with a radius of 3 units is in the origin of the domain.

Introducing picking locations The design-focus of the simulation world is not to learn object manipulation but object-object interaction, i.e. to operate one object by controlled collisions with a second one. Thus, some mechanisms are introduced that simplify the process for the agent to control the position and orientation of a specific object. One of them is the introduction of picking locations, which are a specific number of fixed static points that are marked on the objects. They are visualized as small black dots in Figure 8.1. By accessing these points, the agent is able to move the objects about a fixed distance. In addition, the object can also be rotated by a fixed angle. The rotation axis is given by the chosen picking-location.

[1]`http://box2d.org`

[2]In order to solve an object-placement tasks, a target-object has to be placed into a specified goal-area (within a pre-defined number of time steps).

The target-object offers a single picking location at its center, while the tool provides three picking points.

The interaction range To limit the size of the domain, the ability of the agent to interact with the objects is restricted to a certain area. This *interaction range* is represented by the large circle around the goal area with a radius of 10 units. Within this circle, the agent is able to interact with the objects. A possible analogon to this limited interaction range in the real world is the limited length of a human's or robot's arm that allows them to interact only with objects within a certain reach. To realize this limited area for interaction within the simulation world, a *rope-joint* is implemented. This kind of joint behaves like a rope that is attached between objects. If the distance of the two objects is smaller than the length of the rope, they can move freely. Yet, they are not able to increase their distance further than the length of the rope. Consequently, if the agent is manipulating an object at a chosen picking location within the simulation, a rope-joint is attached at the domains center and the chosen picking location of the object and its length is set to the radius of the interaction range. This prohibits the agent to move the currently chosen object outside of the interaction range. It is, however, possible to move objects outside of the interaction range via mediated interaction, for example by pushing it with a second object.

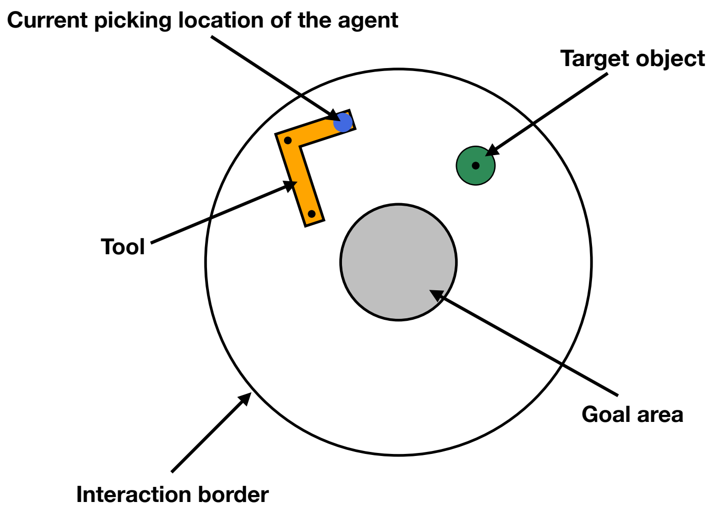


Fig. 8.1.: Schematic description of the learning domain

8.3 Perceiving & acting: defining a suitable state and action space for multi-object interaction scenarios

After the design of an appropriate simulation framework, the next challenge is to combine it with different kinds of reinforcement learning algorithms, described in Chapter 2. Reinforcement learning algorithms need a proper set of states $\mathcal{S}$ that encodes the information the agent receives from the learning domain. Secondly, the algorithms require a set of (usually discrete) pre-defined actions $\mathcal{A}$ that the agent has at its disposal.

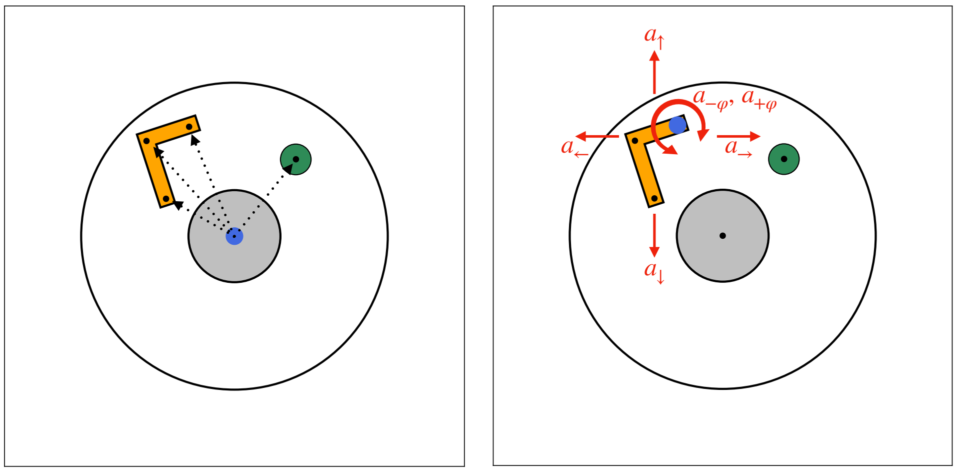
8.3.1 A general set of discrete actions

In the designed learning domain, the available objects can be manipulated by the agent at specific picking-locations. In order to select a picking-location p, an individual action a^p is assigned to each of them. After selecting one of the picking locations, the agent is able to manipulate the corresponding object in the following time step by utilizing a set of six actions

$$A = \{a_{-\varphi}, a_{+\varphi}, a_{\uparrow}, a_{\downarrow}, a_{\leftarrow}, a_{\rightarrow}\},$$

as illustrated in Figure 8.2. For changing the orientation, the agent can choose between two different actions $a_{-\varphi}$ that rotates the object about $-\pi/4$ and $a_{+\varphi}$ that rotates the object about $+\pi/4$ around the current picking location. For changing the object's position, the agent can choose between four different actions $a_{\uparrow}, a_{\downarrow}, a_{\leftarrow}, a_{\rightarrow}$ that move the object about 1 unit in the corresponding direction.

Introducing an additional picking-location Using the presented way of interaction, the agent always has to choose a picking-location that is related to an object, which might bias the learning. To this end, an additional picking location is introduced. It is exploited as an unbiased starting location for the agent. Furthermore, this additional picking location is integrated to be an absorbing state that increases the stability of applied learning algorithms. For the designed learning world, this point is located in the center of the domain. As a result, the full action set $\mathcal{A}$ is built out of the 5 actions a^p for selecting the individual picking locations and the 6 additional actions A for manipulating the object at the selected location, leading to a total number of 11 executable actions.



(a) Choosing a picking location at time step t (b) Manipulating the object at time step $t + 1$ using the actions $A = \{a_{-\varphi}, a_{+\varphi}, a_{\uparrow}, a_{\downarrow}, a_{\leftarrow}, a_{\rightarrow}\}$

Fig. 8.2.: An illustration how the agent interacts with the world. The desired picking location is selected in one step (a), while the first action for manipulating the object at the chosen location is executed in the following time step (b).

Prevent the choice of counterproductive actions To speed up the learning, just a subset of actions $\mathcal{A}(s) \subseteq \mathcal{A}$ is presented to the agent in every state s. This subset $\mathcal{A}(s)$ only includes actions that can actually be executed by the agent within the current state s and are also altering the state of the environment. Consequently, the selected picking-location at that time can not be chosen as the next action, objects outside of the interaction range can not be allocated and objects that are isotropic — like the disc-shaped target object — can not be rotated.

8.3.2 Perceiving the environment

As the agent should resemble a simplified robot, its knowledge of the environment should emulate some kind of sensory measurement. In this work, two contrasting types of sensory measurements are studied: a *simple distance related sensory input*, imitating the measurement of sensors like laser scanners, and a *visual description of the environment given by an image of the current state*.

Distance related sensory input The first sensory input is formulated as a simple relational perception of the world state. It contains enough information to solve tasks that involve the tool, the target and the goal-area. Simultaneously it should prevent a biasing as the target and the goal-area are only connected indirectly through the tool. The sensory input is built out of the six scalar distances between the three picking locations $\boldsymbol{M}_i$ on the mediator-object and the center of the target-object $\boldsymbol{T}$, defined as $|\overrightarrow{M_iT}|$, and the three distances of the picking locations $\boldsymbol{M}_i$ and the domain's origin $\boldsymbol{O}$, given by $|\overrightarrow{M_iO}|$. The distances are visualized in Figure 8.3a as red dotted lines. Additionally, the sensor vector has to be extended by a part that is able to represent the current picking location. Therefore, a binary vector

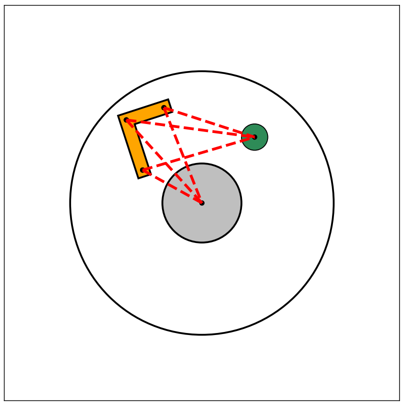
$$\boldsymbol{L} = (l^0, \ldots, l^P)^\top \text{ with } \begin{cases} l^p = 1 & \text{if } p \text{ is chosen picking location} \\ l^p = 0 & \text{else,} \end{cases}$$

is defined. Combined with the distances, the state vector is thus given by

$$\boldsymbol{s}_{\text{dist}} = \left(|\overrightarrow{M_1T}|, |\overrightarrow{M_2T}|, |\overrightarrow{M_3T}|, |\overrightarrow{M_1O}|, |\overrightarrow{M_2O}|, |\overrightarrow{M_3O}|, \boldsymbol{L}\right)^\top . \tag{8.1}$$

The first 6 dimension are continuous, as they are describing the distances between the salient objects (i.e. the target, the tool and the goal) in the current state. The remaining dimensions are discrete and boolean. One of them has to contain a 1 while the others are 0. Both parts are uncorrelated as a change of the distances is not influencing the current picking-location and switching to another picking-location is not changing the distances between the objects.

Sensory input based on raw visual image data The second, much more general sensory input for the agent is an image of the domain, like the illustration in Figure 8.3b. As an additional processing step, the generated images — having an original solution of 300×300 — are downscaled to a solution of 84×84 pixels. Additionally, they are processed through a grayscale filter and after that color-inverted and normalized. Thus, the resulting image vector contains zeroes where the image is black and thus empty. If pixels belong to a salient part of the domain, like an object or the border of the goal- or interaction-area, the image vector contains values grater than zero.


(a) The distances between the mediator-object and the target, and the mediator-object and the goal are visualized as red dotted lines.

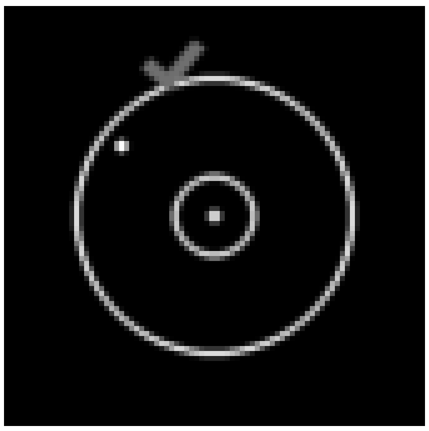

(b) An example image of the visual input, the agent is able to receive as its sensory input.

Fig. 8.3.: Illustrations of the two implemented kinds of sensory inputs that are implemented within the designed simulation framework.

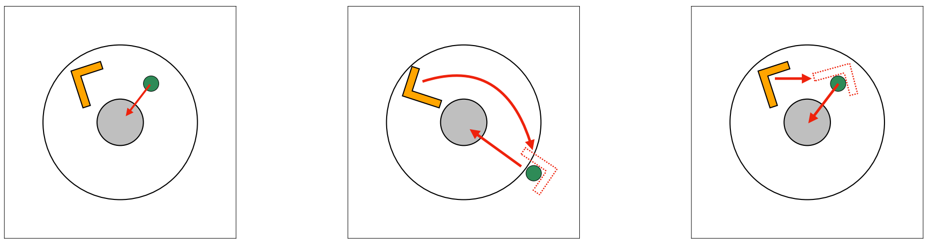
8.4 Designing suitable learning scenarios

The learning domain was designed with a special focus on problems where object-object interaction plays an essential part. This section is now describing three invented episodic learning scenarios that are studied in this work. The chosen scenarios are ranging from simple single-object interaction tasks up to more complex extension-of-reach tasks [2]. The *learning scenario* describes the learning goal, general composition of the environment and the set of possible starting configurations $\mathcal{S}_0$. The specific conditions for a single *learning task* are then sampled from $\mathcal{S}_0$ for each learning trial. In the designed domain, the starting configurations $s_0 \in \mathcal{S}_0$ are given through suitable starting positions and orientations of the objects within the domain at the beginning of each trial.

Single-Object Interaction Scenario At the beginning of a learning episode, the tool and the target are uniformly distributed over the simulation world inside the agent's interaction range. The *agent is able to control both, the tool and the target* and has to learn how to move the disc into the goal area, as shown in Figure 8.4a. After successfully solving the task instance or exceeding the limit of possible interaction-steps per episode, the task starts anew with different initial object positions that are again within the agents interaction range.

Extension-of-Reach Scenario The scenario is structured like the *Single-Object Interaction Scenario* with the additional rule that the target object is distributed only *outside* the border of the agent's interaction range. Now it is only possible for the agent to solve this task by learning to exploit the hook as a tool to pull the disc inside the agent's interaction range as shown in Figure 8.4b. However, the agent can only select picking locations that are within reach (i.e. within the marked circular area). When the target object is placed outside the circle, the agent first has to "discover" that the mediator object can be used to extend the agent's reach beyond the circle boundary.

Tool-Centered Interaction Scenario The third scenario is structured like the *Single-Object Interaction Scenario* with the additional rule that *the agent is able to interact only with the tool* and not with the target. In the scenarios before, the agent was capable of manipulating the target freely within its interaction range and had to rely on the tool only in cases where the target's position was outside of the agent's interaction range. As, in the present case, the agent is not able to control the target object at all, it is forced to always utilize the tool in order to solve the given task (see Figure 8.4c). It is also important to mention that the action set $\mathcal{A}$ is reduced to 10 actions, because the picking location on the target object is omitted.



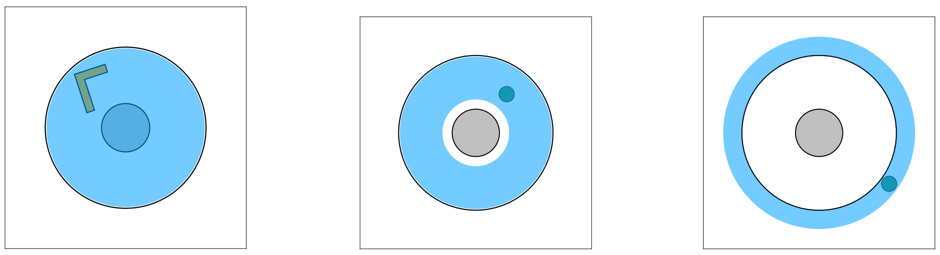
(a) Single-Object Interaction Scenario (b) Extension-of-Reach Scenario (c) Mediated Interaction Scenario

Fig. 8.4.: Illustrations of the different learning scenarios.

The *Single-Object Interaction Scenario* is the easiest one and is also not necessarily involving mediated interaction. It is however connected to the more difficult *Extension-of-Reach Scenario*. After the agent has pulled the target-object into its interaction range, the mediated interaction task can be transformed into a single-object interaction task as the agent is now able to pick the target-object and move it into the goal area without relying on the tool. The next scenario in the severity-hierarchy is the *Tool-Centered Interaction Scenario*. While this scenario narrows down the interaction space of the learner as only the tool can be controlled, it simultaneously raises the difficulty. In the tasks before, the agent was capable of manipulating the target freely within its interaction range and had to rely only on the tool at the beginning of the task, when the target's position was outside of the agent's interaction rage. Now, the agent is not able to control the target-object at all and thus has to master the handling of the given tool on a higher skill level in order to solve the given task.

Start configurations All three learning scenarios are episodic. At the beginning of each episode, the positions of the objects are sampled using suitable probability distributions. Figure 8.5 illustrates the valid sampling regions for the tool and the target. In each scenario, the tool is spawned anywhere within the agent's interaction range with a uniform probability (see Figure 8.5a). The sampled coordinates are then used to set the location of the tool's centroid and its rotation. In a next step it is checked if at least one of the tool's picking-locations is within the interaction range of the agent. If not, new coordinates are sampled. For the Extension-of-Reach Scenario, the target has to be placed slightly outside the agent's interaction range (see Figure 8.5b). For the Single-Object Scenario and the Mediated Interaction Scenario, the target has to be spawned within the interaction range, but not within the goal-area (see Figure 8.5c). After the starting coordinates are sampled and the objects are placed in the simulation world, it is checked if the objects are overlapping and colliding with each other. If this is the case, new positions are sampled and the collision check is executed again.

Learning setup As the learning scenarios are episodic, the agent has to solve the assigned task within a predefined number of time steps. In this work, the maximal number of steps per episode is set to 100 for all three scenarios. The agent is able to execute one action a per step. If it is able to solve the problem by reaching a *terminal state* or exceeding the limited number of steps, the episode ends and a new one is initialized. A terminal state is reached when the agent has placed the target within the defined goal area. In this case, it receives a reward r_{goal}. In order to slightly facilitate the learning process of the interaction between the objects, an additional contact reward can be toggled on, rewarding the agent with $r_{\text{contact}} = r_{\text{goal}} \cdot 0.1$ for the "first" successful collision between tool and target within each episode. To make the learned algorithm more stable, artificial noise is integrated into the system. It forces the agent to execute a random action with a probability of 0.1 and thus adds a stochastic component to the deterministic learning domain which disturbs the agent occasionally with a possible suboptimal action.



(a) Sampling region for the tool

(b) Sampling region for the target **within** the interaction range

(c) Sampling region for the target **outside** the interaction range

Fig. 8.5.: The different sampling regions for the tool and the target. The allowed regions are highlighted in blue. (a) Sampling region for the tool. (b) Sampling region for the target, when a *single-object interaction task* or a *tool-centered* has to be solved. (c) Sampling region for the target object, when an *extension-of-reach task* has to be solved.

8.5 Learning with a distance-related sensory input

In order to test the two scaffolds that were presented in Chapter 4 and 7 within the designed simulation world[3], they have to be combined with eligible reinforcement learning models. For this purpose, this section presents one linear reinforcement learner with two different kinds of function approximators, as well as a deep reinforcement learner. All presented algorithms are relying on the distance-related sensory input that was presented in Section 8.3, i.e. the agent is only capable of perceiving the environment through the distances between the existing objects and the goal-area. Although it is possible to achieve good learning results while utilizing linear function approximators for representing the state vector, it is worth to additionally design and study reinforcement learning with a non-linear function approximator. Despite the fact that it needs longer training time, it is much more stable under changes of its hyperparameters than reinforcement learners using linear state representations, where a small change in the parameter space — especially in the configuration of the applied function approximator — is able to visibly influence the learning performance. Additionally, showing the efficiency of the proposed scaffolding strategies when combined with non-linear reinforcement learners can be seen as a good indicator that the approaches are also improving the learning in more complex learning domains, where the employment of non-linear functions for representing the state vector is inevitable.

[3]The two proposed scaffolds in Chapter 5 and 6 are directly integrated into a suitable learning architecture that was discussed in the respective chapters, leading to the RAA3C model and the HAM.

Outline At first, a linear Q-learner is constructed. For representing the sensory input, two different kind of function approximators (the *Fixed Sparse Representation* and *Gaussian Radial Basis Functions*) are presented. Afterwards, it is explained how to approximate the state-action space for a set of discrete actions. In the last part, a deep Q-learner is designed.

8.5.1 The construction of a linear Q-learner

For learning an ϵ-greedy Q-learner with eligibility traces and linear function approximation (see again Section 2.4) is utilized. In order to gain more insight into the impact of different function approximations on the learning process, two kinds of linear function approximators are introduced for representing the state vector $\boldsymbol{s}$ that was defined in (8.1).

Fixed-Sparse-Representation An easy way to represent the sensor vector is to use a binary representation like the *Fixed Sparse Representation* (FSR) [241]. It is one of the most simple kinds of representations as each state is presented by a plain binary encoding. The FSR tries to balance the coverage and generalization of the state space. In contrast to tabular representations, where each state is represented by exactly one feature, multiple features can be active at the same time. This reduces the overall number of features in the representation to a necessary minimum. It is, however, very sensitive to the information encoded in the state vector.

To create a FSR from a continuous state vector $\boldsymbol{s}$ of length $d = \dim(\boldsymbol{s})$, each dimension s_i is discretized into n_i buckets[4], leading to a total number of $f = \sum_{i=1}^{d} n_i$ features. The number of buckets used for the discretization of each dimension of the state vector $\boldsymbol{s}$ determines the coarseness of the tiled state space.

The complete feature vector

$$\Phi(\mathbf{s}) = [\phi(\mathbf{s})_{11} \dots \phi(\mathbf{s})_{1n_1}, \phi(\mathbf{s})_{21} \dots \phi(\mathbf{s})_{2n_2}, \dots, \phi(\mathbf{s})_{d1} \dots \phi(\mathbf{s})_{dn_d}]^\intercal \,,$$

is then given by a vector of distinct binary features

$$\phi_{ij}(\mathbf{s}) = \begin{cases} 1 & s_i = v_i^j \\ 0 & \text{otherwise} \end{cases} \quad, i = 1, \dots, d \quad j = 1, \dots, n_i$$

[4]Dimensions that are already containing binary information are split into two buckets, indicating if the value is either 0 or 1.

with v_i^j, the jth possible value for the ith state space dimension s_i. For each dimension s_i of an arbitrary state $\boldsymbol{s}$, the feature $\phi_{ij}(\mathrm{s})$ is flipped to 1 where the bucket v_j corresponds to the value s_i.

Radial Basis Functions A more complex but also more efficient approach is to represent the state input by a weighted sum of *Gaussian Radial Basis Functions* (RBFs) [241–243]. Therefore, an arbitrary number of RBFs (see Figure 8.6)

$$\phi_j(s) = \exp\left[-\frac{\|s - \bar{s}_j\|^2}{2\sigma_j^2}\right]$$

are placed in each dimension of the state vector $\boldsymbol{s}$. Depending on the structure of $\boldsymbol{s}$, the center $\bar{s}_j$ of each RBF can either be placed uniformly or randomly in each dimension. In some cases, it also might be beneficial to normalize the RBFs, which leads to

$$\bar{\phi}_i(s) = \frac{\phi_i(s)}{\sum_j^N \phi_j(s)}.$$

In contrast to the FSR, where only one bin could be active in each dimension s_i at the same time, multiple RBFs can contribute to one s_i. This is caused by the continuous decay of each RBF which has its maximum value at the center $\bar{s}_j$ and then decreases according to their standard deviation (or bandwidth) σ_j. A large standard deviation leads to a more flattened RBF that contributes to a larger range of values for s_i, while a small standard deviation creates a more peaked function. The contributing areas of different RBFs are overlapping in most cases, which leads to a smoother and (in most cases) more stable approximation than for example the FSR. The drawback of using RBFs is that the number of functions is strongly increasing with the number of centers placed in each dimension.

How to handle binary dimensions when using RBFs? The RBFs work directly in the continuous space of each dimension s_i of the state vector. This raises the question how to handle cases, like the state vector (8.1), where some dimensions contain continuous and others binary variables. Instead of trying to represent the boolean variables through RBFs, a "hybrid approach" is proposed in this thesis. Therefore, the continuous dimensions of $\boldsymbol{s}$ are represented using RBFs while the binary part of the state vector $\boldsymbol{s}$ is represented using a binary representation, as it can already be seen as a very condensed and efficient feature vector.

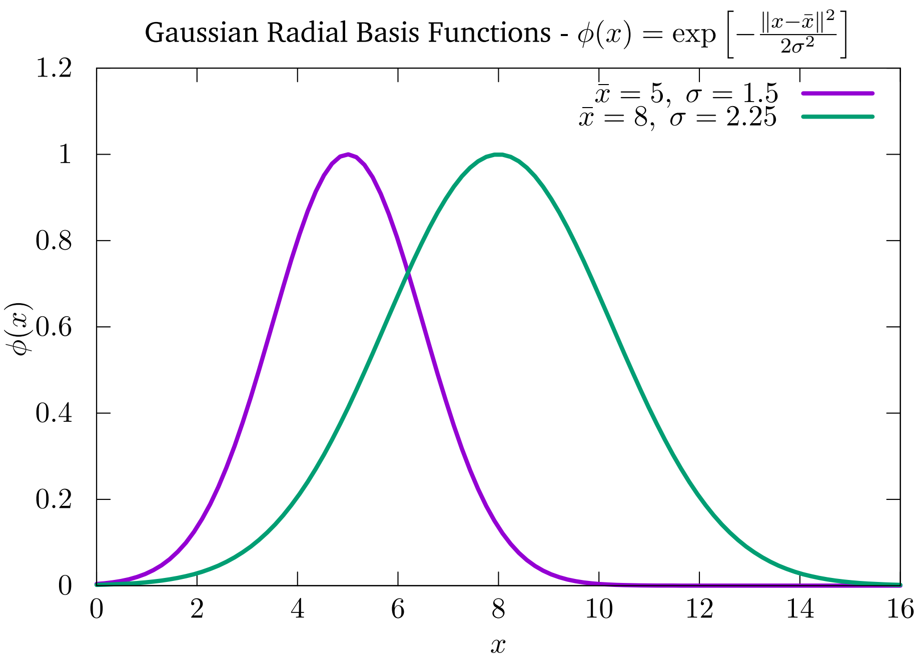


Fig. 8.6.: Illustration of two Gaussian RBFs with different mean and variance. Multiple RBFs are able to contribute to the same value x.

Tuning the parameters The learning parameters for the two linear Q-Learning approaches are individually optimized for each scenario and each representation using random search [244]. A detailed list of the chosen values can be found in Appendix B.1. The optimized parameters for the learning algorithm are the greediness ϵ, the eligibility factor λ, the discount factor γ and the learning rate α. The learning rate is decreasing with each training step t according to the function

$$\alpha(t) = \alpha_0 \cdot \frac{\alpha_B + 1}{\alpha_B + t},$$

that was proposed by Boyan in [245] and is thus called *Boyan decay*. The parameter α_0 is indicating the initial value of the learning rate, while α_B regulates the speed of the decay as illustrated in Figure 8.7. For the Fixed Sparse Representation, the number of bins per dimension was tuned in a distinct parameter search and stays the same in all experiments. The FSR represents the state vector by 310 features. The state representation based on RBFs is normalized and the number of of centers per dimension is tuned together with their standard deviations, leading to 13829 quantized features.

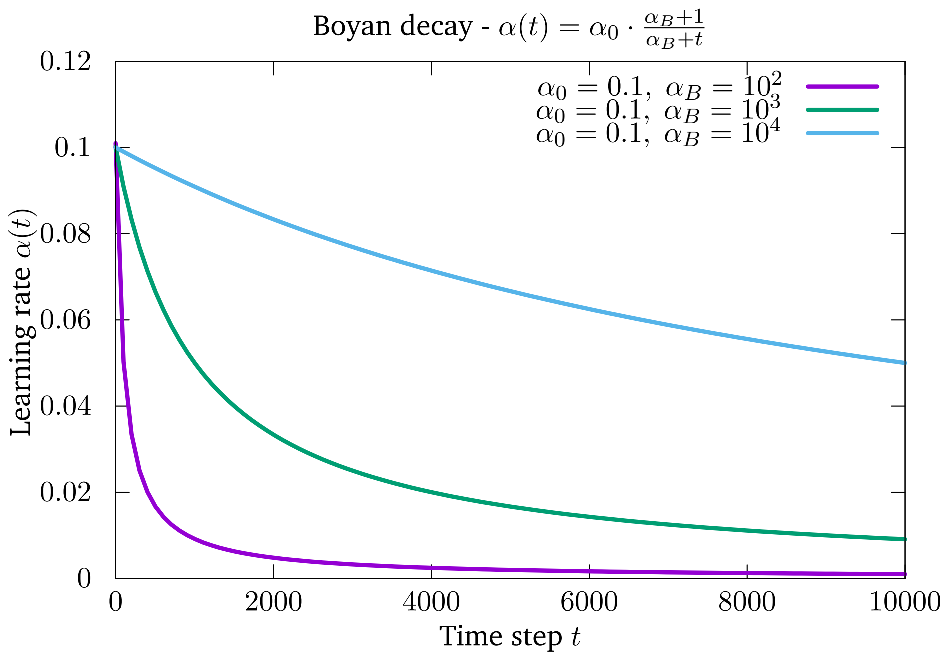


Fig. 8.7.: Illustration of the Boyan decay which is put to use for adjusting the learning rate α during training.

