IPVS-SC

Masterarbeit

Simulation meets Real-World: Deep Reinforcement Learning on Inverted Pendulum System

Linus Bantel

Course of Study:	Artificial Intelligence and Data Science
Examiner:	Prof. Dr. Dirk Pflüger
Supervisor:	Peter Domanski, M.Sc.
Commenced:	June 1, 2023
Completed:	December 1, 2023

Abstract

In this thesis, we investigate the differences between the idealized gymnasium cartpole environment and a real cartpole with the aim to train robust agents in the simulation, that perform well in real-world tasks. In this work, we not only consider the classical upright task, but also the so-called swingup. Models for friction and force are implemented and their effectiveness is evaluated on the real cartpole. The robustness of an agent with regards to changing parameters of the cartpole is also examined and possible solutions presented.

Contents

List of Figures

List of Tables

1 Introduction

The current rise of machine learning is omnipresent. With an expected exponential growth over the next years [Sta], it will be apparent in our everyday life. Many problems are nowadays too complex to be solved by classical algorithms. Ongoing discussions about language models like ChatGPT or Google Bard are just one hint at the importance of machine learning.

Typical areas, where machine learning techniques shine, are computer vision and Natural Language Processing, but there are many more areas, all of them are subject to current research [IBM]. An often, even politically, discussed area is autonomous driving. Classical algorithms are by far not enough to enable safe self-driving cars. There are too many variables and situations that the car needs to analyze and be aware of. Tesla for example uses data from their large fleet all around the world to train the neural networks used in their system called “autopilot“ [Tes]. While this is certainly one way to do it, it is only possible when these large amounts of data are available. And even then, they do not guarantee full coverage.

Another way to collect data is the use of simulations. Therefore, projects like CARLA[DRC+17] are brought to life. Carla aims to enable the training and validation of autonomous driving systems. A simulation enables much faster and therefore cheaper data acquisition. Since a crash in a simulation doesn’t cost any money and, even more importantly, does not harm any life, reinforcement learning can be another viable option. Reinforcement learning is a technique where an agent tries to learn based on the movements it executes. This is infeasible in the real world, however, very well possible in the simulation. A big question mark is the transferability to the real world of such agents, that were solely trained in the simulation. As a consequence, great effort is put into the investigation of real-world capabilities of reinforcement learning algorithms that were trained on simulations.

In the reinforcement learning community, the problem of the cartpole inverted pendulum is considered to be solved. The cartpole is an unstable nonlinear system, that often is used as a first test and benchmark for controlling strategies. which is also the case for reinforcement learning algorithms. The typical benchmark consists of an idealized simulation of the cartpole to allow for comparable results. One major simplification in the environment among others is the lack of any friction. As a starting point, the upright position is chosen. Thus, the classical cartpole problem is to keep the pendulum upright. One can also start in the hanging position with the goal, that the controller moves the cart in such a way, that the pendulum reaches the upright position and is beeing held there. This task, called swingup, is much harder than the classical problem of keeping the pole upright.

With the cartpole simulation being a community-wide benchmark, it is remarkable that controllers are solely tested on the simulation and not on a real cartpole experiment. Besides that, the swingup problem is also rarely mentioned nor used as a test.

Starting from the classical benchmark in the simulation, we examine the transition from the simulation to the real world for both cartpole problems. The simulation is therefore upgraded to a higher degree of realism and the effectiveness examined. Furthermore, the robustness to changing parameters is considered to train an agent with a very high performance on the real cartpole.

2 Related Work

The inverted pendulum cartpole problem has been and still is a fundamental benchmark for reinforcement learning techniques. Since it is considered as being solved, it serves as a first, easy to conduct, experiment to evaluate new techniques and algorithms. Although these tests are carried out only in a simulated environment, the transferability of techniques, especially from simulation to the real world is often neglected.

Starting with classical approaches, this chapter presents related work regarding the control of the cartpole, as well as reinforcement learning-based approaches.

2.1 Controlling the cartpole

Classical and widely known approaches in control theory, that were successfully applied to the cartpole are,e.g., LQR and PID controller. While it is hard to find the origin of such generally and widely used techniques, a lot of research can be found regarding such controllers tested on the cartpole. In the 2015 published paper "Optimal Control of Nonlinear Inverted Pendulum System using PID Controller and LQR [...]"[PTG14] the authors model and design these two controllers and test them on a simulated cartpole. Problems that arise from techniques like PID and LQR are their linear nature. The cartpole however is a nonlinear system. This nonlinear nature makes a controller based on fuzzy theory a great fit. In "Switching-type fuzzy sliding mode control of a cart-pole system"[LS00] a sliding fuzzy controller is used to balance the pole. They did not only test their controller in a simulation but also experimented on a real cartpole. All controllers have in common, that they are designed specifically for the task and knowledge about the system is necessary. Fuzzy controllers can typically generalize much better than, e.g., a PID controller, but some knowledge is still required.

This is where the strength of reinforcement learning algorithms comes into play. In the paper [NPUG17] the authors examine different reinforcement learning algorithms based on the performance of the cartpole problem. This test is, however, carried out on the simulation and only for the classical task of keeping the pole upright. They achieve the swingup with a separate energy-based control strategy and the RL controller takes over for angles smaller than 12°. One single controller that achieves both, the swingup and the balancing of the pole, is preferred and presented in this thesis.

2.2 Reinforcement learning-based techniques

In the paper “Playing Atari with Deep Reinforcement Learning“[MKS+13] the DQN algorithm is presented. They achieved remarkable scores in seven different Atari games while even surpassing a human player in three of them. The Deep Q-Network algorithm serves as a foundation and benchmark in the area of deep reinforcement learning and is also used in this thesis.

In the bachelor thesis [Nay21] a controller was developed that collects data of the real pendulum and synchronizes this data via Dropbox with a PC that uses this data to additionally learn from. The author discusses the importance of the damping of the pole and achieves the swingup successfully. We, however, found the friction of the cart to be much more influential than the damping of the pole. Additionally, we provide a way for the controls to be graded finer.

Dulac-Arnold et al. [DLM+21] propose a real-world reinforcement learning suite that implements nine challenges proposed in their paper. These challenges cover the limited samples of real-world systems, partially stochastic systems, and many more. As one possible solution to increase the robustness of reinforcement agents, the authors suggest perturbating parameters of the simulation stochastically. The technique of perturbating parameters to increase performance and robustness on the real cartpole is evaluated in this thesis and the importance of such a testing suite consolidated.

3 Inverted Pendulum

The inverted pendulum generally describes the setup of a pendulum, on which torque can be applied. The task for such an inverted pendulum system consists of keeping it in the upright position, thus its name “Inverted Pendulum“. For obvious reasons, the pendulum needs to be rigid in order to be able to stay upright. Thus, a string pendulum with a weight attached to it cannot be used. To exert torque on the pendulum there are a multitude of possible setups. Two typical setups are

1. The pendulum axis is rotated off-center around a motor-driven axis
2. The pendulum is attached to a cart which can move left to right

The different setups can be seen in picture Figure 3.1. This thesis covers the control of type 2, also known as a Cartpole or Cartpole Inverted Pendulum. Generally speaking, a function respectivly control strategy needs to be found, that maps the state of the pendulum to an action.

3.1 Physics

To create or find a control strategy to keep the pendulum upright, it can be beneficial to understand the unstable behaviour of a rigid pendulum in the upright position. Afterwards, the differential equations for the whole cartpole setup are presented.

3.1.1 Unstable Upright Position

The main variable describing the pendulum is its angle relative to the centerline. Therefore a differential equation using the angle as input to describe the motion would be desirable. From the classical Newtonian Equation of Motion, we know the equation $F = ma$, setting the accelerating relative to the applied force, with the mass as a scaling factor.

Figure 3.2 shows the general setup of the pendulum.

The force on the pendulum exerted by gravity is described by $F = m * g$. It can be decomposed into two components, acting on the pole. One component, $mg \cos\theta$, is collinear to the pole and does not contribute to the movement. The other component, $mg \sin\theta$, tangential to the circle of movement, is the one accelerating and decelerating the pendulum. Thus $F = -mg \sin\theta$. The minus sign comes from the fact, that the force pulls towards smaller angles. The acceleration can be described via differentiation of the arc, and thus via the angle, see Equation (3.3).

(a) The rotary inverted pendulum by Quanser [Quab] **(b)** The cartpole inverted pendulum by Quanser [Quaa]

Figure 3.1

$$s = l\theta \tag{3.1}$$

$$v = \frac{ds}{dt} = l\frac{d\theta}{dt} \tag{3.2}$$

$$a = \frac{d^2s}{dt^2} = l\frac{d^2\theta}{dt^2} \tag{3.3}$$

Finally, the differential equation can be written as in Equation (3.4).

$$\frac{d^2\theta}{dt^2} = -\frac{m}{l}\sin\theta \tag{3.4}$$

Equation 3.4 can now be used to understand the behaviour of the pendulum. Due to the nature of the sinus function, there are two points of interest. $\theta = 0°$ and $\theta = 180°$. For positive small angles, the right side of equation 3.4 is negative, giving the typical behaviour of a pendulum, falling towards the vertical downward position. The same happens for small negative angles. However, since the flip of the sign of the right side happens at exactly 180 degrees, a force towards smaller angles is also present for angles close to 180 degrees. This fact makes the upright position of the pendulum unstable. In other words, the smallest disturbance from the upright position makes the pendulum fall.

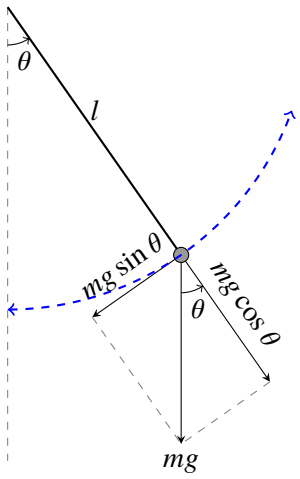


Figure 3.2: Forces acting on the Pendulum. Gravity can be decomposed into a radial component and a tangential one. Only the tangential force contributes to the angular acceleration.

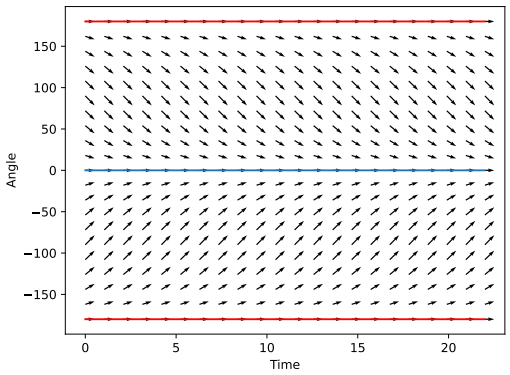


Figure 3.3: Acceleration acting on the pendulum. Red shows the unstable solution, blue the stable one.

3.1.2 CartPole Equation of Motion

Now the cart is introduced into the system, see Figure 3.4. The cart is constrained by the track such that it can only move in one dimension. The cart is accelerated via a force F that is applied to it. Its state is described by its position and velocity. Attached to the cart is the pendulum. The discussion in Section 3.1.1 still applies here, the vertical upright position is unstable.

From now on in this thesis, the angle of the pendulum is measured relative to the upright position. This means a small angle of the pendulum is desired, in the ideal case, an angle of zero degrees. Similar to the cart, the pendulum is described by its position and velocity. Since we are only interested in the angle of the pendulum, the analogous quantity for the description of the pendulum is the angle θ and the angular velocity $\dot{\theta}$. This results in the state, describing the whole cartpole system, $(x, \dot{x}, \theta, \dot{\theta})$.

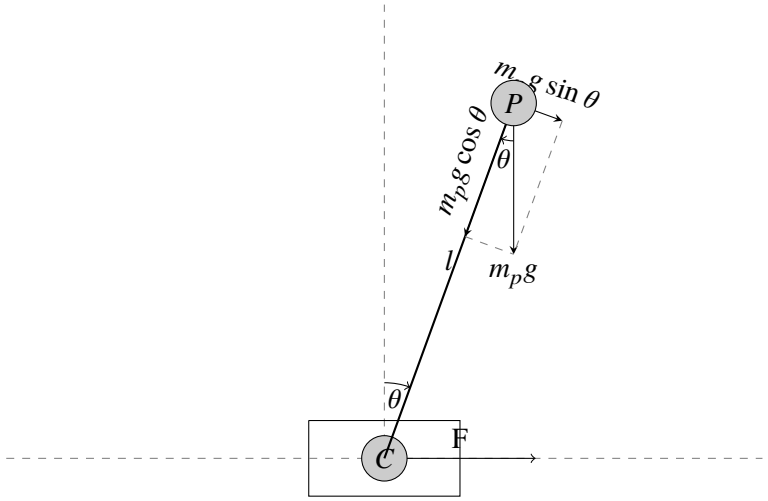


Figure 3.4: Forces acting on the cartpole

Variable	Description	Unit
g	Gravitational Constant	$\frac{m}{s^2}, \frac{N}{kg}$
F	Force applied on the Cart, positive to the right	N
m_c	Mass of the Cart	kg
m_p	Mass of the Pole	kg
l	Length of the Pole	m
θ	Angle of Pole, positive to the right	radians
$\dot{\theta}$	Angular Velocity	$\frac{radians}{s}$
$\ddot{\theta}$	Angular Acceleration	$\frac{radians}{s^2}$
x	Position of the Cart	m
$\dot{x}$	Velocity of the Cart	$\frac{m}{s}$
$\ddot{x}$	Acceleration of the Cart	$\frac{m}{s^2}$

Table 3.1: Physical Quantities of the Cartpole

The system of differential equations describing the cartpole inverted pendulum is given by Equation (3.5) and Equation (3.6)[Gre20]. Note that the equation for the cart acceleration requires the angular acceleration. Thus, Equation (3.5) needs to be evaluated first, as the result is needed in Equation (3.6). One can also derive a similar pair of equations, with the dependency switched around.

$$\ddot{\theta} = \frac{(m_c + m_p)g \sin\theta - \cos\theta (f + m_p l \dot{\theta}^2 \sin\theta)}{(1 + k)(m_c + m_p)l - m_p l \cos^2\theta} \tag{3.5}$$

$$\ddot{x} = \frac{f - m_p l \ddot{\theta} \cos\theta + m_p l \dot{\theta}^2 \sin\theta}{m_c + m_p} \tag{3.6}$$

Table 3.1 describes the variables.

3.1.3 Numerical Integration

Given the differential equations, it is possible to calculate the second derivatives of the position and angle, i.e. the acceleration of the cart and the angular acceleration of the pole. In order to calculate the first and zero order derivatives of the position and angle, one needs a numerical integration, the second part of the simulation.

One of the simplest schemes is the explicit Euler integration, see Equation (3.7). It approximates the integral via left-side boxes, see Figure 3.5. The error of this integration scheme is the area between the boxes and the curve. Reducing the widths of the boxes is often used to derive a formula for continuous integration since the limit for infinitely thin boxes equals the true integral. This is, however, not possible for solving differential equations, since an arbitrarily small step width would result in arbitrarily large computational effort. This error between the numerical integration and the true integral leads to one big problem: typically the energy in the system is not conserved, but rather increased over time, making the simulation unstable. This is also the case for the cartpole equations when using a timestep of $0.02s$. Despite starting in the hanging position, the pole reaches the upright position within a short period of time. Without any forces applied to the cart. The gymnasium environment uses this integration scheme as default. This is fine when starting in the upright position, since the whole energy in the system is much less than during the swingup due to the termination of the simulation when the pole falls.

Luckily, the semi-implicit Euler scheme is already implemented. It is a symplectic scheme and is thus much more suited for the differential equations of the cartpole. In fact, for the same timestep where the explicit Euler is unstable, the semi-implicit Euler is stable. The semi-implicit scheme can be seen in Equation (3.8). Not the difference of the second equation. The position is updated using the already newly calculated velocity.

$$\dot{x}_{new} = \dot{x}_{old} + \ddot{x} * \tau \tag{3.7}$$
$$x_{new} = x_{old} + \dot{x}_{old} * \tau$$
$$\dot{x}_{new} = \dot{x}_{old} + \ddot{x} * \tau \tag{3.8}$$
$$x_{new} = x_{old} + \dot{x}_{new} * \tau$$

3.2 Real-World-Pendulum

Figure 3.6 shows the real cartpole manufactured by Quanser, on which the agents are tested. The original hardware consists of the pendulum itself, as well as a power supply and a PCI card to read the sensor values and control the motor of the pendulum. The main problem with this setup is the PCI card. The PCI standard was last updated in 2004 and is thus lacking support for modern systems.

In a previous project, the hardware of this pendulum was updated. It now consists of a microcontroller, namely the ESP32, whose task it is to read the sensor values of the pendulum and control its motor. The microcontroller is interfaced via a USB-to-Serial connection. This has the advantage, that the pendulum can be accessed by most of the modern programming languages. For Python, this

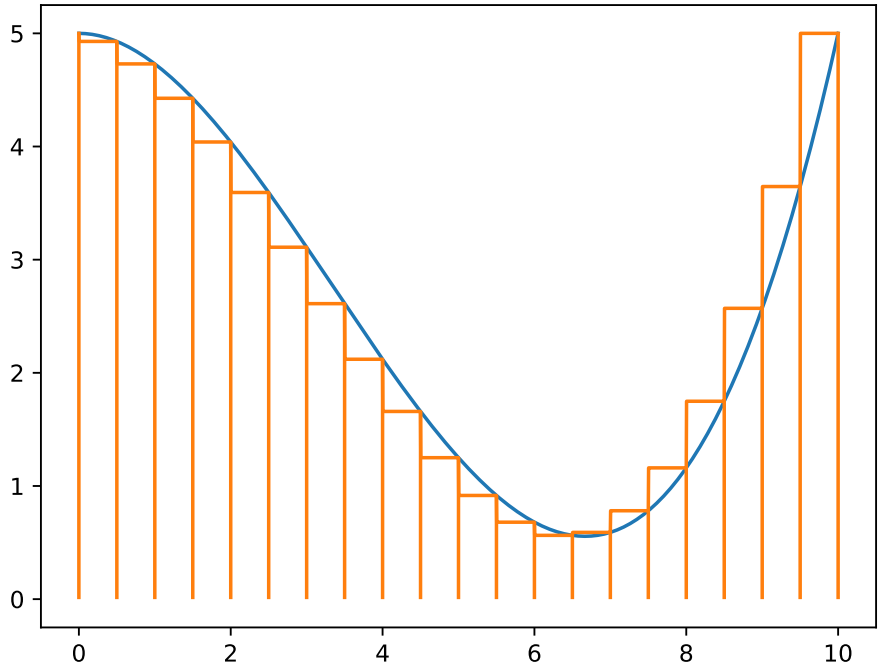


Figure 3.5: Simple Box Integral The difference in error between the boxes and the curve is the Error. For smaller box widths, the approximation quality improves.

is done via the library pyserial and a wrapper class to implement a simple protocol. The sensors used on the cartpole are rotary encoders with a resolution of 4096 steps per revolution. This allows for an angular resolution of $\frac{360^\circ}{4096} \approx 0.0879^\circ$.

Thus there are mainly four important methods, start, stop, getState and setMotor. These can be easily wrapped to make them compatible with the reinforcement learning framework.

3.3 Challenges

The classical cartpole challenge in the context of reinforcement learning consists of keeping the pendulum in the upright position. At the start of the challenge, the pole is therefore already in the upright position. The much harder task, which is also less frequently considered in reinforcement learning literature is the swingup task. At the start, the pendulum is close to the downward hanging position. The corresponding task consists of bringing and keeping the pole in the upright position. The classical cartpole task is considered to be a true subset of the swingup task.

To avoid any ambiguity during the evaluation, the swingup task is always explicitly mentioned when examined.

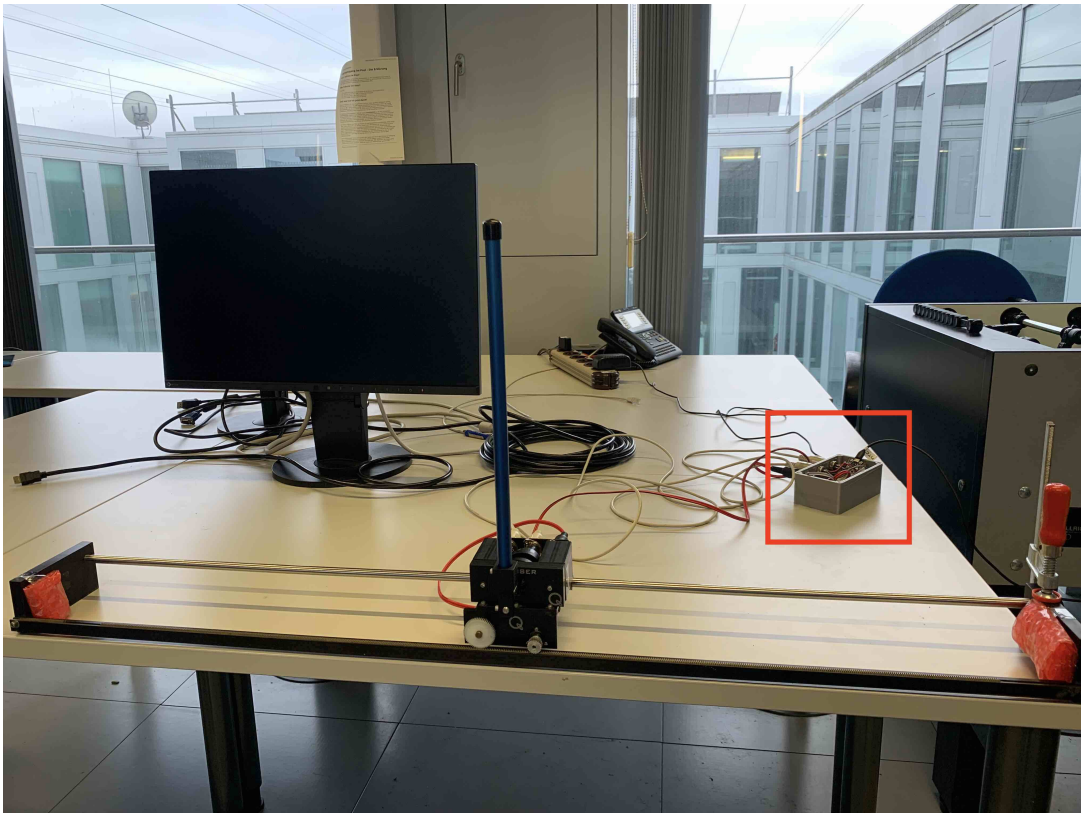


Figure 3.6: The Pendulum by Quanser. The grey box on the table houses the electronics to drive the cart and read the sensors.

4 Reinforcement Learning

Reinforcement learning, besides supervised and unsupervised learning, is one of the three big areas of machine learning. Whereas supervised learning needs input/output pairs in order to train the network, reinforcement learning algorithms generate knowledge from an agent interacting with an environment by taking actions and getting rewarded accordingly. The goal is to maximize the cumulative reward. This basic idea is depicted in Figure 4.1. More precisely, the agent gets the state of the environment and then chooses the action, that it thinks will produce the highest reward. The environment then evaluates the action given the previous state and rewards the agent accordingly. This iterative process is repeated with the goal, that the agent learns the best possible moves given a certain state. This typically starts with a more or less random approach, i.e. the agent simply tries random actions, since it hasn't seen anything yet. With more and more iterations, the agent gets a better idea of the environment and its behaviour and can thus, choose better actions. In an ideal case, the agent converges, such that it always picks the best action.

With the basic concept of reinforcement learning in mind, the following two sections will explain the reinforcement learning algorithm used for controlling the pendulum, namely the Deep Q-Network (DQN) and its predecessor, the Q-learning, which is however not suitable to train on an inverted pendulum system.

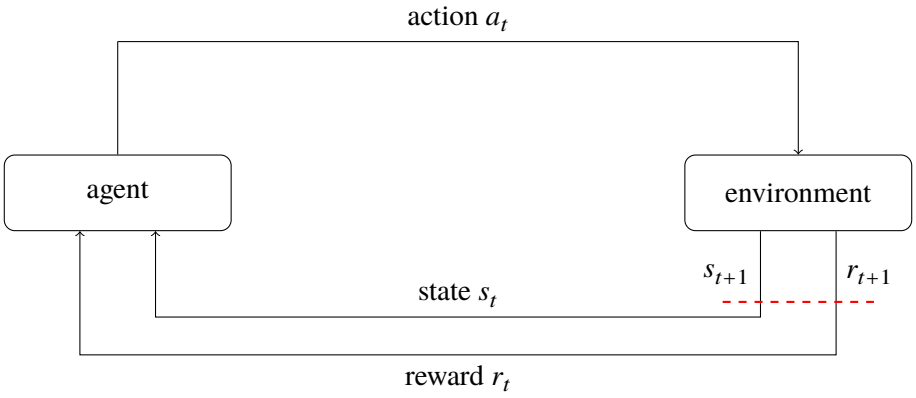


Figure 4.1: Basic Concept of Reinforcement Learning

Table 4.1: Exemplary Q-Table

state \ action	up	down	left	right
$(0,0)$	2	0	0	2
$(1,0)$	2	0	1	0
$(0,1)$	0	1	0	2
$(1,1)$	0	0	0	0

4.1 Reinforcement Learning Agent

4.1.1 Q-Learning

Q-Learning[WD92] is a reinforcement learning algorithm introduced in 1989. It is a model-free algorithm, meaning that the algorithm itself does not build any internal model of the environment. It can solely learn by interacting with the environment. At the core, the algorithm fills a table, also called Q-table, consisting of state/action pairs with the Q-values. The Q-value represents the expected reward when a given state is observed and the corresponding action is chosen. In order to find the best action, the agent can simply look in the table at the row corresponding to the observed state and take the action with the highest Q-value. Table 4.1 shows an exemplary Q-table.

The Q-values can be iteratively approximated using a Bellman equation. It is depicted in Equation (4.1).

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) * Q_(s_t, a_t) + \alpha * (r_t + \gamma * \max_a Q(s_{t+1}, a)) \tag{4.1}$$

With $0 \leq \alpha \leq 1$ as the learning rate, the equation is fundamentally a linear interpolation of the old Q-value and the reward added to the new expected best value.

While this algorithm can theoretically find an optimal control strategy, it has one big drawback. It can only handle discrete states. The pendulum, where its state is given by $(x, \dot{x}, \theta, \dot{\theta})$ is very much continuous. Thus basic Q-learning is not feasible. This is where deep Q-learning comes to the rescue.

4.1.2 Deep Q-Learning

As neural networks can be used to approximate arbitrary functions, they can also be used to approximate the Q-table. Mnih et al. [MKS+13] introduced this idea first in 2013 and successfully trained a network to play Atari games.

Updates are computed the same way using the Bellman equation (Equation (4.1)). The notable difference is, that the neural network is updated using gradient descent with the state and approximated Q-value of the Bellman equation. The deep neural network gets the state as input and has as many output nodes as there are actions. The output nodes shall approximate the q-value of its

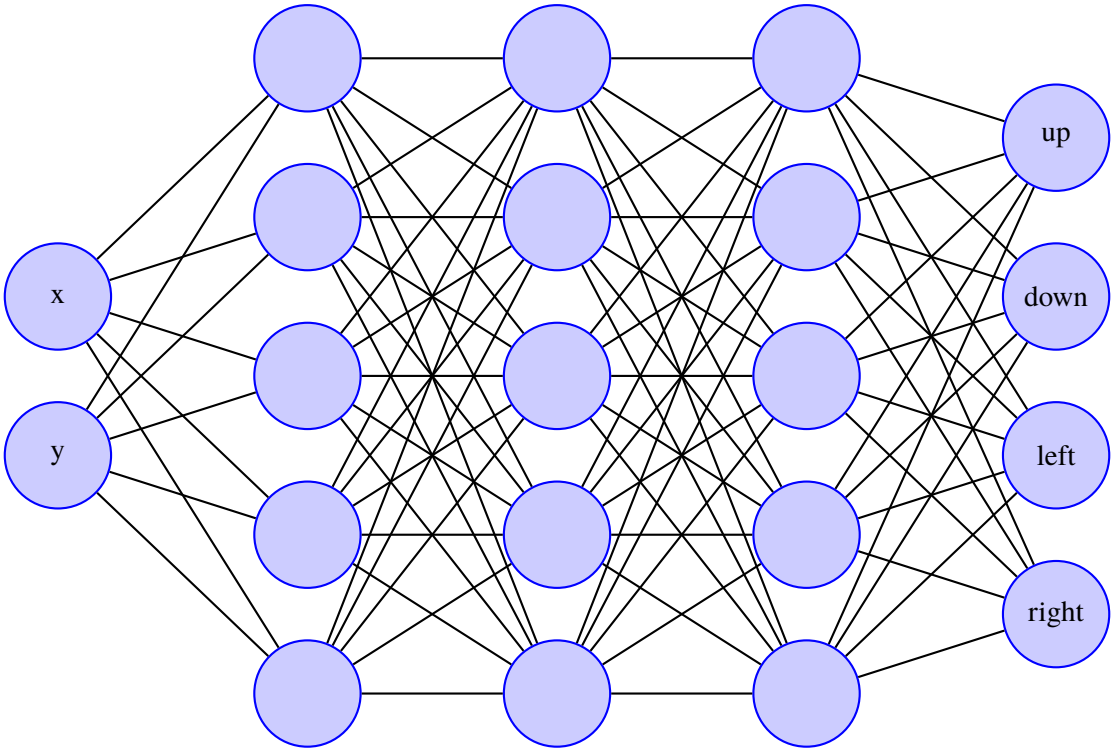


Figure 4.2: Deep Q-Network: Fully connected network to approximate table 4.1

corresponding action, given the input state (see Figure 4.2). While the inputs can now be arbitrary tuples of real values, the output is still discrete. The resulting network is often called Q-network. Learning over many episodes has the goal that the neural network converges similarly to the Q-table. This is, however, often a finicky process where the hyperparameters must be tuned precisely.