8.5.2 Creating a deep Q-Learner

For the case involving the non-linear function approximator, a deep Q-learner as described in Section 2.6 is designed. The invented deep neural network is illustrated in Figure 8.8. It is built out of 3 fully connected layers with 256 neurons each and rectified linear units (ReLUs) as activation functions. Similar to the linear representation based on RBFs, the state vector (8.1) is split into a continuous and a binary part. The continuous part is propagated through the whole network so that suitable features for the representation of the distances between the objects and the goal are learned. The binary part encodes only the current picking location of the agent and is thus uncorrelated to the objects distances. As the binary representation can again be seen as a very condensed and efficient feature vector, it is directly concatenated with the generated features of the last layer.

Tuning the parameters The parameters where optimized using random search and additional manual tuning. They are listed in the Appendix B.2. For the deep Q-Learner, a constant learning rate α is chosen. The greediness is decaying linearly from 1 to 0.1. As proposed in [7], a target network is integrated to compute the gradient that receives a copy of the original networks parameters every τ steps. During training, the weights are updated using the *Adam* algorithm [67].

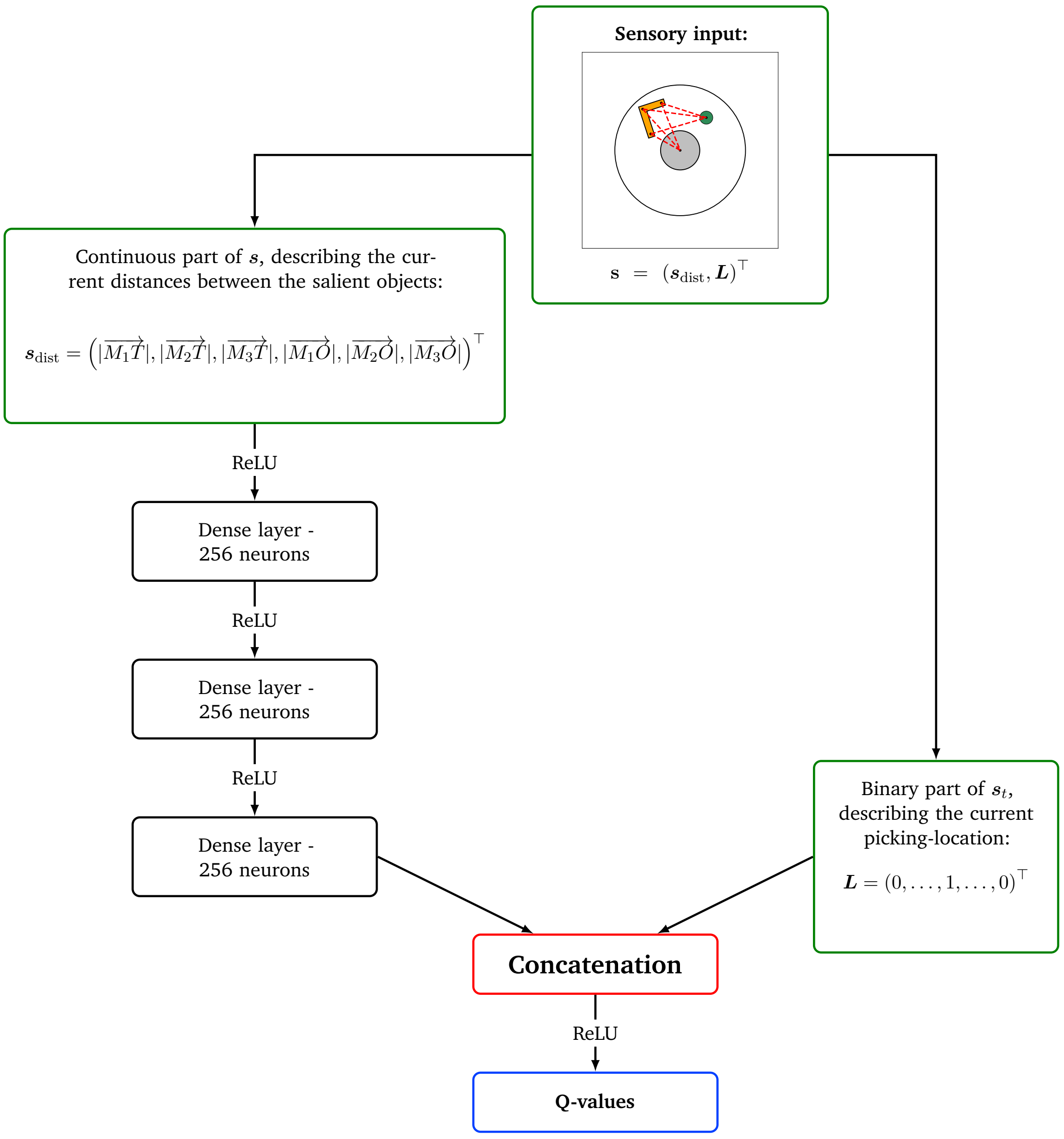

Sensory input:
$\mathbf{s} = (\boldsymbol{s}_{\text{dist}}, \boldsymbol{L})^\top$
Continuous part of $\boldsymbol{s}$, describing the current distances between the salient objects:
$\boldsymbol{s}_{\text{dist}} = \left(|\overrightarrow{M_1T}|, |\overrightarrow{M_2T}|, |\overrightarrow{M_3T}|, |\overrightarrow{M_1O}|, |\overrightarrow{M_2O}|, |\overrightarrow{M_3O}|\right)^\top$
ReLU
Dense layer - 256 neurons
ReLU
Dense layer - 256 neurons
ReLU
Dense layer - 256 neurons
Binary part of $\boldsymbol{s}_t$, describing the current picking-location:
$\boldsymbol{L} = (0, \ldots, 1, \ldots, 0)^\top$
Concatenation
ReLU
Q-values

Fig. 8.8.: Illustration of the designed deep Q-learner The state vector (8.1) is split into a continuous and a discrete binary part. While the continuous part is propagated through the whole network, the binary part is only concatenated with the generated features of the last linear layer.

8.6 Summary

In this chapter, the designed simulation world for studying the scaffolding of *mediated interaction learning* was presented. In order keep the balance between realism and simplicity, the simulation provides a computationally lightweight framework that enables quick optimization and testing for different learning algorithms and scaffolds. Furthermore, three learning scenarios with different levels of difficulty were described.

Designing sensory inputs For characterizing the different states within the environment, two different kinds of sensory signals were designed. The distance-related sensor vector leads to a low-dimensional sensory input that permits the employment of fast and efficient reinforcement learning models with linear function approximators. It can also be used to train deep reinforcement learning models, while keeping the number of trainable weights low. The other sensory measurement that is made available allows the agent to perceive the environment in form of a raw visual image. This high-dimensional sensory input can be combined with modern deep reinforcement learning algorithms that have the ability to learn the extraction of the salient features from this more complex sensory signal.

Set up (non-)linear Q-learners In order to learn to solve the problem scenarios that are presented in Section 8.4, a Q-learner with linear function approximation was set up. In order to gain more insight into the impact of different function approximations on the learning process, two kinds of linear function approximators are taken under consideration for representing the sensor vector s. As the last step, also a Q-learner using a non-linear function approximator is created in the form of a deep Q-network.

The next step After designing a learning domain, as well as suitable learning models, the setup can now be combined with the scaffolds that were presented in the last part of the thesis. At first, the scaffold that is related to "perceptive acting" (see Chapter 4), is tested in order to choose a suitable interaction strategy for learning in the given environment.

9 A first scaffold for learning the "Extension-of-Reach Scenario": determining the best action set

Finding good principles to choose the actions of artificial agents like robots in the most beneficial way to optimize their control over the environment is very much in the focus of current research in the field of intelligent systems. Especially in reinforcement learning, where the agent learns through the direct interaction with the environment, a good choice of actions is essential. In many learning scenarios, the agent is perceiving the environment in a more or less time-discrete way. Every time step, it receives sensory data of the environment that is then processed and used to determine the next action.

Studying the scaffold founded on "perceptive acting" It is possible to enhance the sampling of meaningful sensory data through a suitable reformulation of the agent's interaction strategy by refining its options or degrees of freedom. As a result, this chapter investigates the learning under the influence of different action sets by employing the designed 2D simulation world that was presented in the last Chapter. Therefore, six different interaction strategies are designed in the form of sensorimotor coordinate systems which can all be exploited by the learner as eligible action sets $\mathcal{A}_i$ for learning the given problem. Although some coordinate systems can directly be classified as bad choices via educated guesses, it is not always that easy to estimate their quality and the efficiency of the action set that is connected to it. Additionally, without testing it is hard to determine which choice of coordinate system might be the best. Motivated by the question:

> Are there general features that distinguish action sets that facilitate exploration, learning and control ("good" action sets) from action sets for which exploration, learning and control is more difficult?

a scaffold founded on perceptive acting was proposed in Chapter 4. In order to identify the coordinate system that intensifies the coupling between the agent and the learning domain, it utilizes the concept of *mutual information* [195–197] for the purpose of measuring the amount of control the agent has over the environment. This allows a predictive ranking of different action sets with regard to their influence on the learning performance of an artificial agent. The created ranking is then used as **a permanent scaffold for the learning process by selecting the most suitable action set** that has a high potential to lead to the best learning performance (see also [21]). In order to test the efficiency of the proposed approach, the resulting ranking is compared with the actual learning performance of the agent for solving the "Extension-of-Reach Scenario" using a linear Q-learner while utilizing the given action set $\mathcal{A}_i$.

9.1 Experiments

Six different coordinate systems, illustrated in Figure 9.1, are designed for the given learning domain. They can be used by the agent as action sets $\mathcal{A}_i$ that define in which way the objects can be moved through the environment. Each coordinate system[1] is designed in order to exploit different salient points within the learning domain:

World System — Figure 9.1a The most general movement frame. The objects are moved along a simple cartesian coordinate system with its origin at the center of the domain.

Tool-Fixpoint System — Figure 9.1b An orthogonal tool-focused coordinate system. A cartesian coordinate system is attached to the tool. The origin is fixed at the middle picking location.

Target-Tool-Goal System — Figure 9.1c The non-orthogonal coordinate system connects task relevant points of the domain, which are the target, the tool and the goal. The origin of the coordinate system is attached to the centroid of the tool.

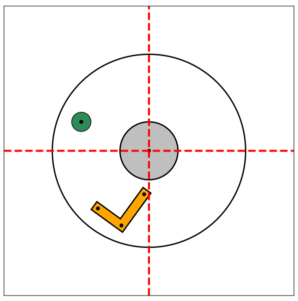
Tool-Centroid System — Figure 9.1d One axis of this orthogonal coordinate system is attached to the centroid and the middle picking-location of the tool.

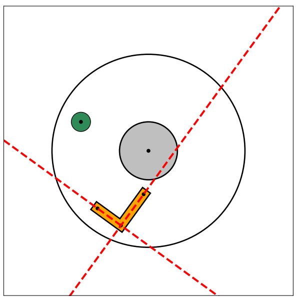
[1]A short video `Coordinate_Systems.mp4` that visualizes policies that are learned by the agent while employing the different coordinate systems, can be found within the supplementary material of this thesis (see also Appendix D).

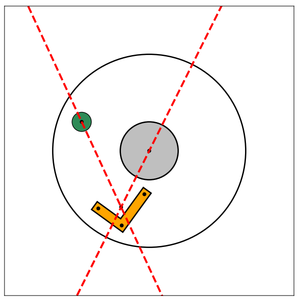
Target-Tool System — Figure 9.1e One axis of the coordinate system is created through connecting the centroid of the target with the centroid of the tool. The other one is attached to two picking-locations of the tool.

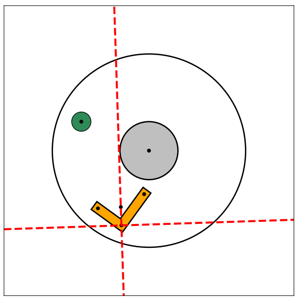
Goal-Tool System — Figure 9.1f The coordinate system has the same structure as (e), but uses the center of the goal instead of the target to define the coordinate axis.

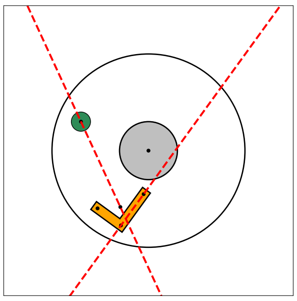
It is also important to notice that all coordinate systems, except the World System, illustrated in Figure 9.1a, are not fixed in the world but alter according to the objects positions.


(a) World System


(b) Tool-Fixpoint System


(c) Target-Tool-Goal System


(d) Tool-Centroid System


(e) Target-Tool System

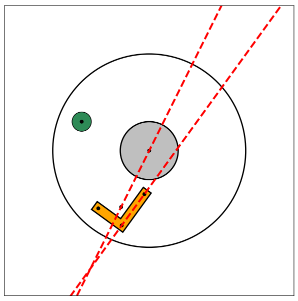

(f) Goal-Tool System

Fig. 9.1.: Illustration of the different coordinate systems within the designed learning domain, representing the different action sets $\mathcal{A}_i$

Learning setup The chosen problem that is used for testing the influence of the designed movement frames and whose learning performance is compared with the information-related ranking $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ is the episodic *Extension-of-Reach Scenario*[2],

[2]Extension-of-Reach Scenario: The task for the agent is to learn to place the target object into the goal area. However, the target object is always placed outside of the agent's interaction range at

described in Section 8.4. The agent has to learn the given task while relying on the relative distances of the salient parts of the domain (i.e. the target, the tool and the center of the goal area) as its sensory information. They are provided by the distance-related sensor vector (8.1).

For learning, the ϵ-greedy Q-learner with eligibility traces, presented in Section 8.5.1, is chosen. Additionally, both presented linear function approximators are utilized for learning. The Fixed-Sparse-Representation (FSR) generates a rather coarse tiling of the state-space with about 310 features, while the representation using Gaussian Radial Basis Functions (RBF) leads to a much finer tiling (about 13829 features). By exploiting these two learners as a benchmark, a good departure point is provided for making general assumptions about the validity to use the mutual information as a measure for ranking the efficiency of the different movement frames with respect to the learning performance.

For this experiment, an exhaustive parameter search is conducted, in which the hyperparameters for the presented Q-learner were individually optimized for each movement frame and used state representation. A list of the chosen parameters is given in Appendix B.1.

Evaluation of the learning performance For evaluating the efficiency of the learning processes of the Extension-of-Reach Scenario for the different movement frames, the average reward per episode $\langle R \rangle$ received by the agent is depicted as a function over the number of learning steps. To compute $\langle R \rangle$, the learning performance under the current policy was evaluated over 100 episodes for each of the 25 evaluated data points. The results is then averaged over 20 distinct learning runs, where the standard deviation of the mean was used as the error. For the purpose of measuring the overall quality of the whole learning process the global reward

$$R_{\text{global}} = \sum_{i}^{T} \langle R_i \rangle \,, \tag{9.1}$$

is defined as the sum over the achieved average reward $\langle R_i \rangle$ evaluated at all learning steps i. The learning process with the highest R_{global} is the one where the agent has gained the most reward and has therefore solved the tasks most efficiently.

Approximating the mutual information In the following experiment, the influence of each coordinate system $\mathcal{A}_i$ on the agent's learning performance is compared

the beginning of each episode. Thus, the agent has to learn to use the available tool for fetching the target object and pulling it in its reach before placing it into the goal area.

with its approximated average mutual information $\langle\mathcal{M}(\mathcal{S},\mathcal{A}_i)\rangle$. For this reason, the agent's state space is tessellated into Voronoi cells as described in Section 4.2. To perform the clustering process, $50 \cdot 10^3$ object-placement configurations for the target and the tool are sampled uniformly over the whole accessible domain. For studying the effect of the coarseness of the tessellated state space, one was created with 100 and a second one with 1000 Voronoi cells.

In these two spaces, $100 \cdot 10^3$ tuples (s, a, s') were counted while the agent interacts with the objects using the action set $\mathcal{A}_i$ and a random policy. Here, s refers to the starting state, while a is the executed action in order to trigger a state transition to state s'. Using this information, the probability densities $\mathrm{P}^a_{ss'}$ and P^a_s can be estimated using (4.1), (4.2) and (4.3). Finally, $\langle\mathcal{M}(\mathcal{S},\mathcal{A}_i)\rangle$ can be approximated using (4.8). For each coordinate system, the resulting $\langle\mathcal{M}(\mathcal{S},\mathcal{A}_i)\rangle$ is averaged over 10 distinct computations where the standard deviation of the mean is used as the error.

Comparing the ranking of the coordinate systems In order to see if the average mutual information is a good choice for determining the most suitable coordinate systems $\mathcal{A}_i$ — more precisely, its associated action set — for the given learning domain, the ranking according to $\langle\mathcal{M}(\mathcal{S},\mathcal{A}_i)\rangle$ is compared with the corresponding learning performance of the agent. Therefore, the global reward R_{global}, defined in (9.1), is employed as a global measure that indicates the overall efficiency of the learning process.

9.2 Results

The agent's time course of learning using a Fixed Sparse Representation of the sensory input is shown in Figure 9.2, while the results for the state representation using Gaussian Radial Basis Functions is illustrated in Figure 9.3. In both plots, the learning performance is highly varying for each used coordinate system. While some of them achieve completely different results within the learning process for the two used state representations, there are 4 coordinate systems that behave similarly. These coordinate systems are the

- World System – Figure 9.1a
- Tool-Fixpoint System – Figure 9.1b
- Target-Tool-Goal System – Figure 9.1c

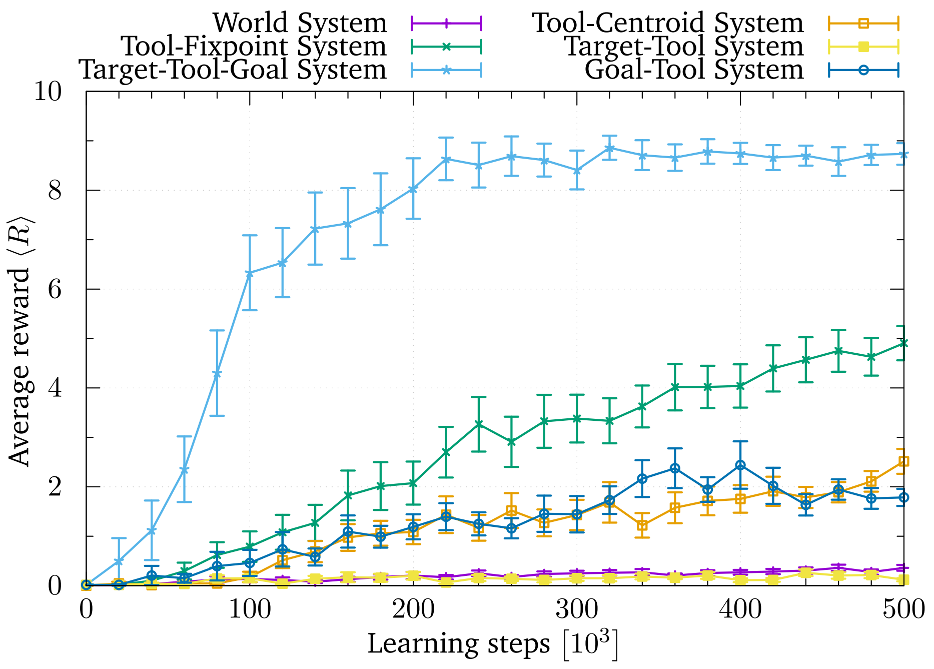


Fig. 9.2.: Course of learning for the "Extension-of-Reach Scenario", using binary features (FSR)

- Goal-Tool System – Figure 9.1f)

and include the best one ("Target-Tool-Goal System") and the one with the second worst performance ("World System") for both state representations.

Comparing learning performance and mutual information The Tables 9.1 and 9.2 are now ranking the coordinate systems according to the global reward R_{global} for both representations and additionally list the respective expected average mutual information over all available states $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$. In order to get a better general view of the correlation of the rankings between $\mathcal{R}_{\text{global}}$ and $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$, the computed average mutual information is color-coded.

If $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ is

- **bold/blue**, the ranking of the average mutual information is deviating about not more than one position from the related $\mathcal{R}_{\text{global}}$.
- red, the ranking is differing about two or more positions from the related $\mathcal{R}_{\text{global}}$.

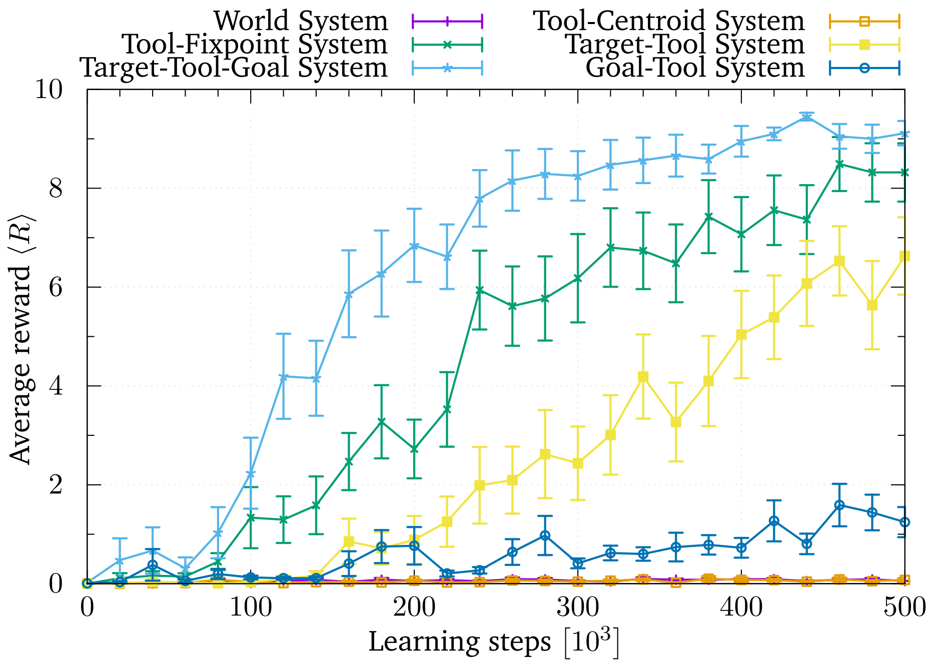


Fig. 9.3.: Course of learning for the "Extension-of-Reach Scenario", using real-valued features (RBF)

The first notable thing, visualized by the color coding, is that a coarser approximation of the state-space leads to results that are aligning better to the ranking of the learning performance. The "Target-Tool-Goal System" that is leading to the best learning performance for both tested reinforcement learners is also the one with the highest average mutual information for both state-space tessellations. Additionally, the "Tool-Centroid System" and the "World System" that are both performing poorly, as seen in Figure 9.2 and 9.3, have also a low ranking when sorted according to their average mutual information. While the discussed coordinate systems are encouraging the hypothesis that the mutual information can be used to make assumptions about the efficiency of specific movement frames, there are also some outliers that break the conformity of the two rankings R_{global} and $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$. The most peculiar outlier is the "Tool-Fixpoint System", which leads to a good learning performance for both different learners, but corresponds to a low average mutual information.

Another point to discuss are coordinate systems that are leading to completely different results during learning for both the FSR and RBF representation. An example is the "Target-Tool System". Using this movement frame, the learner that relies on the FSR is reaching its lowest performance while the learner that puts RBFs to use in order to represent the state is reaching a fair performance that ranks the “Target-Tool System” in the medium range, when R_{global} is used as the ranking criteria.

Coord. systems $\mathcal{A}_i$ - Figure	R_{global}	$\langle\mathcal{M}(\mathcal{S},\mathcal{A}_i)\rangle$ 100 cells	1000 cells
Taget-Tool-Goal System - 9.1c	181.37 ± 11.29	$\mathbf{1.736 \pm 0.013}$	$\mathbf{2.562 \pm 0.123}$
Tool-Fixpoint System - 9.1b	67.97 ± 9.84	1.641 ± 0.012	2.507 ± 0.129
Goal-Tool System -9.1f	32.34 ± 6.58	$\mathbf{1.684 \pm 0.011}$	$\mathbf{2.514 \pm 0.125}$
Tool-Centroid System - 9.1d	29.63 ± 5.62	$\mathbf{1.648 \pm 0.011}$	$\mathbf{2.504 \pm 0.128}$
World System - 9.1a	5.02 ± 1.26	$\mathbf{1.623 \pm 0.014}$	2.505 ± 0.130
Target-Tool System - 9.1e	3.46 ± 1.27	1.691 ± 0.013	2.519 ± 0.123

Tab. 9.1.: The ranking of the different movement frames according to R_{global} for the reinforcement learner using the Fixed-Sparse-Representation.

Coord. systems $\mathcal{A}_i$ - Figure	R_{global}	$\langle\mathcal{M}(\mathcal{S},\mathcal{A}_i)\rangle$ 100 cells	1000 cells
Target-Tool-Goal System - 9.1c	160.14 ± 12.31	$\mathbf{1.736 \pm 0.013}$	$\mathbf{2.562 \pm 0.123}$
Tool-Fixpoint System - 9.1b	115.14 ± 15.09	1.641 ± 0.012	2.507 ± 0.129
Target-Tool System - 9.1e	63.09 ± 13.42	$\mathbf{1.691 \pm 0.013}$	$\mathbf{2.519 \pm 0.123}$
Goal-Tool System -9.1f	15.25 ± 5.11	$\mathbf{1.684 \pm 0.011}$	$\mathbf{2.514 \pm 0.125}$
World System - 9.1a	1.48 ± 0.54	$\mathbf{1.623 \pm 0.014}$	$\mathbf{2.505 \pm 0.130}$
Tool-Centroid System - 9.1d	1.185 ± 0.55	$\mathbf{1.648 \pm 0.011}$	$\mathbf{2.504 \pm 0.128}$

Tab. 9.2.: The ranking of the different movement frames according to R_{global} for the reinforcement learner using the RBF representation.

9.3 Discussion

In this chapter, the scaffolding approach for active perception was utilized in order to investigate the impact of action sets arising from different sensorimotor coordinate frames on the efficiency of learning a mediated-interaction task. By defining actions relative to a coordinate system, the choice of an action set was connected with the choice of a coordinate system. In a next step, the learning performance of different action sets are then evaluated on solving the designed “Extension-of-Reach Scenario” while using reinforcement learning methods. It has been demonstrated that the choice of the movement frame has a visible impact on the quality of learning.

Additionally, the average mutual information $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ is computed for each action set. It measures the reduction of uncertainty of the agent's next state due to the control of the used action set. After the empirical demonstration that different action sets have led to different learning results, their performance ranking was compared with the ranking of their average mutual information within the environment.

Are there general features that are able to rate action sets in the context of learning performance? The results indicate that the mutual information-based measure can yield useful predictions on the aptitude of action sets for the learning process. The findings are also consistent with the expectation that "good" coordinate systems should be those that make uncertainty-reducing actions easy to express. For the task at hand, this turns out to be better achieved with "relational" instead of "absolute" coordinate choices. In this way, the designed approach can be used to rank different options for choosing a coordinate system that is "favorable" for the learning task at hand.

Conclusion Although the rankings of mutual information and global reward are not exactly aligning to each other, there are lots of similarities. It was possible to clearly identify the best action set for the employed learning scenario. There are also indicators leading to the assumption that the action sets with a low ranking with respect to $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ are also performing poorly during learning. In view of this, the concept of mutual information, conditioned on the chosen action set, can be exploited a scaffold that supports the process of sensing and acting within the environment. Through predictive hints on the ranking of the general learning performance while utilizing a specific sensorimotor coordinate system, it is possible to make a pre-selection of potentially good candidates even before starting the learning process.

Future work Based on these findings, further investigations in this matter may lead to a better understanding of the relationship between the mutual information and the agent-environment interaction which then can be used to guide the choice of actions within difficult learning scenarios. In any case, it has additionally to be kept in mind that the computed $\langle \mathcal{M}(\mathcal{S}, \mathcal{A}_i) \rangle$ is only connected to the employed sensory information, the action set and the learning environment, but is completely independent of the learning algorithm, the used state representation and the learning task. As a consequence, one can not expect a full alignment of both rankings but only clues that specific action sets might be more suited for learning than others.

The next step In this chapter a first scaffold was applied to determine a suitable action set for the learning of the “Extension-of-Reach Scenario”. The Figures 9.2 and 9.3 are however indicating that also for the best coordinate system (“Target-Tool-Goal System”) the current learning process has left room for improvement. Hence, the next chapter now investigates if it is possible to further enhance and speed up the learning process by scaffolding the internal representation of the learning model as discussed in Chapter 7.

10 A second scaffold for learning the “Extension-of-Reach Scenario”: structuring the learning process

In addition to facilitating the sampling of good sensory data, it is also possible to support the learning process by directly shaping the internal representation of the given learning model. Especially when dealing with complex tasks, for example those requiring to learn a hierarchy of sub-tasks, most standard learning algorithms are not able to solve the problem to a satisfying level. In this context, *transfer learning* has been introduced as an effective tool for improving the learning of complex tasks, especially when designing a *curriculum* for the pre-trained sequence of source tasks. Guided by the questions:

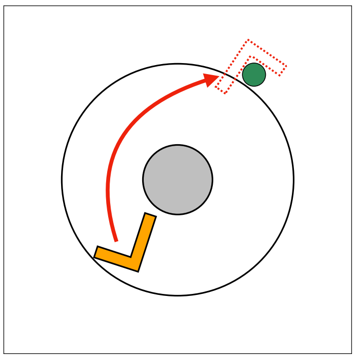
> Can a scaffold based on the combined approach of transfer and curriculum learning augment the learning process of established learning models? How is the integration of key characteristics of the refined concept of scaffolding influencing the learning process?

a scheme for **refining the model’s internal representation through the pre-training of sub-skills using a combined transfer and curriculum learning approach that is enriched by the developed key aspects of scaffolding** was proposed in Chapter 7 (see also [22]). For this scaffold, it is assumed that the agent is able to learn relevant sub-skills within the same domain as the main task, so that both can be handled by the same learning process.