Parameters

Deep Q-learning reinforcement agents have a lot of parameters that can be tuned to achieve better and/or faster policies. In the following, we introduce key parameters of the algorithm, that are important to achieve the cartpole tasks.

Epsilon_greedy: During the learning and exploration phase, the agent must try actions and is rewarded. However, it is fairly possible that the agent might get stuck in a local maximum, i.e. it would have to change the policy in such a way, that it would reduce the reward in the short term, but ultimately overcome it and find a better maximum than the one before. To reduce this problem the epsilon parameter is introduced. It is the probability that a random action is taken instead of the one suggested by the policy. It can also speed up the discovery process.

target_update_period: As introduced, there is only one single network in the DQN algorithm that takes actions and predicts the Q-values for its moves. After one iteration this network gets updated. This can lead to unstable behaviour during the learning process and in the worst case, not achieving the ultimate goal. With this parameter, the idea is that there are two networks, one that decides which actions to take and one that gets updated normally with gradient descent. After target update period amount of iterations, the updated network gets copied on the action-deciding network.

gamma: As in Equation (4.1) seen, there is a factor gamma, the so-called discount factor. It can be described as a factor, which decides whether a short-term or long-term maximum should be achieved. Typical values are close to one, preferring a higher long-term reward, while still being able to break loose from local maxima, which are hard to escape for the algorithm.

4.1.3 Replay Buffer

Deep Q-network reinforcement learning is categorized as a so-called off-policy algorithm. This means that trajectories are stored inside a buffer and can be used multiple times to update the network. Trajectories are tuples consisting of the state, the action taken, the following state and its corresponding reward. This reuse of trajectories makes off-policy algorithms more sample-efficient than on-policy algorithms. The latter ones learn episode by episode and discard all trajectories after each iteration.

The main parameter regarding the replay buffer is its size. It usually has a finite length and can be seen as a queue. If it is full, the oldest trajectories are discarded in order to make room for the new ones. The size of the buffer is thus a tradeoff. A shorter buffer can enable faster learning since old trajectories with worse rewards get replaced faster. However, similar to a low target update period, this can lead to unstable learning behaviour.

4.2 Environment

As seen in Figure 4.1 reinforcement learning does not only consist of the algorithm itself, it needs an environment that the agent can interact with. The environment, as seen by the agent, can be seen as a black box. The only thing, that the agent sees, are consecutive states and the rewards associated with each state. The inner workings, i.e. in the case of this thesis the simulation of the cartpole, are completely hidden from the agent.

In order to make the interaction of the agent with the environment possible, an interface is necessary. For this, OpenAI introduced with gym a simple interface consisting of four functions:

- `reset()`
- `step()`
- `render()`
- `close()`

The `reset()` function resets the environment to a known state and returns it. Variables that are used during the simulation can be reset and altered here between runs. In the `step()` function the main part of the simulation happens. It takes an action and returns the next state, the reward and possible additional information. The `render()` function allows to output a visual representation of the state of the environment. This makes it easy to render a video of the interaction of the agent with the environment. For completeness, the `close()` function closes the environment. In the scope of this thesis, the function is not relevant.

While the interface is still the same, gym is no longer maintained by OpenAI, a fork called gymnasium maintained by the Farama Organisation is a drop-in replacement [22].

Tensorflow has a similar, but not quite the same interface. The main difference of the interface is that gymnasiums API uses typical Python datatypes including numpy-arrays. The TensorFlow API uses tensor objects for interaction. It has the advantage, that these objects can easily be fed into neural networks. Luckily Tensorflow provides a function that wraps gymnasium environments.

5 Setup

As in Chapter 4 presented, reinforcement learning consists of mainly two parts, the environment, which in the case of this thesis is the simulation of the cartpole and the agent, that wraps the DQN and provides the algorithm to train the network. In order to save the trajectories, a replay buffer is also needed.

The main training loop is depicted in Figure 5.1. It consists of two sub-parts, data collection in green and training in yellow. During data collection, the agent predicts an action which is used by the environment to calculate the next state. The trajectory consisting of the two consecutive states, the action and the reward is saved in the replay buffer. During the learning phase, trajectories are sampled from the buffer, typically equally distributed. Using this sampled batch the agent, the DQN respectively, is trained. Having these two parts conceptually separated also allows the collection of more than just one trajectory per learning step.

5.1 Reinforcment Learning Algorithm

The implementation of the reinforcement learning algorithm using TensorFlow can be done with very little effort. The main components that need to be instantiated are the deep Q-network, the agent itself and the replay buffer.

5.1.1 Q-Network

With TensorFlow providing the agent and functions to create neural networks, TensorFlow/Keras functions can be used to create the deep Q-network.

As the state of the cartpole is represented by a simple four-dimensional vector, a typical Q-network would be a densely connected neural network. For a dense neural network, one has the number of layers and their respective amounts of neurons as parameters. For each layer different activation functions can be chosen.

The size of the input layer is dictated by the state of the environment. The output layer is determined by the amount of different actions the environment allows for. For the experiments in this thesis, the typical size of the Q-networks between the input and output layer are $(64, 64)$.

Since consecutive states are time-dependent, the use of a recurrent neural network can also be an option. One big drawback of such a network is the higher computational effort to train them. Convergence is also harder to achieve. For the swingup task, no convergence could be achieved.

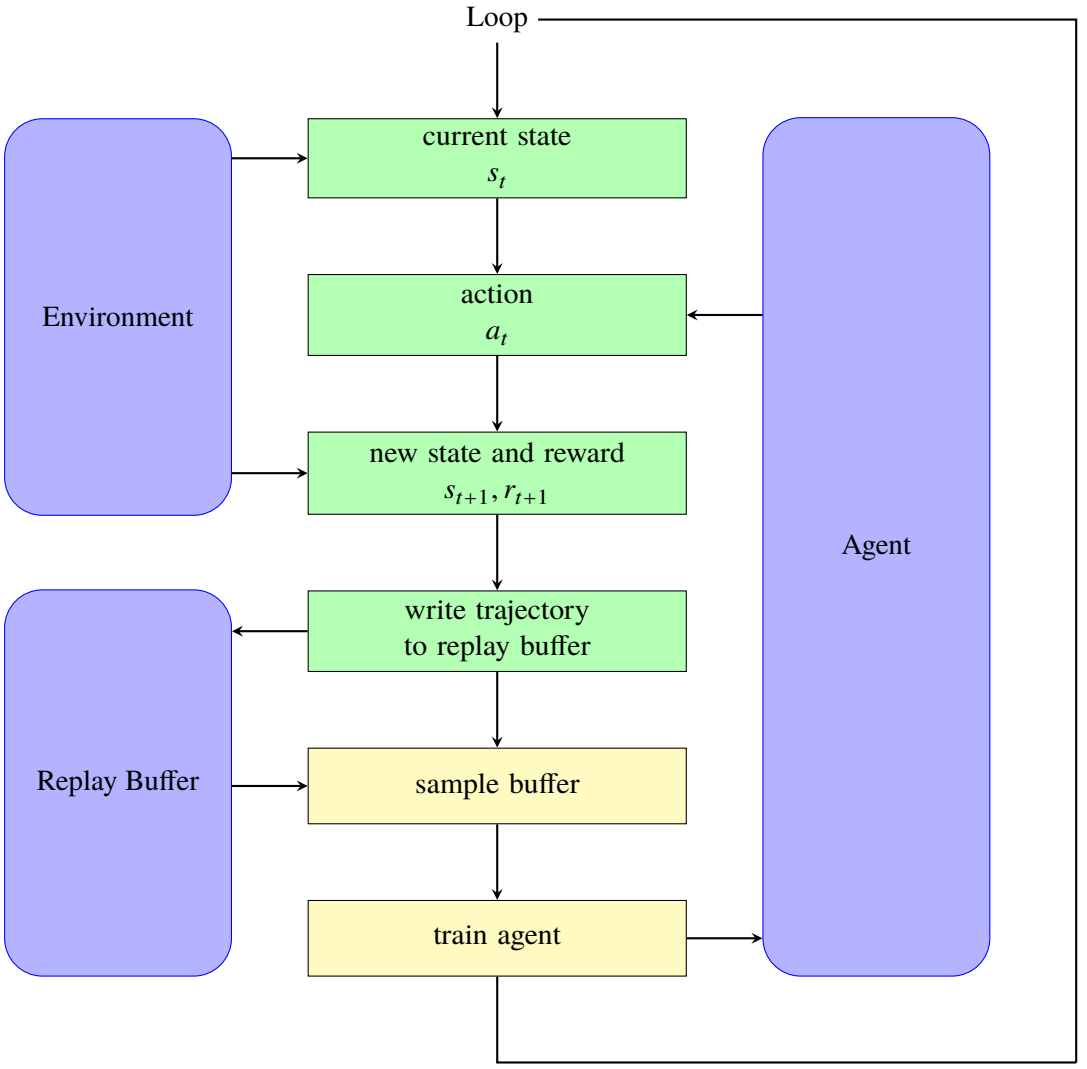


Figure 5.1: A high-level overview over the training loop. **Green:** Data Aquisition. Agent interacts with the environment to create trajectories which are saved in the replay buffer. **Yellow:** Learning. Trajectories are sampled from the replay buffer and the agent is trained on them.

5.1.2 Agent

After the Q-network is defined, the agent can be instantiated. It needs to know the structure of the environment and the amount of actions the environment allows for. Furthermore, it needs to know the Q-network, an optimizer used to update the Q-network and how the loss between prediction and, in the case of DQN, the Bellman equation is calculated. Parameters discussed in Section 4.1.2 are also given to the agent during initialization. The optimizer mainly used for the experiments is the Adam Optimizer ([KB14]). It seems to be the de-facto standard, since it is not only used in many tutorials of TensorFlow regarding reinforcement learning but is also the most used optimizer in OpenAI's baselines [DHK+17] and Google's dopamine [CMG+18]. In the paper on the AdamW optimizer [LH17] it is stated, that their optimizer is an improvement to the original Adam optimizer, with better performance in generalization based on an image classification test. Being only a drop-in replacement for the Adam optimizer, the AdamW is also evaluated.

5.1.3 Replay Buffer

The replay buffer only needs to know the shape of the trajectories that are saved and the size of the buffer.

5.2 Environment

For this thesis, the gymnasium cartpole environment[TTK+23] is used as a foundation to build on. It has the differential equations already implemented as well as the Euler-integration step. Since the classical cartpole task only consists of keeping the pendulum upright with many physical simplifications, some modifications to the original environment need to be made.

5.2.1 Reset-function

The job of the reset function is, as the name implies, resetting the simulation and returning the initial state. As one goal of this thesis is to achieve the swingup of the pendulum, starting every time in the upright position is obviously not a good idea. Starting in the hanging position is one possibility since it resembles the challenge best. However, to speed up the exploration of the possible states of the cartpole, it proved to be beneficial to start with arbitrary angles for the pendulum. For the evaluation of the agent's performance the hanging start is preferred. On the one hand, the full swing-up process shall be evaluated, on the other hand, starting the simulation in very different states randomly, can impact the value of the cumulative reward very strongly and thus making the evaluation of an agent very difficult.

Since this function is always called at the beginning of a run, run-specific parameters can be reset or modified. Later in this chapter physical parameters are determined in order to bring the simulation closer to reality. Models however use always some form of simplifications and parameters needed in these models can only be measured with uncertainty. In an attempt to make the reinforcement agent more robust, random perturbations can be applied to parameters.

It is also possible to train an agent for multiple values of a parameter. In Chapter 6 it will be shown that it is possible to train one single agent to control the cartpole with different lengths of the pole. To achieve this, the parameters of one of the two poles are chosen in the reset function and then used for this run of the simulation.

5.2.2 Step-function

The main part of the simulation takes place inside the step function. For this, the differential Equations (3.6) and (3.5) are used to calculate the new acceleration of the cart, as well as the angular acceleration of the pole. Via an Euler-integration step, corresponding velocities and positions are determined. With the new state present, the reward can be calculated and the tuple of the new state and reward are returned.

Since the real pendulum is typically controlled at 50Hz, the function should also calculate the new state from the previous state after $\tau = \frac{1}{50Hz} = 0.02s$. To possibly improve the stability of the integration step, a finer stepping width can be used.

The original implementation of the cartpole simulation uses some major simplifications. The simulation does not contain any sort of model for friction. Neither the friction between the cart and the track nor between the pendulum and the cart are taken into account. Another simplification is, that there are only two possible actions, left and right. While this is enough to keep the pole straight inside the simulation, the agent can make use of a finer gradation of force levels. This can not only lead to better performance but also protect the hardware from excessive wear and tear.

Reward

The original reward function used in the OpenAI Cartpole example is Equation (5.1). The agent tries to maximize the cumulative reward and therefore the amount of timesteps until the simulation is terminated. This is the case as soon as the pole's angle is greater than a set threshold, as well as when the cart leaves the track, i.e. $abs(x)$ becomes greater than a certain threshold. The original threshold for the angle is 12°, where the two actions, left and right, are enough to keep the pendulum's angle within the boundaries. This reward function, while probably being the most simplistic one, doesn't tell the agent a performance difference between a strongly oscillating cart and one very close to the upright equilibrium.

To tackle this problem, Equation (5.2) is introduced. The closer the pole is to the upright position, the higher the reward. The scaling factor of 0.5 is used in order to keep the reward in the interval $[0, 1]$. Another possible reward function that achieves a similar behaviour is Equation (5.3). While still being on the interval from zero to one, it offers a constant gradient regarding the angle. This can push the agent more towards the optimal solution since the gradient of the cosine is nearly zero for small angles. The non-differentiabilty at $\theta = 0$ however might be kept in mind.

$$r() = 1 \tag{5.1}$$

$$r(\theta) = \cos(\frac{\theta}{2}) \tag{5.2}$$

$$r(\theta) = \frac{\pi - abs(\theta)}{\pi} \tag{5.3}$$

That the angle should influence the reward is trivial. With the original setup, the position of the cart is also present in the cumulative reward, via the termination of the simulation when the cart travels beyond the track limits. It can also be incorporated directly into the reward function. One aspect, which is especially important when the agent learns the swingup, is that it should always be more beneficial to keep the pendulum upright, than having the cart at the desired location while being within boundaries. The cart should not crash into the end of the track. Equation (5.4) is an exemplary reward function.