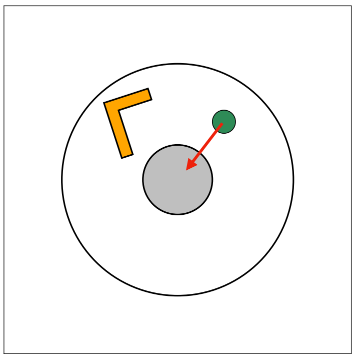
All in all, *four different strategies* were designed. Starting with *Strategy 1* which simplifies the learning process through transfer learning, the four strategies are bottom-up approaches that are integrating up to four key aspects of the concept of scaffolding into the supportive method: *simplification*, *fading*, *transfer of responsibility* and *ongoing diagnosis & assessment*. By applying all strategies on the same learning problem under the same conditions, it is possible to investigate the impact of the different key characteristics of scaffolding on the learning process.

Scaffolding through experience generalization by pre-learning a curriculum based sub-task hierarchy In this study, the “Extension-of-Reach Scenario” within the simulated 2D interaction world that was presented in Chapter 8 was again chosen as the test bed. In the last Chapter 9, learning was improved through the choice of a suitable movement frame. Based on these results, the present goal is to improve the learning process further by employing the ideas from Chapter 7.

Again, the “Extension-of-Reach Scenario” is an appropriate candidate for testing the effectiveness of the proposed transfer learning approach as it can be divided into two distinct stages as shown in Figure 10.1. While the task instances of the first stage (Figure 10.1a) can be solved only through *mediated interaction*, the task instances of the second stage (Figure 10.1b) are solvable through direct, *unmediated interaction* as the agent is able to directly interact with the target object. Stage two, resembling the defined “Single-Object Interaction Scenario” (see Section 8.4), assigns the same problem to the agent, i.e. navigate the target object to the goal, but under easier conditions. It is therefore a good choice to be used as the *source task*. Thus, the designed scaffold can be applied to generalize experiences from source tasks that are solvable through direct, unmediated interaction, to target tasks that require mediated interaction for their solution.



(a) First stage - The tool has to be utilized in order to pull the target into the agent’s interaction range.



(b) Second stage - The target has to be placed into the goal area.

Fig. 10.1.: Sketches of the two stages of the “Extension-of-Reach Scenario”. The second stage (b) is resembling the easier “Single-Object Interaction Scenario”.

10.1 Experiments

The goal of the experiment is to compare the influence of the different strategies on the learning process and to identify their individual impact. Ultimately it is discussed which strategy — and thus which combination of the four key aspects of scaffolding — is the most beneficial technique for speeding up the learning learning process.

Scaffolding the learning process The general idea of all four strategies is to first learn to solve the source tasks to a specific extend. Afterwards the agent directly starts to learn the target task without any delay. This implies that learning the source tasks is **not** rolled out within a separate learning process but treated as a pre-learning routine that is carried out right before learning the target task. Consequently, the learning steps that are needed to solve the source tasks are included within the learning steps of the whole learning run.

10.1.1 Learning and evaluation

The experiments are directly following up to the results of the last chapter. There, it has been shown that the two tested linear Q-learners are both learning best using the "Target-Tool-Goal System". The two learning models[1] that utilize this movement frame are now again employed for the following experiments with the same configuration of hyperparameters[2]. Thus, as in the experiments of the last chapter, the Fixed-Spare-Representation and Gaussian Radial Basis functions are employed as linear function approximators representing the distance-related sensory input (8.1). Additionally, the experiments are conducted for the deep Q-learner that was introduced in Section 8.5.2, in order to investigate if the developed method is also able to facilitate the learning process when a non-linear function approximator like a neural network is used. The used configuration of hyperparameters for the deep Q-learner can be found in Table B.9 in Appendix B.2.

[1]Needless to say, the two learning models are initialized with random weights and trained from the scratch in the experiments that are presented in this chapter.
[2]The parameters configuration of the two learning models can be found in Table B.4 in the Appendix B.1.

In summary, three different models are employed for learning the “Extension-of-Reach Scenario”:

- Linear ϵ-greedy Q-learner with eligibility traces, representing the state vector s via
  - Fixed Sparse Representation
  - Gaussian Radial Basis Functions
- Deep Q-learner

For evaluating the efficiency of the learning processes, the average reward per episode $\langle R \rangle$ is used. The learning performance under the current policy is therefore evaluated over 100 episodes for each of the 25 evaluated data points. The results are then averaged over 20 distinct learning runs, where the standard deviation of the mean is used as the error. In addition to the *global reward* (9.1) that was presented in Section 9.1, a second evaluation metric is now introduced for providing a way to compare the learning process of the “Extension-of-Reach Scenario” without transfer learning and the learning processes using the transfer learning schemes.

Time to Threshold If an artificial agent has to learn the interaction with objects in the real world, there are often many undesired events that may happen because of suboptimal behaviour. The objects might break or the agent itself might be damaged. So it is crucial to reach a “good performance as soon as possible” in order to minimize the chance of these events to happen. This metric, which is introduced in [161] and illustrated in Figure 10.2, measures the number of learning steps that are necessary to reach a certain performance threshold. The choice of a reasonable threshold performance is depending on the studied learning scenario. When learning real-world scenarios using robots, it is a preferable goal that the robot learns to solve them with a good performance as fast as possible to minimize errors and therefore the probability of damaging the environment. Under the assumption that such kind of performance is given by an average success rate of 80% and above, the threshold performance which is measured using the “Time to Threshold” metric is chosen to be at this level. Although this threshold is significantly below optimal success, it is on the one hand high enough to tag the corresponding policies as successful and on the other hand low enough that the performance level can be achieved in all experiments within a reasonable amount of learning steps.

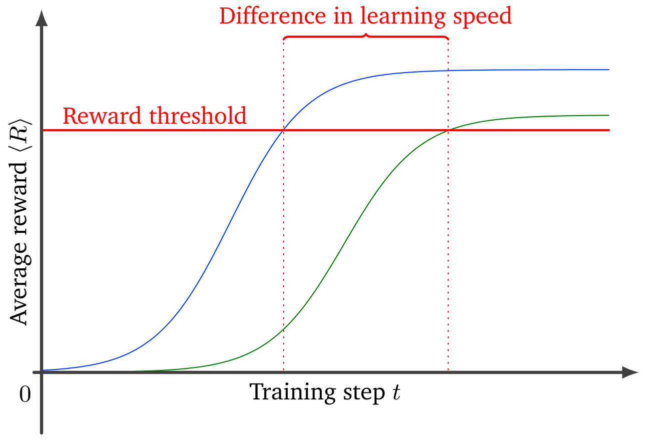

Difference in learning speed
Reward threshold
Average reward $\langle R \rangle$
0
Training step t

Fig. 10.2.: Illustration of the *Time to Threshold* performance metric that is used to determine the difference in learning speed of different approaches

10.1.2 Applying the four transfer learning strategies

While the simplest solution is to randomly generate new source tasks a predefined number of times, given by *Strategy 1*, *Strategy 2* refines this process by generating a fixed set of source tasks. At the beginning of the learning episode, one task is selected randomly and is then removed from the set if it was solved successfully. Further structure is established by *Strategy 3* through the creation of source task prototypes that should be distributed more uniformly over the task space. This is achieved by clustering a large amount of randomly generated source tasks and then selecting the ones that are closest to the prototypes by utilizing a *correlation metric*. In a last step, *Strategy 4* is extending the process by also using the correlation metric to select the next to-be-solved source task.

As a first experiment, the results of the regular transfer learning scheme (*Strategy 1*), presented in Algorithm 1, is evaluated. Therefore, the agent has to solve N_{source} tasks from the "Single-Object Interaction Scenario" (source tasks) that are generated randomly at the beginning of every episode, before starting to learn the main problem. In the following experiments, the source tasks are generated beforehand, as described in the procedures for *Strategy 2 – 4* in Chapter 7. The set C of the to-be-learned source task configurations $c = (\boldsymbol{t}, \boldsymbol{s}) \in \mathcal{C}$ can be decomposed in the task space description $\boldsymbol{t}$ and the sensor space description $\boldsymbol{s}$. Each task can be completely described via the position (x, y) and orientation φ of the target and the tool within the environment, .i.e.

$$\boldsymbol{t} = (x_{\text{target}}, y_{\text{target}}, \varphi_{\text{target}}, x_{\text{tool}}, y_{\text{tool}}, \varphi_{\text{tool}})^\top .$$

This setup is then used for scaffolding the learning according to *Strategy 2*. In order to generate the structured set of source tasks, used in *Strategy 3 and 4*, 10^4 source-task configurations $\boldsymbol{c}_r(\boldsymbol{t}, \boldsymbol{s})$ are sampled. The prototypes P_i that are generated by utilizing the "K-Means clustering algorithm" are then employed to find the most similar sampled source task $\boldsymbol{c}_i$ by either using the *cosine similarity* (7.1) or the *Pearson correlation coefficient* (7.2). During learning, these source task configurations are randomly presented to the agent at the beginning of each episode as described in Algorithm 2 until the agent has solved the problem N_{source} times. At last, the transfer learning scheme is extended by *Strategy 4*, i.e. by further structuring the learning of the source task as described in Algorithm 3.

10.2 Results

In order to get an impression of the difference between the regular learning and the different kinds of transfer strategies, the learning curves for selected learning processes are plotted. In every plot, the regular learning is illustrated, together with the fastest configuration using the simple *Strategy 1* and the configuration that is leading to the fastest learning for the given learner by taking all four strategies under consideration. Here, the fastest learner is given by the configuration that reaches the performance threshold of 80% within the smallest number of learning steps. If more than one configuration is learning with nearly the same speed, the one with the higher global reward R_{global} is chosen[3].

Results for linear Q-learning relying on the FSR Figure 10.3 shows the results for the linear Q-learner using the Fixed Sparse Representation. All learning curves, with and without transfer learning, are able to achieve an average reward of $\langle R \rangle \approx 9$. This corresponds to a probability of about 90% to successfully solve the assigned extension-of-reach task. Using a transfer strategy, however, is significantly speeding up the learning time that is needed in order to reach the performance threshold. While there is only a slight visible difference between the best achieved results (*Strategy 2*) and the results of *Strategy 1*, their learning curves can be clearly separated from the one illustrating the slower learning process of the regular learning.

[3]An illustrative video `SkillTransfer.mp4` with examples of learned policies can be found in the supplementary material of this thesis (see also Appendix D).

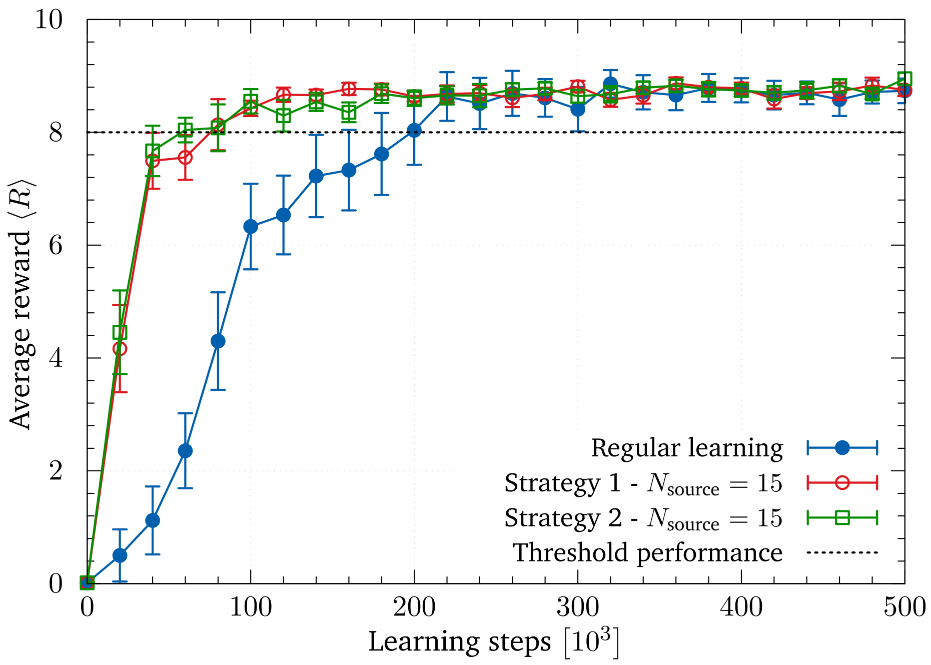


Fig. 10.3.: Evaluation of the learning process for the "extension-of-reach task", using the binary representation (FSR). The threshold performance is placed at an average success rate of 80%.

Results for linear Q-learning relying on RBFs The next Figure 10.4 shows the learning curves for the learner using the real-valued representation (RBF). Again all three learning curves are able to achieve an average reward of $\langle R \rangle \approx 9$. In contrast to the leaner using the binary representation (FSR), there is a visible difference in the learning performance for *Strategy 1* and *4*. It is a good illustration that further structuring the selection process of the source task can lead to an additional speed up of the learning.

Results for the deep Q-learner Similar observations can be made when looking at Figure 10.5 that illustrates the results for the deep learner. As the deep Q-learner has much more weights to train than the linear Q-learners, the learning is slower than for the other two cases. Nevertheless, applying *Strategy 3* of the developed scaffold enables the agent to reach the threshold performance after $\approx 280 \cdot 10^3$ learning steps and a final performance of $\langle R \rangle > 9$ within the limit of $500 \cdot 10^3$ steps. The regular learning reaches only a final performance of $\langle R \rangle \approx 6$ which corresponds to a probability of just 60% to successfully solve the task.

For a more detailed evaluation, the results of all 4 Strategies for different numbers of source task configurations N_{source} are listed in Table 10.1 for the binary represen-

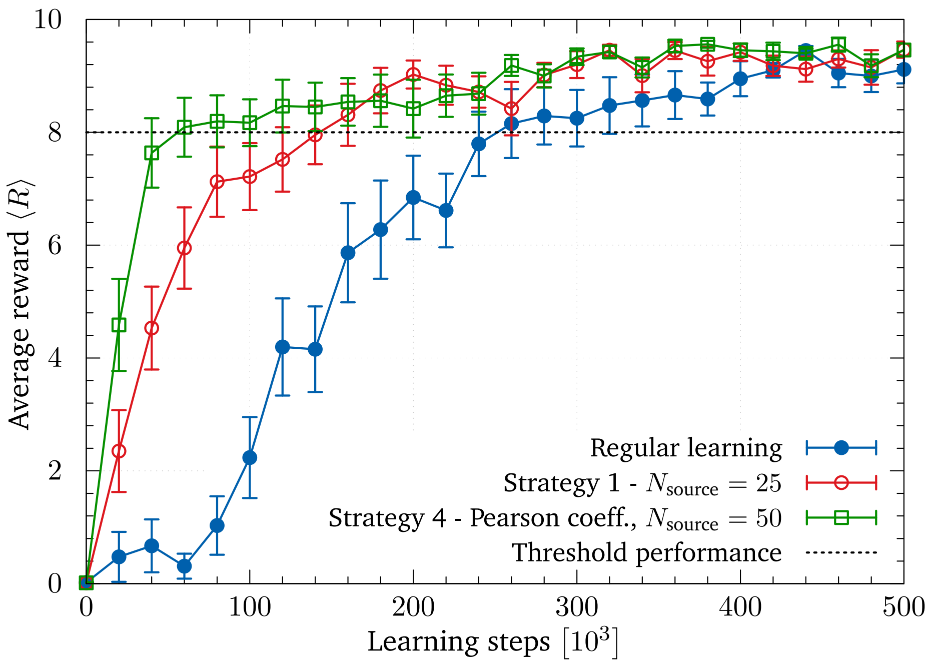


Fig. 10.4.: Evaluation of the learning process for the "extension-of-reach task" for the RBF representation. The threshold performance is placed at an average success rate of 80%.

tation (FSR), in Table 10.2 for the real-valued representation (RBF) and in Table 10.3 for the deep Q-network (DQN).

Strategy 1 At first, the results of *Strategy 1* are listed. The first notable point that can be seen in both tables of the linear function approximators is that the agent has just to solve the source task one single time ($N_{source} = 1$) to get a favorable seeding of the Q-values that is good enough to increase the overall performance, measured by R_{global}. Increasing the number of source task problems further optimizes the learning by increasing R_{global} for all three tested learners. While the deep learner is not able to reach the performance threshold of an 80% success rate using regular learning, this first simple but nevertheless effective strategy is able to already support the learning process to a level that allows the agent to reach the performance threshold after about $300 \cdot 10^3$ steps. The highest global reward R_{global}can then be achieved for $N_{source} = 15$ (FSR), $N_{source} = 25$ (RBF) and $N_{source} = 50$ (DQN). If N_{source} gets large, the learning process becomes too specialized on the source task. This phenomenon leads to a decrease of R_{global} and the learning time to reach the performance threshold up to a point where the transfer learning process undermines the learning of the target task. The extreme case of $N_{\text{source}} = 5000$ suppresses the

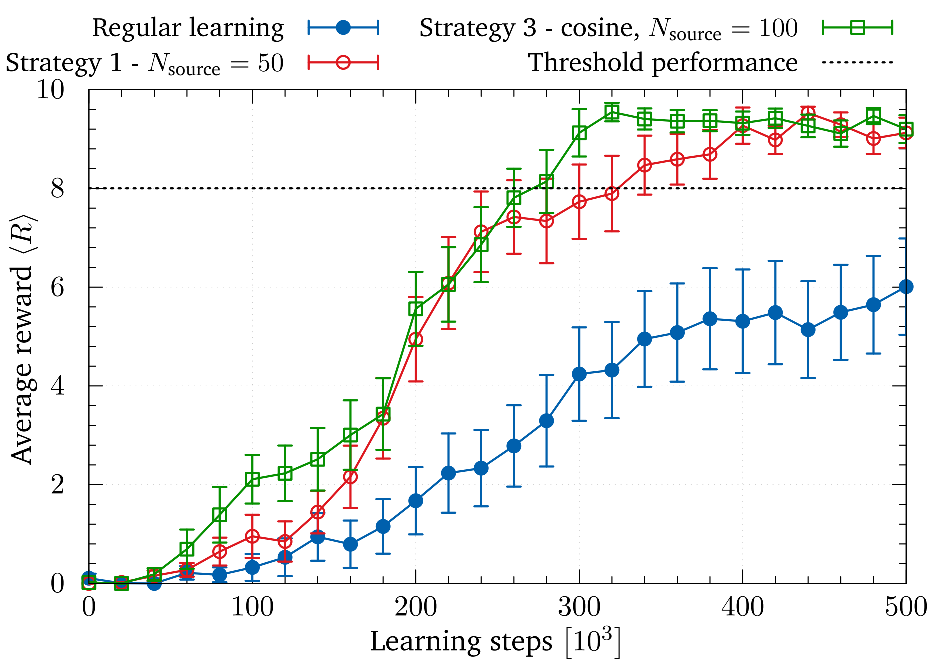


Fig. 10.5.: Evaluation of the learning process for the "extension-of-reach task" for the deep Q-learner. The threshold performance is placed at an average success rate of 80%.

learning of the target task for the learning process with the binary representation and the deep neural network.

Strategy 2 The listed results for *Strategy 2* show that this more structured strategy is able to further speed up the learning for both linear Q-learners. For the learner that utilizes the binary representation (FSR), this *Strategy 2* also leads to the best global result by speeding up the learning process about more than 3 times. In this way, the required learning time that is needed to reach the performance threshold is reduced from $\approx 200 \cdot 10^3$ learning steps to about $\approx 60 \cdot 10^3$ for $N_{\text{source}} = 15$.

Strategy 3 The next strategy further refines *Strategy 2* by integrating a methodized process for generating the set of source tasks. Within this process the *cosine* (7.1) and the *Pearson correlation coefficient* (7.2) are used in order to measure the similarity of the different source task instances (see Section 7.1.2). The listed results demonstrate that the choice of the similarity metric influences the overall performance of the learning process. While this strategy is not able to further improve the learning results of the linear Q-learner with the binary representation, it again leads to better results for both the Q-learner with the real-valued representation and the deep Q-learner. *Strategy 3* is able to achieve the fastest learning speed for the deep Q-learner. It is possible to reach the performance threshold after $\approx 220 \cdot 10^3$ learning steps for $N_{\text{source}} = 100$ source tasks, while employing the cosine as the similarity metric. Using the Pearson correlation coefficient instead leads to a slightly slower learning ($\approx 250 \cdot 10^3$ learning steps) but to the highest value of $R_{\text{global}} = 153.75 \pm 8.39$.

Strategy 4 Up until now, the currently to-be-solved source tasks were picked randomly from the created set. This can be changed by an ordered selection process, described by *Strategy 4*. Using the Pearson correlation coefficient, this strategy leads to the best results for the Q-learner that relies on the RBF representation. The learning process is now more than four times faster, accelerating the learner to reach the learning threshold after $60 \cdot 10^3$ learning steps instead of $260 \cdot 10^3$. While for the fastest learner for the FSR is provided by *Strategy 2*, a slightly higher global reward $R_{\text{global}} = 210.95$ can be achieved using this strategy with $N_{\text{source}} = 25$. The learning, however, is about 17% slower when applying this configuration by reaching the performance threshold after $70 \cdot 10^3$ steps. When studying the deep Q-learner, the learning — compared to the results when applying *Strategy 3* — is about $40 \cdot 10^3$ steps slower when employing the cosine and about $30 \cdot 10^3$ steps when utilizing the Pearson correlation.

Evaluating the strategies with respect to the global reward R_{global} *Strategy 4* generates the highest R_{global} for the learning with linear function approximators, while for the deep Q-learner *Strategy 3* leads to a slightly better result. Applying *Strategy 4*, however, results in the most learning runs that are able to reach the learning threshold of 80%. In order to get better global view on the ranking with respect to the performance gain that is achieved by the different strategies, Figure 10.6 sorts the conducted learning runs according to R_{global}.

The influence of the correlation metric on the learning process One thing that has been ignored so far is that the efficiency of *Strategy 3* & *4* is also depending on the used correlation metric. In addition to the analysis of the results in Table 10.1, 10.2 and 10.3, the three plots in Figure 10.6 aim to illustrate that the choice of the metric is visibly influencing the learning process. While it is hard to make a general statement which of the two tested metrics seems to be the better choice in general, the plots imply that, in terms of R_{global}, the cosine is a better selection for *Strategy 3* and the Pearson correlation coefficient for *Strategy 4*.

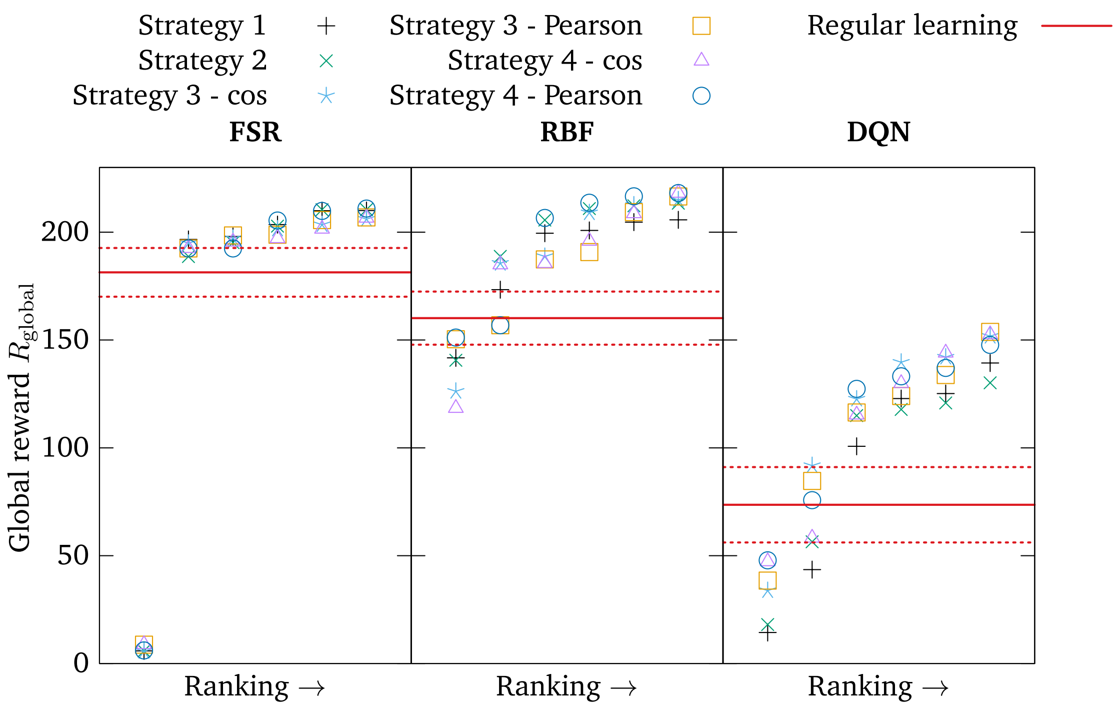


Fig. 10.6.: Comparison of the global reward R_{global} for the different learners. The plot lists the global rewards for all conducted learning runs, i.e. all 4 strategies, both correlation metrics and all tested N_{source}, sorted from the highest R_{global} to the lowest.

Used scheme	**Metric**	$\mathbf{N_{source}}$	$\mathbf{R_{global}}$	**Steps to 80% threshold** $[10^3]$
Regular learning		0	181.37 ± 11.29	≈ 200
Strategy 1		1	197.99 ± 8.83	≈ 140
		15	$\mathbf{210.03 \pm 4.69}$	$\approx \mathbf{80}$
		25	209.97 ± 5.07	≈ 80
Regular transfer		50	203.51 ± 4.93	≈ 100
No Set		100	196.62 ± 6.94	≈ 160
		5000	5.85 ± 2.47	–
Strategy 2		1	188.72 ± 10.31	≈ 160
		15	$\mathbf{210.47 \pm 4.84}$	$\approx \mathbf{60}$
		25	210.44 ± 5.37	≈ 60
Regular transfer		50	202.795 ± 5.46	≈ 100
Random Set		100	195.15 ± 6.43	≈ 120
		5000	5.87 ± 2.26	–
Strategy 3		1	197.14 ± 8.95	≈ 140
		15	206.03 ± 7.75	≈ 100
		25	203.49 ± 9.04	≈ 130
Regular transfer	cos	**50**	$\mathbf{201.49 \pm 7.54}$	$\approx \mathbf{70}$
Clustered Set		100	194.89 ± 7.34	≈ 100
		5000	5.645 ± 2.45	–
Strategy 3		1	192.47 ± 10.69	≈ 200
		15	198.50 ± 9.28	≈ 140
		25	$\mathbf{206.83 \pm 5.78}$	$\approx \mathbf{100}$
Regular transfer	pearson	50	205.61 ± 6.94	≈ 110
Clustered Set		100	198.80 ± 6.46	≈ 120
		5000	8.74 ± 4.14	–
Strategy 4		1	197.14 ± 8.95	≈ 140
		15	192.55 ± 10.86	≈ 220
		25	$\mathbf{206.64 \pm 5.77}$	$\approx \mathbf{80}$
Scaffolded transfer	cos	50	201.55 ± 7.79	≈ 160
Clustered Set		100	195.54 ± 6.92	≈ 180
		5000	8.905 ± 3.60	–
Strategy 4		1	192.47 ± 10.69	≈ 200
		15	209.85 ± 5.98	≈ 100
		25	$\mathbf{210.95 \pm 4.46}$	$\approx \mathbf{70}$
Scaffolded transfer	pearson	50	205.40 ± 5.44	≈ 80
Clustered Set		100	192.46 ± 8.63	≈ 160
		5000	6.075 ± 2.31	–

Tab. 10.1.: Evaluation of the learning process over $500 \cdot 10^3$ learning steps, using the Q-Learner with the binary representation (FSR). The best results for every approach are marked with **bold** letters, while the best global result is additionally highlighted.

Used scheme	Metric	N_{source}	R_{global}	Steps to 80% threshold $[10^3]$
Regular learning		0	160.14 ± 12.31	≈ 260
		1	173.33 ± 17.64	≈ 280
Strategy 1		15	204.67 ± 9.88	≈ 160
		25	$\mathbf{205.73 \pm 9.20}$	$\approx$ **140**
Regular transfer		50	199.5 ± 12.00	≈ 170
No Set		100	200.81 ± 12.12	≈ 140
		5000	141.74 ± 11.08	≈ 420
		1	188.79 ± 12.97	≈ 260
Strategy 2		**15**	$\mathbf{212.24 \pm 8.34}$	$\approx$ **100**
		25	205.63 ± 11.16	≈ 100
Regular transfer		50	213.44 ± 7.65	≈ 120
Random Set		100	210.94 ± 7.79	≈ 100
		5000	140.595 ± 11.85	–
		1	185.58 ± 16.11	≈ 260
Strategy 3		15	209.03 ± 9.67	≈ 140
		25	212.61 ± 7.35	≈ 100
Regular transfer	cos	50	188.75 ± 16.86	≈ 180
Clustered Set		**100**	$\mathbf{214.99 \pm 8.14}$	$\approx$ **80**
		5000	126.22 ± 11.81	–
		1	156.87 ± 17.63	≈ 320
Strategy 3		15	190.78 ± 14.94	≈ 180
		25	187.38 ± 13.71	≈ 200
Regular transfer	pearson	**50**	$\mathbf{216.57 \pm 7.06}$	$\approx$ **90**
Clustered Set		100	209.29 ± 9.64	≈ 110
		5000	150.37 ± 11.75	≈ 260
		1	185.58 ± 16.11	≈ 260
Strategy 4		15	195.94 ± 15.40	≈ 180
		25	185.00 ± 17.64	≈ 280
Scaffolded transfer	cos	50	208.70 ± 11.06	≈ 100
Clustered Set		**100**	$\mathbf{218.17 \pm 6.94}$	$\approx$ **90**
		5000	118.425 ± 15.14	–
		1	156.87 ± 17.63	≈ 320
Strategy 4		15	206.60 ± 9.44	≈ 120
		25	216.62 ± 7.57	≈ 60
Scaffolded transfer	pearson	**50**	$\mathbf{218.14 \pm 7.67}$	$\approx$ **60**
Clustered Set		100	213.65 ± 7.86	≈ 100
		5000	151.15 ± 9.71	≈ 240

Tab. 10.2.: Evaluation of the learning process over $500 \cdot 10^3$ learning steps, using the Q-Learner with the representation using radial basis functions (RBF). The best results for every approach are marked with **bold** letters, while the best global result is additionally highlighted.

Used scheme	Metric	N_{source}	R_{global}	Steps to 80% threshold $[10^3]$
Regular learning		0	73.61 ± 17.47	–
Strategy 1 Regular transfer No Set		1	43.54 ± 15.44	–
		15	100.73 ± 17.88	–
		25	122.87 ± 15.62	≈ 400
		50	$\mathbf{139.33 \pm 12.23}$	$\approx$ **330**
		100	125.165 ± 12.2	≈ 300
		5000	14.375 ± 7.38	–
Strategy 2 Regular transfer Random Set		1	56.43 ± 17.07	–
		15	117.81 ± 17.11	–
		25	114.97 ± 15.35	≈ 420
		50	120.82 ± 13.09	$\approx$ **350**
		100	$\mathbf{130.15 \pm 14.08}$	≈ 370
		5000	18.07 ± 8.06	–
Strategy 3 Regular transfer Clustered Set	cos	1	91.77 ± 19.26	–
		15	122.62 ± 18.09	–
		25	139.685 ± 13.36	≈ 280
		50	142.07 ± 13.90	–
		100	$\mathbf{151.75 \pm 8.90}$	$\approx$ **220**
		5000	33.81 ± 10.26	–
Strategy 3 Regular transfer Clustered Set	pearson	1	84.665 ± 20.27	–
		15	124.02 ± 16.98	≈ 360
		25	133.76 ± 16.99	≈ 400
		50	116.42 ± 15.95	≈ 360
		100	$\mathbf{153.75 \pm 8.39}$	$\approx$ **250**
		5000	38.46 ± 12.34	–
Strategy 4 Scaffolded transfer Clustered Set	cos	1	58.28 ± 18.18	–
		15	115.34 ± 17.78	–
		25	129.93 ± 16.25	≈ 280
		50	144.1 ± 14.19	≈ 300
		100	$\mathbf{152.59 \pm 10.40}$	$\approx$ **260**
		5000	47.31 ± 16.77	–
Strategy 4 Scaffolded transfer Clustered Set	pearson	1	75.75 ± 19.36	–
		15	137.05 ± 16.50	≈ 340
		25	$\mathbf{147.745 \pm 14.92}$	$\approx$ **280**
		50	127.36 ± 16.43	≈ 360
		100	133.19 ± 15.22	≈ 340
		5000	47.84 ± 14.746	–

Tab. 10.3.: Evaluation of the learning process over $500 \cdot 10^3$ learning steps, using the Deep Q-Learner (DQN). The best results for every approach are marked with **bold** letters, while the best global result is additionally highlighted.

10.3 Discussion

In this chapter, the designed strategies from Chapter 7 are employed in order to facilitate the learning process of a “mediated-interaction task”. By combining transfer and curriculum learning with up to four different key aspects of scaffolding, the internal representation of the learning model is refined during the learning process in order to cope with the hierarchical aspect of the given scenario.

Can a combined approach of transfer and curriculum learning improving the learning performance? Within this simplified, yet physically realistic toy world, the integration of transfer learning is able to speed up the learning process of the Extension-of-Reach Scenario by about an order of magnitude. Significant improvements can already be found by combining the most simple *Strategy 1*, i.e. temporary simplifying the learning process, with a reinforcement learner using Q-learning with linear and non-linear function approximation.

Does adjoining more key aspects of scaffolding into the supporting approach enhance the improvement of the learning process? For all three learners, a different combination of key aspects is leading to the fastest learning result. Thus, increasing the number of key aspects is not necessarily improving the learning process. While for the FSR, *Strategy 2* (*simplification* and *fading*) is already providing the best results, the deep Q-learner needs more scaffolded support to reach its best performance, leading to *Strategy 3* (*simplification*, *fading* and *transfer of responsibility*). The learner using the real-valued linear state representation (RBF) receives the highest amount of improvement through the most structured learning process, guided by *Strategy 4* (*simplification*, *fading*, *transfer of responsibility* and *ongoing diagnosis & assessment*).

Nevertheless, the results are indicating that, out of all four strategies, *Strategy 4* — embedding all four key characteristics of scaffolding — is the most stable technique. It leads to the learning processes with the highest global reward for both linear Q-learners and also provides the fastest learner for the real-valued and the second fastest learner for the binary representation. Although the deep learner is neither achieving the fastest learning nor the learning process with the highest global reward when utilizing this strategy, the results are nonetheless outperforming *Strategy 1* and *Strategy 2*. When employing *Strategy 3* together with the cosine metric, the deep learner is only reaching the performance threshold for two out of five source task configurations (i. e. $N_{\text{source}} = 25$ and $N_{\text{source}} = 100$). Using the cosine metric in *Strategy 4,* three out of five configurations are reaching the threshold. Moreover, *Strategy 4* seems also to be the most stable approach that leads to good learning performances when a high number of source tasks N_{source} is used.

Conclusion While the proposed approach was evaluated only within a simplified but physically realistic toy world, the obtained results can be viewed as encouraging for bringing the general idea to speed up the learning of object interaction scenarios by efficiently pre-learning relevant sub-skills to more complex 3D domains and also to real-world robot learning tasks. A straightforward *simplification* approach (*Strategy 1*) was already able to lead to a visible speed up of the learning process and an improvement of the learning performance. Further structuring the source task selection with more techniques that are related to scaffolding (*Strategy 2-4*) led to an additional performance gain, totalling to a speed-up of almost one order of magnitude for attaining nearly optimal performance. The impact strength of the advanced strategies is however depending on the used kind of function approximator. Additionally, the results direct to the conclusion that the impact of the correlation metric is also depending on the applied state representation. The fastest approach for the FSR was given by relying on a fixed set of source tasks (*Strategy 2*). As all source-tasks within the set (regardless of their difficulty) have to be solved by the agent, a *fading* like process is created. *Strategy 3* is then advancing the generation process of source-tasks by distributing the tasks and thus including a smoother transfer of responsibility to the artificial learner. Combined with the *cosine correlation metric*, this strategy leads to the fastest learner when training with the deep learning model. When learning with an RBF representation, *Strategy 4* combined with the *Pearson correlation metric* is the best choice for achieving a fast learning of the task. It replaces the uniform distribution of choosing the next source task by a deterministic one that selects the source task which is most similar to the last solved, adding a form of *ongoing diagnosis & assessment*.

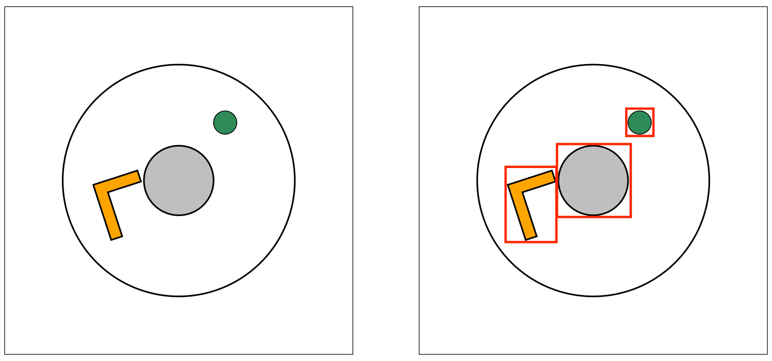
As a last remark it should be mentioned that the performed studies are indicating a dependency of the scaffolds effectiveness not only on the used kind of representation but also on the quality of the utilized baseline. If the learning performance of the agent is excellent from the start, the learning speed-up of the proposed learning strategies is much smaller as for the case when, for example good hyperparameters are not well known.

Future work There are many open questions that could be investigated in future work, like the impact of the presented or similar transfer learning strategies on learning algorithms different from Q-learning. In this work, the number of key aspects of scaffolding is increased from *Strategy 1* to *Strategy 4*. It was, however, not tested how the learning is influenced when combining two or three key aspects in different combinations. In addition the presented approach has at least two more issues that have yet to be solved. The first problem is that one does not know beforehand which correlation function is the best. The second problem is that the learning performance is also depending on the number of source-tasks N_{source}. While the results are encouraging the assumption that a choice $N_{\text{source}} \in [1, 100]$ should lead to a visible improvement of the learning process, the best N_{source} has to be found by testing.

The next step This chapter concludes the application of a scaffold based on transfer learning and curriculum learning on the process of learning an "Extension-of-Reach Scenario" with distance-related sensory input. This kind of input is leading to a low dimensional sensor-vector that is nevertheless able to characterize the state of the world well enough for solving the given problem. It is, however, not a very general way to describe the world and might also be complicated to be realized in reality as a set of sensors has to be set up in order to measure the related distances between the objects. A much better way, which is often used nowadays, would be to exploit the raw visual information of the environment. As this kind of high dimensional sensory information contains much more information than the distance related sensory-vector, larger models and longer training times are required to process the information. The next chapter is discussing one way to learn the "Tool-Centered Interaction Scenario" with this kind of sensory input while employing the *recurrent attention asynchronous advantage actor-critic (RAA3C) model* (see again Chapter 5) in order to reduce the necessary amount of visual information that has to be processed for achieving a satisfying learning performance.

Scaffolding the learning process through “active visual perception”: an attention based approach

In the last years, a rising trend of resorting to raw camera images as a more general way for artificial agents to perceive their environment could be observed in the machine learning community. Examples can not only be found in robotics [42, 43, 246, 247], but also when learning in computer- and board-game environments [7, 31, 32, 38, 39, 168, 248]. However, one common problem that appears when learning tasks using visual data is that most approaches are processing the raw image of the whole scene instead of concentrating only on the salient features that are important for solving the current task, as for example shown in Figure 11.1. This leads to learning architectures with huge amounts of weights and thus to long training times as larger features have to be processed.



(a) The Simulation world

(b) Highlighting the important parts of the environment

Fig. 11.1.: The figures illustrate the difference between processing the whole scene and processing only the salient parts of the environment for a mediated-interaction task within the designed simulation world. While (a) illustrates the whole simulation world, (b) highlights the salient parts, necessary for solving the task.

Scaffolding the learning process by efficiently reducing the information load A possible solution for this problem was proposed in Chapter 5. The designed RAA3C model is an *asynchronous advantage actor-critic architecture* that resorts to a *recurrent model of visual attention* as **a permanent scaffold for efficiently reducing the amount of information the agent needs to process for solving the given task by adapting its visual attention**. Therefore, the RAA3C has the ability to learn to actively search the visual scene for salient parts while relying only on a small glimpse-like pixel window. During this search, the information contained within the glimpses are stored in a memory network, built out of multiple LSTM networks. The combined features are then used for learning the given task.

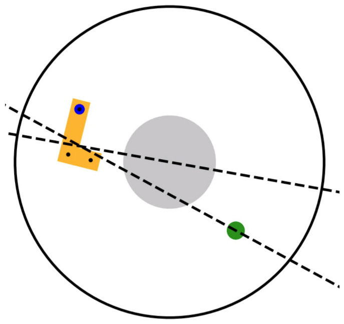
This chapter is now testing the RAA3C model by learning a mediated-interaction scenario within the designed simulation world in order to answer the question:

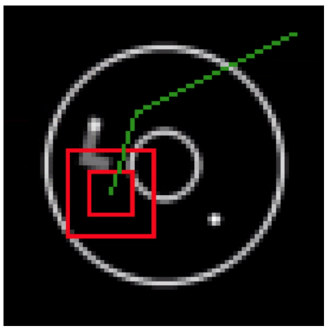
> Is it possible to scaffold the learning process of complex learning problems by using *visual attention* in order to implement *active perception* into the learning process of artificial agents that rely on visual input as its sensory information?

Learning scenario To test the efficiency of the RAA3C architecture, the learner has to solve the "Tool-Centered Interaction Scenario" that was explained in detail in Section 8.4. In this scenario, the tool and the target are both sampled from uniform distributions within the interaction range of the agent at the beginning of the episode. The agent, however, is only able to control the tool in order to move the target into the defined goal area. Thus, the agent has to learn where to look in order to find and identify both the tool and the target using a limited number of glimpses.

Again, the "Target-Tool-Goal System" (see Figure 9.1) is used as the underlying movement frame. The distance-related sensory input (8.1) that was used in the last chapters is now replaced by a raw visual image of the scenery as explained in Section 8.3. The simulation world (Figure 11.2a) is thus represented via a downscaled 84×84 pixel image that is processed through a grayscale filter, followed by a color inversion. In the resulting image, as for example given by Figure 11.2b, the salient features of the domain are presented by grey and white pixels, while the empty space is black. When exploiting the whole image of the environment, the agent would have to process 7056 pixels, which contain a lot more information when compared to the previously utilized distance-related input.

At each learning step, the agent is able to collect a fixed number of glimpses by moving its virtual eye over the image of the current scene as illustrated in Figure 11.2. After successfully solving the exercise or exceeding a fixed limit of 100 interaction-steps, the task starts anew with different object positions. When the agent is able to solve the task by navigating the target object into the goal area via pushing or pulling it with the tool, it receives a reward of $r = 1$ and $r = 0$ else[1]. Additionally, the agent gets a *contact reward* of $r_{\text{contact}} = 0.1$ *once* every episode for the first contact of the target and the tool.


(a) Simulation world


(b) Inverted grey scale image of the environment

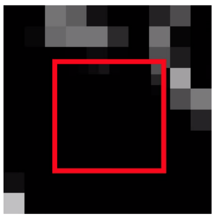

(c) The corresponding glimpse

Fig. 11.2.: Image (a) illustrates the simulation world. The blue dot (currently placed on the tool) indicates the current active picking location that the agent is using. Image (b) shows a preprocessed version of the domain, in which the agent selects the glimpses. The green line visualizes the movement history of the agents artificial eye for collecting previous glimpses. Image (c) shows the current glimpse.

11.1 Experiments

By learning to solve the described "Tool-Centered Interaction Scenario", the quality of the designed RAA3C model is tested. In particular, the following questions were addressed:

- Is the RAA3C architecture able to learn the given task, although it receives only limited information about the environment through the glimpses?
- Is the developed policy for creating the next glimpses better than a simple random policy?

[1]Although the reinforcement learning approaches in Chapter 9 and 10 were successfully trained with a reward of $r = 10$, it seems better to use a smaller reward for more complex models like the RAA3C in order to prevent large gradients during training.

- How is the amount of information that is provided by the implemented *context network* influencing the learning process?

In a first experiment, the agent has to learn to solve the assigned tasks using 3, 6, 8 and 10 glimpses of 10×10 pixels. An image of the full state of the environment, scaled down to 40×40 pixels, is taken as the input image for the context network[2]. In order to test the efficiency of the developed policy for creating the glimpses, the model is trained again while using a random policy for generating locations for the next glimpse. For these training runs, the weighting parameter η that controls how much capacity should be allocated for learning, the classification and how much for learning the location policy is set to 0. Thus, the term that is rating the current location policy is omitted within the update rule (5.1). Next, the model is trained again with 6 glimpses, while additionally using images with 10×10 pixels and 20×20 pixels as the input for the context network[3] and also omitting the context network by simply initializing the corresponding LSTM network with zeros in order to test its impact on the learning performance. As a last step, the learning performance is tested when the context network is not used to initialize *LSTM 3*, but *LSTM 2* or *LSTM 1*.

Training The model is always trained for 8000 episodes[4]. It handles 8 workers that are independently exploring their own copy of the environment and computing their own gradient using an experience buffer with a maximal size of $N_{\mathrm{EB}} = 8$. The master network then updates its weights using the *Adam optimizer* [67]. Every 10 episodes, each worker receives a new copy of weights from the master network. More information about the hyperparameters used for learning can be found in Appendix B.3.

Evaluation For evaluating the efficiency of the learning processes, the average reward per episode $\langle R \rangle$ that is achieved by the agent is measured. In contrast to the last experiments, $\langle R \rangle$ is now taken as a function of episodes instead of learning steps. In order to compute $\langle R \rangle$, the learning performance under the current policy is evaluated over 100 episodes for each of the 11 evaluated data points. At the end, four learning runs are averaged, where the standard deviation of the mean is used as the error.

[2]A short video `RAA3C.mp4` of the learned policy for the model using 6 glimpses and a 40×40 context image can be found in the supplementary material of this thesis (see also Appendix D).

[3]A short reminder: The features of the context network are used to initialize the hidden states of the last LSTM within the memory network. The generated features are employed only by the location network and not by the action network.

[4]As the episodes are limited to 100 steps, the agent is trained for $\leq 8 \cdot 10^5$ steps per worker.

11.2 Results

The results for solving the designed learning problem while using 3, 6, 8 and 10 glimpses are presented in Figure 11.3. While the four configurations are starting with a very poor policy, they all seem to reach a stable success rate after about 2000 episodes. The average reward of each configuration is depending on the used number of glances. While the model is receiving an average reward of about $\langle R \rangle = 0.2$ when 3 glances are used, the one with 6 glances is able to achieve a maximal average reward of $\langle R \rangle \approx 0.8$, corresponding to an average success rate of roughly 70%. When trained on 8 glimpses, the average reward drops down again to about $\langle R \rangle \approx 0.45$. The performance of the model is further dropping down for 10 glimpses, leading to the learning runs with the lowest learning performance of $\langle R \rangle \approx 0.15$.

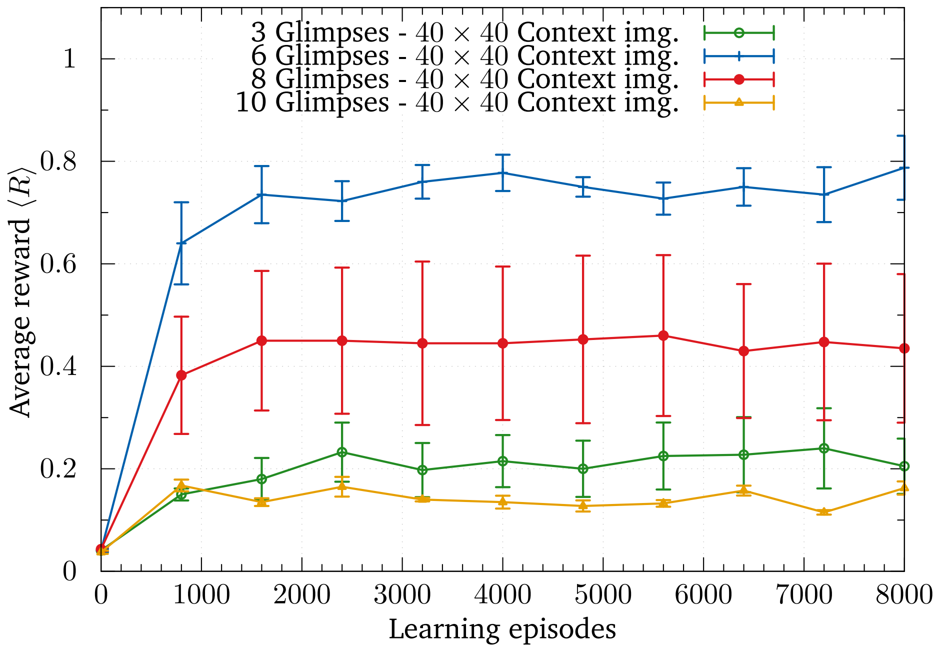


Fig. 11.3.: The average reward $\langle R \rangle$ of the proposed model for 3, 6, 8 and 10 glimpses per state s_t is plotted over the number of training episodes.

Table 11.1 is comparing the learning performance for the models that are using the location network for learning to generate the position of the next glance and the models using a random policy for generating the location of all glances. Therefore, the highest average reward $\langle R \rangle$ that was measured during the performance runs is listed. Additionally, Figure 11.4 visualizes the whole learning runs for 6 and 8 glimpses.

While for 3 glances, the model with the random policy leads just to a slightly worse learning performance, a huge gap can be observed for 6 glances. In this case, the learning performance for this configuration is halved when a random policy is used. For 8 and 10 glances, the learning performances for both location policies — the learned and the random — are nearly on the same level.

	Highest $\langle R \rangle$	
# **Glimpses**	$\pi_{\mathbf{loc}}$	$\pi_{\mathbf{rloc}}$
3	0.24 ± 0.078	0.194 ± 0.028
6	0.788 ± 0.063	0.383 ± 0.116
8	0.465 ± 0.154	0.480 ± 0.156
10	0.168 ± 0.011	0.243 ± 0.069

Tab. 11.1.: The table lists the highest average reward $\langle R \rangle$ for the training runs with the learned location policy $\pi_{\mathbf{loc}}$ and the random location policy $\pi_{\mathbf{rloc}}$.

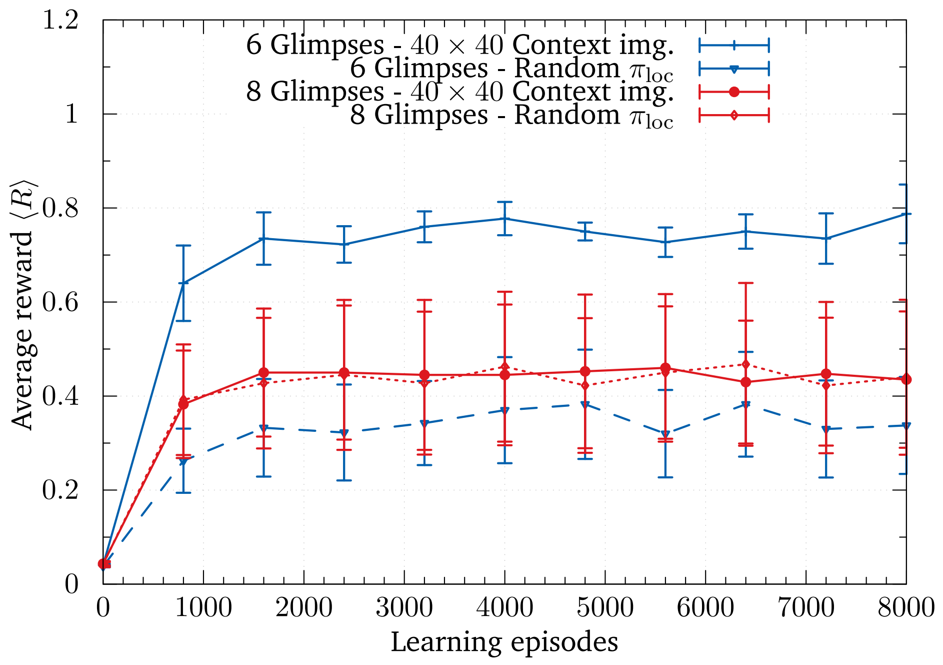


Fig. 11.4.: The graph compares the learning performance of the models with the case when a random location policy is used.

In order to test the influence of the context network on the learning process of the model, Figure 11.5 shows the learning performance for the agent, using a fixed number of 6 glimpses, while varying the size of the context image and also omitting the context network. Without the context network and initializing the corresponding LSTM network also with zeros, the learning seems to stop at an average reward of $\langle R\rangle \approx 0.18$. While a context image of 10×10 pixels leads only to an average reward of $\langle R\rangle \approx 0.3$, it can be raised to $\langle R\rangle \approx 0.5$ by doubling the image size. By again doubling the size of the image, the learning performance of the model further increases and now achieves an average reward of $\langle R\rangle \approx 0.8$, known from the previous results.

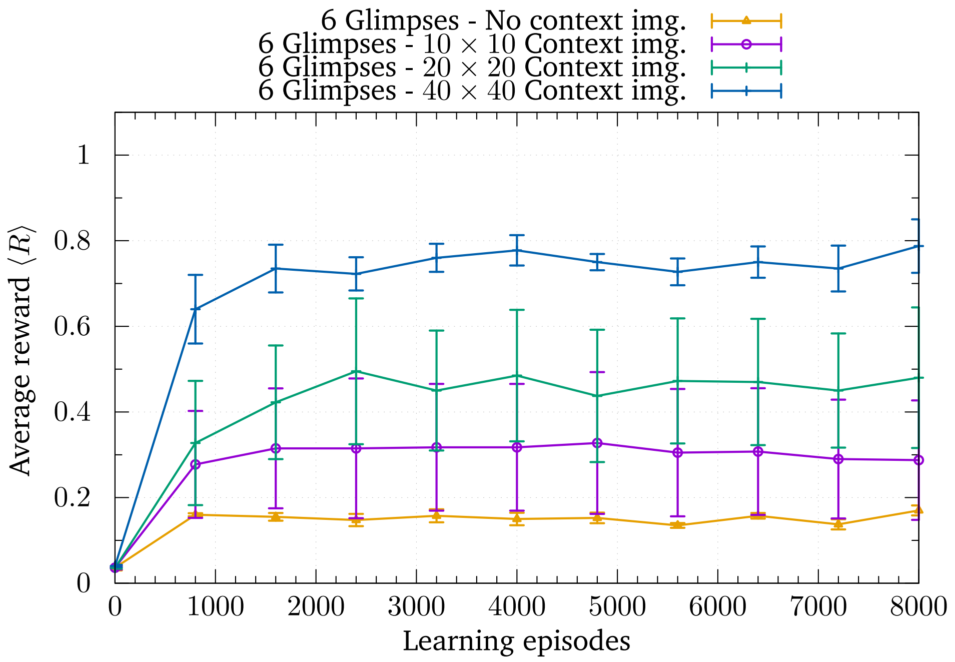


Fig. 11.5.: The graph shows the learning performance using 6 glimpses for different sizes of context images and also when the context network is omitted.

In addition to the previous results, Figure 11.6 illustrates the performance of the learning runs in which the LSTM that is initialized by the context network is varied. Although one might expect an improvement of the learning performance when the context network is used to initialize *LSTM 1* or *LSTM 2*, both configurations are leading to a performance drop of about 25% and more.

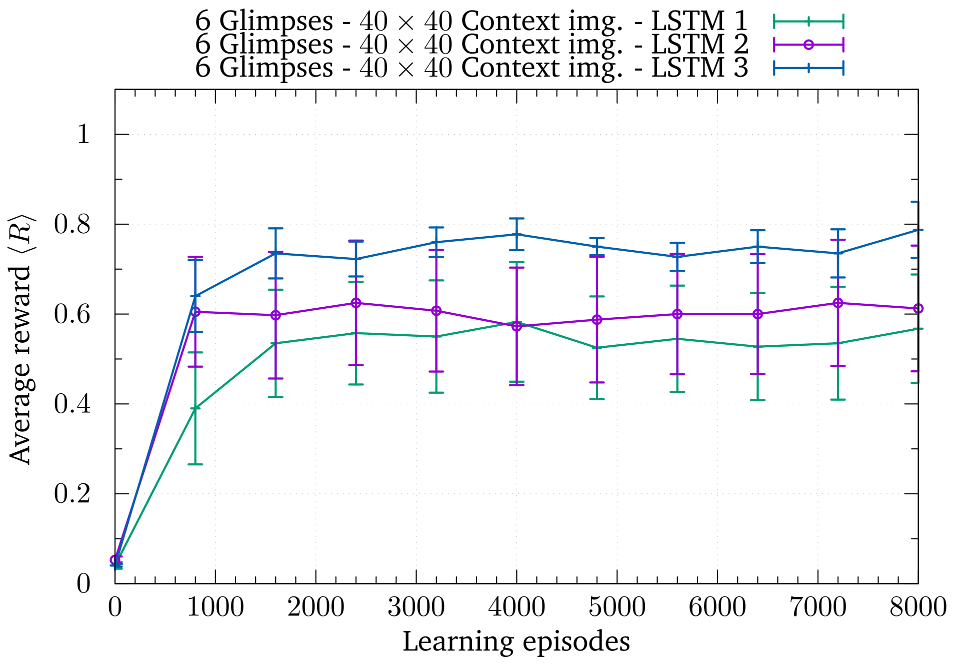


Fig. 11.6.: The graph shows the learning performance using 6 glimpses when the context network is used to initialize the internal states of *LSTM 1-3*.