Depending on the state of the cartpole, different reward functions can be used. Nayante[Nay21] added a constant value to the reward when in the upright position, to boost the gradient. Such a function might look like Equation (5.5).

$$r(x, \theta) = \cos(\frac{\theta}{2}) * \frac{x_{max} - abs(x)}{x_{max}} \tag{5.4}$$

$$r(x, \theta) = \cos(\frac{\theta}{2}) * \frac{x_{max} - abs(x)}{x_{max}} + \begin{cases} 10 \text{ for } abs(\theta) < \theta_{thr} \\ 0 \text{ else} \end{cases} \tag{5.5}$$

5.2.3 Meassuring Physical Parameters

In order to bring the simulation of the cartpole closer to the real one, the behaviour of the real pendulum is recorded and suitable models are fitted to the data. The investigated physical quantities are friction, which occur between the pole and the cart, as well as the cart and the track. The force the motor applies to the cart is also examined.

Pole Friction

The main effects that slow the pendulum down are the friction of the bearing that connects the pole with the cart and the air resistance, aka. viscous friction, of the pole itself. Different models exist for simulating these effects. A typical model for these dampening effects on a pendulum is an exponential decay [Mol04]. While at first sight, this seems a fitting model, this dampening model does only fit harmonic oscillators, the general pendulum, however, is not a harmonic oscillator. A harmonic oscillation is present if the restoring force is proportional to the displacement from the neutral position. As seen in Figure 3.2, the force is described as $F = mg \sin\theta$. Thus $F \sim \theta$ is only approximately true for small angles θ measured from the hanging position.

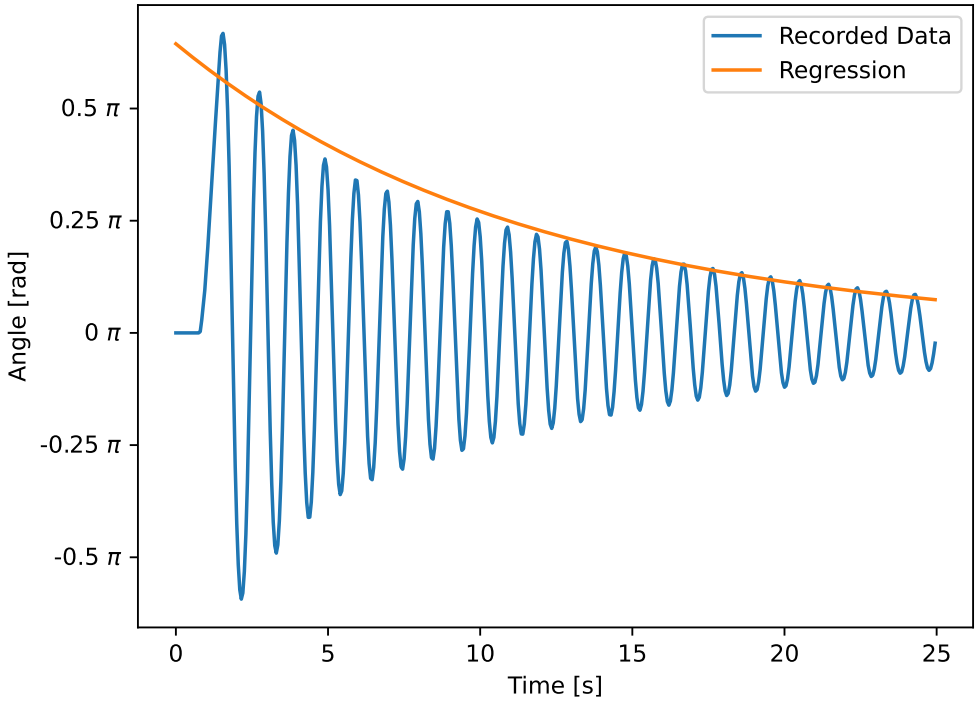


Figure 5.2: Record of the pendulum displacement. Fit of one single exponential function to model the energy loss in the system.

During the swing-up process, air resistance has a much greater effect than in the upright position. This additionally motivates to model bearing friction and air resistance separately. The friction coefficient can be approximated by measuring the decay for very small displacements since the air resistance is then nearly zero due to its quadratic nature [Wik23].

In Chapter 6 we can see, that the exponential decay model proves to be effective. Figure 5.2 shows the exponential decay.

Cart-Track Friction

The main friction happens on the track where the teeth of the gear rack and pinion gears meet, as well as the linear ball bearings that guide the cart.. The gear rack is not lubed, the linear guide just slightly, thus dry friction is present. This is the same as the case for the pendulum described in [Mol04], i.e. a constant friction force, only depending on the direction of the movement and the normal force pushing the pendulum down. [PJK+14] elaborates more on this. In Figure 5.3 on the left the roll-out of the cart can be seen. For this, the unpowered car got pushed by hand and the position was recorded. The derivatives were calculated using central differences. Furthermore, the acceleration curve was scaled down by a factor of 10. While not perfectly true, a roughly constant breaking force can be seen confirming the constant model for dry friction. Figure 5.3 on the right

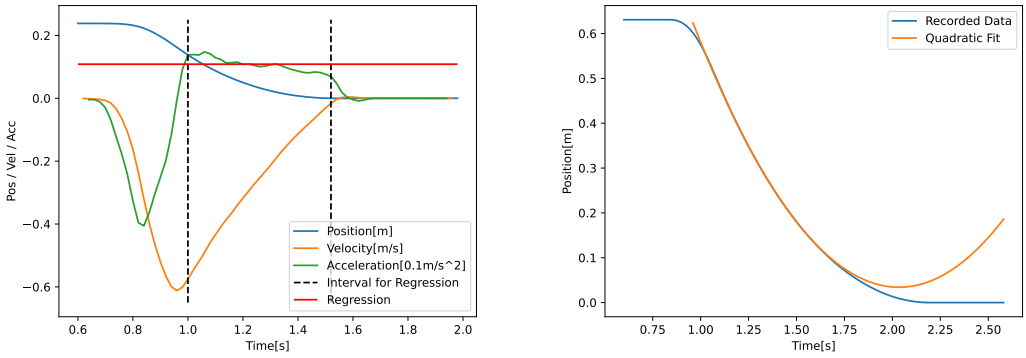


Figure 5.3: Left: Motion of the cart without the pole. The cart was pushed by hand. Blue: Position, Orange: Velocity, Green: Acceleration. A constant can be used to approximate the breaking phase. **Right:** The quadratic fit to the motion.

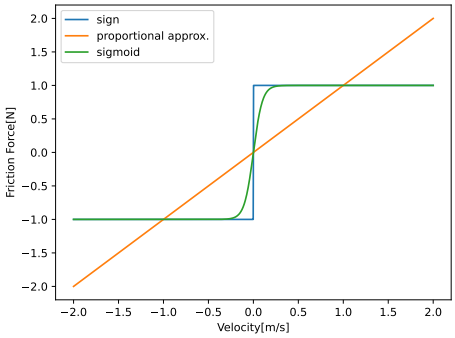
shows a quadratic fit to the cart motion confirming the model further more. Both, the quadratic regression and the constant regression on the acceleration yield a pretty similar result with less than 3% deviation.

Modelling Friction

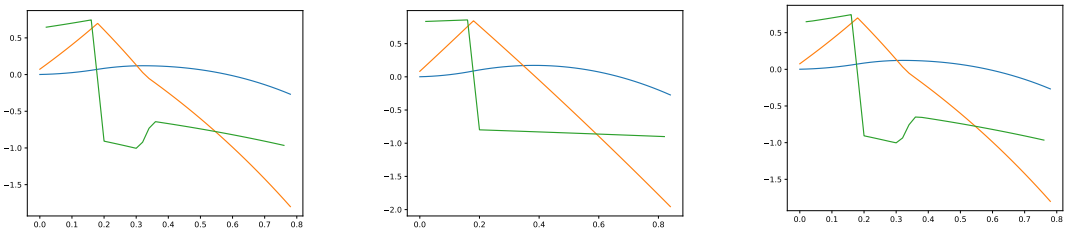
The general equation for modeling dynamic friction is pretty simple as seen in Equation (5.6) [PJK+14]. The coefficient for friction is μ, F_{norm} is the force the two interacting surfaces are pressed together with. Experimentally the product μF_{norm} was determined in the previous sections.

$$F = \mu F_{norm} sign(\dot{x}) \tag{5.6}$$

This formula however can pose a problem when used in differential equations and time integration solvers as it is not continuous for $\dot{x} = 0$. This means, that at the turning point of the cart, where the direction changes, the friction force would be applied in the wrong direction for one timestep. [Gre20] suggests the approximation as seen in Equation (5.7). To better approximate the constant nature of the original equation, a scaled sigmoid function is also tested. It ensures nearly constant friction for high velocities while still being continuous around slow velocities. Using the parameter ρ, Equation (5.8) can be tuned to the system for best performance. Figure 5.4 shows the behaviour of the cart for the different friction functions in a simulation. For this, the cart accelerates for some fixed time to one side, then the direction of acceleration is switched. In the green acceleration curve, one can see the jumps the acceleration makes, when the direction of the velocity changes and thus the direction of the friction force. For the green curve in the middle, this is not the case, since the friction is proportional to the velocity.



(a) Three different friction models for the cart track.



(b) $F = \mu F_{norm} sign(\dot{x})$ **(c)** $F = \mu\dot{x}$ **(d)** $F = \mu F_{norm}\left(\frac{2}{1+e^{-\dot{x}*100}} - 1\right)$

Figure 5.4: The effects of different friction functions for the cart. **Blue:** Position, **Orange:** Velocity, **Green:** Acceleration

$$F = \mu\dot{x} \tag{5.7}$$

$$F = \mu F_{norm}\left(\frac{2}{1 + e^{-\dot{x}*\rho}} - 1\right) \tag{5.8}$$

Force-Curve

Measuring the force curve enables a more precise simulation when using finer gradation of force values. The applied forces were measured by setting a motor control value and measuring the acceleration. Since the cart only measures its position, the derivatives were taken to get the acceleration and with the known mass of the cart, the force can be calculated via $F = ma$.

A problem that arises with this setup: since the cart rests on the pinion gear and the linear guide, the measured force of the cart is influenced by the friction. Using the constant friction force of the cart determined earlier, the force curve can be offset to resemble the actual force more closely.

Figure 5.5 shows the measurements and a possible regression using a quadratic model. Note that the zig-zag of the curve implies, that the cart is accelerating differently depending on the direction.

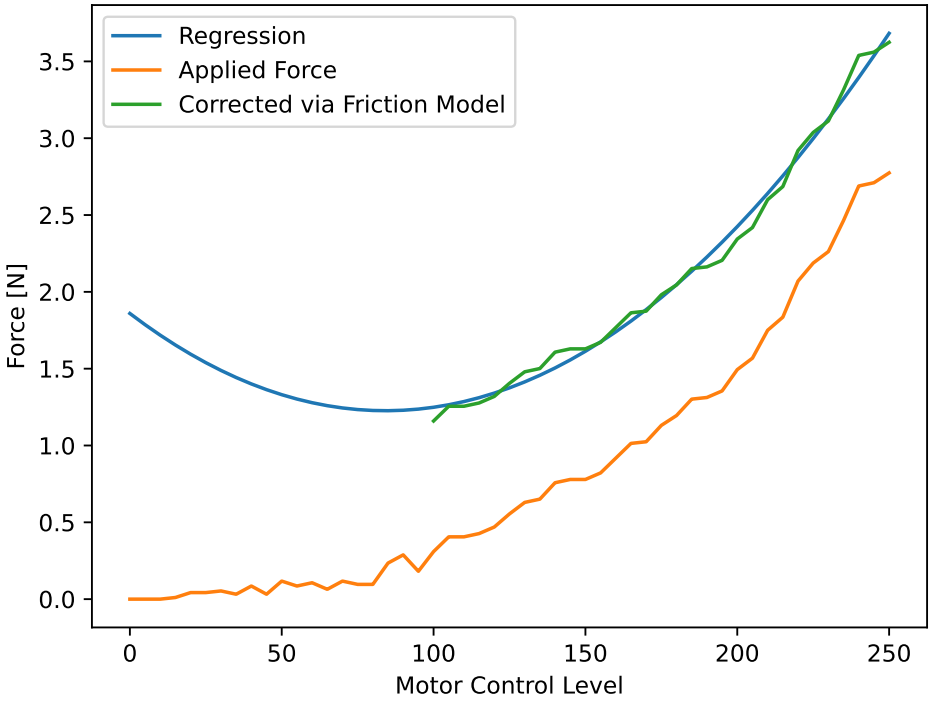


Figure 5.5: The measured forces at the cartpole and a regression using a quadratic model.

As displayed in [PJK+14], there is also the effect of static friction. It is a force required to overcome the stillstand. It is most often greater than the force of dynamic friction. This force is most likely the reason, why for small motor control values the cart doesn't move at all. A simple approach to bypass modeling static friction is not allowing for such small motor control values. The higher the value, the smaller the impact on the cart motion itself.

5.3 Real Cartpole

In order to control the real cartpole, the environment gets replaced with the functions to interact with the real cartpole. There are however some changes to the control flow necessary. While during simulation the loop calls the step-function of the environment as fast as the computer allows for it, the frequency of interaction with the pendulum is dictated by the pendulum itself. The control flow can be seen in Figure 5.6. There are two main differences compared to the setup when training on the simulation (Figure 5.1). The first difference is the blocking call to receive the current state from the pendulum. The agent returns an action based on this state. Following this, the action gets then mapped to a motor control value between -255 and 255 and finally sent to the pendulum. These 3 steps are marked as the green nodes. They are enough to evaluate an agent on the pendulum. The three steps depicted in yellow color are necessary to train the agent. While it is not very feasible

to train the agent on the real pendulum from the ground up, pre-training on the simulation and fine-tuning the agent in the real world with few episodes might enable a better performance on the real cartpole.