11.3 Discussion

This chapter evaluates the proposed approach for scaffolding vision-based reinforcement learning problems by reducing the amount of information the agent needs to process for solving the given task. The designed *recurrent attention advantage actor-critic model* takes a sequence of fovea-like glimpses as an input instead of the high-dimensional image of the whole state of the environment.

Is the RAA3C architecture able to learn the given mediated interaction task, although it receives only limited information about the environment through the glimpses? The artificial agent had to learn to solve the "Tool-Centered Interaction Scenario", presented in Section 8.4, while relying on raw visual data as the sensory input. Although the policies that are learned within the 8000 episodes are all not optimal, the result for 6 glances can be seen as a learning success. The first results within this chapter are demonstrating that it is able to solve the given task with an average success rate of about 70%. Other observations indicate that less glimpses seems not to be able to provide enough information to generate a good picture of the current state while too many glimpses exceed the memory networks capacity and are again undermining the learning.

As each glimpse is built out of two pixel patches with a size of 10×10 pixels, the RAA3C needs only to process a sequence of sensory inputs of size 200 plus the glimpse location. The number of input features is thus about 35 times smaller than the number of features when using the original 84×84 grayscale image. The price that has to be paid is given by the recurrent *memory network* of the RAA3C that is required to store and filter the information of the executed glimpses that are collected during each training step.

Is the learned location policy for creating the next glimpses better than a simple random policy? An interesting observation that can be made when visualizing the agent's learned behaviour is that it often takes glimpses at locations where only black pixels are present. While this looks like an inefficient or random behaviour at the first glance, the conducted experiments are demonstrating that the learned policy $\pi_{\mathbf{loc}}$ for generating the glimpses is outperforming a random policy. Hence, $\pi_{\mathbf{loc}}$ is a necessary part for learning to solve the given task. It is also a signal that the agent needs the confirmation that both objects are absent at the specific location in order to generate a sufficient model of the present state of the environment.

How is the amount of information that is provided by the implemented context network influencing the learning process? One important ingredient of the architecture is the context network. The results are indicating that a bigger context image leads to a better location policy, although it initializes just the hidden states of one LSTM network that is also connected only to the location network and not to the action network. Additionally, its information content seems to strongly influence the quality of the location policy. As the input to the context network is a rescaled version of the whole domain and the resolution of this image seems to influence the resulting location policy, one might argue that the RAA3C also has to rely on a high high-dimensional sensory input for achieving satisfying learning results. The context image is, however, processed only through one linear layer that is downscaling the resulting feature vector to 64 dimensions. The features for learning the action policy are solely generated by using the information from the glimpses. It is also an interesting observation that the learning process gets worse when the context network is used to initialize the internal states of an LSTM network other than *LSTM 3*.

Conclusion While the model still needs to be improved further, this work is a promising starting point for further investigation of reinforcement learning using an attention guided visual input. As the average success rate of about 80% is not optimal, it might be possible to improve it either by training the model for more than 8000 episodes or by searching for a better set of hyperparameters. While in the given approach only linear layers and LSTMs are used, it is an interesting next step to integrate convolutional layers as in [218] in order to improve the models performance. Another idea is to test different kinds of recurrent modules like the *gating recurrent unit* [77, 78]. A further direction could also be to think about a way to combine the *recurrent model of visual attention* with other kinds of reinforcement learning algorithms like a *deep Q-learner* [7].

The next step The presented findings show a way to shape the learning process by reducing the amount of information the agent needs to handle through the replacement of the environment's full visual input by a sequence of small glimpses. The idea is not restricted to the field of computer vision. As discussed in Chapter 6, it can also be ported to the field of "haptic robotics" in order to enable a robot to perceive its surroundings by relying only on sequential tactile sensor measurements. In the next chapter, this approach is tested within a simulated robotics environment.

12 A scaffold for enabling “active haptic perception”: learning efficient haptic exploration

In addition to the ability to process visual information, a second key skill for both robots and humans to discriminate and handle unknown or recognize familiar objects is given by haptic exploration. From early on, humans reliably acquire sophisticated sensorimotor capabilities for active exploratory touch and directed manual exploration that associate surfaces and object properties with their spatial locations. The lack of good real-world interaction models, along with very restricted sensors and a scarcity of suitable training data to leverage machine learning methods has so far rendered haptic exploration a largely underdeveloped skill for robots. This is in marked contrast to vision, where deep learning approaches and an abundance of available training data have triggered huge advances.

Based on the observation that visual perception is a touch-like process [136], Chapter 6 has introduced an approach that enables a robot to learn efficient haptic exploration. It is not only inspired by human perception, but also exploits again a recurrent attention model as its basis in order to effectively extract and accumulate the information from a sequence of tactile data. The current chapter employs this approach in order to create **a scaffold for efficient haptic exploration via facilitating active haptic perception** by putting the designed *haptic attention model* (HAM) to use. In order to test the efficiency of the model, a simulated robot learns to explore and to classify different kinds of geometric objects.

Outline The focus of this chapter lies on answering the question:

> Is it possible to scaffold the learning process of haptic problems by using *recurrent attention* in order to learn synthetic exploratory procedures for robots by optimizing motor control?

Therefore, the developed simulation setup is described at first. The following section then addresses the coupling of the proposed HAM. After giving a detailed explanation of the conducted experiments, the results are presented and then discussed.

12.1 Designing the simulation world

The experimental setup is designed to acquire tactile signals with a tactile sensor array mounted on a robot arm, while exploring a stimulus. It inspired by the concept of modelling the functionality of a finger performing haptic interaction with fingertip-sized objects, similar to the approach presented in [229]. The simulation environment is created using `Gazebo`[1] as illustrated in Figure 12.1. The Communication of the different parts within the simulation, including the HAM, is performed via a `ROS-interface`[2].

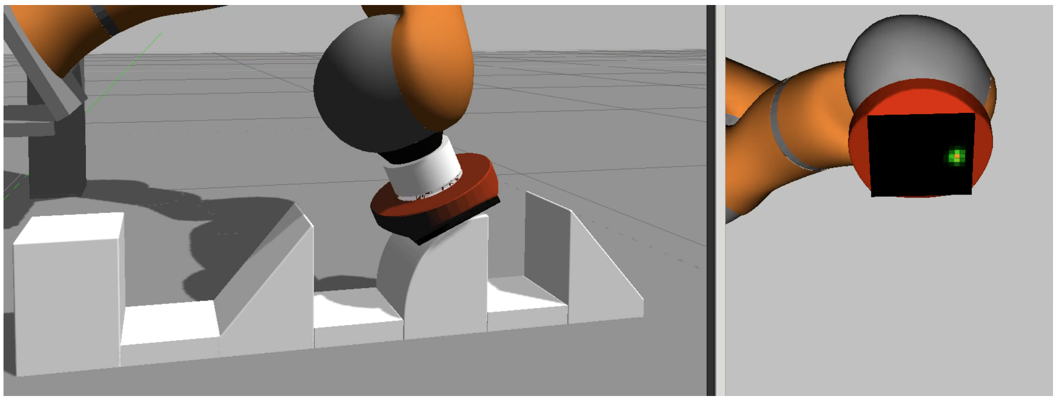


Fig. 12.1.: Simulation of the KUKA robot arm (7 degrees of freedom) with the Myrmex sensor array attached to the end-effector. On the left side of the image, the robot performs a haptic glance by establishing contact with one of the objects. The right side shows the corresponding visualization of the resulting tactile measurement.

12.1.1 The three building blocks of the simulation world

The simulation environment is constructed out of three parts: the tactile sensor, the stimulus material and the robot.

Tactile sensor For exploring the different stimuli the simulated tactile sensor is built out of a square-shaped 16×16 array of pressure-sensitive elements. The design matches the tactile sensor, called *Myrmex*, that was presented in [249]. The Myrmex provides a high-speed data acquisition and exhibits a very low first-contact threshold. Contacts at collision are estimated by Gazebo's physics engine `ODE` according to inter-penetration of objects (intrinsic compliance) and to default local surface parameters. Each contact defined by its position

[1] `http://gazebosim.org/`
[2] `http://www.ros.org/`

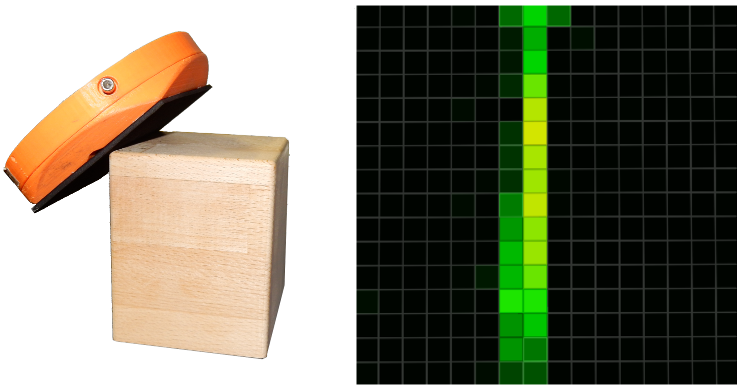
and force vector generates a Gaussian distribution around the contact center with amplitude depending only on the normal force. The standard deviation is arbitrarily fixed to mimic the deformation of the sensitive foam on the real sensor. As a result, the simulated version of the sensor emulates the contact distribution over the array through the use of a mixture of Gaussian distributions around the contact points obtained through Gazebo, weighted according to the local contact force. Mixing the distributions creates a 16×16 tactile pressure image, that is represented as an array of floating point values contrary to the real sensor with only 4096 levels of pressure.

In Figure 12.2 the tactile image for a contact with an edge are shown for the real Myrmex sensor and the simulated sensor. Due to the limitations of the collision library `libccd` used by the ODE simulation engine of Gazebo, only two contact points are generated at a time[3]. Consequently, it is not possible to produce an edge in the resulting tactile image. On the contrary, the real sensor produces a tactile image in which the expected line of contact is visible. When therefore the collision with an edge is measured, as it is illustrated in Figure 12.2a, the expected pressure profile would have the shape of a line. However, due to the limitations of the collision library, it will appear only as two contact points in simulation as shown in Figure 12.2b.

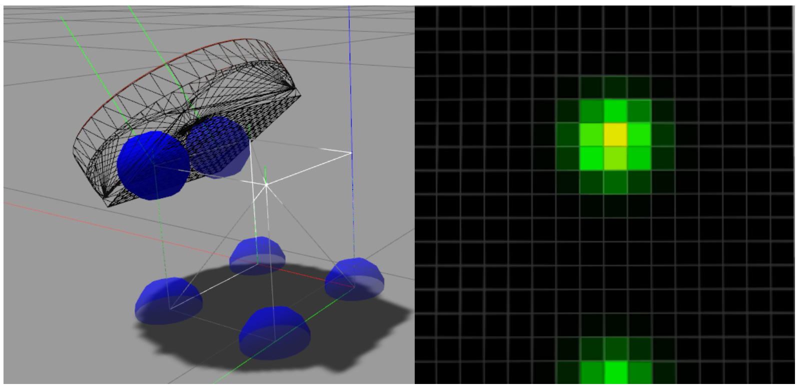
Stimulus material The second building block is a static set of the 3D objects that should be explored and then classified. Therefore, each of the objects is assigned to a different class label that has to be identified correctly by the robot. As this stimulus material should, like the tactile sensor, exist both in simulation and as a real-world object, building blocks from the "Modular Haptic Stimulus Board"[4], introduced by Moringen et al. [250, 251], are employed. Figure 12.3 shows an example of four object classes that were also ported in the simulated experiment as shown in Figure 12.1.

[3]A short explanatory video `HAM_1.mp4` can be found within the supplementary material of this thesis (see also Appendix D).

[4]An explanatory video of the "Modular Haptic Stimulus Board" can be found at `https://www.youtube.com/watch?v=CftpCCrIAuw`.



(a) Collision detection — real Myrmex sensor



(b) Collision detection — simulated Myrmex sensor

Fig. 12.2.: A comparison of the measurements between simulated and real Myrmex sensor when contacting an edge. (a) shows the tactile image of the simulated Myrmex. (b) shows the real sensor and the measured tactile image.

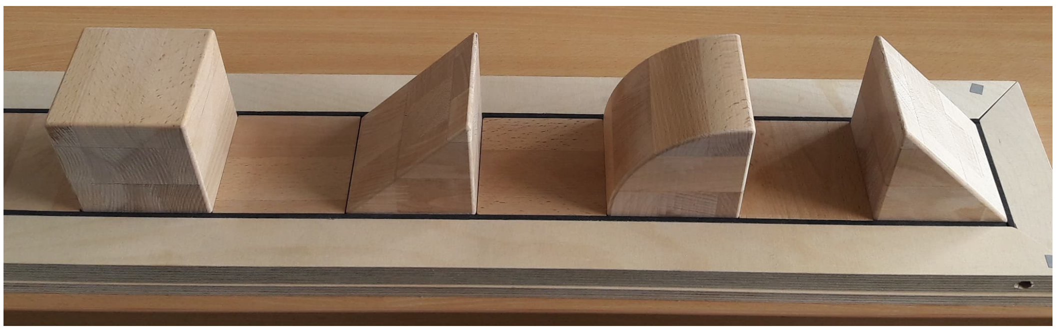


Fig. 12.3.: Exemplary objects employed in an identification task.

Robot arm The robotic setup consist of a KUKA LRW4 robot arm with 7 degrees of freedom ensuring a range of motion similar to a human arm. Its end-effector is equipped with the tactile sensor Myrmex mounted on an ATI force-torque sensor as shown in Fig. 12.1. The whole robotic system was recreated in simulation, using Gazebo and a simulated light-weight robot controller providing impedance control in joint space. The real-time control loop consisting of a Cartesian controller and of a Cartesian trajectory controller (interpolating motions and monitoring deviation), is exactly the same for the real world robot, and permits to validate the safety mechanism and the algorithms in the virtual environment first.

The purpose of the setup is to explore the stimuli with the sensing surface in a safe manner. Robotic interactions with the environment always require great care to avoid damage due to unintended high contact forces. Therefore, unplanned contacts are usually not desired. Since the exploration procedure is guided by the learning system, the various sensor poses executed on the robot are not known in advance. Moreover, for more realism, the shapes to explore are also unknown, which forbids any planning for obstacle avoidance. Hence, the robot arm should move and rely on events to react accordingly when touching the environment. The motion of the robot is stopped by a tactile event, which is a successful data acquisition. It is also stopped by a too high force on contact between the end-effector and the environment, or by a too large deviation between the desired joint target and the actual in case of a contact with other robot body parts, both latter events being considered as a failed data acquisition.

12.1.2 Implementing essential control primitives

Corresponding to the definition of the *haptic glance* in Section 6.1, it is implemented as a movement downwards towards the object, while sustaining a given pose, until a contact is established. The target pose of the tactile sensor can be described by a vector $\boldsymbol{l} = (x_g, y_g, z_g, e_1, e_2, e_3)$ that is constructed out of the three variables x_g, y_g and z_g that are defining its position in the global coordinate frame and the three Euler angles (e_1, e_2, e_3) defining its orientation. As a further simplification of the designed simulation framework only two of the six parameters can be modified by the learner: the position along the x-axis (x_g) and the angle around the y-axis (e_2). For the sake of readability, the alterable position x_g is called x and the angle e_2 is called φ in the following text. Thus, each haptic glance that is executed within the Gazebo simulation is represented by a pose (x, φ) of the tactile sensor. The remaining variables y_g, e_1 and e_3 are staying constant, while z_g is controlled by an external *haptic glance controller*.

Haptic glance controller The *haptic glance controller* is the interface between the learning framework and the robot simulation. It is designed as a state machine that receives a target pose for each individual haptic glance from the learner and is in charge of actually moving the sensor within the simulation world, stopping it when colliding with objects and resetting its position.

The new exploration pose is then executed by following a sequence of three states.

1. The sensor is moved to the pose (x, φ) above the objects, while z remains at the constant pre-defined level.
2. A slow downwards motion is queried, while monitoring the high-force, deviation and tactile pressure events.
3. On any of the events, the state-machine switches to the last state, moving the sensor away from the object. In the case of a tactile event, the data is transmitted back to the HAM, completing one haptic glance.

12.1.3 The classification task

The full experimental setup is visualized in Figure 12.4. During training and classification, one out of the four objects is always presented to the agent within its associated exploration zone.

Exploration zones Exploration zones are pre-defined regions in front of the robot with their own local coordinate systems, in which the objects are placed for exploration. After specification of the exploration zone, two out of six pose parameters of the tactile sensor can be modified by the HAM: the position x along the x-axis within the coordinate frame of the associated exploration zone, and the orientation angle φ around the y-axis. Before the execution of a haptic glance, the sensor is placed at the specified pose. The height of the sensor is set to a predefined value that guarantees a free floating of the sensor above the given exploration zone without any collisions. In order to establish the contact, the sensor is moved down along the z-axis. A pressure vector $\boldsymbol{p}$ is recorded once any sensitive cell of the sensor reaches a pre-defined threshold.

Executing haptic glances The agent explores the restricted object space with the sensor by performing a predefined number of haptic glances. While the first glance is always random, all following ones are generated using the HAM. Therefore a new position $x \in [-1, 1]$ and a new orientation $\varphi \in [-0.3\pi, +0.3\pi]$ are provided to the agent. The coordinate pair (x, φ) is then transformed into the coordinate system of the active exploration zone and then passed to the haptic glance controller for generating the associated pressure vector $\boldsymbol{p}$. The tuple $(x, \varphi, \boldsymbol{p})$ is then used by the HAM, together with the already stored information of the previously executed glances, for either generating the location-orientation pair of the next haptic glace or for classifying the active object.

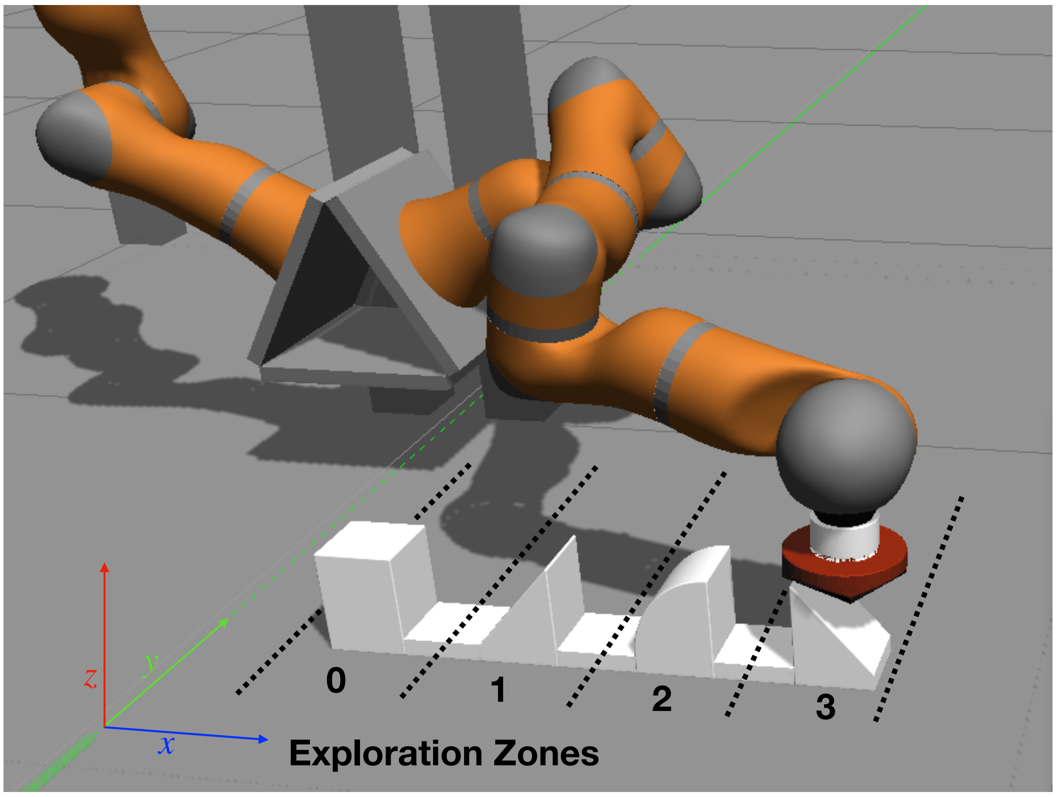


Fig. 12.4.: The experimental setup contains four objects whose positions are static. The Myrmex sensor gathers information about the objects by performing haptic glances at position x and orientation φ around around the y-axis.

12.1.4 Creation of the dataset

Learning within the simulation can be done only with a speed of $2\times$ real-time. It is however possible to speed up the learning by exploiting the fact that the sensor is interacting within the same regions of the location-orientation space multiple times, leading to almost the same pressure readings. In order to enable a fast and efficient evaluation of the model for different configurations of parameters like the number of used haptic glances, the whole location-orientation space that can be accessed by the sensor is tesselated. As a next step, a cache of haptic glances is generated that are then stored in a dataset $\mathcal{D}_o$ for learning.

For each object o, the dataset is produced by recording tuples $d_o = (\boldsymbol{p}, x, \varphi)$ of the normalized pressure data $\boldsymbol{p}$, together with the associated location x and orientation φ of the sensor. In order to generate data of each object that is independent of its positioning, the location data $x \in [-1, 1]$ is given within the location space of the exploration zone. After reaching the associated exploration zone with the robot, the recording of the data points starts at $x = -1$ with the orientation $\varphi = -0.3\pi$. The whole orientation space is now covered by recording φ while incrementing it with a step size of $\Delta_\varphi = \pi \cdot 0.05$. The location is then incremented by $\Delta_x = 0.05$ and the recording of the orientations starts anew at $\varphi = -0.3\pi$. The described procedure leads to 41×41 pre-recordings per object (41 orientation recordings per position) and full a dataset with a size of about 6724 data points.

During training, the model generates location-orientation pairs (x, φ) for which the associated pressure vector $\boldsymbol{p}$ is directly extracted from the dataset. This is performed by taking the data point d_o that best matches (x, φ), instead of re-measuring the pressure vector in simulation.

A larger dataset for better model evaluation In order to train the model within an admissible amount of time, the creation of the dataset using the full robot simulation is a necessary intermediate step. However, the production of the training data comes in line with some drawbacks. At first, it takes a lot of time to record it. A second problem is that the existence of noise within the pressure measurements is an inevitable phenomenon. The sensor either might get stuck at some edge or is not able to properly record the pressure data because of a disadvantageous pose. This and some other simulation-related issues are the reason for the recording of a relatively coarse dataset. It is a good choice for validating the functionality and generalizability of the designed haptic attention model. Furthermore, it is also necessary for creating a pre-trained model that can then be used for solving the classification task within the simulation environment. In spite of everything, it might not be a good choice for a more theoretical study of the model's features as analysing the characteristics of the different network modules. For the latter case, a larger and much finer dataset with $161 \cdot 10^3$ data points was created in [24] by omitting the robot arm within the simulation and also shrinking the (now floating) simulated Myrmex for a better exploration process. The most important results when training on the large dataset can be found in the Appendix C, while a detailed evaluation is given in [24].

12.2 Implementation of the haptic attention model

The implemented version of the *haptic attention model (HAM)* is proposed in Section 6.2 and illustrated in Figure 6.1. A vector $\boldsymbol{s} = (\boldsymbol{p}, x, \varphi)^\top$ consisting of the sensor pose (x, φ) and the associated pressure profile acquired by the Myrmex sensor while performing a haptic glance in Gazebo is used as the sensory input for the network. The 16×16 pressure matrix is flattened to a normalized pressure vector $\boldsymbol{p}$ with $\dim(\boldsymbol{p}) = 256$. First, the input is processed through the *tactile network*, which combines the recorded pressure profile $\boldsymbol{p}$ with its associated location x and orientation φ into one single feature vector. The features $\boldsymbol{s}$ are then propagated through the memory network. It then provides features to the *location network* that in turn generates a new pose. After a certain number of glances, the features that are generated by the LSTM are used by the *classification network* for predicting a class label for the current object. Although the classification of the object can be done within each haptic glance, the one after the final glance is usually the classification result of one's interest.

Training The HAM is trained using stochastic gradient descent with Nesterov momentum [61–63] and the update rules (6.1), (6.2) from Section 6.2.1. A detailed list of the used parameters can be found in the Appendix B.4, leading to a model with a total number of 741248 trainable weights. For each training step, a new batch of size 64 is generated, where the to-be-classified objects o are uniformly chosen from the set of available objects. It is important to notice that a used batch of training data is given only as a uniformly sampled list of objects. All data is then generated on the fly during training by performing the haptic glances within the object's "exploration zone". At first, all haptic glances are executed for one object, together with the classification and the computation of the gradients. The data is then saved, before interacting with the next object. If all objects of one batch are explored, the saved gradients are accumulated and used to update the model. This technique can be used for training on a robotic platform without further modifications.

The learning rate α_t decays exponentially every T training steps to $\alpha_{\min}$ with a decay-factor of δ_α according to

$$\alpha_t = \min\left(\alpha_{\min}, \alpha_0 \cdot e^{-\delta_\alpha t}\right) \text{ with } t = t + 1 \text{ every } T \text{ training steps.}$$

Table B.12 in Appendix B.4 lists the hyperparameters that are used for all experiments. The parameters are chosen according to *random search* [244] with a fixed number of 3 glances, followed by additional manual tuning. The weights of all layers are initialized using *He normal initialization* [252] with a bias of 0.

12.3 Experiments

The conducted experiments are split in two parts. In the first part, the HAM is trained by using the created dataset. It is evaluated how well the model is able to learn the classification task for different numbers of glances, while training only on the limited pre-recorded data. The model is always trained for 10^4 steps. In order to measure the performance after a certain number of training steps, the training is paused. Next, the accuracy for correctly classifying the given object in 100 newly generated batches using the currently available policy is estimated. To obtain a statistically correct measure of the accuracy, each experiment is repeated 10 times. For the final evaluation, the mean accuracy is averaged over all experiments with the standard deviation of the mean as the error.

In the second part, the trained model has to generalize beyond the recorded data by classifying the four objects within the fully-fledged simulation. Therefore, the best performing models are utilized, where each was trained on a specific number of glances. The agent has then to classify all of the four objects 20 times within the simulation. The resulting classification accuracy is then averaged over four distinct trials with the standard deviation of the mean as the error.

12.4 Results

The evaluated results for training and testing the *haptic attention model* on the pre-recorded dataset can be found in the second column of Table 12.1. Additionally, Figure 12.5 visualizes the learning performance for the training on 1, 2, 3 and 6 haptic glances. The model reaches an accuracy of about 84% after just one random glance. It then continuously improves when more glances can be executed. Granting the model just one more glance leads to an accuracy of more than 96%.

	Performance	
# Glances	Dataset	Simulation
1	0.849 ± 0.001	0.803 ± 0.0709
2	0.969 ± 0.001	0.906 ± 0.0225
3	0.988 ± 0.001	0.941 ± 0.014
6	0.994 ± 0.001	0.978 ± 0.018
8	0.997 ± 0.000	$\mathbf{0.997 \pm 0.005}$
10	0.998 ± 0.000	0.978 ± 0.018

Tab. 12.1.: List of measured classification performances, recorded in simulation using the best model, pre-trained on the recorded dataset. The results are averaged over 4 trials.

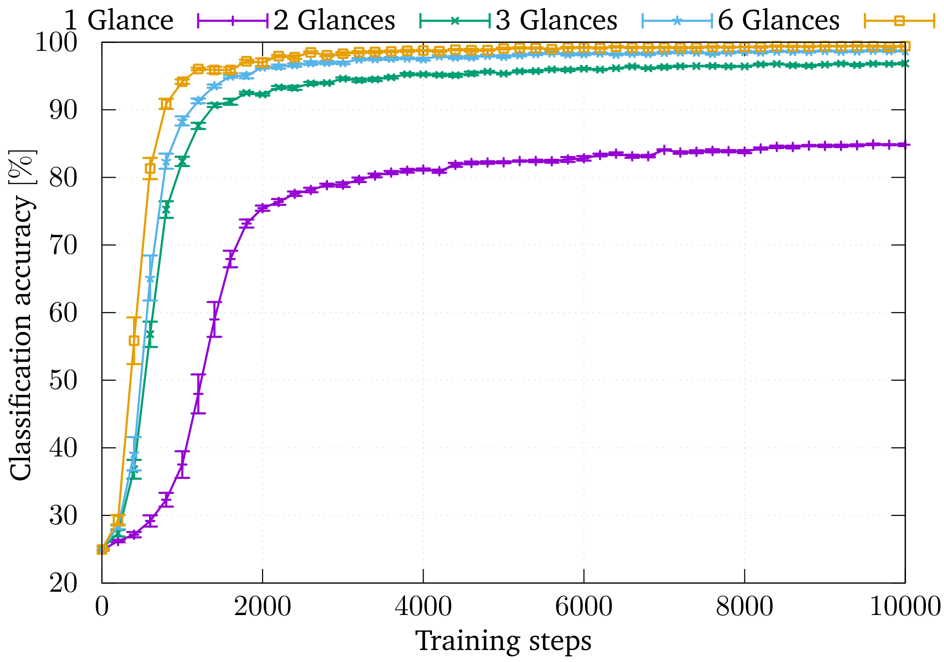


Fig. 12.5.: Classification accuracy of the designed approach. The classification accuracy during the training is visualized for the model while it is trained to classify using different numbers of glances.

Testing the model on a simulated robot arm with an attached tactile sensor After successfully training a model that is able to classify the four objects with high accuracy while using only the limited data of the pre-recorded dataset, the learned model is tested within simulation[5]. The results are listed in the last column of Table 12.1. It is now possible to compare the results evaluated within the simulation with the ones that are computed while relying only on the dataset. One can directly see that the performance measured within the simulation is smaller for less glances. It then gradually increases when more haptic glances can be applied and then reaches nearly the same performance for 8 glances as measured on the dataset. Also the classification accuracy in the simulation is smaller for 1 to 6 glances, even the use of one single haptic glance per object leads to an accuracy of about 80%. While adding a second glance increases the success rate about 10%, the third one adds only a gain of $\approx 4\%$ that further decreases with every additional glance added. Nevertheless, an accuracy of more than 99% can be reached for this simple task when 8 glances are used. For 10 glances, the accuracy is slightly dropping to about 98%.

12.5 Discussion

In this chapter, the theoretical ideas from Chapter 6 were tested to verify the hypothesis that existing attention mechanisms which are able to scaffold "active visual perception" can also be utilized for scaffolding the process of "active haptic perception" in tactile-based learning scenarios. Therefore, a variant of the "recurrent model of visual attention" was utilized in order to learn the control of sequences of haptic glances for efficiently classifying four different objects. A simulation of an actuated robot that is equipped with tactile sensing was exploited in order to verify the success of the approach with a setup mimicking haptic interaction performed with one human finger.

Main results Using the integrated attention mechanism of the HAM as a scaffold for sensor-control, the developed architecture is able to learn suitable *exploratory procedures* with respect to its own constraints and the spatio-temporal resolution of the acquired data. These results may be limited by the simplicity of the 3D shapes that are employed in the experiments. As the considered objects should be characterizable through a one-dimensional curvature, there are, however, not many other objects that could be chosen.

Despite a relatively small training set and training time with respect to the weights, the network shows good generalization performance as demonstrated by the results

[5]A short video `HAM_2.mp4` can be found within the supplementary material (also see Appendix D).

achieved in simulation. This is in line with findings in literature, where large numbers of weights are not necessary leading to overfitting [253]. Additionally, much time was spent on optimizing the hyperparameters of the model and the process of data acquisition.

Future work: towards more complex applications As the application of the presented method is not restricted to the chosen morphology, it can be applied to control more complex setups within future works. In this work, the implemented model has been tested with a classification front-end, i.e. the task of the HAM was to efficiently identify objects. Due to the modularity of its architecture, the object classifier can be substituted with a different front-end to perform other objectives, such as haptic search or fault diagnosis. Furthermore, it could be a possibility that an improved variant of the model can also be applied to perform not only contour exploration, but also other types of haptic exploration, such as squeezing for rigidity identification, or texture identification.

Future work: towards real-world learning Another stream of possible improvement would be to induce the transition from the simulated robot to a real-world setup. While it is possible to recreate the experimental setup while using a real KUKA robot[6] as shown in Figure 12.6, technical and safety issues have to be resolved. Additionally, a new dataset has to be created using the illustrated setup with predictable safe poses. Training with this real-world dataset should show how well the model can deal with the noise within the data that is inevitably present when working with a real robotic setup.

To enable a rigorous exploratory behavior, the original HAM model yields a stochastic location policy resulting in a behavior that could be described casually as jumpy. A first attempt to regulate this stochasticity for more smooth and efficient trajectories in Cartesian space as well as a more focused exploration of the local shape features can be found in [254].

Training solely on a pre-recorded data set might not be sufficient for more complicated tasks, while a full training within the online simulation is likely to be time consuming. One option could be to use a *transfer learning* approach [162] by first training the model on a pre-recorded dataset and then adding refinement to the learned policy by training the same model for a smaller number of training steps

[6]A short video `KUKA.mp4` that shows the early testing off the control scheme in a real world setup can be found within the supplementary material of this thesis (also see Appendix D).

directly on the simulated robot setup. Potentially, it might even be possible to employ a variant of the transfer learning approach that was presented in Chapter 7.

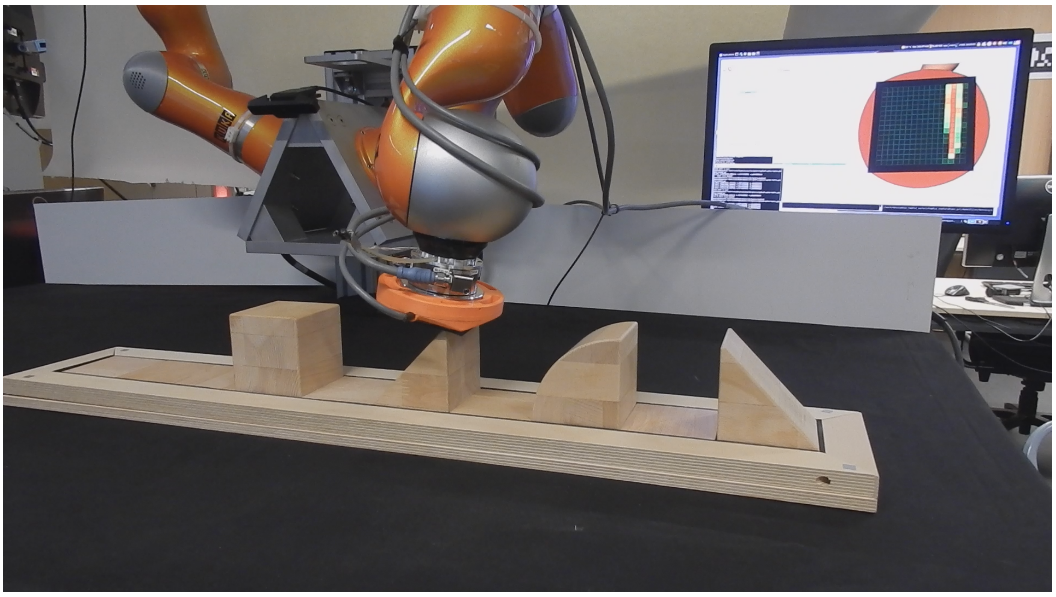


Fig. 12.6.: Photo of the real-world setup, where the robot arm is performing a haptic glance using the Myrmex sensor. The resulting pressure measurement is shown on the monitor in the background.

Part IV

Conclusion

13 Summary, conclusion & outlook

At present, the accustomed assistance of robots in daily life tasks is a challenging matter. Especially when the task forces the robot to physically interact with its surroundings and to adapt on changes, many demands have to be faced. Not only fast learning of new assigned duties is required, but also the ability to learn in ways that enable a human to easily instruct the robot when facing new problems. This thesis has addressed the described issues by analysing and exploiting the potential of a concept taken from educational psychology, called **scaffolding**. In that context, the main theme of this thesis was to **propose scaffolding as a general guiding principle that exploits available meta-knowledge from the field of machine learning for developing approaches that are able to improve and accelerate the learning process of artificial agents**. The emerging key-aspects were then exploited as a computational skeleton for a possible research agenda. As a result, **four new scaffolding approaches** were proposed, analysed and discussed throughout the remaining part of this thesis.

Outline The present chapter now summarizes the conclusions that were drawn out of the conducted studies (Section 13.1), followed by an overarching discussion in Section 13.2. In the end, the gathered information is used for proposing some continuative ideas for future research in Section 13.3.

13.1 Four scaffolding approaches — a summary

In Chapter 2, reinforcement learning was presented as a class of very efficient and frequently employed machine learning algorithms that were inspired by biological agents. Regardless of their effectiveness, they are also bound by common problems like long training times and the need of huge amounts of training data. As a possible remedy, scaffolding was first introduced as a theory from educational psychology in the beginning of Chapter 3. It was then related to different approaches in machine learning in order to show that many of the key concepts were already used and also led to promising results. Based on the gathered insights, a refined principle of scaffolding for artificial agents was formulated. Therefore, the original key aspects were either reinterpreted or — if needed — refined. Condensed in a mindmap (see Figure 3.5), the key aspects were identified as *refining the learner's attention*, *simplification*, *ongoing diagnosis and assessment*, *modelling & demonstration* and *fading & transfer of responsibility* and were thus not much differing from the five key aspects of the original theory. The proposed concept was then utilized for identifying parts of the learning process of artificial agents that were likely to provide a good starting point for scaffolds. These parts turned out to be the agent's perception of the learning domain on the one hand and the initial part of the learning process on the other hand. With this information in mind, four different kinds of scaffolds for reinforcement learners were designed, together with suitable testbeds:

Perceptive acting — Chapters 4 & 9 Finding good principles to choose the actions of artificial agents in the most beneficial way to optimize their control of the environment is an important aspect for facilitating a fruitful learning process. Especially in reinforcement learning, where the agent learns through the direct interaction with the environment, a good choice of actions is essential. A new approach was proposed that is scaffolding existing learning frameworks by choosing a favorable set of actions according to their *mutual information*. This **permanent scaffold for the learning process by selecting the most suitable action set** allows a predictive ranking of different action sets with regard to their influence on the learning performance of an artificial agent. In the conducted experiments, it was shown that a favorable choice of the action set is able to significantly improve the learning process without explicitly modifying the learning algorithm itself. It was then demonstrated that the mutual information-based measure can yield useful predictions on the aptitude of action sets for the time course of learning.

Active visual perception — Chapter 5 & 11 Humans are sensing the world in an active way. Instead of perceiving the environment as a whole image, only parts of the scenery in the form of image sections, called “glimpses”, are combined in order to get an accumulated understanding of the visible scene. This can be seen as a first indicator that *active visual perception* might be a promising approach for creating **a permanent scaffold that enables the integration of efficient active visual perception** when solving complex learning scenarios with reinforcement learning. As a result, a new approach was presented that embeds the perception of the environment through active visual perception into an *asynchronous advantage actor-critic architecture*. Instead of using the full visual information of the scene, the resulting model accumulates the foveal information of controlled glimpses and is thus able to reduce the complexity of the network. Using the designed model, an artificial agent was able to solve a challenging "mediated interaction scenario". To learn the given problem, the agent "actively" searches for salient points within the environment by taking a limited number of fovea-like glimpses. It then uses the accumulated information to decide which action to take next.

Active haptic perception — Chapter 6 & 12 In addition to the ability to process visual information, a second key skill for both robots and humans to discriminate and handle unknown or recognize familiar objects is given by haptic exploration. This part focuses on **designing a scaffold for learning one central and important haptic skill**: the discrimination of unknown object shapes through a sequence of actively controlled haptic contacts between a sensor and the object surface. The proposed *haptic attention architecture* simultaneously optimizes main perception-action loop components: feature extraction, integration of features over time, and the control strategy, while continuously acquiring new tactile data online. For testing, the designed haptic meta-controller was combined with an rigid tactile sensor array attached to a robot arm that moves in a physics-driven simulation environment. Performing object contour exploration that has been optimized for its own sensor morphology and classifying four differently shaped objects, it achieves results close to 100%.

Scaffolding the structure of the learning process — Chapter 7 & 10 Basic linear and deep reinforcement learning architectures are lacking the ability to simplify a given learning problem by splitting it into a hierarchy of simpler sub-tasks and then learn their essential sub-skills. Therefore, a scheme for **temporary scaffold the learning by refining the model's internal representation through the pre-training of sub-skills using a combined transfer and curriculum learning approach that is enriched by the developed key aspects of scaffolding** was proposed. The focus was set on the case where transfer learning is applied to generalize experiences from source tasks that are solvable through direct, unmediated interaction, to target tasks that require mediated interaction for their solution. The findings indicate that the designed scaffold, employed in this context, leads to a significant acceleration of learning. The invented approach can be used to easily extend and improve many existing learning architectures without the need to modify them.

13.2 Conclusion

This work has demonstrated that a refined concept of scaffolding can be employed as a guiding principle for developing new approaches in machine learning that support the training process of artificial agents on different problems. For this purpose, scaffolding in machine learning was formulated in a broader sense than it has been done in educational psychology.

In a following step, the new principle was then used to design four scaffolds that could either enable learning or improve the learning process of the assigned task within the constructed testbeds. These approaches are not only good candidates for new insights into the learning process of reinforcement learning agents. They might also be able to set the direction to new and interesting perspectives within the field of artificial learning.

Scaffolding the structure of the learning process by implementing refined key aspects taken from the original theory has successfully demonstrated its positive effect on the learning process. Additionally, it was highlighted that it is also a good approach to design not only temporary — as it was stated in the original psychological theory — but also permanent scaffolds that are centered on supporting specific parts of the learning process.

A second mentionable observation is the importance of perception for the learning process. While the way of perceiving the world is often seen as a distinct feature of the agent that is defined by the programmer, the conducted studies encourage the formulation of two statements: The first one is that a deeper study of available interaction strategies might lead to a choice of action primitives that are able to significantly improve the learning process as they augment the agent's perception (*perceptive acting*). Secondly, it seems also beneficial to design a deeper connection between the learning model and the collection of sensory input by integrating the ability to learn an efficient "active" perception of the environment into the learning model. This can not only be used as a way for reducing the information load of the often very high dimensional sensory input. It can also endow the agent with new abilities like learning to generate exploratory procedures for an efficient haptic exploration of the environment.

13.3 Recommendations for future research

While for the presented scaffolds some recommendations for further investigations were given within the individual discussions, it is also important to make suggestions for continuing the study of the hidden potential within the invented principle: *scaffolding the learning process of machines*. In this regard, an interesting idea would be to start a deeper study on how the different aspects of the theory are influencing the learning process of artificial learners: Which of them are the most promising ones in general for supporting the learning process of machines? What are the limitations? What can we learn from results achieved in similar experiments that are conducted not with artificial learners but with human participants? By designing experiments that can be compared to ones with human participants it might also be possible to study scaffolding in the opposite direction: Is it possible to speed up the learning process of humans by transferring scaffolds for machine learning into the human realm?

Within this work, only the scaffolding of perception and the internal representation of the model were discussed. Thus, as a last idea for further research, it could be a promising attempt to identify additional parts of the learning process as potential candidates for a supporting scaffold. They might again lead to valuable insights into the learning process and to new approaches that narrow the gap to a procedure enabling fast and save real-time learning for robots.

Bibliography

[1] Elisabetta Visalberghi and Loredana Trinca. "Tool use in capuchin monkeys: Distinguishing between performing and understanding". In: *Primates* 30.4 (Oct. 1989), pp. 511–521 (cit. on pp. 1, 99).

[2] Alexander Stoytchev. "Behavior-Grounded Representation of Tool Affordances". In: *IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3060–3065 (cit. on pp. 1, 99, 106).

[3] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, José Santos-Victor, and Luc De Raedt. "Learning relational affordance models for robots in multi-object manipulation tasks." In: *ICRA* (2012), pp. 4373–4378 (cit. on pp. 1, 99).

[4] Lorenzo Jamone, Emre Ugur, Angelo Cangelosi, et al. "Affordances in Psychology, Neuroscience, and Robotics: A Survey". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.1 (Mar. 2018), pp. 4–25 (cit. on pp. 1, 99).

[5] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. "Neuroscience-Inspired Artificial Intelligence". In: *Neuron* 95.2 (July 2017), pp. 245–258 (cit. on pp. 2, 30, 69).

[6] Andrew N. Meltzoff, Patricia K. Kuhl, Javier Movellan, and Terrence J. Sejnowski. "Foundations for a New Science of Learning". In: *Science* 325.5938 (July 2009), pp. 284–288 (cit. on p. 2).

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533 (cit. on pp. 2, 9, 25, 29, 69, 113, 145, 154, 206).

[8] Nuttapong Chentanez, Andrew G. Barto, and Satinder P Singh. "Intrinsically Motivated Reinforcement Learning". In: *Advances in Neural Information Processing Systems 17*. Ed. by L K Saul, Y Weiss, and L Bottou. MIT Press, 2005, pp. 1281–1288 (cit. on p. 2).

[9] Satinder Singh, Richard L. Lewis, Andrew G. Barto, and Jonathan Sorg. "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective". In: *IEEE Transactions on Autonomous Mental Development* 2.2 (June 2010), pp. 70–82 (cit. on p. 2).

[10] Alec Solway, Carlos Diuk, Natalia Córdova, et al. "Optimal Behavioral Hierarchy". In: *PLoS Comput Biol* 10.8 (Aug. 2014), e1003779–10 (cit. on p. 2).

[11] Emanuel Todorov. "Efficient computation of optimal actions". In: *Proceedings of the National Academy of Sciences* 106.28 (July 2009), pp. 11478–11483 (cit. on p. 2).

[12]Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, et al. “Unifying Count-Based Exploration and Intrinsic Motivation”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 1471–1479 (cit. on pp. 2, 42, 53).

[13]Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, et al. “Reinforcement Learning with Unsupervised Auxiliary Tasks”. In: *CoRR* abs/1611.05397 (2016). arXiv: 1611.05397 (cit. on pp. 2, 42, 53).

[14]Martin A. Riedmiller, Roland Hafner, Thomas Lampe, et al. “Learning by Playing - Solving Sparse Reward Tasks from Scratch”. In: *CoRR* abs/1802.10567 (2018). arXiv: 1802.10567 (cit. on p. 2).

[15]Jacob Andreas, Dan Klein, and Sergey Levine. “Modular Multitask Reinforcement Learning with Policy Sketches”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 166–175 (cit. on pp. 2, 45).

[16]Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, et al. “FeUdal Networks for Hierarchical Reinforcement Learning”. In: *CoRR* abs/1703.0 (2017) (cit. on pp. 2, 45).

[17]Jennifer Hammond. *Scaffolding: Teaching and learning in language and literacy education.* ERIC, 2001 (cit. on pp. 2, 31, 32).

[18]Jennifer Hammond and Pauline Gibbons. “What is scaffolding”. In: *Teachers’ Voices*. Ed. by Anne Bruns and Helen de Silva Joyce. National Centre for English Language Teaching and Research, Macquarie University, 2005, pp. 8–16 (cit. on pp. 2, 31, 32).

[19]Janneke van de Pol, Monique Volman, and Jos Beishuizen. “Scaffolding in Teacher–Student Interaction: A Decade of Research”. In: *Educational Psychology Review* 22.3 (2010), pp. 271–296 (cit. on pp. 2, 31–33).

[20]Janet Mannheimer Zydney. “Scaffolding”. In: *Encyclopedia of the Sciences of Learning*. Ed. by Norbert M Seel. Boston, MA: Springer US, 2012, pp. 2913–2916 (cit. on pp. 2, 31–33, 38).

[21]Sascha Fleer and Helge Ritter. “Comparing Action Sets: Mutual Information as a Measure of Control”. In: *Artificial Neural Networks and Machine Learning – ICANN 2017*. Cham: Springer International Publishing, Oct. 2017, pp. 68–75 (cit. on pp. 3, 4, 59, 118).

[22]Sascha Fleer and Helge Ritter. “Skill Transfer for Mediated Interaction Learning”. In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. Nov. 2018, pp. 1–8. DOI: 10.1109/HUMANOIDS.2018.8624951 (cit. on pp. 3, 4, 127).

[23]Sascha Fleer and Helge Ritter. “Solving a Tool-Based Interaction Task Using Deep Reinforcement Learning with Visual Attention”. In: *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization*. Ed. by Alfredo Vellido, Karina Gibert, Cecilio Angulo, and José David Martín Guerrero. Cham: Springer International Publishing, 2019, pp. 231–240 (cit. on pp. 3, 4, 71).

[24] Sascha Fleer, Alexandra Moringen, Roberta L. Klatzky, and Helge Ritter. “Learning efficient haptic shape exploration with a rigid tactile sensor array”. In: *PLOS ONE* 15.1 (Jan. 2020), pp. 1–22. DOI: `10.1371/journal.pone.0226880` (cit. on pp. 3, 4, 80, 162, 163, 211).

[25] Richard S. Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. second edtion. MIT Press, 2018 (cit. on pp. 9–11, 19, 46).

[26] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (July 1959), pp. 210–229. DOI: `10.1147/rd.33.0210` (cit. on p. 9).

[27] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress”. In: *IBM Journal of Research and Development* 11.6 (Nov. 1967), pp. 601–617. DOI: `10.1147/rd.116.0601` (cit. on p. 9).

[28] Gerald Tesauro. “TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play”. In: *Neural Computation* 6.2 (1994), pp. 215–219. DOI: `10.1162/neco.1994.6.2.215`. eprint: `https://doi.org/10.1162/neco.1994.6.2.215` (cit. on p. 9).

[29] Gerald Tesauro. “Temporal Difference Learning and TD-Gammon”. In: *Commun. ACM* 38.3 (Mar. 1995), pp. 58–68. DOI: `10.1145/203330.203343` (cit. on p. 9).

[30] Gerald Tesauro. “Programming backgammon using self-teaching neural nets”. In: *Artificial Intelligence* 134.1 (2002), pp. 181–199. DOI: `https://doi.org/10.1016/S0004-3702(01)00110-2` (cit. on p. 9).

[31] David Silver, Aja Huang, Chris J Maddison, et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489 (cit. on pp. 9, 29, 145).

[32] David Silver, Julian Schrittwieser, Karen Simonyan, et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359. DOI: `10.1038/nature24270` (cit. on pp. 9, 145).

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: `1312.5602` (cit. on pp. 9, 25).

[34] Vlad Firoiu, William F. Whitney, and Joshua B. Tenenbaum. “Beating the World’s Best at Super Smash Bros. with Deep Reinforcement Learning”. In: *CoRR* abs/1702.06230 (2017). arXiv: `1702.06230` (cit. on p. 9).

[35] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. “ViZDoom: A Doom-based AI research platform for visual reinforcement learning”. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. Sept. 2016, pp. 1–8. DOI: `10.1109/CIG.2016.7860433` (cit. on p. 9).

[36] Shehroze Bhatti, Alban Desmaison, Ondrej Miksik, et al. “Playing Doom with SLAM-Augmented Deep Reinforcement Learning”. In: *CoRR* abs/1612.00380 (2016). arXiv: `1612.00380` (cit. on p. 9).

[37] Guillaume Lample and Devendra Singh Chaplot. "Playing FPS games with deep reinforcement learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (cit. on p. 9).

[38] Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. "Control of Memory, Active Perception, and Action in Minecraft". In: *CoRR* abs/1605.09128 (2016). arXiv: 1605.09128 (cit. on pp. 9, 145).

[39] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. "A deep hierarchical approach to lifelong learning in Minecraft". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (cit. on pp. 9, 145).

[40] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, et al. "StarCraft II: A New Challenge for Reinforcement Learning". In: *CoRR* abs/1708.04782 (2017). arXiv: 1708.04782 (cit. on p. 9).

[41] P. Kormushev, S. Calinon, and D. G. Caldwell. "Robot motor skill coordination with EM-based Reinforcement Learning". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2010, pp. 3232–3237. DOI: 10.1109/IROS.2010.5649089 (cit. on p. 9).

[42] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373 (cit. on pp. 10, 69, 145).

[43] Sergey Levine, Peter Pastor, Peter Pastor, et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection". In: *The International Journal of Robotics Research* 37.4-5 (June 2017), pp. 421–436 (cit. on pp. 10, 79, 145).

[44] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. "Uncertainty-Aware Reinforcement Learning for Collision Avoidance". In: *CoRR* abs/1702.01182 (2017). arXiv: 1702.01182 (cit. on p. 10).

[45] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 6292–6299. DOI: 10.1109/ICRA.2018.8463162 (cit. on pp. 10, 46).

[46] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, et al. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In: *CoRR* abs/1709.10087 (2017). arXiv: 1709.10087 (cit. on pp. 10, 46).

[47] S. Gu, E. Holly, T. Lillicrap, and S. Levine. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 3389–3396. DOI: 10.1109/ICRA.2017.7989385 (cit. on p. 10).

[48] Richard Bellman. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684 (cit. on p. 10).

[49] Richard Bellman. *Dynamic programming*. Princeton, N. J: Princeton University Press, 1957 (cit. on pp. 12, 14).

[50] Christopher John Cornish Hellaby Watkins and Peter Dayan. “Q-learning”. In: *Machine Learning* 8.3-4 (May 1992), pp. 279–292 (cit. on pp. 14, 17).

[51] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. PhD thesis. King’s College, Cambridge, 1989 (cit. on p. 15).

[52] Michail G. Lagoudakis and Ronald Parr. “Least-squares policy iteration”. In: *Journal of Machine Learning Research* 4.Dec (2003), pp. 1107–1149 (cit. on p. 17).

[53] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. “Policy Gradient Methods for Reinforcement Learning with Function Approximation”. In: *Advances in Neural Information Processing Systems 12, {[NIPS} Conference, Denver, Colorado, USA, November 29 - December 4, 1999]* 07932 (1999), pp. 1057–1063. DOI: 10.1.1.37.9714 (cit. on p. 18).

[54] A. G. Barto, R. S. Sutton, and C. W. Anderson. “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (Sept. 1983), pp. 834–846. DOI: 10.1109/TSMC.1983.6313077 (cit. on p. 19).

[55] Richard S. Sutton. “Temporal credit assignment in reinforcement learning”. English. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2016-05-11. PhD thesis. University of Massachusetts, 1984, p. 223 (cit. on p. 19).

[56] Ronald J Williams. “Toward a theory of reinforcement-learning connectionist systems”. In: *Technical Report NU-CCS-88-3, Northeastern University* (1988) (cit. on p. 20).

[57] Ronald J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine Learning* 8.3-4 (May 1992), pp. 229–256 (cit. on p. 20).

[58] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on p. 22).

[59] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. “What is the best multi-stage architecture for object recognition?” In: *2009 IEEE 12th International Conference on Computer Vision*. Sept. 2009, pp. 2146–2153. DOI: 10.1109/ICCV.2009.5459469 (cit. on p. 23).

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on pp. 23, 71).

[61] Yurii Nesterov. “A method for solving the convex programming problem with convergence rate $O(1/k^2)$”. In: *Dokl. Akad. Nauk SSSR*. 1983, pp. 543–547 (cit. on pp. 23, 163).

[62] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Applied Optimization. Springer US, 2013 (cit. on pp. 23, 163).

[63] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. “On the importance of initialization and momentum in deep learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1139–1147 (cit. on pp. 23, 163).

[64] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159 (cit. on p. 23).

[65] Matthew D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *CoRR* abs/1212.5701 (2012). arXiv: `1212.5701` (cit. on p. 23).

[66] Tijmen Tieleman and Geoffrey Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31 (cit. on p. 23).

[67] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv.org* (Dec. 2014). arXiv: `1412.6980` (cit. on pp. 23, 113, 148, 207, 208).

[68] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536. DOI: `10.1038/323533a0` (cit. on pp. 23, 24).

[69] F Rosenblatt. *The perceptron: A probabilistic model for information storage and organization in the brain.* US, 1958. DOI: `10.1037/h0042519` (cit. on p. 24).

[70] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. DOI: `10.1007/BF02551274` (cit. on p. 24).

[71] Jürgen Schmidhuber. “Deep Learning in Neural Networks: An Overview”. In: *CoRR* abs/1404.7828 (2014). arXiv: `1404.7828` (cit. on p. 24).

[72] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: `10.1038/nature14539` (cit. on p. 24).

[73] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on p. 24).

[74] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). arXiv: `1502.03167` (cit. on p. 24).

[75] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 24).

[76] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *MIT Press* 9.8 (Nov. 1997), pp. 1735–1780 (cit. on pp. 24, 70).

[77] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078 (cit. on pp. 24, 154).

[78] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555 (cit. on pp. 24, 154).

[79] Herbert Jaeger and Harald Haas. "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". In: *Science* 304.5667 (2004), pp. 78–80. DOI: 10.1126/science.1091277. eprint: https://science.sciencemag.org/content/304/5667/78.full.pdf (cit. on p. 24).

[80] Herbert Jaeger. "Echo state network". In: *Scholarpedia* 2.9 (2007), p. 2330. DOI: 10.4249/scholarpedia.2330 (cit. on p. 24).

[81] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations". In: *Neural Computation* 14.11 (2002), pp. 2531–2560. DOI: 10.1162/089976602760407955. eprint: https://doi.org/10.1162/089976602760407955 (cit. on p. 24).