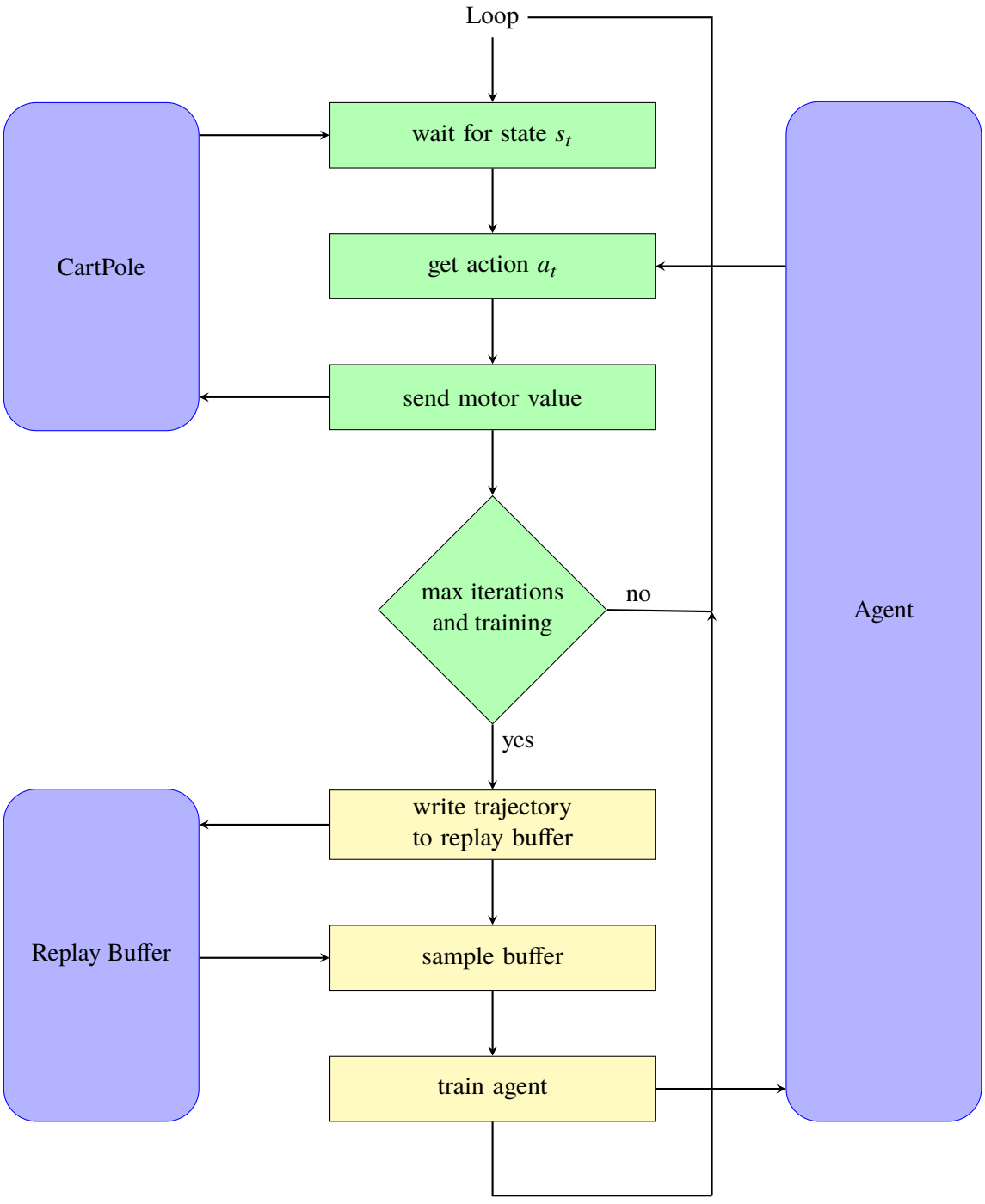


Figure 5.6: Control loop of the real cartpole. **Green:** Main control loop where state and action are evaluated. **Yellow:** Optional training of the agent with data from the real cartpole.

6 Evaluation

With the experimental setup established the performance of the reinforcement learning algorithm can be evaluated. The agents are trained for one of the two tasks, keeping the pole upright and the swingup. Note that, technically, keeping the pole upright is a true subset of the swingup task. The agent is evaluated during the training on the simulated environment. After training, the agent can be tested on the real cartpole. The order, in which the evaluation results are presented roughly reflects the order of the experiments during the thesis.

For the evaluation of an agent, the cumulative reward (so-called return) is used. To make the comparison between agents not only easier but also fair, the evaluation always uses the same reward function, irrespective of the reward definition in training. Especially when experimenting with the reward function the agent is trained with, an easy comparison of the agents can be made by looking at their respective plots.

With the cartpole, there are two primary goals. The first one is keeping the pendulum upright, i.e., the angle θ should be as close to zero as possible. The second goal consists of keeping the cart at a specified position on the track. In this evaluation, the desired position of the cart is always $x = 0$, being at the middle of the track. Including the derivatives of the angle and the cart position in the reward function is not necessary for the evaluation, since higher velocities automatically imply, that the respective positions are not always close to zero. Equation (6.1) shows the function that is used to evaluate the performance. The function returns values on the interval $[0, 1]$. The agent is evaluated over ten-second runs not only in the simulation but also on the real cartpole. With a $50Hz$ simulation respective control-loop rate, this yields 500 steps and thus a maximum score of 500 per run. Practically this is not achievable since the pendulum would need to be perfectly straight and the cart in the middle. For the swingup task, a score of close to 500 is not achievable, since the pole starts in the hanging position and the agent needs several moves to add enough momentum to the pole to bring it to the upright position. It can also be ambiguous, whether the agent even achieved the swingup. One agent might bring the pole very high very soon during the ten seconds but never achieves the swingup. Another agent might take longer to reach the same amplitude as the other agent but succeeds in the swingup task right before the end. Both agents can yield a similar return. Since only the latter one accomplished the swingup task, it can be beneficial to introduce a way to differentiate this. For this, the reward from Equation (6.2) is introduced, inspired by [Nay21]. This increases the theoretical maximum reward to 5000. Scores over 500 indicate, that the pole was in the upright position. Generally speaking, scores above 2000 are good, above 2500 the agent performs very well.

$$r(x, \theta) = \cos\left(\frac{\theta}{2}\right) * \frac{x_{max} - abs(x)}{x_{max}} \tag{6.1}$$

$$r(x, \theta) = \cos\left(\frac{\theta}{2}\right) * \frac{x_{max} - abs(x)}{x_{max}} + \begin{cases} 5 \text{ for } abs(\theta) < 5^\circ \\ 0 \text{ else} \end{cases} + \begin{cases} 5 \text{ for } abs(\theta) < 5^\circ \wedge abs(\dot{\theta}) < 6^\circ \\ 0 \text{ else} \end{cases} \tag{6.2}$$

6.1 Baseline

For a first baseline performance, the slightly adapted gymnasium cartpole environment is used. While the environment only uses the action left and right, a third action is introduced that doesn't exert any force on the pendulum. This mimics the real cart, when no voltage is applied to the motor. The other changes include setting the physical static parameters of the simulated cartpole to match the ones of the real cartpole. Here we adapt the weight of the components, the length of the pole and the track, as well as the force, that is applied to the cart. The original environment simply uses a value of 1 as the reward function and terminates, if the angle of the pole becomes greater than 12°or the cart leaves the track. With this reward function, the swingup task is not possible and thus not evaluated here.

Figure 6.1 shows the reward over the training of 30000 episodes using the parameters of Table 6.1. The deviation of rewards in the simulation is rather large with many very strong dips in performance. With the reward function only being one, as long as the pendulum is operated within the limits of position and angle, the agent is neither rewarded for staying near the center of the track, nor for keeping the angle at zero degrees. This behaviour can also be seen on the real pendulum, depicted in Figure 6.2a. The y-axis values represent the angle of the pole. The position and action are scaled linearly in such a way, that their maximum amplitudes are ± 10 units. This is done to see the meaningful behaviour of all three quantities in one single plot. It is clear, that the agent can achieve the classical cartpole task. Due to the lack of feedback via the reward function, problems can still arise. The angle of the pole oscillates heavily between 2.5°, as well as the actions. While this doesn't pose any problem in the simulation, the real cartpole can suffer from wear. This can not only lead to a change in the physical parameters, mainly friction coefficients, but it can also lead to parts, like the pinion gear, completely breaking down with the need to replace it.

While Figure 6.2a looks already promising, the more general picture can be seen in Figure 6.2b. Both of these agents produced a similar reward in the simulation, yet one completely outperforms the other on the real cartpole. A closer look at these two graphs can give an idea, why such a deviation in real-world performance can happen. Agent a) tries to keep the cart around the center of the track while allowing for some deviation of the pole. Agent b) has its priority more on keeping the pole upright. Probably due to friction in the real system, the agent wasn't able to bring the pole to the other side of the vertical axis. Thus the cart drifts to the left until the agent decides, that it is too close to the axis limit and decides to turn the cart around. The pole falls.

A first step that can eliminate this problem is to feed the agent with less ambiguous rewards.

Table 6.1: Hyperparameters for the baseline evaluation

NN	(64,64)
target update rate	200
replay buffer size	15000
batch size	128
Reward function	1
Actions	-1,0,1

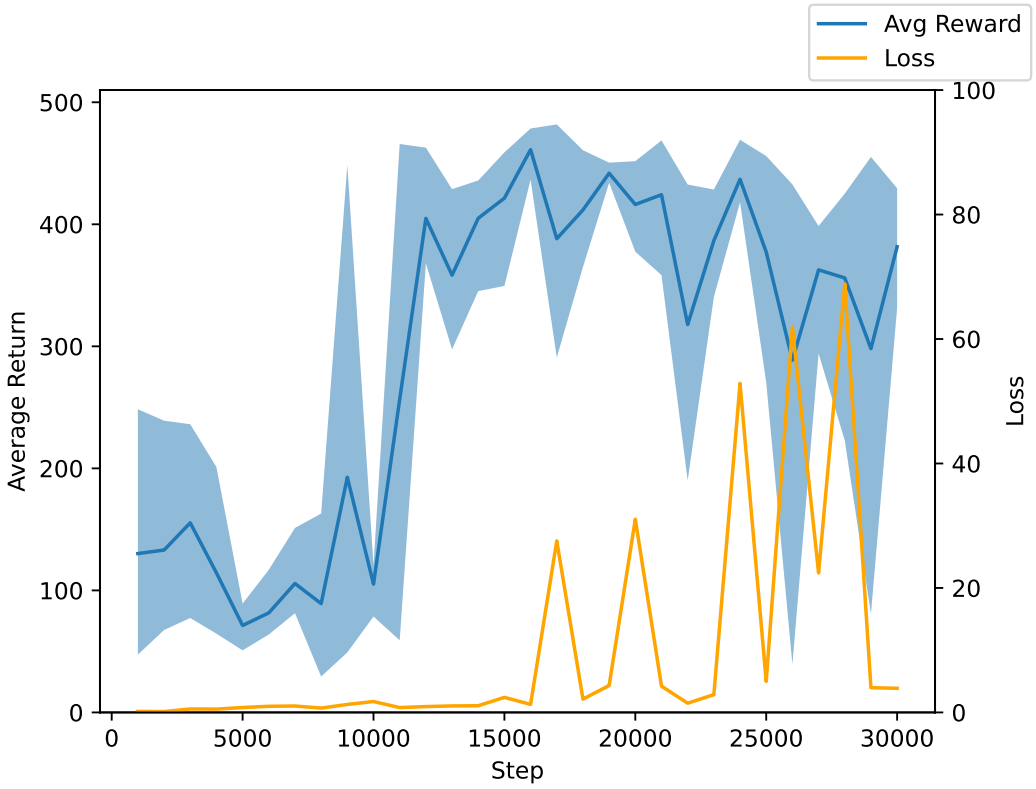


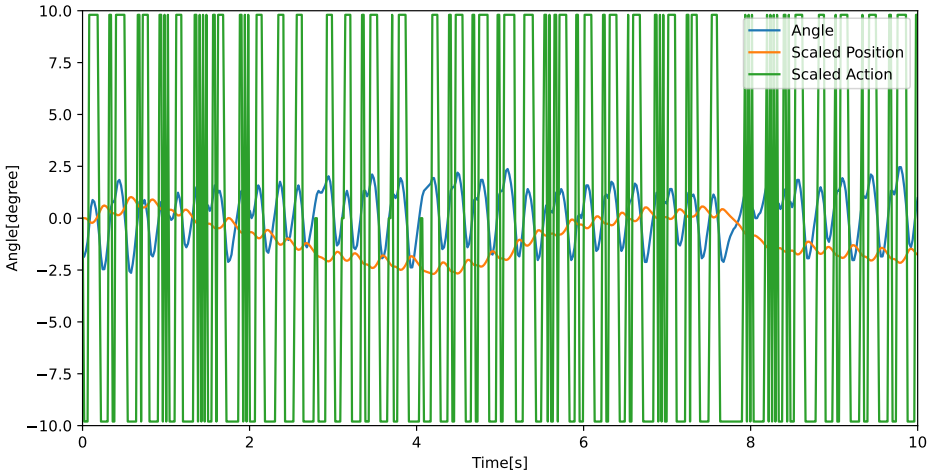
Figure 6.1: Baseline performance of the cartpole.

6.2 Advanced Setup

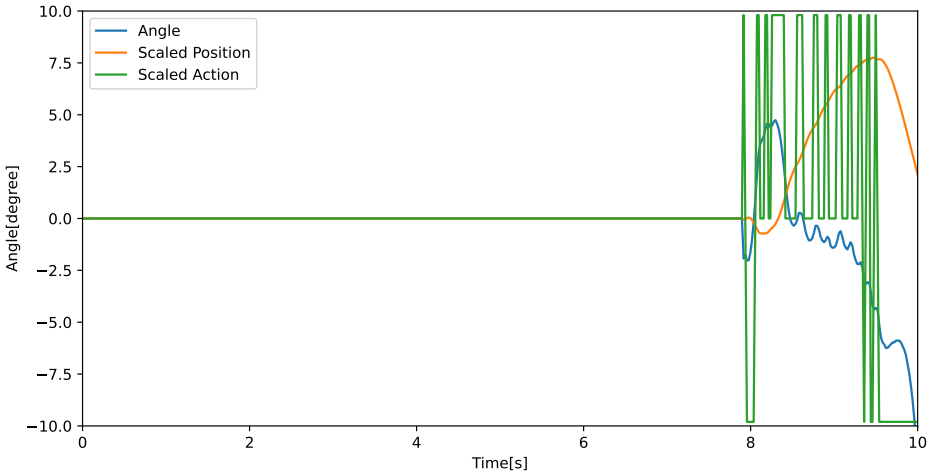
In order to give the agent better feedback about position and angle, the reward function is improved. With the previously mentioned points, the reward function from Equation (6.1) is used. It is to be expected, that the average reward will improve since the agent tries to optimize the exact function, that it is evaluated with. In Figure 6.4a it is apparent, that the first peak above 400 points is much earlier. 6000 episodes compared to nearly 10000. Besides that, the average reward is noticeably higher and the deviation much smaller. The performance on the real cartpole is also notably higher with approximately 450 for some agents. Again, this is not the general case.

An interesting policy can be seen in Figure 6.3a. The agent tries to keep the angular velocity of the pole low. This allows the agent to do very few but precise correction actions.

With this setup, it is theoretically possible to train the agent to achieve the swingup. Using the standard reward from the classical task, the agent was not able to achieve a swingup. Even not in the simulation. Therefore, Equation (6.2) is not only used for the evaluation but also for the training. While the environment is kept nearly identical, the difference is in the reset function. The environment is not reset to the hanging position of the pole, but rather to any random position between -180 and 180 degrees. Only starting from the hanging position, the swingup could not be learned with this setup.