[82] Long-Ji Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine Learning* 8.3 (May 1992), pp. 293–321. DOI: 10.1007/BF00992699 (cit. on p. 25).

[83] Andrew W. Moore and Christopher G. Atkeson. "Prioritized sweeping: Reinforcement learning with less data and less time". In: *Machine Learning* 13.1 (Oct. 1993), pp. 103–130. DOI: 10.1007/BF00993104 (cit. on p. 25).

[84] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015) (cit. on p. 25).

[85] R. Liu and J. Zou. "The Effects of Memory Replay in Reinforcement Learning". In: *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Oct. 2018, pp. 478–485. DOI: 10.1109/ALLERTON.2018.8636075 (cit. on p. 25).

[86] Hado Van Hasselt, Adaptive Computation Group, and Centrum Wiskunde. "Double Q-learning". In: *Nips* (2010), pp. 1–9 (cit. on p. 26).

[87] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-learning". In: *arXiv:1509.06461 [cs]* 2 (2015), pp. 1–5. DOI: 10.1016/j.artint.2015.09.002. arXiv: 1509.06461 (cit. on p. 26).

[88] Ziyu Wang, Nando de Freitas, and Marc Lanctot. "Dueling Network Architectures for Deep Reinforcement Learning". In: *arXiv* 9 (2016), pp. 1–16. DOI: 10.1109/MCOM.2016.7378425. arXiv: 1511.06581 (cit. on p. 26).

[89] Matteo Hessel, Joseph Modayil, Hado van Hasselt, et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence* (Apr. 2018) (cit. on p. 26).

[90]Hao Yi Ong, Kevin Chavez, and Augustus Hong. “Distributed Deep Q-Learning”. In: *CoRR* abs/1508.04186 (2015). arXiv: 1508.04186 (cit. on p. 26).

[91]Arun Nair, Praveen Srinivasan, Sam Blackwell, et al. “Massively Parallel Methods for Deep Reinforcement Learning”. In: *arXiv* (2015), p. 14. DOI: 10.1109/IJCNN.2010.5596468. arXiv: 1507.04296 (cit. on p. 26).

[92]Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *arXiv.org* (Feb. 2016). arXiv: 1602.01783v2 [cs.LG] (cit. on pp. 26, 71).

[93]N. Munawaroh. “The influence of teaching methods and learning environment to the student’s learning achievement of craft and entrepreneurship subjects at vocational high school”. In: *International Journal of Environmental & Science Education* 12.4 (2017), pp. 665–678 (cit. on p. 29).

[94]Max Wertheimer. “Untersuchungen zur Lehre von der Gestalt. II”. In: *Psychological Research* 4.1 (1923), pp. 301–350 (cit. on p. 30).

[95]W. Metzger. “Gesetze des Sehens, Frankfurt am Main In: Waldemar Kramer. W. Metzger (Eds).(2006)”. In: *Laws of Seeing* (1975) (cit. on p. 30).

[96]Been Kim, Emily Reif, Martin Wattenberg, and Samy Bengio. “Do Neural Networks Show Gestalt Phenomena? An Exploration of the Law of Closure”. In: *CoRR* abs/1903.01069 (2019). arXiv: 1903.01069 (cit. on p. 30).

[97]Samuel Ritter, David G. T. Barrett, Adam Santoro, and Matt M. Botvinick. “Cognitive Psychology for Deep Neural Networks: A Shape Bias Case Study”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, June 2017, pp. 2940–2949 (cit. on p. 30).

[98]Jürgen Schmidhuber. “Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook”. Diplomarbeit. München: Technische Universität München, 1987 (cit. on p. 30).

[99]Christophe Giraud-Carrier, Ricardo Vilalta, and Pavel Brazdil. “Introduction to the Special Issue on Meta-Learning”. In: *Machine Learning* 54.3 (Mar. 2004), pp. 187–193. DOI: 10.1023/B:MACH.0000015878.60765.42 (cit. on p. 30).

[100]Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012 (cit. on p. 30).

[101]Ricardo Vilalta and Youssef Drissi. “A Perspective View and Survey of Meta-Learning”. In: *Artificial Intelligence Review* 18.2 (June 2002), pp. 77–95. DOI: 10.1023/A:1019956318069 (cit. on p. 30).

[102]Lev S. Vygotsky, Abdul Rahman Embong, and Nazri Muslim. *Mind in society: The development of higher psychological*. Cambridge, MA: info: Cambridge: Harvard University Press, 1978, 1978 (cit. on pp. 32, 88).

[103]Andrey I Podolskiy. “Zone of Proximal Development”. In: *Encyclopedia of the Sciences of Learning*. Ed. by Norbert M Seel. Boston, MA: Springer US, 2012, pp. 3485–3487 (cit. on pp. 32, 88).

[104] David Wood, Jerome S. Bruner, and Gail Ross. “The role of tutoring in problem solving”. In: *Journal of Child Psychology and Psychiatry* 17.2 (Apr. 1976), pp. 89–100 (cit. on p. 32).

[105] Jerome Bruner. “Vygotsky: A Historical and Conceptual Perspective”. In: *Culture communication, and cognition: Vygotskian perspectives*. Ed. by James V Wertsch and Center for Psychosocial Studies. New York: Cambridge University Press, 1985, pp. 21–34 (cit. on p. 32).

[106] Debra Myhill and Pauline Warren. “Scaffolds or straitjackets? Critical moments in classroom discourse”. In: *Educational Review* 57.1 (Feb. 2005), pp. 55–69 (cit. on p. 32).

[107] Roy D. Pea. “The Social and Technological Dimensions of Scaffolding and Related Theoretical Concepts for Learning, Education, and Human Activity”. In: *Journal of the Learning Sciences* 13.3 (July 2004), pp. 423–451 (cit. on pp. 33, 34, 38, 47).

[108] Sadhana Puntambekar and Roland Hubscher. “Tools for Scaffolding Students in a Complex Learning Environment: What Have We Gained and What Have We Missed?” In: *Educational Psychologist* 40.1 (Mar. 2005), pp. 1–12 (cit. on pp. 33, 37).

[109] C. Addison Stone. “Should We Salvage the Scaffolding Metaphor?” In: *Journal of Learning Disabilities* 31.4 (1998), pp. 409–413 (cit. on p. 33).

[110] C. Addison Stone. “The Metaphor of Scaffolding: Its Utility for the Field of Learning Disabilities”. In: *Journal of Learning Disabilities* 31.4 (1998), pp. 344–364 (cit. on p. 33).

[111] G. Wells. *The Meaning Makers: Children Learning Language and Using Language to Learn*. Portsmouth, NH: Heinemann, 1986 (cit. on p. 33).

[112] Leo Van Lier. “Interaction in the language classroom: Awareness, autonomy and authenticity”. In: *London: Longrnan* (1996) (cit. on p. 33).

[113] Elliot Soloway, Mark Guzdial, and Kenneth E. Hay. “Learner-centered Design: The Challenge for HCI in the 21st Century”. In: *interactions* 1.2 (Apr. 1994), pp. 36–48. DOI: 10.1145/174809.174813 (cit. on p. 36).

[114] Lindy Crawford, Kristina N. Higgins, and Barbara Freeman. “Exploring the Use of Active Electronic Support Tools by Students with Learning Disabilities”. In: *Learning Disabilities: A Multidisciplinary Journal* 18.3 (2012), pp. 135–144 (cit. on p. 36).

[115] Lindy Crawford, Kristina N. Higgins, Jacqueline N. Huscroft-D’Angelo, and Lindsay Hall. “Students’ use of electronic support tools in mathematics”. In: *Educational Technology Research and Development* 64.6 (Dec. 2016), pp. 1163–1182. DOI: 10.1007/s11423-016-9452-7 (cit. on p. 36).

[116] Ricardo S. Alonso, Javier Prieto, Óscar García, and Juan M. Corchado. “Collaborative learning via social computing”. In: *Frontiers of Information Technology & Electronic Engineering* 20.2 (Feb. 2019), pp. 265–282. DOI: 10.1631/FITEE.1700840 (cit. on p. 36).

[117] Shari L. Jackson. “The Design of Guided Learner-adaptable Scaffolding in Interactive Learning Environments”. In: *Proceedings of the 1996 International Conference on Learning Sciences*. ICLS ’96. Evanston, Illinois: International Society of the Learning Sciences, 1996, pp. 575–576 (cit. on p. 36).

[118] C. Addison Stone. “The Metaphor of Scaffolding: Its Utility for the Field of Learning Disabilities”. In: *Journal of Learning Disabilities* 31.4 (1998). PMID: 9666611, pp. 344–364. DOI: 10.1177/002221949803100404. eprint: https://doi.org/10.1177/002221949803100404 (cit. on p. 36).

[119] Roger Azevedo and Allyson F. Hadwin. “Scaffolding Self-regulated Learning and Metacognition – Implications for the Design of Computer-based Scaffolds”. In: *Instructional Science* 33.5 (Nov. 2005), pp. 367–379. DOI: 10.1007/s11251-005-1272-9 (cit. on p. 37).

[120] Roger Azevedo, Jennifer G. Cromley, Fielding I. Winters, Daniel C. Moos, and Jeffrey A. Greene. “Adaptive Human Scaffolding Facilitates Adolescents’ Self-regulated Learning with Hypermedia”. In: *Instructional Science* 33.5 (Nov. 2005), pp. 381–412. DOI: 10.1007/s11251-005-1273-8 (cit. on p. 37).

[121] Allyson Fiona Hadwin, Lori Wozney, and Oonagh Pontin. “Scaffolding the Appropriation of Self-regulatory Activity: A Socio-cultural Analysis of Changes in Teacher–student Discourse about a Graduate Research Portfolio”. In: *Instructional Science* 33.5 (Nov. 2005), pp. 413–450. DOI: 10.1007/s11251-005-1274-7 (cit. on p. 37).

[122] Ikseon Choi, Susan M. Land, and Alfred J. Turgeon. “Scaffolding Peer-questioning Strategies to Facilitate Metacognition During Online Small Group Discussion”. In: *Instructional Science* 33.5 (Nov. 2005), pp. 483–511. DOI: 10.1007/s11251-005-1277-4 (cit. on p. 37).

[123] Sadhana Puntambekar and Agni Stylianou. “Designing Navigation Support in Hypertext Systems Based on Navigation Patterns”. In: *Instructional Science* 33.5 (Nov. 2005), pp. 451–481. DOI: 10.1007/s11251-005-1276-5 (cit. on p. 37).

[124] Janette R. Hill and Michael J. Hannafin. “Cognitive strategies and learning from the world wide web”. In: *Educational Technology Research and Development* 45.4 (Dec. 1997), pp. 37–64. DOI: 10.1007/BF02299682 (cit. on p. 37).

[125] Barbara A. Greene and Susan M. Land. “A Qualitative Analysis of Scaffolding Use in a Resource-Based Learning Environment Involving the World Wide Web”. In: *Journal of Educational Computing Research* 23.2 (2000), pp. 151–179. DOI: 10.2190/1GUB-8UE9-NW80-CQAD. eprint: https://doi.org/10.2190/1GUB-8UE9-NW80-CQAD (cit. on p. 37).

[126] Roger Azevedo, Jennifer G. Cromley, and Diane Seibert. “Does adaptive scaffolding facilitate students’ ability to regulate their learning with hypermedia?” In: *Contemporary Educational Psychology* 29.3 (2004), pp. 344–370. DOI: https://doi.org/10.1016/j.cedpsych.2003.09.002 (cit. on p. 37).

[127] Sadhana Puntambekar and Janet L Kolodner. “Toward implementing distributed scaffolding: Helping students learn science from design”. In: *Journal of Research in Science Teaching* 42.2 (2005), pp. 185–217 (cit. on p. 38).

[128] Gavriel Salomon. "No distribution without individuals' cognition: a dynamic interactional view". In: *Distributed Cognitions: Psychological and Educational Considerations*. Ed. by Gavriel Salomon. Cambridge University Press, 1996, pp. 110–138 (cit. on p. 38).

[129] Andrea Lockerd Thomaz and Cynthia Breazeal. "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance". In: *Association for the Advancement of Artificial Intelligence - 2006*. Boston, MA, 2006, pp. 1000–1005 (cit. on p. 40).

[130] Andrea Lockerd Thomaz. "Socially guided machine learning". PhD thesis. Computer Science Department Faculty Publication Series, 2006 (cit. on p. 40).

[131] W. Bradley Knox and Peter Stone. "Interactively Shaping Agents via Human Reinforcement: The TAMER Framework". In: *Proceedings of the Fifth International Conference on Knowledge Capture*. K-CAP '09. Redondo Beach, California, USA: ACM, 2009, pp. 9–16. DOI: 10.1145/1597735.1597738 (cit. on p. 40).

[132] Ana C. Tenorio-Gonzalez, Eduardo F. Morales, and Luis Villaseñor-Pineda. "Dynamic Reward Shaping: Training a Robot by Voice". In: *Advances in Artificial Intelligence – IBERAMIA 2010*. Ed. by Angel Kuri-Morales and Guillermo R. Simari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 483–492 (cit. on p. 40).

[133] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. "Deep tamer: Interactive agent shaping in high-dimensional state spaces". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018 (cit. on p. 40).

[134] James J. Gibson. *The Ecological Approach To Visual Perception*. Psychology Press, Sept. 1986 (cit. on pp. 41, 99).

[135] J. Kevin O'Regan and Alva Noë. "A sensorimotor account of vision and visual consciousness". In: *Behavioral and Brain Sciences* 24.5 (2001), pp. 939–973. DOI: 10.1017/S0140525X01000115 (cit. on p. 41).

[136] Alva Noë. *Action in perception.* Representation and mind. Cambridge, MA, US: MIT Press, 2004, pp. viii, 277–viii, 277 (cit. on pp. 41, 80, 155).

[137] Joaquín M. Fuster. "The prefrontal cortex—an update: time is of the essence". In: *Neuron* 30.2 (2001), pp. 319–333 (cit. on p. 41).

[138] Joaquín M. Fuster. "The cognit: A network model of cortical representation". In: *International Journal of Psychophysiology* 60.2 (2006). Models and Theories of Brain Function with Special Emphasis on Cognitive Processing, pp. 125–132. DOI: https://doi.org/10.1016/j.ijpsycho.2005.12.015 (cit. on p. 41).

[139] Jeannette Bohg, Karol Hausman, Bharath Sankaran, et al. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291 (cit. on pp. 41, 43, 53).

[140] Karl Friston, James Kilner, and Lee Harrison. "A free energy principle for the brain". In: *Journal of Physiology-Paris* 100.1-3 (July 2006), pp. 70–87 (cit. on p. 42).

[141] Karl Friston. "The free-energy principle: a rough guide to the brain?" In: *Trends in Cognitive Sciences* 13.7 (July 2009), pp. 293–301 (cit. on p. 42).

[142]Karl Friston. “The free-energy principle: a unified brain theory?” In: *Nature Reviews Neuroscience* 11.2 (Jan. 2010), pp. 1–12 (cit. on pp. 42, 61).

[143]A. S. Klyubin, D. Polani, and C. L. Nehaniv. “Empowerment: a universal agent-centric measure of control”. In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 1. Sept. 2005, 128–135 Vol.1. DOI: 10.1109/CEC.2005.1554676 (cit. on pp. 42, 53).

[144]Christoph Salge, Cornelius Glackin, and Daniel Polani. “Empowerment – an Introduction”. In: *arXiv.org* (Oct. 2013). arXiv: 1310.1863v2 [cs.AI] (cit. on pp. 42, 53).

[145]Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. “Revisiting active perception”. In: *Autonomous Robots* 42.2 (Feb. 2018), pp. 177–196. DOI: 10.1007/s10514-017-9615-3 (cit. on p. 43).

[146]Laurent Itti and Christof Koch. “Computational modelling of visual attention”. In: *Nature Reviews Neuroscience* 2.3 (2001), pp. 194–203 (cit. on pp. 43, 69).

[147]John K. Tsotsos, Scan M. Culhane, Winky Yan Kei Wai, et al. “Modeling visual attention via selective tuning”. In: *Artificial Intelligence* 78.1-2 (Oct. 1995), pp. 507–545 (cit. on p. 43).

[148]Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. “Recurrent Models of Visual Attention”. In: *arXiv.org* (June 2014). arXiv: 1406.6247v1 [cs.LG] (cit. on pp. 43, 69, 73, 74, 81).

[149]C. J. Tsikos and R. K. Bajcsy. “Segmentation via manipulation”. In: *IEEE Transactions on Robotics and Automation* 7.3 (June 1991), pp. 306–319. DOI: 10.1109/70.88140 (cit. on p. 43).

[150]Howard R. Nicholls and Mark H. Lee. “A Survey of Robot Tactile Sensing Technology”. In: *The International Journal of Robotics Research* 8.3 (1989), pp. 3–30. DOI: 10.1177/027836498900800301. eprint: https://doi.org/10.1177/027836498900800301 (cit. on p. 43).

[151]Lorenzo Natale, Giorgio Metta, and Giulio Sandini. “Learning haptic representation of objects”. In: *International Conference on Intelligent Manipulation and Grasping*. 2004 (cit. on p. 43).

[152]A. Petrovskaya and O. Khatib. “Global Localization of Objects via Touch”. In: *IEEE Transactions on Robotics* 27.3 (June 2011), pp. 569–585. DOI: 10.1109/TRO.2011.2138450 (cit. on p. 43).

[153]P. Hebert, T. Howard, N. Hudson, J. Ma, and J. W. Burdick. “The next best touch for model-based localization”. In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 99–106. DOI: 10.1109/ICRA.2013.6630562 (cit. on p. 43).

[154]S. Javdani, M. Klingensmith, J. A. Bagnell, N. S. Pollard, and S. S. Srinivasa. “Efficient touch based localization through submodularity”. In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 1828–1835. DOI: 10.1109/ICRA.2013.6630818 (cit. on p. 43).

[155] Jeremy A Fishel and Gerald E Loeb. “Bayesian Exploration for Intelligent Identification of Textures”. In: *Frontiers in Neurorobotics* 6 (June 2012) (cit. on p. 43).

[156] Gerald E. Loeb and Jeremy A. Fishel. “Bayesian Action & Perception: Representing the World in the Brain”. In: *Frontiers in Neuroscience* 8 (2014), p. 341. DOI: `10.3389/fnins.2014.00341` (cit. on p. 43).

[157] Vivian Chu, Ian McMahon, Lorenzo Riano, et al. “Robotic learning of haptic adjectives through physical interaction”. In: *Robotics and Autonomous Systems* 63 (2015). Advances in Tactile Sensing and Touch-based Human Robot Interaction, pp. 279–292. DOI: `https://doi.org/10.1016/j.robot.2014.09.021` (cit. on p. 43).

[158] S. Dragiev, M. Toussaint, and M. Gienger. “Gaussian process implicit surfaces for shape estimation and grasping”. In: *2011 IEEE International Conference on Robotics and Automation*. May 2011, pp. 2845–2850. DOI: `10.1109/ICRA.2011.5980395` (cit. on p. 43).

[159] Min Jeong Kim, Mina Choi, Yong Bum Kim, et al. “Exploration of unknown object by active touch of robot hand”. In: *International Journal of Control, Automation and Systems* 12.2 (Apr. 2014), pp. 406–414. DOI: `10.1007/s12555-013-0328-x` (cit. on p. 43).

[160] J. Bohg, M. Johnson-Roberson, M. Björkman, and D. Kragic. “Strategies for multi-modal scene exploration”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2010, pp. 4509–4515. DOI: `10.1109/IROS.2010.5652967` (cit. on p. 43).

[161] Matthew E. Taylor and Peter Stone. “Transfer Learning for Reinforcement Learning Domains : A Survey”. In: *Journal of Machine Learning Research* 10 (2009), pp. 1633–1685. DOI: `10.1007/978-3-642-27645-3`. arXiv: `0803.0476` (cit. on pp. 44, 87, 130).

[162] Matthew E. Taylor and Peter Stone. “An Introduction to Intertask Transfer for Reinforcement Learning”. In: *AI Magazine* 32.1 (2011), p. 15. DOI: `10.1609/aimag.v32i1.2329` (cit. on pp. 44, 48, 87, 88, 167).

[163] R. S. Woodworth and E. L. Thorndike. “The influence of improvement in one mental function upon the efficiency of other functions. (I).” In: *Psychological Review* 8.3 (1901), pp. 247–261. DOI: `10.1037/h0074898` (cit. on p. 44).

[164] Burrhus Frederic Skinner. *Science and human behavior*. 92904. Simon and Schuster, 1953 (cit. on pp. 44, 48).

[165] Lisa Torrey, Trevor Walker, Jude Shavlik, and Richard Maclin. “Using Advice to Transfer Knowledge Acquired in One Reinforcement Learning Task to Another”. In: *Machine Learning: ECML 2005*. Ed. by João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 412–424 (cit. on p. 44).

[166] George Konidaris and Andrew G. Barto. “Autonomous shaping: knowledge transfer in reinforcement learning”. In: *the 23rd international conference* (2006), pp. 489–496 (cit. on p. 44).

[167] Matthew E. Taylor, Peter Stone, and Yaxin Liu. “Transfer learning via inter-task mappings for temporal difference learning”. In: *Journal of Machine Learning Research* 8.Sep (2007), pp. 2125–2167 (cit. on p. 44).

[168] Irina Higgins, Arka Pal, Andrei Rusu, et al. “DARLA: Improving Zero-Shot Transfer in Reinforcement Learning”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, June 2017, pp. 1480–1490 (cit. on pp. 44, 87, 145).

[169] Matthew E. Taylor and Peter Stone. “Cross-domain transfer for reinforcement learning”. In: *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 879–886 (cit. on p. 44).

[170] D. Marmanis, M. Datcu, T. Esch, and U. Stilla. “Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks”. In: *IEEE Geoscience and Remote Sensing Letters* 13.1 (Jan. 2016), pp. 105–109. DOI: 10.1109/LGRS.2015.2499239 (cit. on p. 44).

[171] H. Shin, H. R. Roth, M. Gao, et al. “Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning”. In: *IEEE Transactions on Medical Imaging* 35.5 (May 2016), pp. 1285–1298. DOI: 10.1109/TMI.2016.2528162 (cit. on p. 44).

[172] Samiksha Mahajan. “Hierarchical reinforcement learning in complex learning problems: a survey”. In: *International Journal of Computer Science and Engine Science and Engineering* 2.5 (2014), pp. 72–78 (cit. on p. 44).

[173] Peter Dayan and Geoffrey E Hinton. “Feudal Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 5*. Ed. by S. J. Hanson, J. D. Cowan, and C. L. Giles. Morgan-Kaufmann, 1993, pp. 271–278 (cit. on p. 45).

[174] Richard S. Sutton, Doina Precup, and Satinder Singh. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial Intelligence* 112.1 (1999), pp. 181–211. DOI: 10.1016/S0004-3702(99)00052-1. arXiv: arXiv:1011.1669v3 (cit. on p. 45).

[175] Jonathan Sorg and Satinder Singh. “Linear Options”. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*. AAMAS ’10. Toronto, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 31–38 (cit. on p. 45).

[176] Hengshuai Yao, Csaba Szepesvari, Richard S. Sutton, Joseph Modayil, and Shalabh Bhatnagar. “Universal Option Models”. In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 990–998 (cit. on p. 45).

[177] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. “Universal Value Function Approximators”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1312–1320 (cit. on p. 45).

[178] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation". In: *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. 2016, pp. 3675–3683. DOI: 10.1023/A:1025696116075. arXiv: 1604.06057 (cit. on p. 45).

[179] Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The Option-Critic Architecture". In: *CoRR* abs/1609.05140 (2016). arXiv: 1609.05140 (cit. on p. 45).

[180] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. "Meta Learning Shared Hierarchies". In: *CoRR* abs/1710.09767 (2017). arXiv: 1710.09767 (cit. on p. 45).

[181] D. Meger, J. C. G. Higuera, A. Xu, P. Giguère, and G. Dudek. "Learning legged swimming gaits from experience". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. May 2015, pp. 2332–2338. DOI: 10.1109/ICRA.2015.7139509 (cit. on p. 46).

[182] Jun Morimoto and Christopher G Atkeson. "Minimax differential dynamic programming: An application to robust biped walking". In: *Advances in neural information processing systems*. 2003, pp. 1563–1570 (cit. on p. 46).

[183] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 7559–7566. DOI: 10.1109/ICRA.2018.8463189 (cit. on p. 46).

[184] A. Giusti, J. Guzzi, D. C. Cireşan, et al. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots". In: *IEEE Robotics and Automation Letters* 1.2 (July 2016), pp. 661–667. DOI: 10.1109/LRA.2015.2509024 (cit. on p. 46).

[185] Andrew Y. Ng and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670 (cit. on p. 46).

[186] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: ACM, 2004, pp. 1–. DOI: 10.1145/1015330.1015430 (cit. on p. 46).

[187] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. "Autonomous Helicopter Aerobatics through Apprenticeship Learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639. DOI: 10.1177/0278364910371999. eprint: https://doi.org/10.1177/0278364910371999 (cit. on p. 46).

[188] Todd Hester, Matej Vecerik, Olivier Pietquin, et al. "Deep Q-learning from demonstrations". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018 (cit. on p. 46).

[189] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48 (cit. on pp. 47, 88).

[190] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. "Source task creation for curriculum learning". In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 566–574 (cit. on pp. 47, 88).

[191] Yuxin Wu and Yuandong Tian. "Training agent for first-person shooter game with actor-critic curriculum learning". In: *Proceedings of the 5th International Conference on Learning Representations*. 2016 (cit. on pp. 47, 88).

[192] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. "Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. Melbourne, Australia: AAAI Press, 2017, pp. 2536–2542 (cit. on pp. 47, 48, 87, 88).

[193] Niels Justesen and Sebastian Risi. "Automated Curriculum Learning by Rewarding Temporally Rare Events". In: *CoRR* abs/1803.07131 (2018) (cit. on pp. 47, 88).

[194] Sanmit Narvekar and Peter Stone. "Learning Curriculum Policies for Reinforcement Learning". In: *CoRR* abs/1812.00285 (2018). arXiv: `1812.00285` (cit. on pp. 47, 48, 87, 88).

[195] Claude E. Shannon and Warren Weaver. "The mathematical theory of communication". In: *Urbana, IL: University of Illinois Press* (1963) (cit. on pp. 60–62, 118).

[196] Douglas Samuel Jones. *Elementary information theory*. Clarendon Press, 1979 (cit. on pp. 60, 62, 118).

[197] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012 (cit. on pp. 60, 62, 118).

[198] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006 (cit. on pp. 64, 93).

[199] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297 (cit. on pp. 64, 93).

[200] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. DOI: `10.1109/TIT.1982.1056489` (cit. on pp. 64, 93).

[201] Georges Voronoi. "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs." In: *Journal für die reine und angewandte Mathematik* 134 (1908), pp. 198–287 (cit. on p. 65).

[202] Franz Aurenhammer. "Voronoi Diagrams – Survey of a Fundamental Geometric Data Structure". In: *ACM Comput. Surv.* 23.3 (Sept. 1991), pp. 345–405. DOI: `10.1145/116873.116880` (cit. on p. 65).

[203] Barry N. Boots, Sung Nok Chiu, Atsuyuki Okabe, and Kokichi Sugihara. *Spatial tessellations: concepts and applications of Voronoi diagrams; 2nd ed.* Wiley Series in Probability and Statistics. Chichester: Wiley, 2000 (cit. on p. 65).

[204] George Miller. “Note on the bias of information estimates”. In: *Information theory in psychology: Problems and methods* (1955) (cit. on p. 66).

[205] B Efron and C Stein. “The Jackknife Estimate of Variance”. In: *The Annals of Statistics* 9.3 (1981), pp. 586–596 (cit. on p. 66).

[206] S. P. Strong, Roland Koberle, Rob R. de Ruyter van Steveninck, and William Bialek. “Entropy and Information in Neural Spike Trains”. In: *Physical Review Letters* 80.1 (Jan. 1998), pp. 197–200 (cit. on p. 66).

[207] András Antos and Ioannis Kontoyiannis. “Convergence properties of functional estimates for discrete distributions”. In: *Random Structures and Algorithms* 19.3-4 (2001), pp. 163–193 (cit. on p. 66).

[208] Liam Paninski. “Estimation of Entropy and Mutual Information”. In: *Neural Computation* 15.6 (June 2003), pp. 1191–1253 (cit. on p. 66).

[209] G. Basharin. “On a Statistical Estimate for the Entropy of a Sequence of Independent Random Variables”. In: *Theory of Probability & Its Applications* 4.3 (1959), pp. 333–336. DOI: `10.1137/1104033`. eprint: `https://doi.org/10.1137/1104033` (cit. on p. 66).

[210] Bernard Harris. *The statistical estimation of entropy in the non-parametric case*. Tech. rep. WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER, 1975 (cit. on p. 66).

[211] Samuel Zahl. “Jackknifing An Index of Diversity”. In: *Ecology* 58.4 (1977), pp. 907–913 (cit. on p. 66).

[212] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. “A Brief Survey of Deep Reinforcement Learning”. In: *arXiv.org* (Aug. 2017). arXiv: `1708.05866v2 [cs.LG]` (cit. on p. 69).

[213] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, et al. “Reinforcement Learning with Unsupervised Auxiliary Tasks”. In: *arXiv.org* (Nov. 2016). arXiv: `1611.05397 [cs.LG]` (cit. on p. 69).

[214] Mary Hayhoe and Dana Ballard. “Eye movements in natural behavior”. In: *Trends in Cognitive Sciences* 9.4 (Apr. 2005), pp. 188–194 (cit. on p. 69).

[215] Stefan Mathe and Cristian Sminchisescu. “Action from still image dataset and inverse optimal control to learn task specific visual scanpaths”. In: *Advances in neural information processing systems*. 2013, pp. 1923–1931 (cit. on p. 69).

[216] Laurent Itti, Christof Koch, and Ernst Niebur. “A model of saliency-based visual attention for rapid scene analysis”. In: *IEEE Transactions on pattern analysis and machine intelligence* 20.11 (1998), pp. 1254–1259 (cit. on p. 69).

[217] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 69).

[218] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. "Multiple Object Recognition with Visual Attention". In: *CoRR* abs/1412.7755 (Dec. 2014). arXiv: `1412.7755` (cit. on pp. 70, 71, 73, 74, 77, 81, 154).

[219] K. Fukushima. "Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position-Neocognitron-". In: *IEICE Technical Report, A* 62.10 (1979), pp. 658–665 (cit. on p. 70).

[220] Yann LeCun, B. Boser, J. S. Denker, et al. "Handwritten digit recognition with a back-propagation network". English (US). In: *Advances in Neural Information Processing Systems (NIPS 1989), Denver, CO*. Ed. by David Touretzky. Vol. 2. Morgan Kaufmann, 1990 (cit. on p. 70).

[221] Hugo Larochelle and Geoffrey E Hinton. "Learning to combine foveal glimpses with a third-order Boltzmann machine". In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 1243–1251 (cit. on pp. 73, 74, 77, 78, 85).

[222] Nicholas Leonard. *Recurrent Model of Visual Attention*. `http://torch.ch/blog/2015/09/21/rmva.html`. Accessed: 2018-03-28. 2015 (cit. on p. 73).

[223] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. "Incorporating second-order functional knowledge for better option pricing". In: *Advances in neural information processing systems*. 2001, pp. 472–478 (cit. on p. 74).

[224] John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *CoRR* abs/1506.02438 (2015) (cit. on p. 78).

[225] Christian Szegedy, Wei Liu, Yangqing Jia, et al. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). arXiv: `1409.4842` (cit. on p. 79).

[226] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.0 (2015). arXiv: `1512.03385` (cit. on p. 79).

[227] Allison M. Okamura and Mark R. Cutkosky. "Feature Detection for Haptic Exploration with Robotic Fingers". In: *The International Journal of Robotics Research* 20.12 (2001), pp. 925–938. DOI: `10.1177/02783640122068191`. eprint: `https://doi.org/10.1177/02783640122068191` (cit. on p. 79).

[228] Ricardo Martins, João Filipe Ferreira, and Jorge Dias. "Touch attention Bayesian models for robotic active haptic exploration of heterogeneous surfaces". In: *CoRR* abs/1409.6 (2014). arXiv: `1409.6226` (cit. on p. 79).

[229] Stephen Tian, Frederik Ebert, Dinesh Jayaraman, et al. "Manipulation by Feel: Touch-Based Control with Deep Predictive Models". In: *CoRR* abs/1903.04128 (2019). arXiv: `1903.04128` (cit. on pp. 79, 81, 156).

[230] Charles Spence and Alberto Gallace. "Recent developments in the study of tactile attention". In: *Canadian journal of experimental psychology = Revue canadienne de psychologie experimentale* 61.3 (Sept. 2007), pp. 196–207. DOI: `10.1037/cjep2007021` (cit. on p. 80).

[231] Hilary Kalagher and Susan S. Jones. “Young children’s haptic exploratory procedures”. In: *Journal of Experimental Child Psychology* 110.4 (2011), pp. 592–602. DOI: `https://doi.org/10.1016/j.jecp.2011.06.007` (cit. on p. 80).

[232] Roberta L. Klatzky, Susan J. Lederman, and Julie M. Mankinen. “Visual and haptic exploratory procedures in children’s judgments about tool function”. In: *The Development of Haptic Perception* 28.3 (2005), pp. 240–249 (cit. on p. 80).

[233] Roberta L. Klatzky, Susan J. Lederman, and Catherine L. Reed. “There’s more to touch than meets the eye: the salience of object attributes for haptics with and without vision”. In: *Journal of Experimental Psychology* (1987) (cit. on p. 80).

[234] Roberta L Klatzky and Susan J Lederman. “Identifying objects from a haptic glance”. In: *Perception & Psychophysics* 57.8 (Nov. 1995), pp. 1111–1123. DOI: `10.3758/BF03208368` (cit. on p. 80).

[235] G. Rouhafzay and A. Cretu. “Object Recognition From Haptic Glance at Visually Salient Locations”. In: *IEEE Transactions on Instrumentation and Measurement* (2019), pp. 1–11. DOI: `10.1109/TIM.2019.2905906` (cit. on p. 81).

[236] B J Frey and D Dueck. “Clustering by Passing Messages Between Data Points”. In: *Science* 315.5814 (Feb. 2007), pp. 972–976 (cit. on p. 93).

[237] J. Roger Bray and J. T. Curtis. “An Ordination of the Upland Forest Communities of Southern Wisconsin”. In: *Ecological Monographs* 27.4 (Oct. 1957), pp. 325–349 (cit. on p. 93).

[238] Karl Pearson. “Note on Regression and Inheritance in the Case of Two Parents”. In: *Proceedings of the Royal Society of London* 58 (1895), pp. 240–242 (cit. on p. 94).

[239] Robert W. Shumaker, Kristina R. Walkup, and Benjamin B. Beck. *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press, 2011 (cit. on p. 99).

[240] Jessica A. Sommerville, Elina A. Hildebrand, and Catharyn C. Crane. “Experience matters: The impact of doing versus watching on infants’ subsequent perception of tool-use events.” In: *Developmental Psychology* 44.5 (2008), pp. 1249–1256. DOI: `10.1037/a0012296` (cit. on p. 99).

[241] Alborz Geramifard, Thomas J. Walsh, Stefanie Tellex, et al. “A Tutorial on Linear Function Approximators for Dynamic Programming and Reinforcement Learning”. In: *Foundations and Trends® in Machine Learning* 6.4 (2013), pp. 375–451 (cit. on pp. 110, 111).

[242] John Moody and Christian J Darken. “Fast Learning in Networks of Locally-Tuned Processing Units”. In: *Neural Computation* 1.2 (June 1989), pp. 281–294 (cit. on p. 111).

[243] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010 (cit. on p. 111).

[244] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305 (cit. on pp. 112, 164, 209).

[245] Justin A. Boyan. “Technical Update: Least-Squares Temporal Difference Learning”. In: *Machine Learning* 49.2 (Nov. 2002), pp. 233–246. DOI: 10.1023/A:1017936530646 (cit. on p. 112).

[246] Deirdre Quillen, Eric Jang, Ofir Nachum, et al. “Deep Reinforcement Learning for Vision-Based Robotic Grasping: A Simulated Comparative Evaluation of Off-Policy Methods”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 6284–6291. DOI: 10.1109/ICRA.2018.8461039 (cit. on p. 145).

[247] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. “Generalization through Simulation: Integrating Simulated and Real Data into Deep Reinforcement Learning for Vision-Based Autonomous Flight”. In: *CoRR* abs/1902.03701 (2019). arXiv: 1902.03701 (cit. on p. 145).

[248] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, et al. “FeUdal Networks for Hierarchical Reinforcement Learning”. In: *CoRR* abs/1703.01161 (2017) (cit. on p. 145).

[249] Carsten Schurmann, Risto Koiva, Robert Haschke, and Helge Ritter. “A modular high-speed tactile sensor for human manipulation research”. In: *2011 IEEE World Haptics Conference (WHC 2011)*. IEEE, 2011, pp. 339–344 (cit. on p. 156).

[250] Alexandra Moringen, Witali Aswolinskij, Gereon Buescher, et al. “Modeling Target-Distractor Discrimination for Haptic Search in a 3D Environment”. In: BioRob. 2018 (cit. on p. 157).

[251] Alexandra Moringen, Robert Haschke, and Helge Ritter. “Search Procedures during Haptic Search in an Unstructured 3D Display”. In: *IEEE Haptics Symposium*. 2016 (cit. on p. 157).

[252] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034 (cit. on pp. 164, 208, 209).

[253] Tomaso A. Poggio, Kenji Kawaguchi, Qianli Liao, et al. “Theory of Deep Learning III: explaining the non-overfitting puzzle”. In: *CoRR* abs/1801.00173 (2018). arXiv: 1801.00173 (cit. on p. 167).

[254] Alexandra Moringen, Sascha Fleer, and Helge Ritter. “Scaffolding Haptic Attention with Controller Gating”. In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation*. Ed. by Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis. Cham: Springer International Publishing, 2019, pp. 669–684 (cit. on p. 167).

Appendices

A

Pseudocode

Algorithm 4: Linear greedy Q-learning with eligibility traces

```
Data: greediness $\epsilon$, Maximal number of Training steps $T$
$t = 0$
Set learning rate $\alpha_t$
Initialize weights $\boldsymbol{w}_t$
while $t < T$ do
    Initialize episode
    Initialize eligibility trace $\boldsymbol{e}_t = \boldsymbol{0}$
    $\mathrm{terminal} = \mathrm{False}$
    repeat
        Receive state $s_t$
        if $\mathrm{random\ uniform} \leq \epsilon$ then
            Choose and perform random action $a_t \in \mathcal{A}(s_t)$
            $\boldsymbol{e}_t = \boldsymbol{0}$
        else
            Choose and perform action $a_t = \mathrm{argmax}_{a' \in \mathcal{A}(\int_\sqcup)} Q(s_t, a'; \boldsymbol{w}_t)$
            $\boldsymbol{e}_t \leftarrow \gamma\lambda\boldsymbol{e}_t$
        end
        $\boldsymbol{e}_t \leftarrow \boldsymbol{e}_t + \nabla_{\boldsymbol{w}_t} Q(s_t, a_t; \boldsymbol{w}_t)$
        Receive state $s_{t+1}$ and reward $r_t$
        if *goal achieved or end of episode* then
            $\mathrm{terminal} = \mathrm{True}$
        end
        $\delta = \begin{cases} r_t & \text{if terminal} == \text{True} \\ r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \boldsymbol{w}_t) & \text{if terminal} == \text{False} \end{cases}$
        $\boldsymbol{w}_t \leftarrow \boldsymbol{w}_t + \alpha_t \left[\delta - Q(s_t, a_t; \boldsymbol{w}_t)\right] \cdot \boldsymbol{e}_t$
        $t \leftarrow t + 1$
        Adjust learning rate $\alpha_t$
    until $\mathrm{terminal} == \mathrm{True}$
end
```

Algorithm 5: Deep Q-learning

```
Data: Maximal number of training steps $T$, Capacity $N$ of replay buffer $B$, steps
      until updating target-weights $F$
$t = 0$
Exploration parameter $\epsilon = 1$
Initialize weights $\boldsymbol{w}_t$
Initialize target-weights $\boldsymbol{w}_t^- = \boldsymbol{w}_t$
Initialize replay Buffer $B$
while $t < T$ do
    Initialize episode
    $\mathrm{terminal} = \mathrm{False}$
    repeat
        Receive state $s_t$
        if $\mathrm{random\ uniform} \leq \epsilon$ then
            Choose and perform random action $a_t \in \mathcal{A}(s_t)$
            $\boldsymbol{e}_t = \mathbf{0}$
        else
            Choose and perform action $a_t = \mathrm{argmax}_{a' \in \mathcal{A}(\int_{\sqcup})} Q(s_t, a'; \boldsymbol{w}_t)$
        end
        Receive state $s_{t+1}$ and reward $r_t$
        if goal achieved or end of episode then
            $\mathrm{terminal} = \mathrm{True}$
        end
        Store tuple in replay buffer $(s_t, a_t, r_t, s_{t+1}, \mathrm{terminal}) \rightarrow B$
        Create mini-batch $b$ through sampling from replay buffer $B$
        Optimize network parameters $\boldsymbol{w}_t$ by performing a gradient descent step
          on $[\delta - Q(s_t, a_t; \boldsymbol{w}_t)]^2$ using learning rate $\alpha_t$ and all tuples
          $(s_t, a_t, r_t, s_{t+1}, \mathrm{terminal}) \in b$ with
          $\delta = \begin{cases} r_t & \text{if terminal == True} \\ r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \boldsymbol{w}_t^-) & \text{if terminal == False} \end{cases}$
        $t \leftarrow t + 1$
        Adjust learning rate $\alpha_t$
        Adjust exploration parameter $\epsilon_t$
        if $t \mod F$ then
            $\boldsymbol{w}_t^- = \boldsymbol{w}_t$
        end
    until $\mathrm{terminal} == \mathrm{True}$
end
```

Algorithm 6: Asynchronous advantage actor-critic algorithm

```
Data: Maximal number of episodes $E$, replay buffer $B$, update rate $u$, episodes
      until updating target-weights $\tau$
Episodes $e = 0$
Initialize global weights $\boldsymbol{w}_g$ and $\boldsymbol{w}_{g_v}$, worker weights $\boldsymbol{w}$ and $\boldsymbol{w}_v$ and replay
  Buffer $B$
while $e < E$ do
    Initialize episode, clear replay buffer $B$ and training steps $t = 0$
    $\text{terminal} = \text{False}$
    repeat
        Receive state $s_t$
        Sample and perform action $a_t$ from $\pi(a_t|s_t;\boldsymbol{w})$
        Receive state $s_{t+1}$ and reward $r_t$
        Store tuple in replay buffer $(s_t, a_t, r_t, s_{t+1}, \hat{V}(s_t)) \rightarrow B$
        if goal achieved or end of episode then
            $\text{terminal} = \text{True}$
        end
        if $t \mod u$ or $\text{terminal} == \text{True}$ then
            $R = \begin{cases} 0 & \text{if terminal} == \text{True} \\ \hat{V}(s_t;\boldsymbol{w}_v) & \text{if terminal} == \text{False} \rightarrow \text{bootstrap} \end{cases}$
            for $b \in \{t-1 \dots t - \text{length}(B)\}$ do
                $R_t = r_b + \gamma \cdot R$
                Accumulate gradients for $\boldsymbol{w}$ and $\boldsymbol{w}_v$:
                $\Delta\boldsymbol{w} \leftarrow \Delta\boldsymbol{w} + \left[R_t - \hat{V}(\boldsymbol{s}_t;\boldsymbol{w}_v)\right] \nabla_{\boldsymbol{w}} \log\left(\pi(a_t|s_t;\boldsymbol{w})\right)$
                $\Delta\boldsymbol{w}_v \leftarrow \Delta\boldsymbol{w}_v + \nabla_{\boldsymbol{w}_v} \left[R_t - \hat{V}(\boldsymbol{s}_t;\boldsymbol{w}_v)\right]^2$
            end
            Asynchronously update global weights $\boldsymbol{w}_g$ and $\boldsymbol{w}_{g_v}$ using learning
              rate $\alpha_e$
        end
        $t \leftarrow t + 1$
    until $\text{terminal} == \text{True}$
    $e \leftarrow e + 1$
    Adjust learning rate $\alpha_e$
    if $e \mod \tau$ then
        $\boldsymbol{w} = \boldsymbol{w}_g$
    end
end
```

B

Used learning parameters

This chapter lists the hyperparameters for the various algorithms that were put to use in this thesis.

B.1 Linear Q-learning

B.1.1 State representations

The hyperparameters for the state representation were learned using 20 different parameter configurations while the learning parameters of the Q-learner stayed constant. For every configuration, the experiment was repeated 10 times.

Fixed Sparse Representation (FSR) The Fixed Sparse Representation is representing the state vector

$$\boldsymbol{s}_{\text{dist}} = \left(|\overrightarrow{M_1T}|, |\overrightarrow{M_2T}|, |\overrightarrow{M_3T}|, |\overrightarrow{M_1O}|, |\overrightarrow{M_2O}|, |\overrightarrow{M_3O}|, \boldsymbol{L} \right)^\top ,$$

that was first defined in (8.1), using 50 bins in the first six continuous dimensions. The binary dimensions $\boldsymbol{L}$ are all represented using two bins per dimension.

Radial Basis Functions (RBF) The state representation based on radial basis functions represents the first three real-valued dimensions of the state vector (8.1), i.e. the distances between the target object and the tool's picking locations, via five uniformly placed radial basis functions per dimension. The other three real-valued dimensions of (8.1), incorporating the distances between the picking locations of the tool and the domain's origin, are expressed via three uniformly placed radial basis functions per dimension. The standard deviation of each RBF is set to

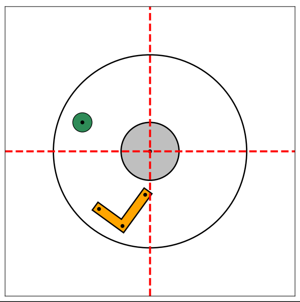
$\sigma = 1$. Additionally the RBFs are normalized according to (8.5.1). This kind of state representation segments the real valued state vector into 13829 quantized features.

B.1.2 Learning the “Extension-of-Reach Scenario” using different coordinate systems

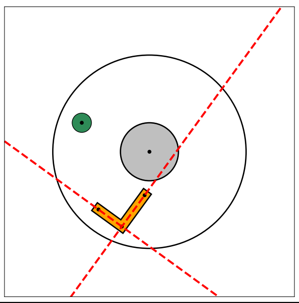
This section lists the chosen hyperparameters for learning the “Extension-of-Reach Scenario” when either the FSR or RBFs were used for representing the state vector. For every coordinate system (see Chapter 9 and Figure 9.1), the whole task was learned using 40 different parameter configurations while varying the parameters listed in Table B.1. For every configuration, the experiment was repeated 20 times. In the end, the parameter configuration was chosen that maximizes the accumulated average reward that was achieved during learning. The search range for every optimized parameter is listed in Table B.1.

Parameter	**Search range**
Greediness ϵ	uniform $[0, 1]$
Initial learning rate α_0	$10^{\text{uniform}[0.05,1]}$
Boyan decay-factor α_B	$10^{\text{uniform}[1,5]}$
Discount factor γ	uniform $[0, 1]$
Eligibility decay-factor λ	uniform $[0, 1]$

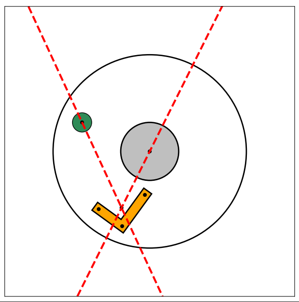
Tab. B.1.: The table lists the search-ranges for the hyperparameters of the linear Q-learner

Coordinate system	**Parameter**	**FSR**	**RBF**
	ϵ	0.014	0.001
	α_0	0.622	0.069
	α_B	24.289	10.183
	γ	0.931	0.297
	λ	0.072	0.748

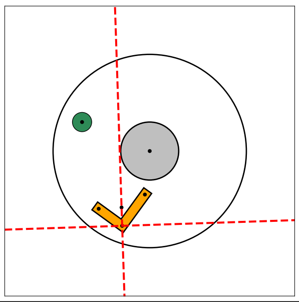
Tab. B.2.: List of the learning parameters for the “Extension-of-Reach Scenario” using the “World System”

Coordinate system	Parameter	FSR	RBF
	ϵ α_0 α_B γ λ	0.100 0.607 73.287 0.967 0.606	0.057 0.399 8942.364 0.976 0.811

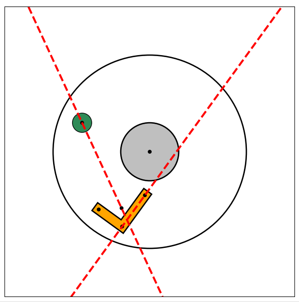
Tab. B.3.: List of the learning parameters for the "Extension-of-Reach Scenario" using the "Tool-Fixpoint System"

Coordinate system	Parameter	FSR	RBF
	ϵ α_0 α_B γ λ	0.76 0.913 59.52 0.988 0.191	0.0572 0.206 6927.641 0.930 0.630

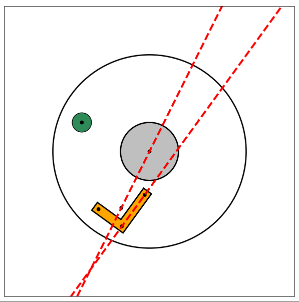
Tab. B.4.: List of the learning parameters for the "Extension-of-Reach Scenario" using the "Target-Tool-Goal System"

Coordinate system	Parameter	FSR	RBF
	ϵ α_0 α_B γ λ	0.176 0.323 40.27 0.982 0.315	0.227 0.118 7548.765 0.551 0.697

Tab. B.5.: List of the learning parameters for the "Extension-of-Reach Scenario" using the "Tool-Centroid System"

Coordinate system	Parameter	FSR	RBF
	ϵ α_0 α_B γ λ	0.997 0.063 10.346 0.238 0.534	0.035 0.991 15289.540 0.940 0.250

Tab. B.6.: List of the learning parameters for the “Extension-of-Reach Scenario” using the “Target-Tool System”

Coordinate system	Parameter	FSR	RBF
	ϵ α_0 α_B γ λ	0.089 0.313 40.183 0.955 0.329	0.019 0.089 4518.031 0.927 0.297

Tab. B.7.: List of the learning parameters for the “Extension-of-Reach Scenario” using the “Goal-Tool System”

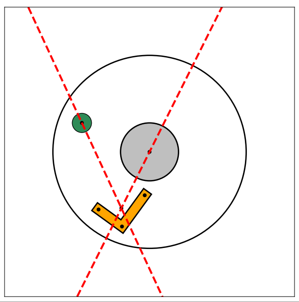
B.2 Deep Q-learning

The hyperparameters for the deep Q-learner are tuned only for the “Target-Tool-Goal System” while learning to solve the “Extension-of-Reach Scenario”. The whole task was learned using 40 different parameter configurations. The search-range of the hyperparameters is listed in Table B.8. In the end, the parameter configuration was chosen that maximizes the accumulated average reward that was achieved during learning. The best configuration of parameters for the deep Q-learner is listed in Table B.9.

For the deep Q-learner, a constant learning rate α is used. The greediness is decaying linearly from 1 to 0.1. As proposed in [7], a target network is used to compute the gradient which receives a copy of the original networks parameters every τ steps. The number of transition tuples stored in the replay buffer for experience replay is given by $\mathcal{E}$, while $\mathcal{E}_{\text{start}}$ defines the minimal number of tuples that are required to start the learning process. During training, the weights are updated using the Adam algorithm [67].

Parameter	**Search range**
$\mathcal{E}$	$10^{\text{uniform}[3,6]}$
$\mathcal{E}_{\text{start}}$	$10^{\text{uniform}[3,5]}$
ϵ-decay stop	$10^{\text{uniform}[2,6]}$
τ	$10^{\text{uniform}[1,3]}$
α	$10^{\text{uniform}[-5,-1]}$
γ	uniform $[0,1]$
Number of layers	uniform $\{3,4,5,6\}$
Number neurons	$10^{\text{uniform}[\log_{10}(32),\, log_{10}(128)]}$

Tab. B.8.: The table lists the search-ranges of the hyperparameters for the deep Q-learner

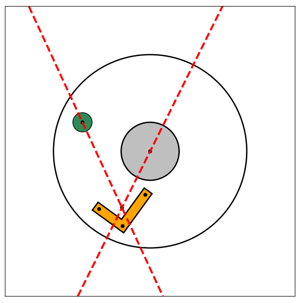
Coordinate system	**Parameter**	**Value**
	$\mathcal{E}$	10^5
	$\mathcal{E}_{\text{start}}$	$5 \cdot 10^3$
	ϵ-decay stop	$5 \cdot 10^5$
	Batch size	32
	τ	200
	α	$25 \cdot 10^{-5}$
	γ	0.95

Tab. B.9.: List of the learning parameters for the deep Q-learner

B.3 Recurrent attention advantage actor-critic model

The parameters for the RAA3C model are chosen by educated guess and manual tuning while learning the “Tool-Centered Interaction Scenario” and utilizing the “Target-Tool-Goal System”. They are summarized in Table B.10.

The weights of all layers are initialized using *He normal initialization* [252] with a bias of 0.01. The model is trained with 8 workers using the *Adam optimizer* [67] and an experience buffer with a maximal size of $N_{\text{EB}} = 8$. The weights of each worker are updated every $\tau = 10$ episodes. The discount factor is chosen to be $\gamma = 0.99$. The learning rate is starting at $\alpha_0 = 10^{-5}$. It decays exponentially every $T = 5$ episodes to $\alpha_{\text{min}} = 10^{-8}$ with a decay-factor of $\delta_\alpha = 0.99$ according to

Coordinate system	**Parameter**	**Value**
	β	1
	N_{EB}	8
	α_0	10^{-5}
	α_{min}	10^{-8}
	δ_α	0.99
	γ	0.99
	τ	10
	T	5 episodes

Tab. B.10.: List of the learning parameters for the RAA3C model

B.4 Haptic attention model

For finding good parameters, the HAM was trained using 40 different parameter configurations. The search-range of the hyperparameters is listed in Table B.11. The parameters are chosen according to *random search* [244] with a fixed number of 3 glances, followed by additional manual tuning. In the end, the parameter configuration was chosen that led to the highest accumulated average reward achieved during learning. The best configuration of parameters is listed in Table B.12. The weights of all layers are initialized using *He normal initialization* [252] with a bias of 0.

Parameter	**Search range**
Initial learning rate α_0	$10^{\text{uniform}[-5,-1]}$
Learning rate decay δ_α	$10^{\text{uniform}[\log_{10}(0.1),log_{10}(1)]}$
Learning rate decay steps T	$10^{\text{uniform}[2,4]}$
Weighting factor β	$10^{\text{uniform}[\log_{10}(0.1),log_{10}(1)]}$
Number neurons — LSTM	$10^{\text{uniform}[\log_{10}(128),log_{10}(512)]}$
Number neurons — Linear layer	$10^{\text{uniform}[\log_{10}(32),log_{10}(128)]}$

Tab. B.11.: The table lists the search-ranges of the hyperparameters for the HAM

Parameter	**Value**
Batch size	64
α_0	$8 \cdot 10^{-4}$
δ_α	0.97
T	800 training steps
α_{min}	10^{-6}
Used optimizer	SGD with Nesterov momentum
Momentum	0.9
β	0.4

Tab. B.12.: List of learning parameters for the HAM

C

Floating Myrmex sensor: experimental results

The main results for training the HAM on the large data set, presented in [24], are summarized in Table C.1. The classification performance for the full *haptic attention model* using an LSTM unit and a location network to generate new sensor poses can be found in the column $\boldsymbol{\pi_{\mathrm{LSTM}}}$. Replacing the location network with a random policy leads to the results that are marked with $\boldsymbol{\pi_{\mathrm{rloc}}}$.

# Glances	1. π_{LSTM}	2. π_{rloc}
1	0.547 ± 0.002	0.547 ± 0.002
2	$\mathbf{0.831 \pm 0.002}$	0.753 ± 0.002
3	$\mathbf{0.910 \pm 0.001}$	0.858 ± 0.001
4	$\mathbf{0.942 \pm 0.001}$	0.917 ± 0.001
6	$\mathbf{0.977 \pm 0.001}$	0.971 ± 0.001
8	0.988 ± 0.001	0.990 ± 0.000
10	0.994 ± 0.001	0.995 ± 0.000

Tab. C.1.: The table lists the best measured classification performances after $10 \cdot 10^3$ training steps. The full meta-controller model π_{LSTM} contains all trained components including the LSTM module and the location network. The random location policy approach π_{rloc} substitutes the location network with a random location generator.

D Supplementary material

`CoordinateSystems.mp4` A short video that visualizes policies that are learned by the agent while employing the different coordinate systems (see Chapter 4 & 9). In the video, the agent tries to solve the "extension-of-reach task".

`RAA3C.mp4` A short video of a learned policy when utilizing the RAA3C model presented in the Chapters 5 & 11. In the video, the agent tries to solve a "tool-centered interaction task" using 6 glimpses and a 40×40 context image.

`SkillTransfer.mp4` An illustrative video, explaining the skill transfer approach that is presented in the Chapters 7 & 10.

`HAM_1.mp4` A short video that further illustrates the mode of operation of the used collision library and the resulting way the tactile data is measured by the simulated Myrmex sensor (see Chapter 12).

`HAM_2.mp4` A short video that shows the learned policy of the HAM (see Chapter 6 & 12) on a simulated robot arm with an attached tactile sensor.

`KUKA.mp4` A short video that shows the early testing off the control scheme in a real world setup (See the paragraph about future work in Section 12.5). The KUKA robot arm performs one haptic glance per object. Only the pressure threshold and the stiffness parameters are re-adjusted in comparison to the simulation.