(a) A cherry-picked checkpoint can perform very well. Average reward around 480.

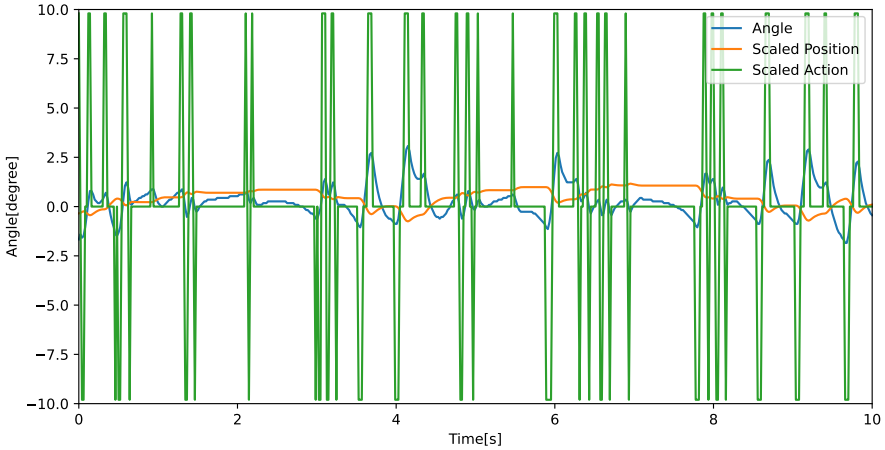


(b) Agent with a similar performance in the simulation. No continuous hold possible.

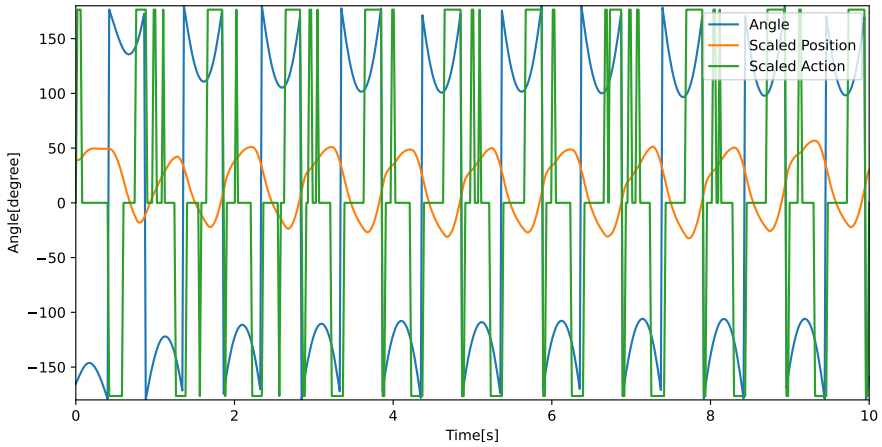
Figure 6.2: Behaviour of the agent on the real cartpole, trained on the baseline environment.

The learning curve of the agent on the environment can be seen in Figure 6.4b. The agent was trained for 300000 episodes with a buffer size of 150000 and a target update period of 5000. The agent is not converging to a policy, however, there are some spikes where one performed very well in the simulation. These well-performing agents however are not able to reproduce this on the real cartpole. Most of them struggle to bring the pole up as seen in Figure 6.3b. Some did achieve the swingup, however not within the time limit of 500 steps, respective ten seconds. Generally speaking, it is very hard to find suitable hyperparameters for the swingup task.

Recall that the agent still has only three actions to choose from. Left, no force, and right. With a finer gradation, one could enable the agent to bring the pendulum closer to the upright position. Additionally, the wear on the mechanical parts can be reduced. In Figure 6.5 the learning curve of



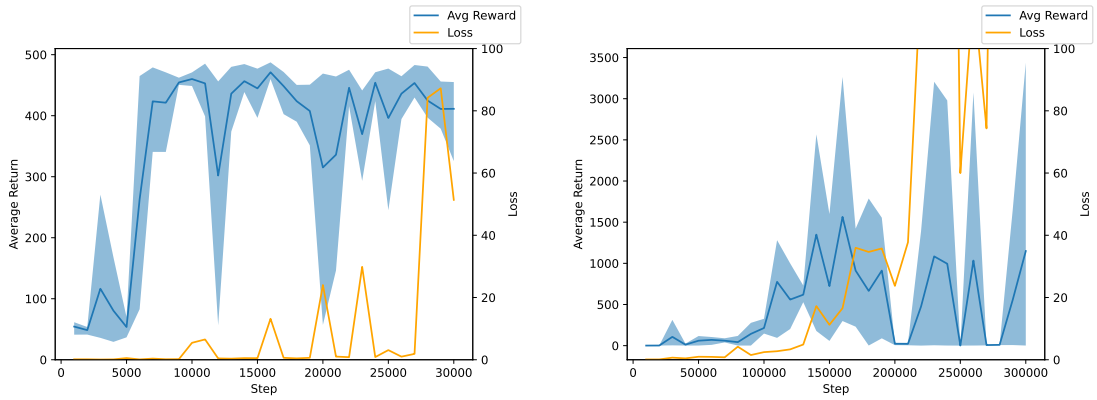
(a) Interesting behaviour of an agent. Minimizes angular velocity around the upright position. Few, but precise actions are enough to keep the pole upright.



(b) Swingup: The agent cannot get enough energy into the system to bring the pole to the upright position.

Figure 6.3: Behaviour of the agent on the real cartpole using the advanced reward functions.

an agent with 9 actions can be seen. The actions represent a force of -4,-3,...,3,4 Newtons in the simulation. The reward in the simulation seems to have fewer negative outliers. The performance on the real cartpole is increased slightly. Note, that there is still a rather large deviation in the performance of agents. In Figure 6.6 can be seen, that the agent not only stays closer around the center of the track, but it also keeps the amplitude of the oscillations of the pole smaller compared to the baseline test in Figure 6.2. One caveat however: the agent learned the actions to corresponding force magnitudes of 0, 1, 2, 3 and 4 newtons, the respective motor control values were 0, 100, 150, 200, 250. The latter is nonlinear. With a linear spacing of 0, 62.5, 125, 187.5, 250 the performance of the same agent drops significantly, some fail to keep the pole upright.



(a) Upright task **(b)** Swingup task

Figure 6.4: Learning curve with the reward from Equation (6.1)

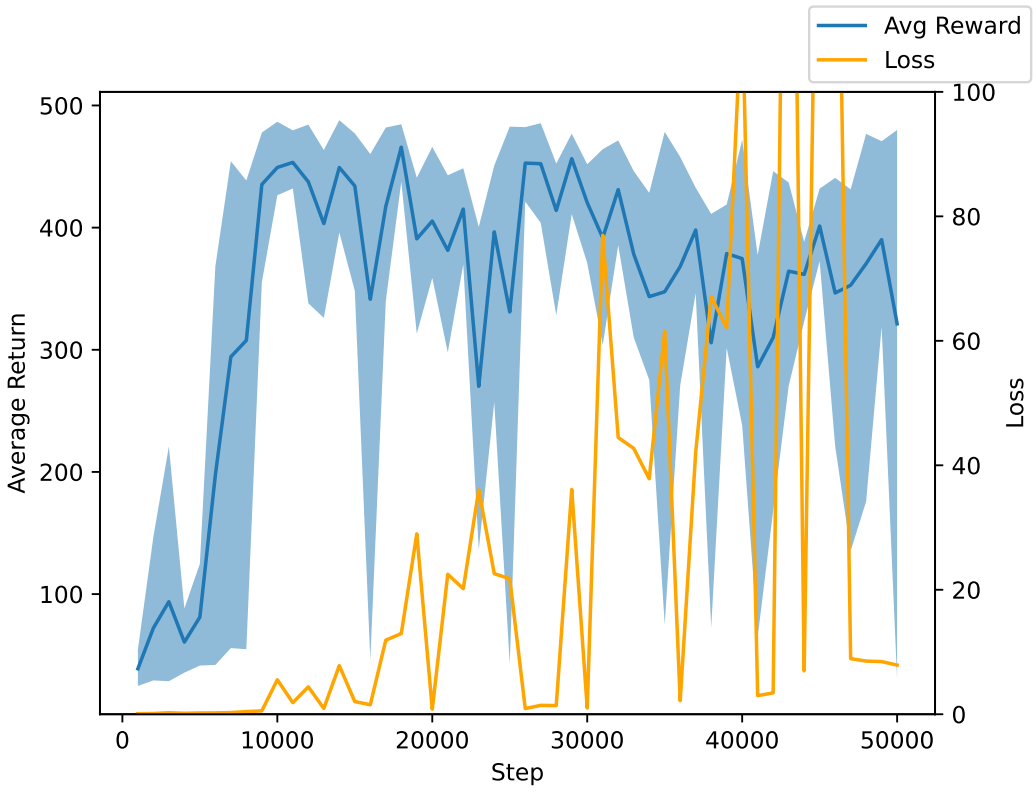


Figure 6.5: Learning curve with a force gradation of 4 steps.

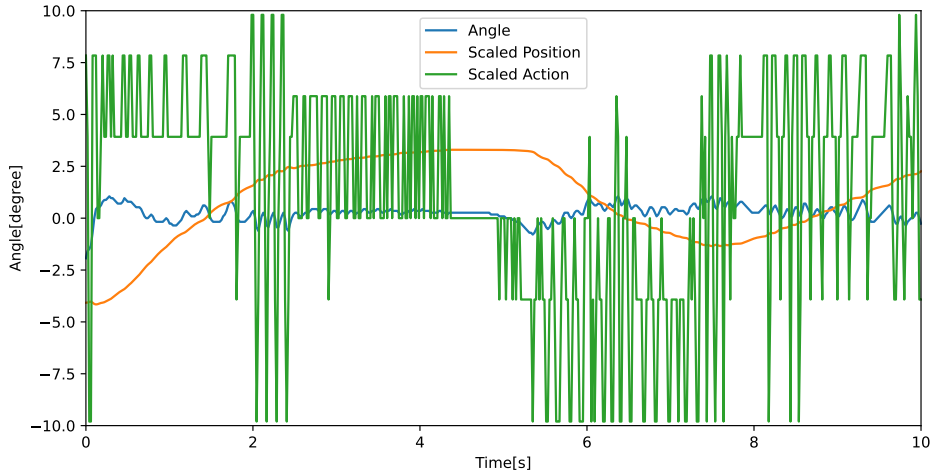


Figure 6.6: Controlling behaviour of the agent on the real cartpole

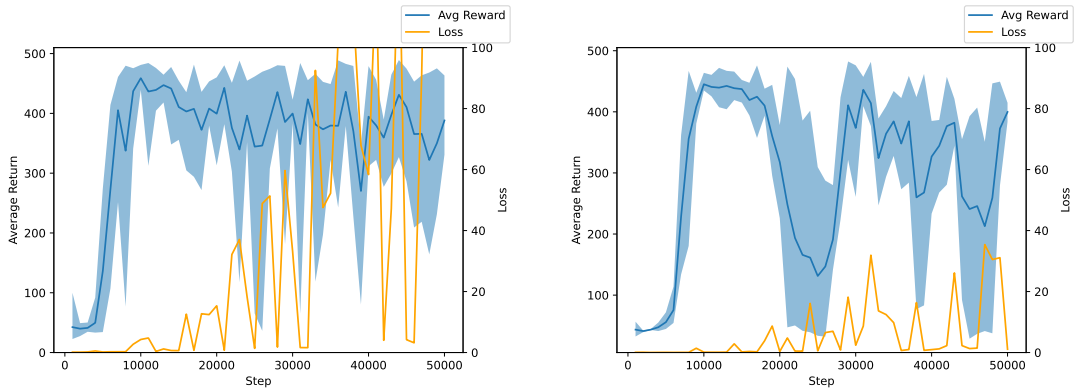
In Chapter 5 the setup of the agent was discussed and the choice of different optimizers. The last experiments were all done using the standard Adam optimizer. In Figure 6.7a the learning process using the AdamW optimizer can be seen. The setup is, besides the different optimizer, the same as the one used in Figure 6.5. While the graph is a bit smoother than with the Adam optimizer, the loss is notably higher. For the following evaluations, the Adam optimizer is therefore used. The rising loss is generally a problem when training an agent on the cartpole. When the agent explores the environment and learns to control the pendulum one could expect the loss to converge, ideally to zero. With the AdamW optimizer, this is more apparent than with the original Adam optimizer. A way to significantly reduce the growth of the loss over time is to choose higher values for the target update period parameter. This comes with the drawback, that the agent learns generally slower, since the neural network, that is used to predict the actions, is updated less frequently.

Figure 6.7b shows the learning curve, where in the reward function the cosine term got replaced with $\frac{\pi - abs(\theta)}{\pi}$. The performance is dropping perceptible compared to the original cosine term.

6.3 Closing the Reality Gap

In the experiment with multiple force levels, the match from force to motor control value was just eyeballed. Chapter 5 elaborated, on how a force curve, which matches motor control values with the actual applied force, can be measured. Due to the way it is measured, it is inherently influenced by the friction force of the cart. It was also discussed how one can implement this force in the simulation.

The next experiment therefore uses the motor control values of 0,100,150,200 and 250. Inside the simulation, these values are converted to actual force values using the quadratic regression from Figure 5.5. The friction of the cart is modeled using the sigmoid approximation with the parameter $\rho = 100$, the linear approximation as well as the “real“ step-function.



(a) Performance of the AdamW optimizer

(b) Training with a linear theta component as in Equation (5.3)

Figure 6.7

The learning curve doesn't show any significant differences compared to the previous ones. The interest now lies more on the performance discrepancies between the simulation and the real world. For this, random agents are picked with varying performance in the simulation and tested on the real cartpole. The ratio between the cumulative reward of the real cartpole and the simulation is documented. This means, values greater than one represent an agent, that performed better on the real cartpole. Ten agents are picked and evaluated as described. Figure 6.8 shows the performance of the eyeballed force gradation with no friction, the force curve together with the three different friction models for the cart as well as the sigmoid and step model for the cart together with the exponential model for the pole and the force curve.

While the linear friction model performs very badly, the real step function performs better than the sigmoid function. The simulation is further improved by adding the exponential model to simulate the resistance of the pole.

Swingup

With the simulation resembling the real cartpole much better, the swingup task is now considered again. The training parameters, that achieved the swingup in the stock simulation did not prove to be working with the more realistic simulation. The training parameters used to achieve the performance from Figure 6.9 can be seen in Table 6.2. The optimizer used is the AdamW optimizer.

The agent's behaviour is now much better compared to the stock environment. The upright position is reached before the 250-step mark. The behaviour is depicted in Figure 6.10a. The agent also knows precisely for which angles the pole still can be corrected to bring it to the upright position or whether the better action would be to bring the pole to the fall and move it up from the other side. It can bring the pole to the upright position after being tipped over by just one single fall (Figure 6.10b).

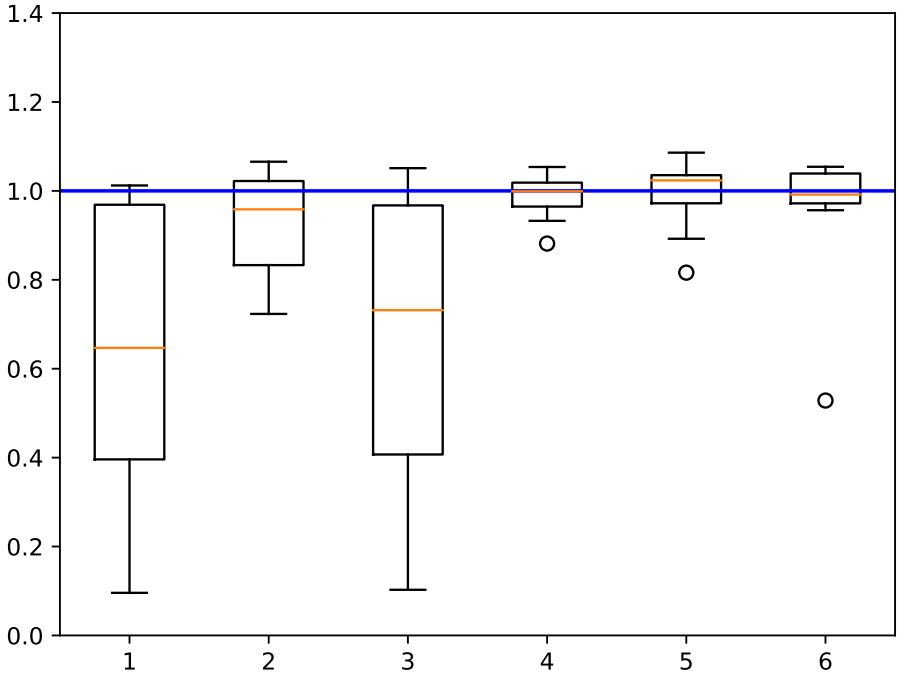


Figure 6.8: Classical task: Comparison of the deviation between simulation and real-world for different setups. The more realistic force curve together with the friction model show a general closer performance. **1:** Eyeballed force curve, no friction **2-4:** Measured force curve, sigmoid/linear/step friction model for cart **5,6:** force curve, sigmoid/step friction for cart, exponential decay for pole friction

Table 6.2: Hyperparameters for the swingup task

NN	(64,64)
learning rate	0.001
gamma	0.99
epsilon	0.1
target update period	2000
buffer size	200000
batch size	256

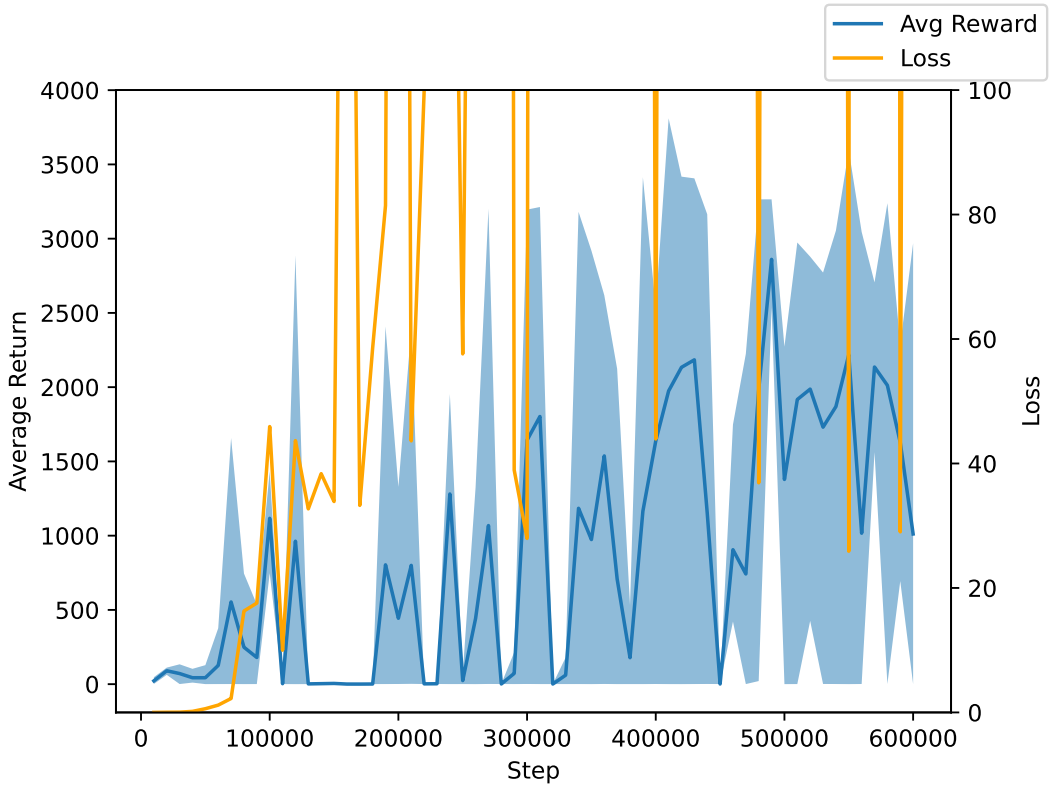


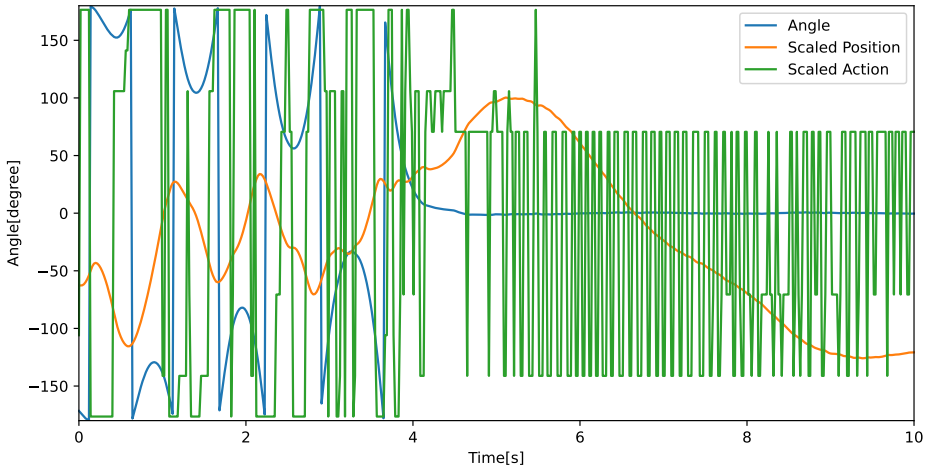
Figure 6.9: Learning curve of the swingup task with the realistic simulation.

Table 6.3: Pole Parameters

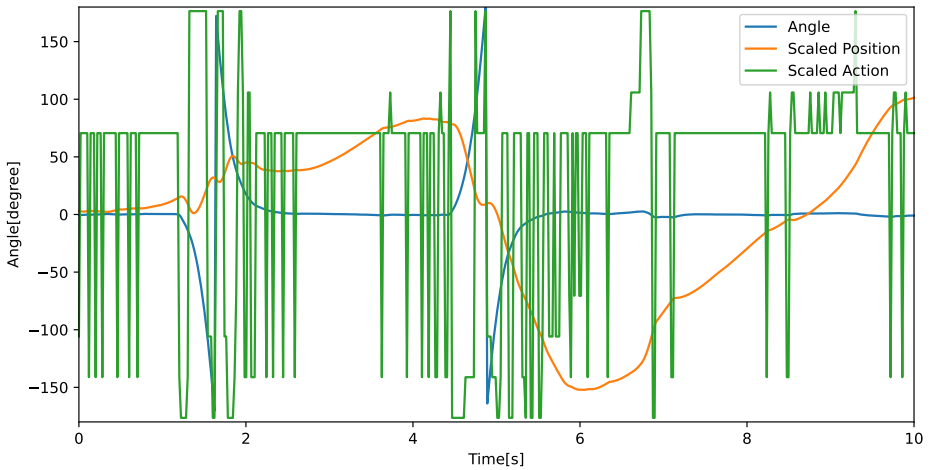
	Medium Pole	Large Pole
Weight	0.127kg	0.23kg
Length	0.355m	0.66m

6.4 Altering physical parameters

All experiments until now were based on the cartpole using the medium pole provided by Quanser. There is also a large pole provided with the parameters depicted in Table 6.3. The training on the large pole can be seen in Figure 6.11. The agent does not achieve nearly as much cumulative reward as an agent trained for the medium pole. One reason might be, that due to the larger pole, more energy needs to be introduced into the system. The motor is not changed between the setups, thus the swingup takes longer. But the swingup on the real cartpole is still possible, as seen in Figure 6.12.



(a) Behaviour of the agent without any disturbance. The agent achieves a fast swingup.



(b) Behaviour of the agent when the pole is tipped over by hand. Only one swing is necessary to bring the pole upright again.

Figure 6.10: Swingup: Behaviour of the agent trained on the simulation with friction and the force curve.

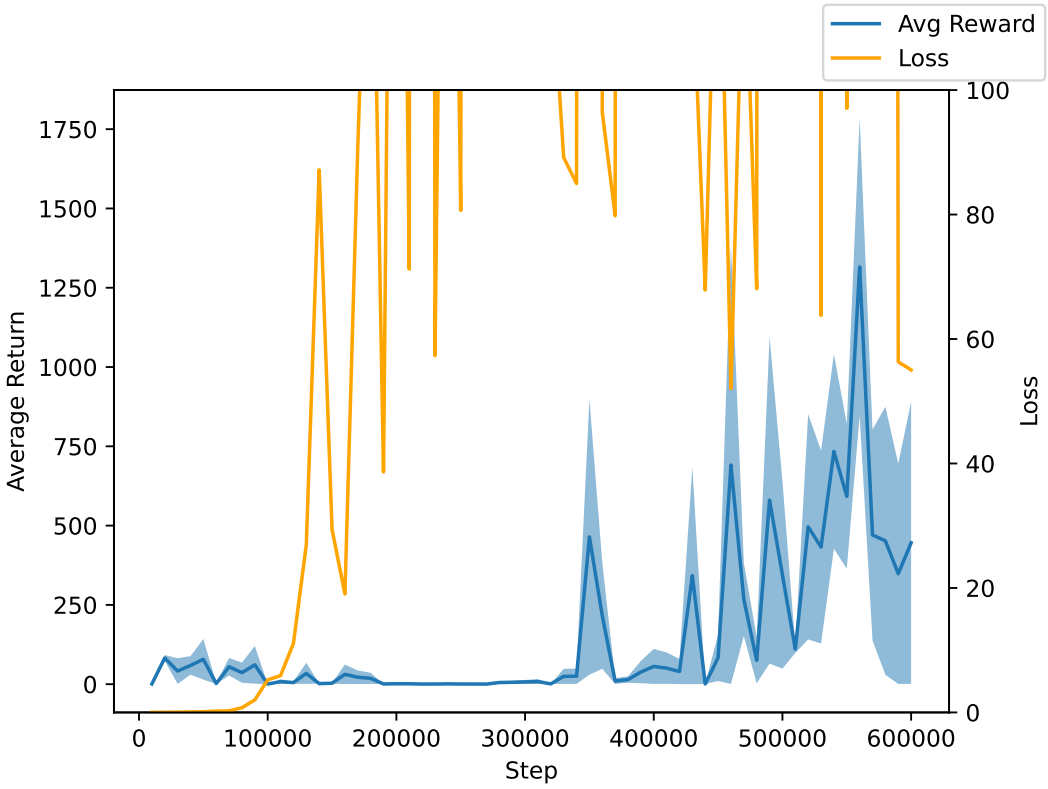


Figure 6.11: Learning curve of the swingup using the large pole.

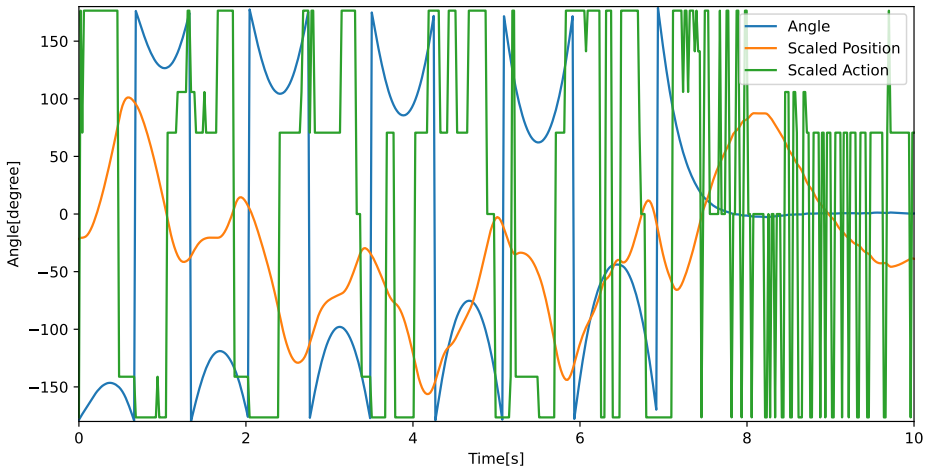
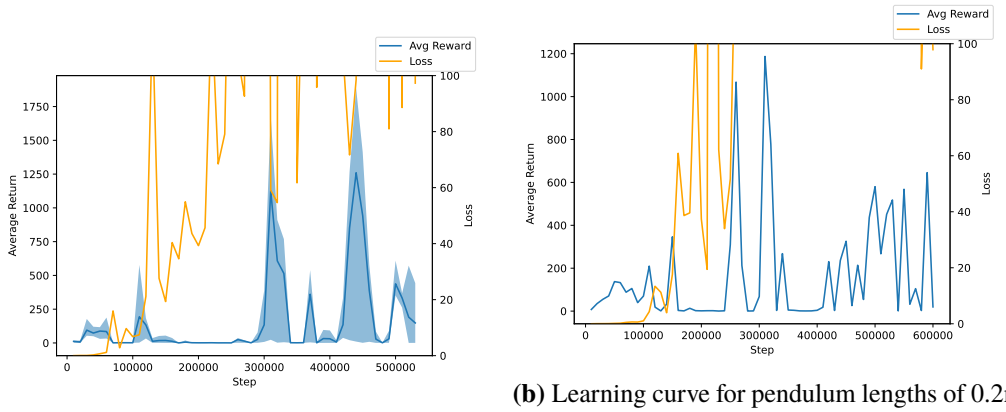


Figure 6.12: Behaviour of the agent trained on the large pole.



(a) Learning curve when training on both, the medium and the large pole.

(b) Learning curve for pendulum lengths of 0.2m to 1m. The weight is calculated via the density of the poles.

Figure 6.13

Now one can, regarding the robustness, raise the question if a single agent be trained to achieve the swingup for both, the medium and the large pole. For this, the reset function chooses a random binary value. A value of zero will start the simulation with the medium pole, and a value of one means the large pole is chosen. The resulting learning curve can be seen in Figure 6.13a. The agent has a much harder time achieving consistently the swingup. With cherry-picking the right agent, it is possible to find one, that is able to succeed in the swingup task with both poles. This is however not the general case.

Until now, the state of the cartpole was always described by $(x, \dot{x}, \theta, \dot{\theta})$. In some real-world scenarios, e.g., when a robot picks something up, not only does its positions and velocities change, but also, in the case of this example, its weight. To investigate this, the state is supplemented with the length of the pole. More precisely, during the reset of the environment, a random length of the pole is chosen on the interval $[0.2m, 1m)$. The poles provided by Quanser roughly have a weight-to-length ratio of $0.7\frac{kg}{m}$. The length of the pole is continuously provided as a 5th quantity in the state that the agent sees. For the evaluation on the real cartpole, the length of the pole that is used during evaluation is also added to the original state. The performance in the simulation is very bad with some spikes, as seen in Figure 6.13b. This bad performance is also the reason, why this experiment was not carried out multiple times. While the agent was able to bring the medium pole on the real cartpole to the upright position, the large pole barely saw angles below 90°.

Dulac-Arnold et al.[DLM+21] propose in their work, that perturbating parameters can lead to better performing agents in real-world scenarios. Their observation was, that with a wider range of perturbation, the performance of agents drop. Their tests, however, were only focused on the pole length of the cartpole environment. Also, no real-world tests were carried out. The length of the pole is certainly a very important quantity of the cartpole system and challenging to train as seen before. Training on different pole lengths aims at enabling an agent to generalize better in order to balance different poles. Conceptually there is a second reason why perturbating parameters can prove to be useful. When training an agent for a specific setup, the pole's length and weight can be measured very easily with high accuracy. Parameters like friction are much harder to measure. To counteract the problem of uncertainty in certain quantities, perturbating these parameters can prove useful. In the case of the cartpole, these quantities are mainly the two friction coefficients

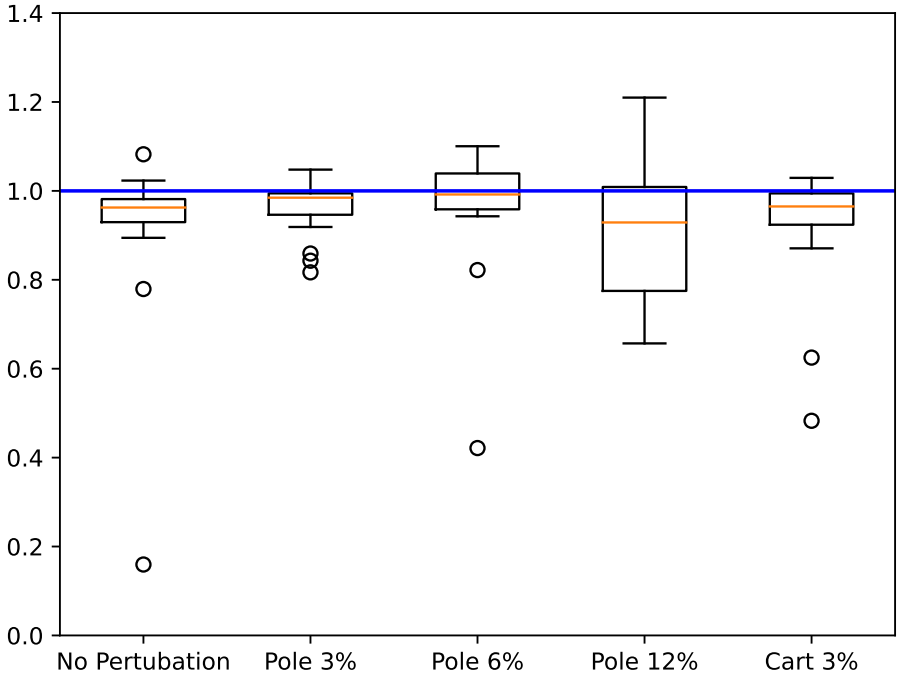


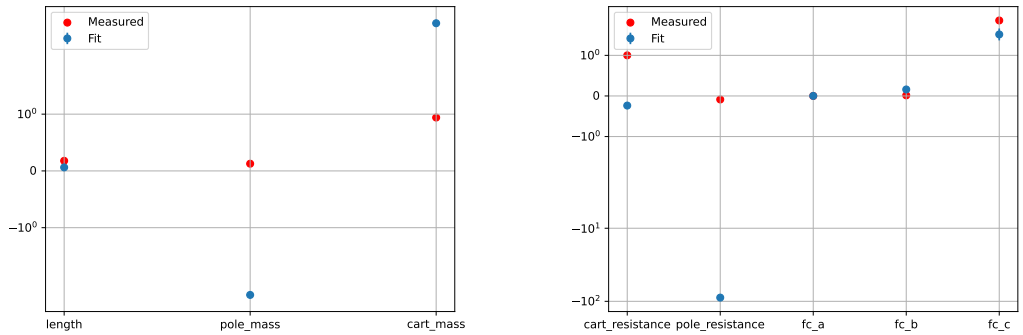
Figure 6.14: Random Perturbations of the coefficients of the friction models.

of the cart and the pole. Figure 6.14 shows the performance comparison between the simulation and the real world. The perturbations were uniformly distributed with the mean value being the measured one. The perturbations of the pole coefficients prove to be helpful. For a perturbation of six percent, the average performance ratio is one, which is the ideal case. A perturbation of 12% is too large. The deviation is rather large, in both directions. Perturbating the coefficient of the cart friction with three percent does not show any significant difference from the control measurement. This can mean, that the friction coefficient is sufficiently accurately determined.

6.5 Additional Experiments

6.5.1 Parameter Estimation

In Chapter 5 the methodology was presented, how the parameters were measured. This was only possible by measuring the dynamic parameters more or less isolated. In the case of the cartpole these are the cart friction, pole friction and the force curve. Static parameters are provided by the manufacturer Quanser. As presented, this yielded good results. The need for measuring isolated parameters raises the question, whether it is possible to extract parameters from state transitions of the cartpole. More precisely, tuples of (s_t, a_t, s_{t+1}) are recorded, two consecutive states and the action that lead to the latter state.



(a) Estimating weight of the cart, pole and the pole's length.

(b) Estimating the friction coefficients + the coefficients of the force curve.

Figure 6.15: Attempt to estimate parameters from recorded behaviour of the real cartpole.

The Python module scipy[VGO+20] has a function called `curve_fit` that allows for parameter estimation using arbitrary functions. For this experiment, the used function is a slightly modified step function from the cartpole environment. The goal is to predict the parameters in such a way, that the step function closely predicts the recorded behaviour of the real cartpole. In Figure 6.15 the results of this experiment can be seen. In Figure 6.15a friction and force curve parameters are fixed. The goal is to estimate the weight of the cart, the weight of the pole as well as its length. This was not successful. Neither was the second experiment with the switched parameters as seen in Figure 6.15b. Some estimations are completely off. For example, the prediction of the mass of the pole or the cart friction is negative. The pole resistance coefficient, which is negative due to the exponential decay model, is off by 3 to 4 orders of magnitude.

7 Conclusion and Outlook

The evaluation showed, that it is possible to train agents solely in the simulation, without sacrificing much real-world performance. While the classical cartpole task is very undemanding with regard to the realism of the simulation, introducing a gradation of the applied force already needs knowledge about the real cartpole to achieve similar levels of performance. Additionally introducing friction can greatly improve the performance.

For the swingup task, friction is much more influential. The cart does not only need to move with much greater amplitude to achieve the swingup, air resistance also comes into play as the velocity of the pole and cart are much higher during the swingup process, than at the upright position.

Finding suitable hyperparameters that enable a convergence of the agent to a good-performing policy can be very challenging. While the classical task is much easier to learn than the swingup, agents still suffer from a rising loss for both tasks. Parameters that typically solve this problem for the DQN algorithm, like the target update period, also had the side effect that the learning process was much slower, sometimes without any progress. All in all, it seems, that the DQN algorithm is very sensitive to its hyperparameters.

Changing the parameters of the cartpole, namely the size and weight of the pole, can enable agents to be more robust. The experiments showed, that an agent can be trained on at least two poles by randomly choosing one at the start of the simulation. Again this is very easy for the classical upright task and much harder for the swingup task with only very few reasonable performing agents. Increasing abstraction by training an agent for pole densities and feeding the pole length as an additional parameter into the neural network showed promising results, being often very close to the upright position. Again, the classical task is easily controlled with this.

Outlook

All agents in this thesis were trained on a simulation based on differential equations. While this can offer advantages like easily changing parameters when switching to a different cartpole setup, disadvantages were presented in the form of finding suitable models and determining parameters in order to close the reality gap. The data-driven approach using machine learning would be to generate data from the real cartpole and use this data to approximate the ground truth simulation. Estimating parameters did not work at all. Therefore, human experts are still necessary to find models for, e.g., friction. Ideally, one could learn some kind of neural network that can predict the next state simply from the current state of the cartpole and the action. An agent trained in such a simulation is expected to perform very well.

Besides that, there is still much room for improvement in the simulation. Effects like static friction are ignored. The dampening of the pole is approximated by an exponential decay. The underlying friction forces are ignored. The force, the pole exerts on the cart is only considered in the horizontal

direction. While technically being fixed in the vertical direction, the pole still exerts a force on the cart that influences its friction force. The graphs from Section 5.2.3 still show some deviation between measurement and model.

Having the cartpole problem intensively studied now, interesting questions regarding control theory arise for different setups as well. Typical tasks in our everyday life are the stabilization of drones, whether when recording images and videos or flying small commercially available drones ourselves. Drones are generally controlled very well using classical control theory without the need for machine learning techniques. The need for them in self-driving cars, together with the aforementioned problems and complexity, needs to be intensively studied in the future.

Bibliography

[22] *Announcing The Farama Foundation*. Oct. 2022. URL: https://farama.org/Announcing-The-Farama-Foundation (cit. on p. 27).

[CMG+18] P. S. Castro, S. Moitra, C. Gelada, S. Kumar, M. G. Bellemare. "Dopamine: A Research Framework for Deep Reinforcement Learning". In: (2018). URL: http://arxiv.org/abs/1812.06110 (cit. on p. 31).

[DHK+17] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, P. Zhokhov. *OpenAI Baselines*. https://github.com/openai/baselines. 2017 (cit. on p. 31).

[DLM+21] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, T. Hester. "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis". In: *Machine Learning* 110.9 (Apr. 2021), pp. 2419–2468. ISSN: 1573-0565. DOI: https://doi.org/10.1007/s10994-021-05961-4 (cit. on pp. 14, 53).

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16 (cit. on p. 11).

[Gre20] C. D. Green. *Equations of Motion for the Cart and Pole Control Task*. 2020. URL: https://sharpneat.sourceforge.io/research/cart-pole/cart-pole-equations.html (cit. on pp. 18, 35).

[IBM] IBM. *What is machine learning*. URL: https://www.ibm.com/topics/machine-learning (cit. on p. 11).

[KB14] D. P. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization". In: (2014). DOI: https://doi.org/10.48550/arXiv.1412.6980 (cit. on p. 31).

[LH17] I. Loshchilov, F. Hutter. "Decoupled Weight Decay Regularization". In: (2017). DOI: https://doi.org/10.48550/arXiv.1711.05101 (cit. on p. 31).

[LS00] T.-H. S. Li, M.-Y. Shieh. "Switching-type fuzzy sliding mode control of a cart–pole system". In: *Mechatronics* 10.1–2 (Feb. 2000), pp. 91–109. ISSN: 0957-4158. DOI: https://doi.org/10.1016/S0957-4158(99)00053-7 (cit. on p. 13).

[MKS+13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. "Playing Atari with Deep Reinforcement Learning". In: (2013). DOI: https://doi.org/10.48550/arXiv.1312.5602 (cit. on pp. 14, 24).

[Mol04] M. I. Molina. "Exponential versus linear amplitude decay in damped oscillators". In: (2004). DOI: https://doi.org/10.48550/arXiv.physics/0407080 (cit. on pp. 33, 34).

[Nay21] Y. Nayante. "Reinforcement Learning am realen undsimulierten Cart-Pole-Swing-Up Pendulumim Vergleich". MA thesis. TH Köln, 2021 (cit. on pp. 14, 33, 41).

[NPUG17] S. Nagendra, N. Podila, R. Ugarakhod, K. George. “Comparison of reinforcement learning algorithms applied to the cart-pole problem”. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2017, pp. 26–32. DOI: 10.1109/ICACCI.2017.8125811 (cit. on p. 13).

[PJK+14] D. K. Péter, H. János, D. S. Krisztián, B. Attila, D. T. Péter. *Chapter 8. Models of Friction*. 2014. URL: https://www.mogi.bme.hu/TAMOP/robot_applications/ch07.html (cit. on pp. 34, 35, 37).

[PTG14] L. B. Prasad, B. Tyagi, H. O. Gupta. “Optimal Control of Nonlinear Inverted Pendulum System Using PID Controller and LQR: Performance Analysis Without and With Disturbance Input”. In: *International Journal of Automation and Computing* 11.6 (Dec. 2014), pp. 661–670. ISSN: 1751-8520. DOI: https://doi.org/10.1007/s11633-014-0818-1 (cit. on p. 13).

[Quaa] QuanserCIP. *Quanser Inverted Pendulum*. URL: https://www.quanser.com/products/linear-servo-base-unit-inverted-pendulum/ (cit. on p. 16).

[Quab] QuanserRIP. *Rotary Inverted Pendulum by Quanser*. URL: https://www.quanser.com/products/rotary-inverted-pendulum/ (cit. on p. 16).

[Sta] Statista. *Machine Learning - Worldwide*. URL: https://www.statista.com/outlook/tmo/artificial-intelligence/machine-learning/worldwide (cit. on p. 11).

[Tes] Tesla. *AI and Robotics*. URL: https://www.tesla.com/de_de/AI (cit. on p. 11).

[TTK+23] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, O. G. Younis. *Gymnasium*. Mar. 2023. DOI: 10.5281/zenodo.8127026. URL: https://zenodo.org/record/8127025 (visited on 07/08/2023) (cit. on p. 31).

[VGO+20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 55).

[WD92] C. J. C. H. Watkins, P. Dayan. “Q-learning”. In: *Machine Learning* 8.3-4 (May 1992), pp. 279–292. DOI: https://doi.org/10.1007/BF00992698 (cit. on p. 24).

[Wik23] Wikipedia contributors. *Drag (physics) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Drag_(physics)&oldid=1180851419. [Online; accessed 24-October-2023]. 2023 (cit. on p. 34).

All links were last followed on November 28, 2023.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature