Machine Learning Applications

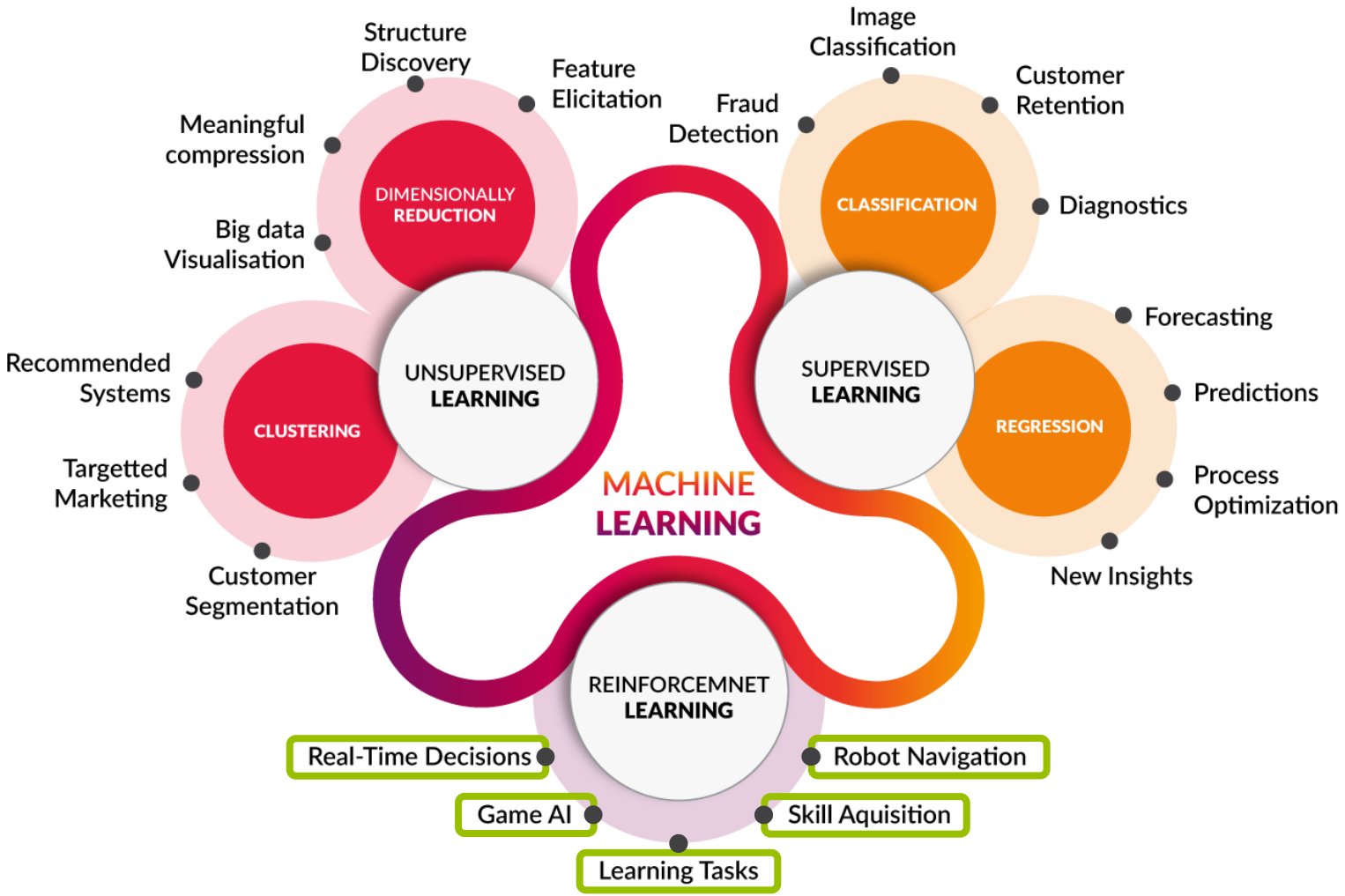
Optimized Control of Cross-Linked Energy Systems by Means of Reinforcement Learning

Prof. Dr.-Ing. M. Weigold, Heiko Ranzau, M.Sc.

Darmstadt, 14.02.2020

Agenda

Reinforcement Learning in the Machine Learning Ecosystem



Objectives

After today you should know:

Potentials and applications of Deep Reinforcement Learning

Basics of **Reinforcement Learning** (RL) and **Deep Reinforcement Learning** (DRL)

Influencing factors on the **performance** of DRL Algorithms

Tools that enable **interaction** between **current DRL algorithms** and **powerful simulations**

Basics of industrial **supply systems**

An **application case** at the **ETA-Factory** (control of supply systems)

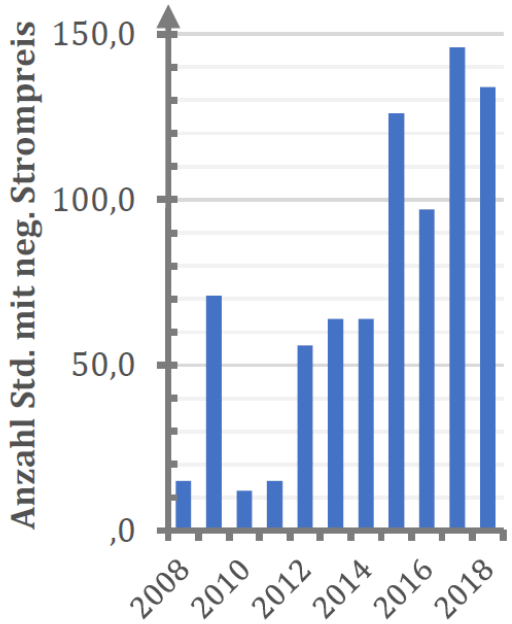
Agenda

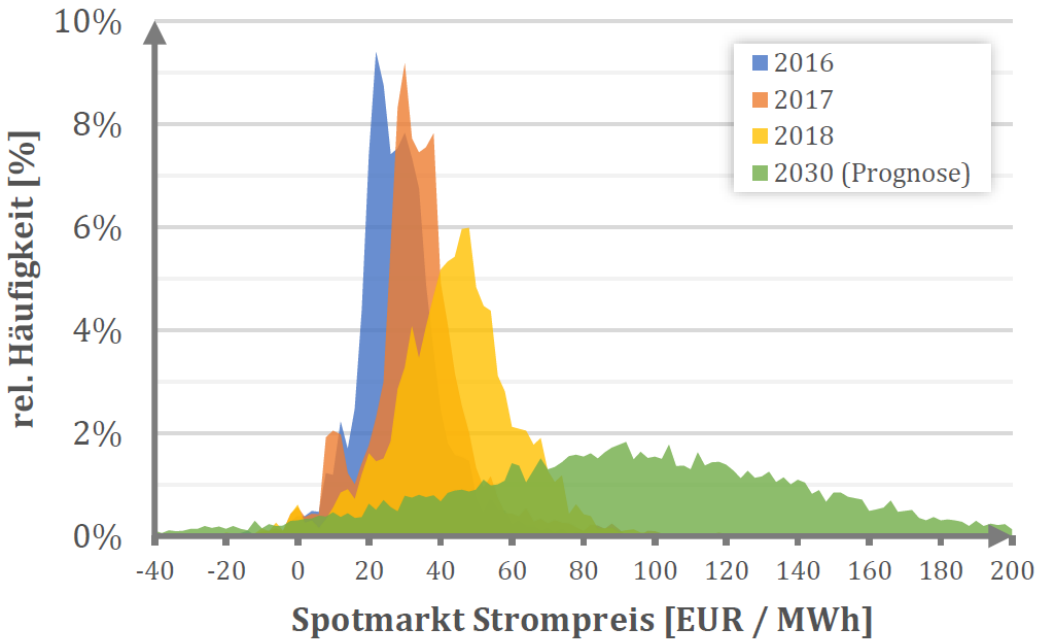
Motivation for using DRL

Why automate highly complex control?

Current situation

1. Electricity prices get **more volatile** with the **increase of renewable energy** production methods





Number of hours with **negative prices** on the day-ahead electricity market (left) and **distribution** in the histogram of **spot market electricity prices** (right)
Source: Panten, N., 2019, Deep Reinforcement Learning zur Betriebsoptimierung hybrider industrieller Energienetze

Motivation for using DRL

Why automate highly complex control?

Current situation

1. Electricity prices get **more volatile** with the **increase of renewable energy** production methods
2. In order to adjust to this fact, industrial systems need to become more energy flexible (**demand side management**)

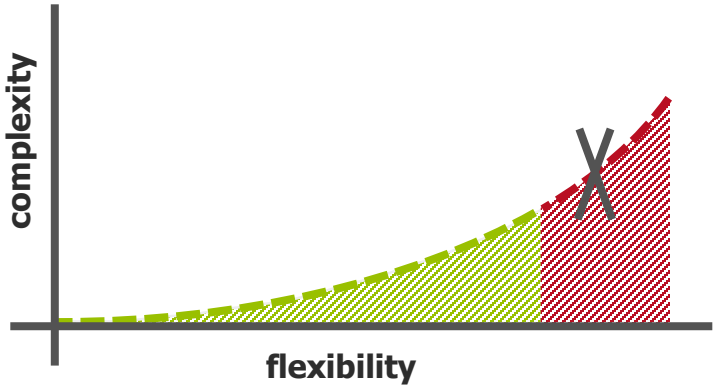
KOPERNIKUS PROJEKTE
SynErgie
Die Zukunft unserer Energie

Motivation for using DRL

Why automate highly complex control?

Current situation

1. Electricity prices get **more volatile** with the **increase of renewable energy** production methods
2. In order to adjust to this fact, industrial systems need to become more energy flexible (**demand side management**)
3. With **higher flexibility** most often comes **higher complexity** that is harder to control **efficiently**



Possible to control **efficiently** with conventianal control methods

Very hard to control **efficiently** with conventianal control methods

Motivation for using DRL

Why automate highly complex control?

Current situation

1. Electricity prices get **more volatile** with the **increase of renewable energy** production methods
2. In order to adjust to this fact, industrial systems need to become more energy flexible (**demand side management**)
3. With **higher flexibility** most often comes **higher complexity** that is harder to control **efficiently**

Can DRL solve this problem?

Motivation for using DRL

What **can** DRL already do?

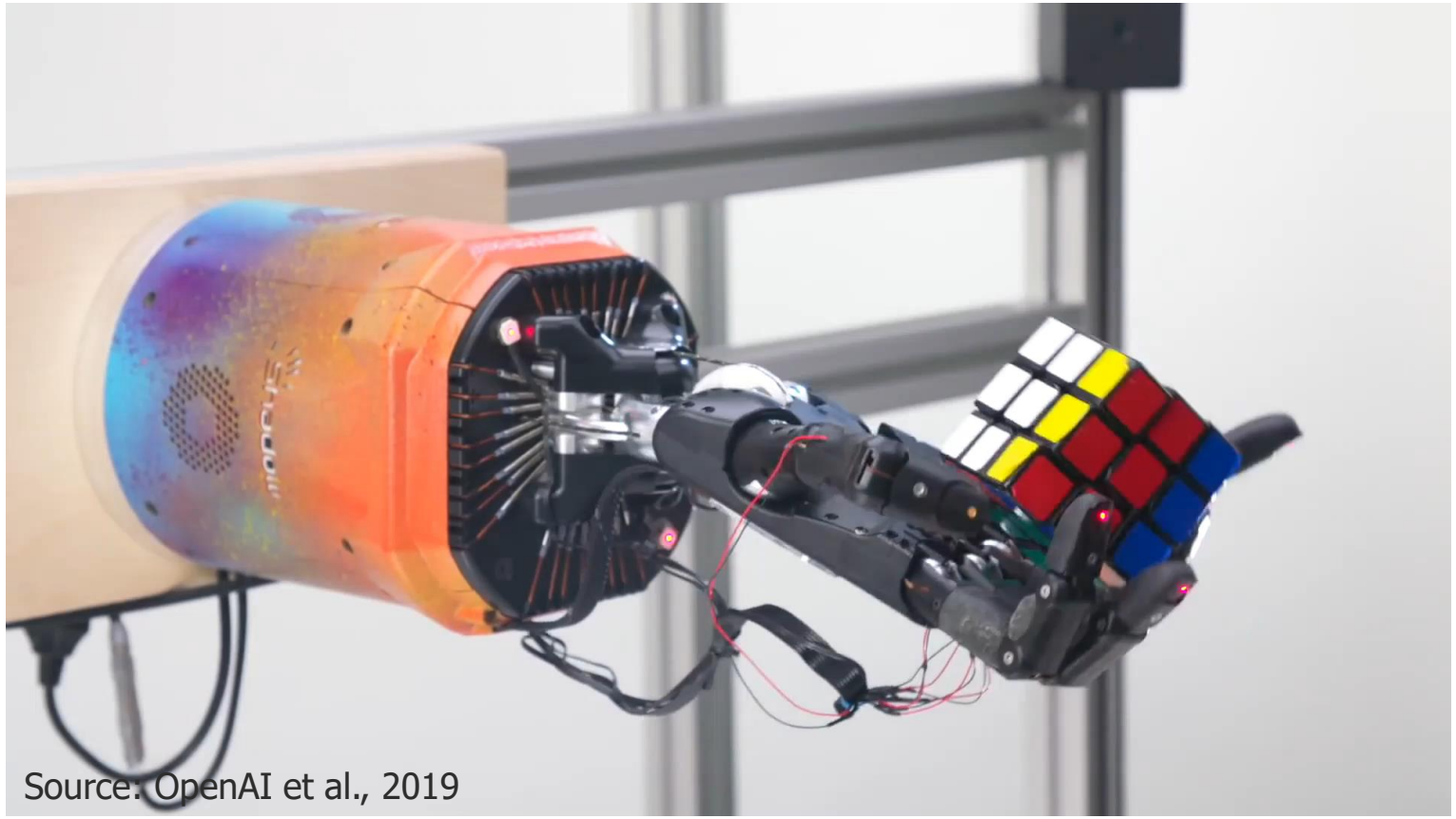
OpenAI 5 was able to **beat the best human players** in the highly complex game of Dota 2 and wins **99,4 % of public matches**!

Paper: "Dota 2 with Large Scale Deep Reinforcement Learning" (2019)
https://arxiv.org/abs/1912.06680

PTW: can we use this to tackle the problem of controlling highly complex industrial supply systems?

Motivation for using DRL

What **can** DRL already do?



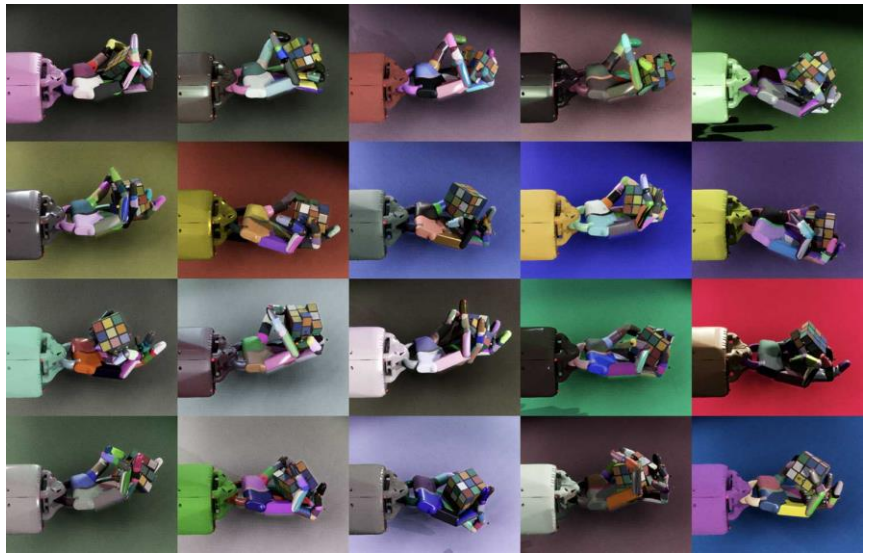
Source: OpenAI et al., 2019

- **OpenAI** used DRL to **learn robot-hand dexterity** to a degree that it is possible to **solve a rubrics cube** with it.
- **Paper**: "Solving Rubik's Cube with a Robot Hand" (2019) https://arxiv.org/abs/1910.07113

PTW: we can use this to tackle the problem of controlling highly complex industrial supply systems**!**

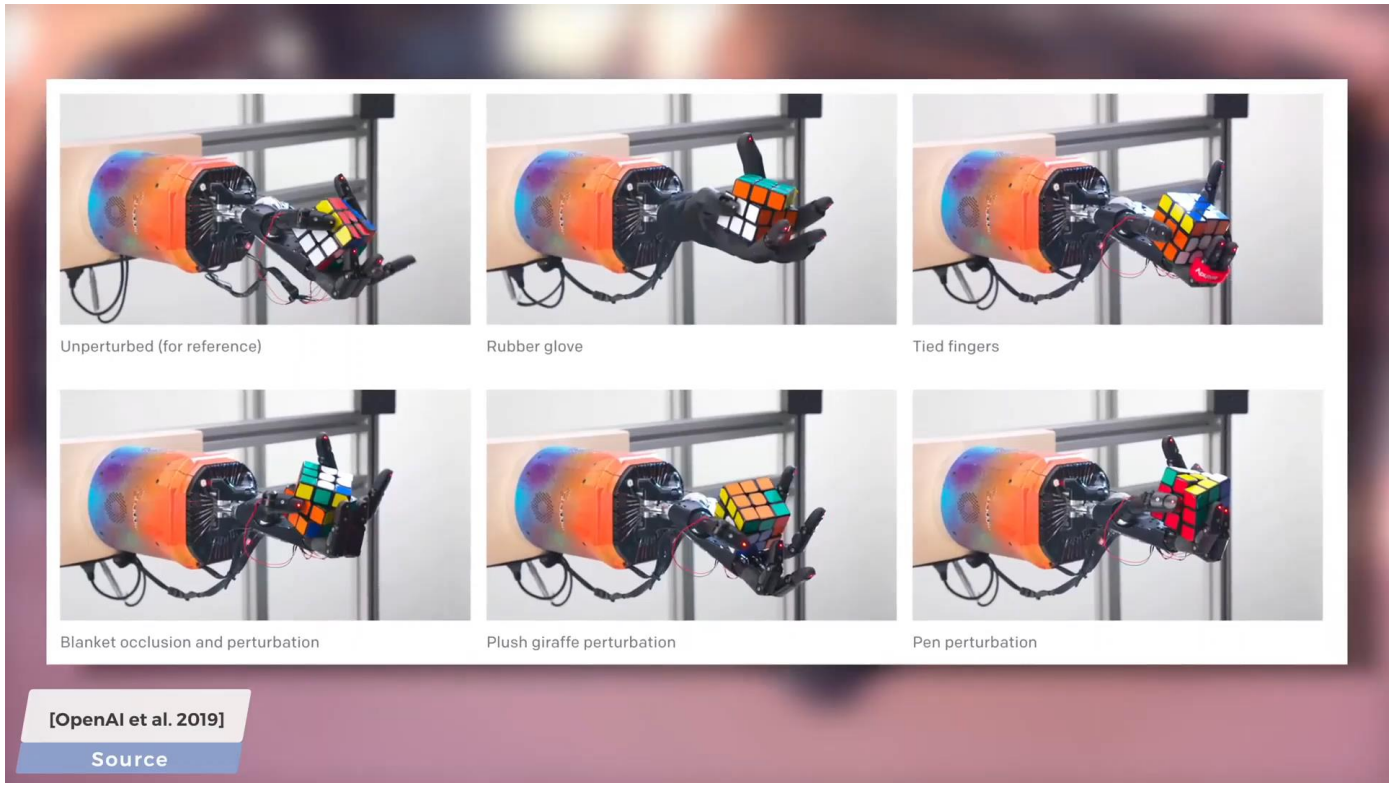
Motivation for using DRL

What **can** DRL already do?



OpenAI: "**Domain randomization** enables networks trained solely in simulation **to transfer to a real robot.**"

Source: OpenAI et al., 2019



Motivation for using DRL

What **can** DRL already do?

OpenAI is a **research organization** with the goal of "**discovering** and **enacting** the path to **safe artificial general intelligence**."

https://openai.com/

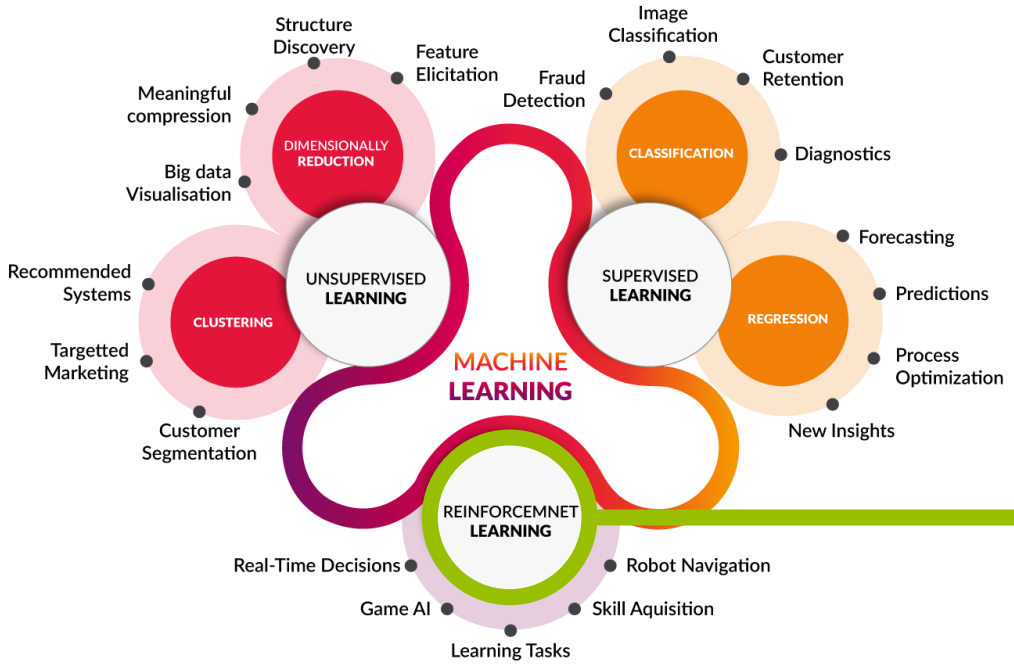
So the control of **highly complex** systems **is possible**, but

- how does it work?
- how can we use it for our control problem (industrial supply systems)?

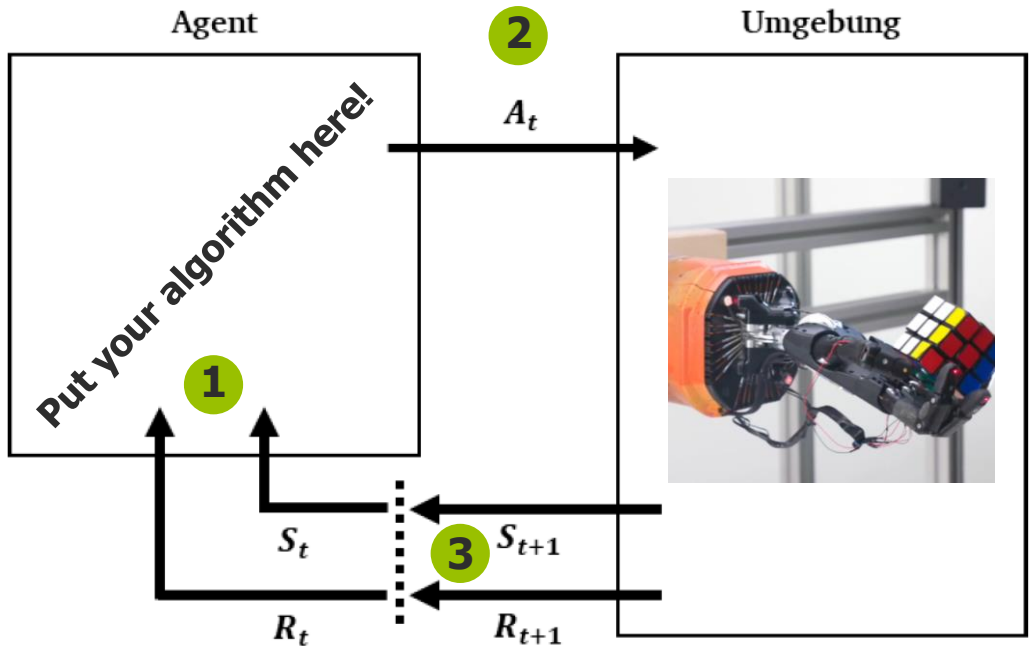
Agenda

DRL Basics

How to create a self learning system?



General RL Scheme



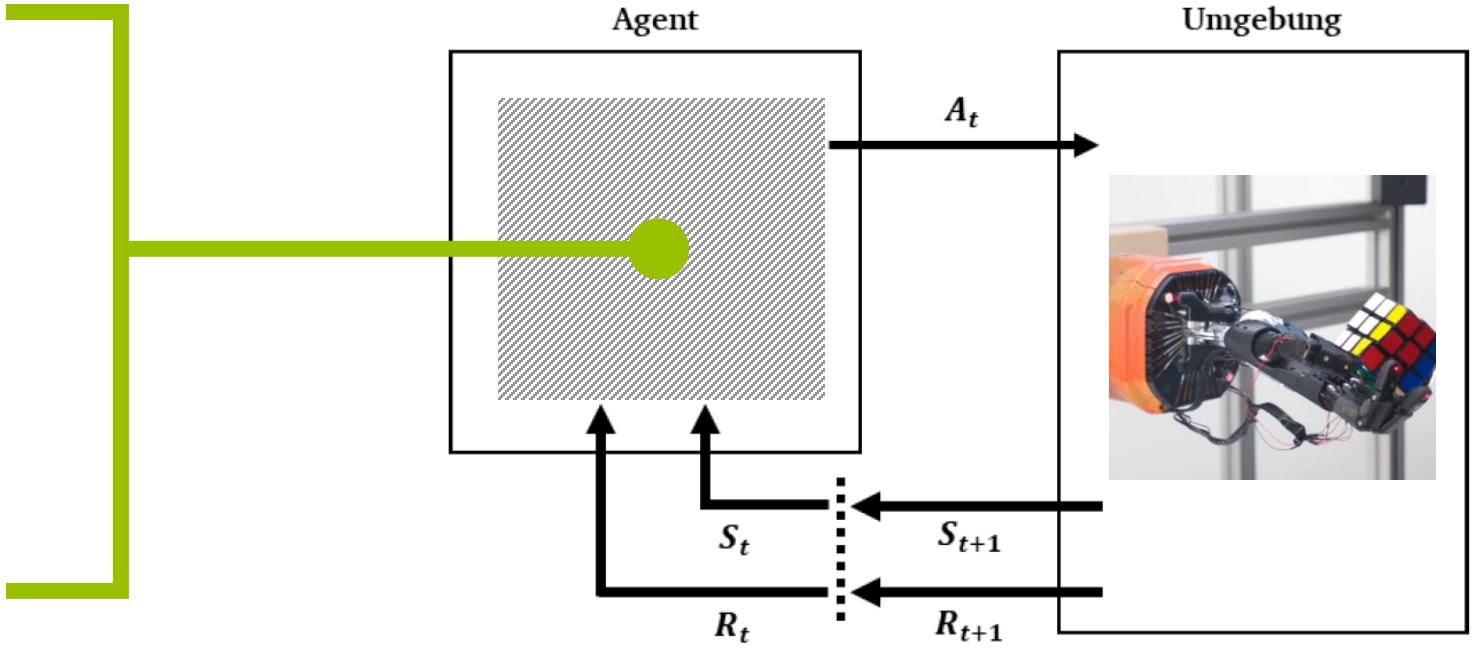
Through **Markov Decision Processes**, the problem of **sequential** decision-making under uncertainty can be mathematically modelled

DRL Basics

How to create a self learning system?

- We need to find a function, that **maps states (S_t)** to **actions (A_t)**
- In Reinforcement Learning, this is called a **policy (π)**.
- The policy needs to be able to **improve** by receiving a **numeric reward / punishment** (R_t) that depends on the system state.

General RL Scheme



This can be done with Q-Learning!

DRL Basics

How to create a self learning system? Example: Q-Learning

- **Q-learning** was introduced by Chris Watkins in **1989**. A convergence proof was presented by Watkins and **Dayan** in **1992.**

- In **2014 Google DeepMind** patented an application of Q-learning to deep learning, titled "deep reinforcement learning" or "deep Q-learning" that can **play Atari 2600 games at expert human levels.** (Deepmind was bought by Google for ~ 500 Mio. $)

The objective of Q-learning is to **find a policy** that is optimal in the sense that the **expected return** over **all successive time steps** is the **maximum** achievable.

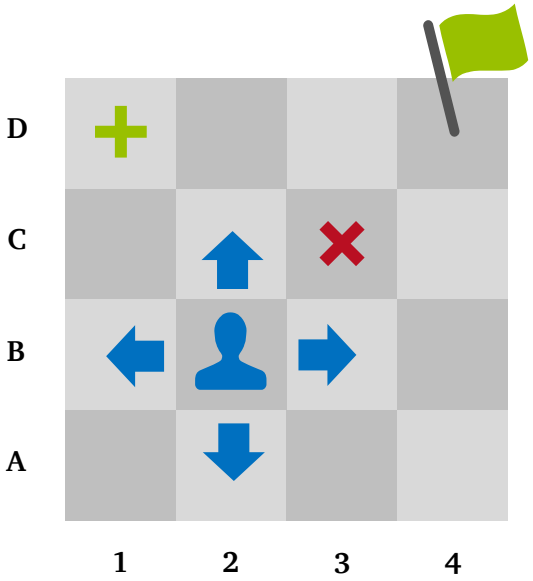
DRL Basics

Q-Learning – An example

How can we find that policy?
(remember: the policy maps **states** to **actions**!)

Q-learning separates this problem into **two sub-problems**:

1. Find Q-values **for every state-action pair**
2. Choose an **action regarding to the Q-value**



+20 & restart

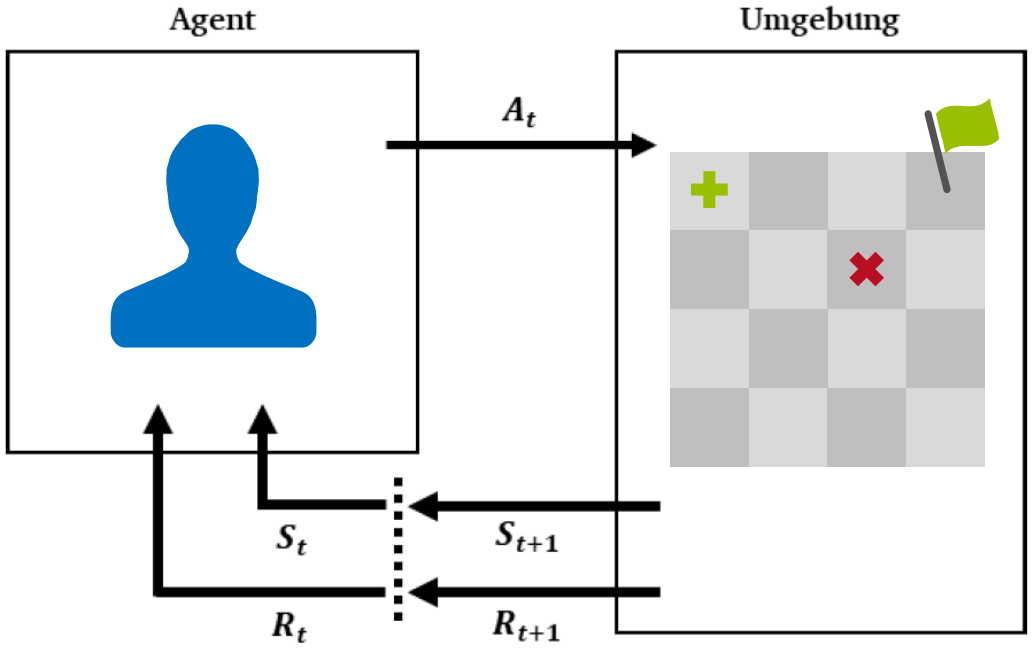
+5

-5

-1

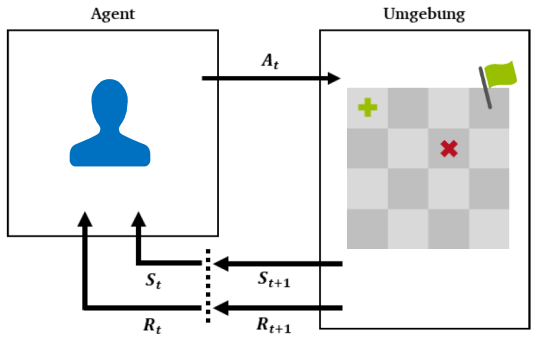
max. 10 steps, then restart

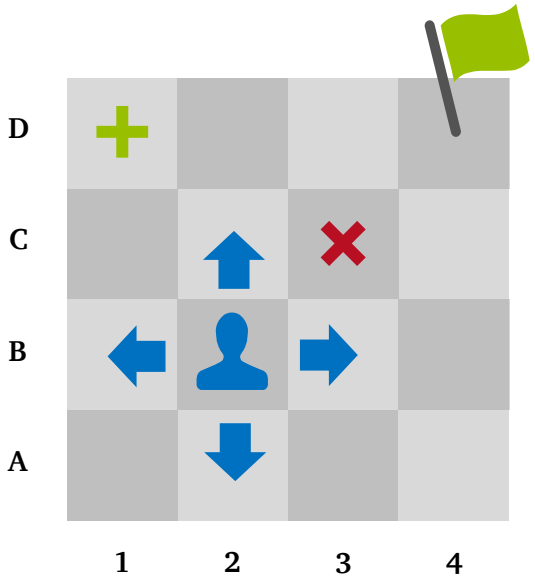
General DRL Scheme



DRL Basics

Q-Learning, Q-table and greedy policy





Actions

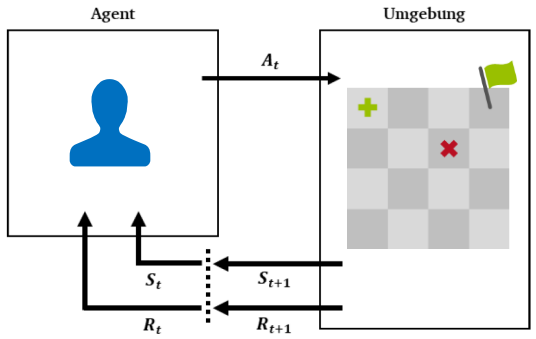
States	➡	⬆	⬅	⬇
A1	$q_*(A1, r)$	$q_*(A1, u)$	$q_*(A1, l)$	$q_*(A1, d)$
A2	$q_*(A2, r)$	$q_*(A2, u)$	$q_*(A2, l)$	$q_*(A2, d)$
A3	$q_*(A3, r)$	$q_*(A3, u)$	$q_*(A3, l)$	$q_*(A3, d)$
A4	$q_*(A4, r)$	$q_*(A4, u)$	$q_*(A4, l)$	$q_*(A4, d)$
B1	$q_*(B1, r)$	$q_*(B1, u)$	$q_*(B1, l)$	$q_*(B1, d)$
B2	$q_*(B2, r)$	$q_*(B2, u)$	$q_*(B2, l)$	$q_*(B2, d)$
⋮	⋮	⋮	⋮	⋮

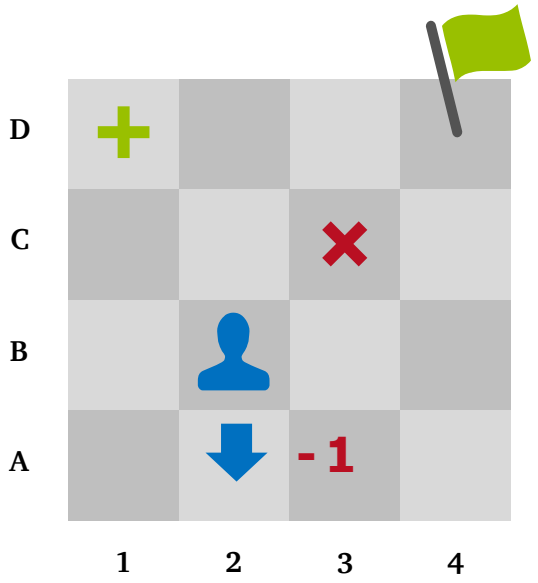
! If we would know optimal Q-values, we could use a **greedy policy** and just take **the actions with the highest values**

The Q-value is the **expected return** from **taking a given action in a given state** and **following the given policy** thereafter.

DRL Basics

Q-Learning, Q-table





Actions

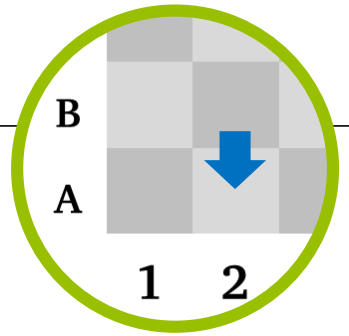
States	→	↑	←	↓
A1	0	0	0	0
A2	0	0	0	0
A3	0	0	0	0
A4	0	0	0	0
B1	0	0	0	0
B2	0	0	0	0
⋮	⋮	⋮	⋮	⋮

! We take the **random** **action** "down", go from **B2** → **A2** and receive the reward **-1**.

New, empty Q-table

DRL Basics

Q-Learning, Q-table calculation



Actions

States	→	↑	←	↓
A1	0	0	0	0
A2	0	0	0	0
A3	0	0	0	0
A4	0	0	0	0
B1	0	0	0	0
B2	0	0	0	**0**
⋮	⋮	⋮	⋮	⋮

Bellman optimality equation

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

expected reward

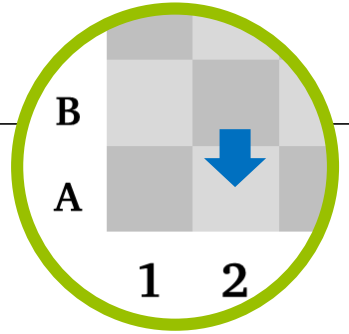
maximum expected discounted return

$$q_*(s,a) - q(s,a) = loss$$

This is an **iterative**, "learning" process! The knowledge about the environment and which action to take **gets stored in the Q-table** as **Q-values**. For this, we have to solve the **Bellman optimality equation**.

DRL Basics

Q-Learning, Q-table calculation



Hyperparameters	Value
α (learning rate)	0,7
γ (discount factor)	0,99

Actions

States	→	↑	←	↓
A1	0	0	0	0
A2	0	0	0	0
A3	0	0	0	0
A4	0	0	0	0
B1	0	0	0	0
B2	0	0	0	**0**
⋮	⋮	⋮	⋮	⋮

$$\max_{a'} q(s', a') = \max(0,0,0,0) = 0$$

$$q^{new}(s,a) = (1-\alpha)\underbrace{q(s,a)}_{\text{old value}} + \alpha\overbrace{\left(R_{t+1} + \gamma \max_{a'} q(s', a')\right)}^{\text{learned value}}$$

$$\begin{aligned} &= (1-0.7)(0) + 0.7\left(-1 + 0.99\left(\max_{a'} q(s', a')\right)\right) \\ &= (1-0.7)(0) + 0.7(-1 + 0.99(0)) \\ &= 0 + 0.7(-1) \\ &= -0.7 \end{aligned}$$

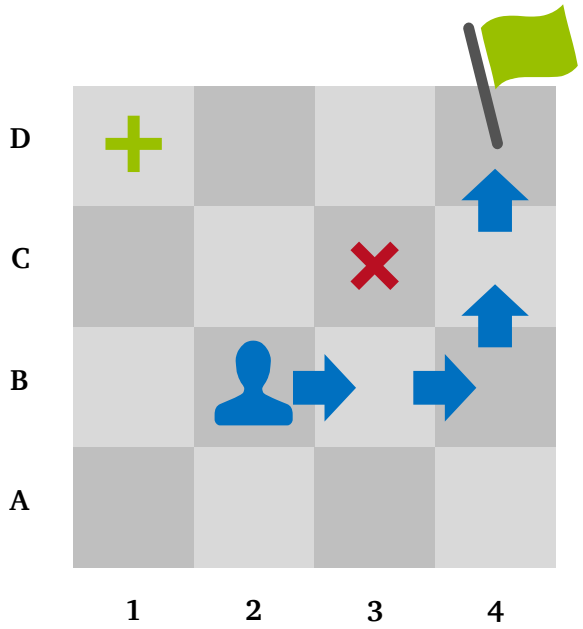
This is an **iterative**, "learning" process! The knowledge about the environment and which action to take when **gets stored in the Q-table**.

DRL Basics

Q-Learning – Exploration vs. Exploitation

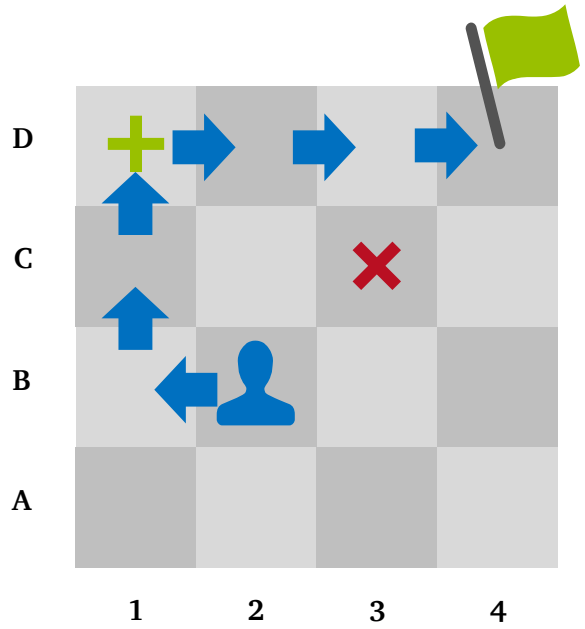
Does it make sense to use the greedy policy all the time?

- **Pro**: Yes, because we want to maximize the cumulative reward!
- **Contra**: No, because then we will take the first solution that works and don't explore more profitable ones!



with greedy policy

Sum of rewards: **17**



With ϵ-greedy policy

Sum of rewards: **21**

! If we immediately follow the **first** working path we find, we **might never find** the extra points (or even the flag)!

DRL Basics

Q-Learning – Exploration vs. Exploitation

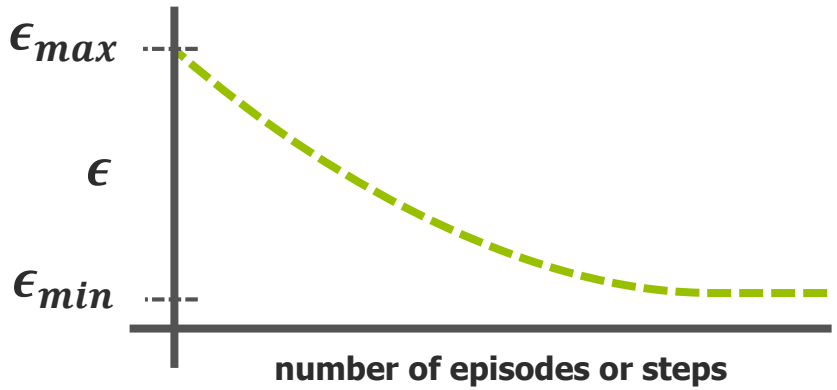
We need to find a **balance** between **exploration and exploitation**!

Exploration is the act of **exploring the environment** to **gather information** about it

Exploitation is the act of exploiting the information that is **already known** in order **to maximize the return**

We can do this by using the ϵ**-greedy policy!**

The ϵ**-greedy policy**, like the "pure" greedy policy, chooses the action with the best Q-value, **but** with the **probability** ϵ it chooses a **random action**.

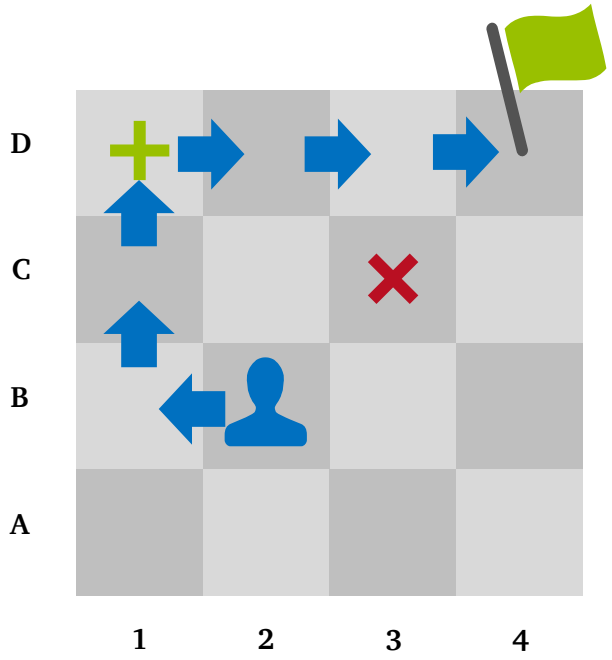


DRL Basics

Q-Learning, Optimal Q-table and Hyperparameters

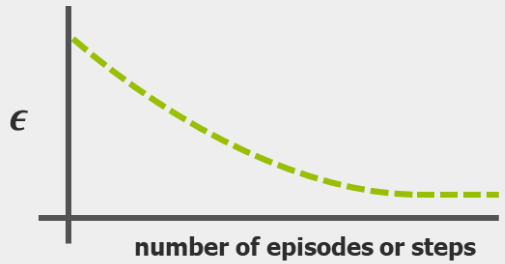
States	Actions →	Actions ↑	Actions ←	Actions ↓
A1	0	3	-2	-2
A2	-1	2	2,5	-2
A3	-1,5	1	3	-2
A4	-2	2,5	2	-2
B1	2	4	-2	-3
B2	-1	3	4	-2
⋮	⋮	⋮	⋮	⋮

Trained, optimal Q-table



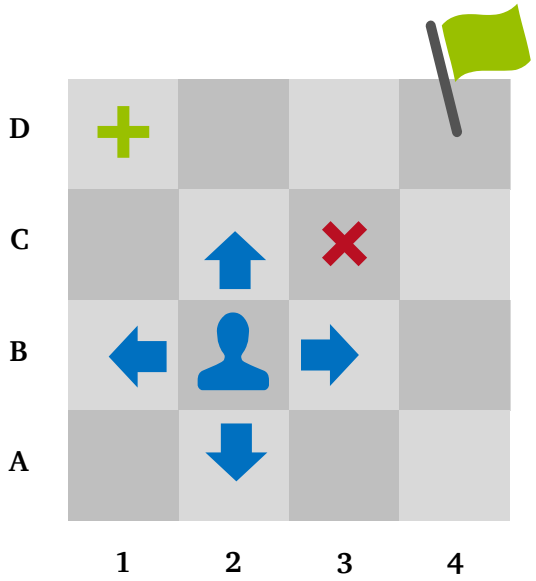
Optimal actions

Hyperparameters	Value
α (learning rate)	0,7
γ (discount factor)	0,99
ϵ (exploration rate)	$f(s)$
ϵ_{max} (max. expl.-rate)	1
ϵ_{min} (max. expl.-rate)	0,01
ϵ_{γ} (expl.- decay rate)	0,01



DRL Basics

Reinforcement Learning, Simple vs. complex Environments



+20 & restart **+5** **-5**

-1 **max. 10 steps, then restart**

VS

What is different here?

Most environments **relevant to real application** cases are **much more complex** than our gridworld-example!
It is **not possible** to gather and store the information **about all states and actions**. How is this problem solved?

DRL Basics

The Deep-Q Algorithm (how to turn RL into DRL)

State A2

Actions

States	→	↑	←	↓
A1	0	3	-2	-2
A2	-1	2	2,5	-2
A3	-1,5	1	3	-2
A4	-2	2,5	2	-2
B1	2	4	-2	-3
B2	-1	3	4	-2
⋮	⋮	⋮	⋮	⋮

$q(A2, r)$
$q(A2, u)$
$q(A2, l)$
$q(A2, d)$

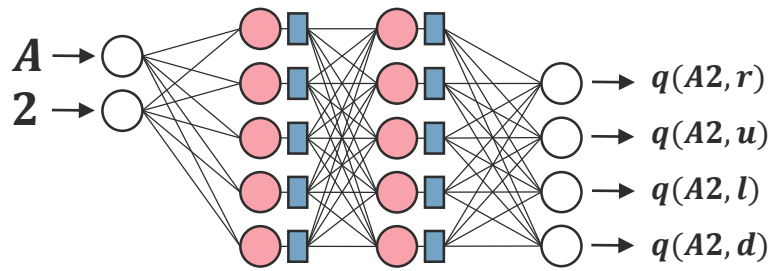
ϵ-greedy policy → **Actions**

Reinforcement Learning

+ Deep Neural Networks =

Deep Reinforcement Learning

State A2



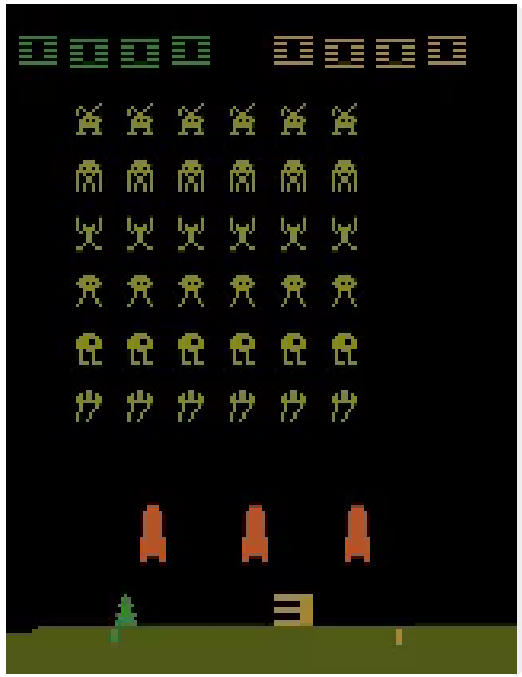
ϵ-greedy policy → **Actions**

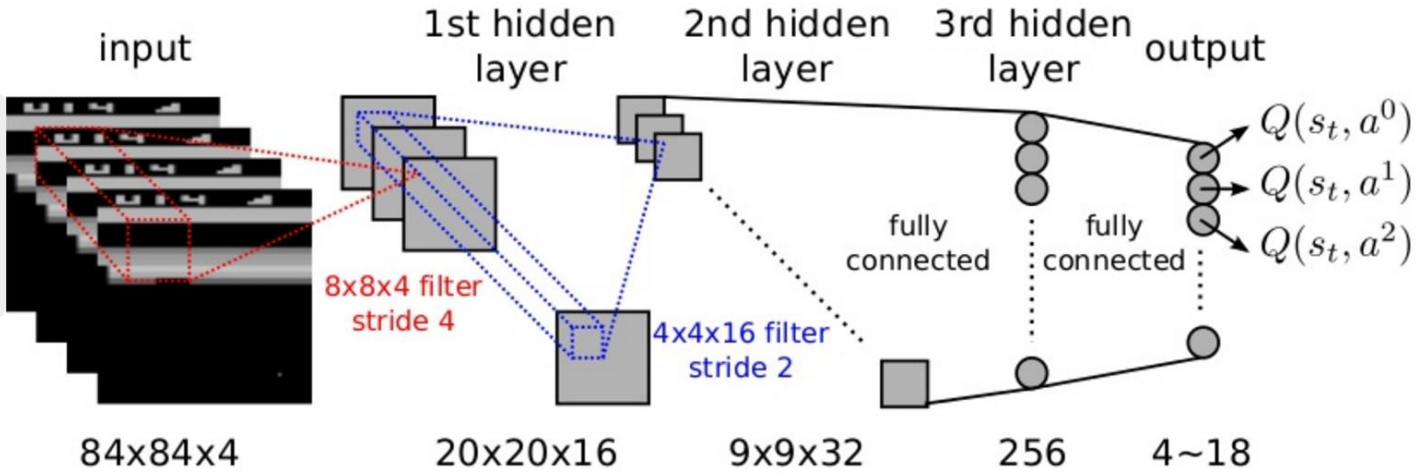
Deep Neural Network

! Deep neural networks can be considered as **universal function approximators**

DRL Basics

The Deep-Q Algorithm (how to turn RL into DRL)



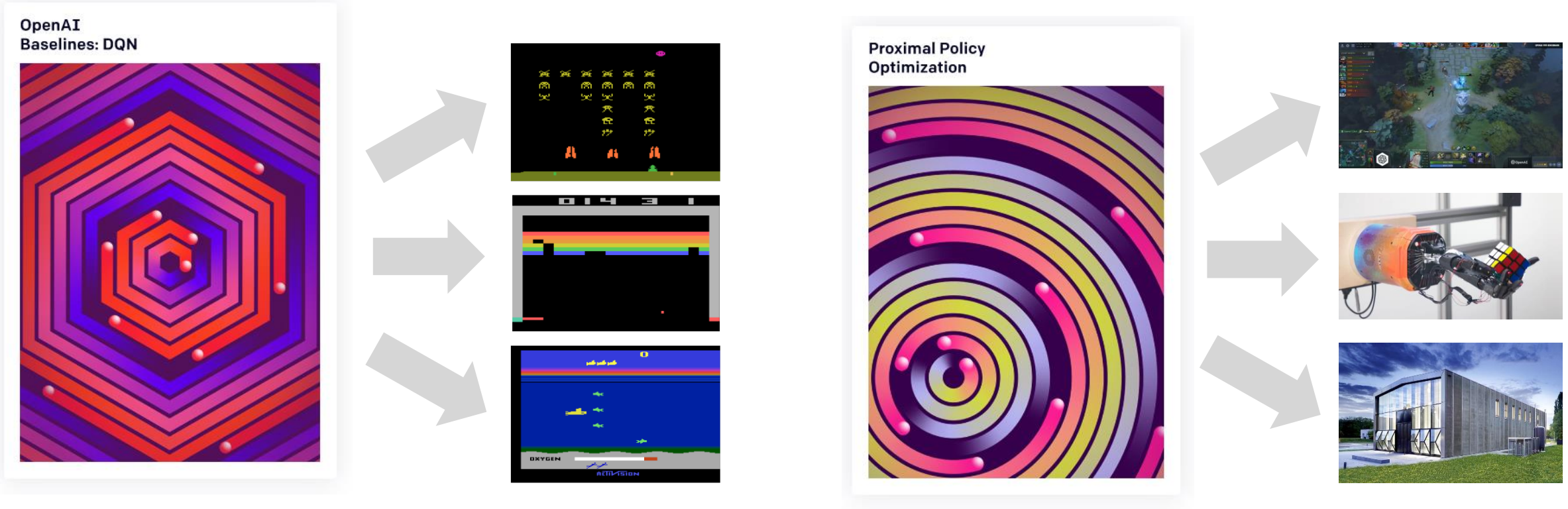


Paper: "Playing Atari with Deep Reinforcement Learning" (2013)

https://arxiv.org/abs/1312.5602

DRL Basics

Advanced algorithms



- **OpenAI-Baselines:** https://github.com/openai/baselines
- **Stable-Baselines:** https://stable-baselines.readthedocs.io/en/master/

Agenda

Application Case

Highly connected systems allow for greater efficiency

Today: Isolated optimization of different sub-systems of a factory

Building

25%

Quelle: Prof. Dipl.-Ing. J. Eisele

Process chain

20%

Machine

30%

Savings

< 30 %

Our vision: Holistic factory optimization including all sub-systems

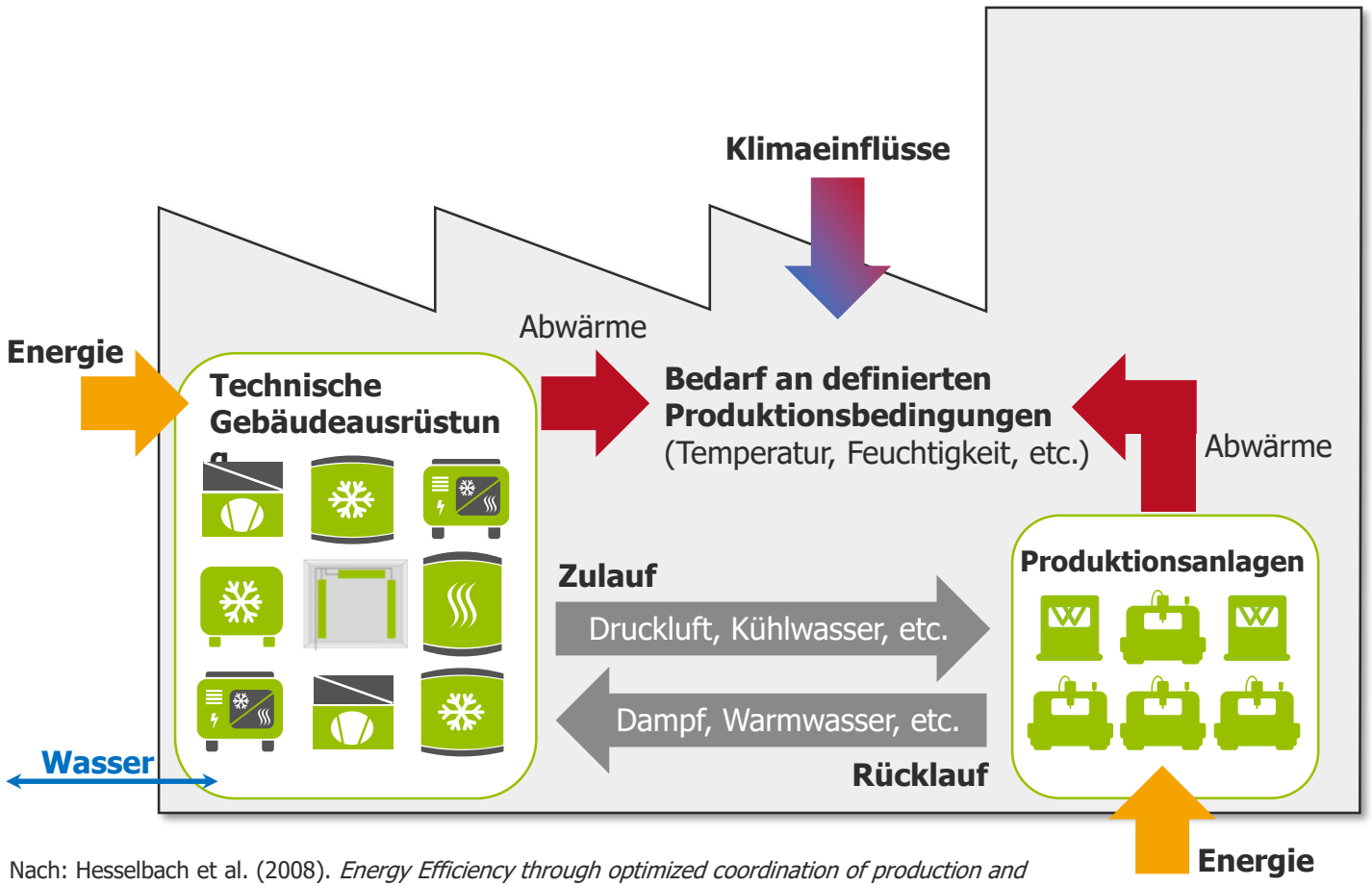
Potential

~ 40 %

Synergies by energy controlling and recovery measures

Application Case

Industrial supply systems



Nach: Hesselbach et al. (2008). *Energy Efficiency through optimized coordination of production and technical building services.*

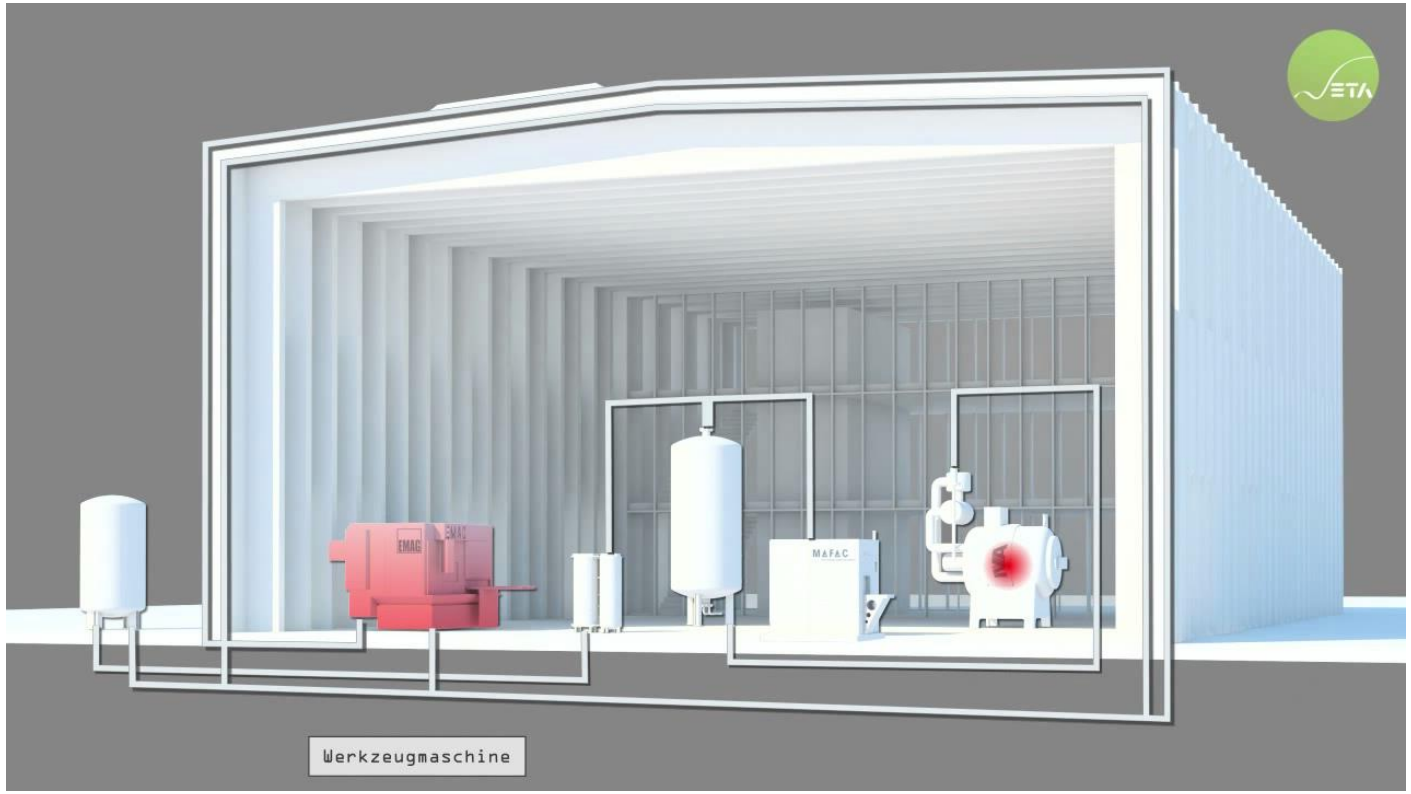
System boundary around the factory

Interaction of

- Production facilities
- Supply systems
- Building
- Interfaces with the environment

Application Case

Industrial supply systems at the ETA-factory – an example

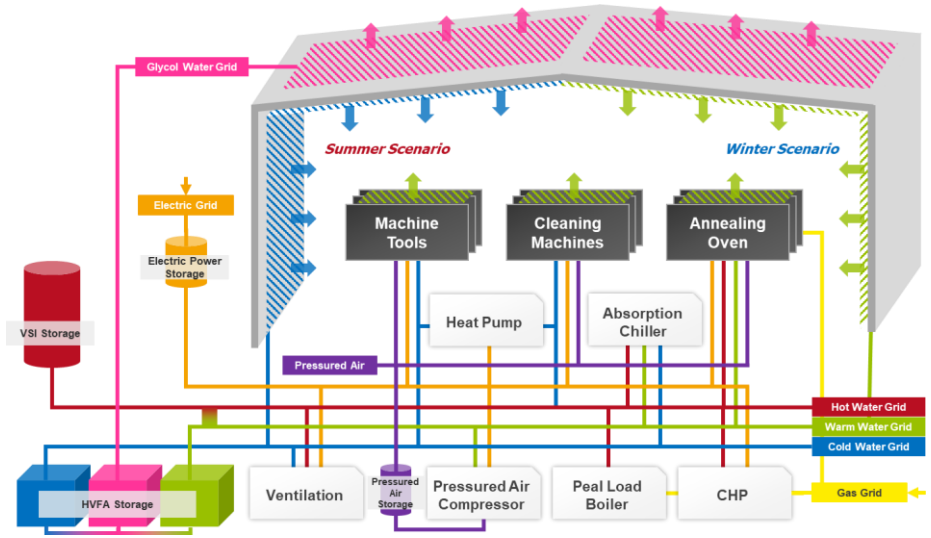


- **Interaction** between **machines**, **storages** and the **building**
- **Waste heat** can be recovered and used for heating cleaning machines

- This is **ahead** of common practice in the industry and can offer **promising strategies** for **brownfield** and **greenfield** applications!

Application Case

What can we win by applying DRL to control industrial supply systems?



It is hard to find the optimal control strategy

- Supply Systems can become **very complex** with **increasing interlinkage**
- Storages allow for the **temporal decoupling** of **supply and demand** but **increase the complexity** even further
- **Fluctuating energy prices** have to be taken into account to choose optimal actions
- **External influences** like solar irradiance, temperature and air humidity have a **strong influence** on the supply systems

- The **non-linear system dynamics** and multiple **semi-random influences** create a **unique** challenge for every new application case

Agenda

Implementation

Conditions for the right framework

Open Source software

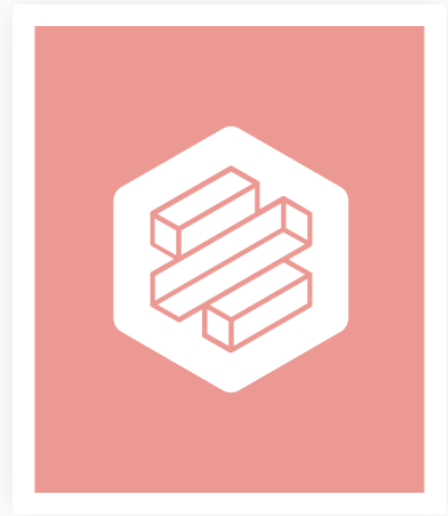
State of the art performance

Easy to learn and **widely available** programming language with many **free libraries**

Accurate Digital Twin of the supply systems

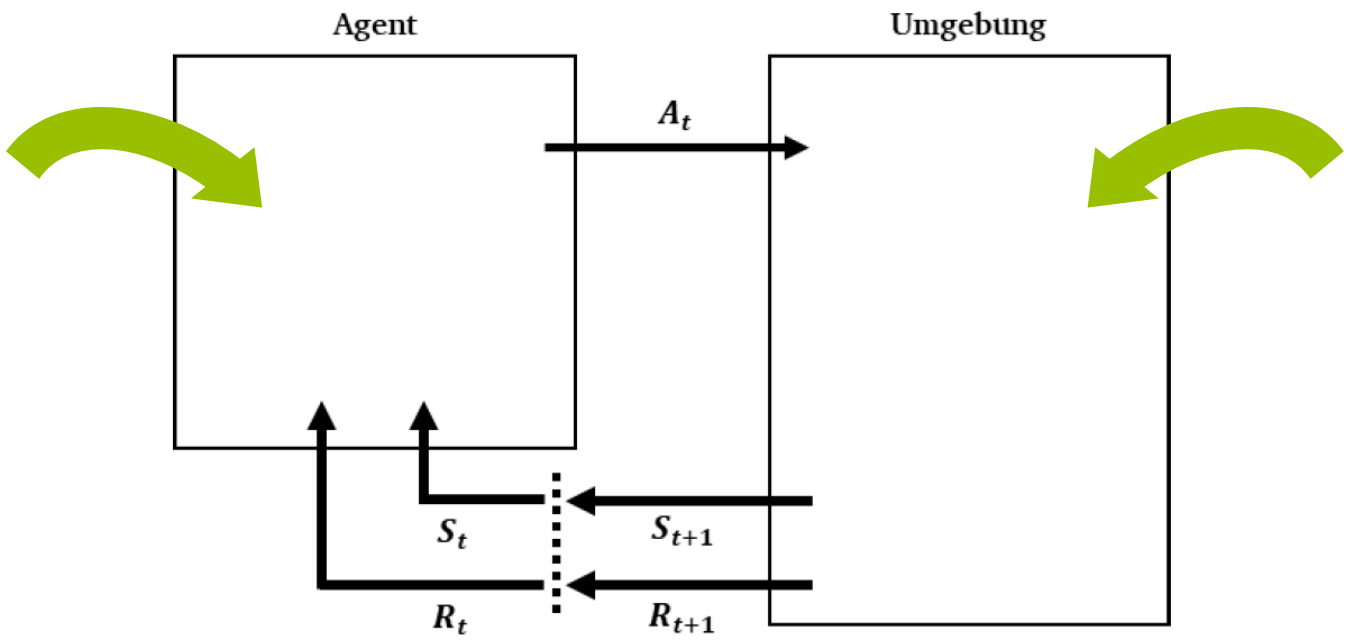
Implementation

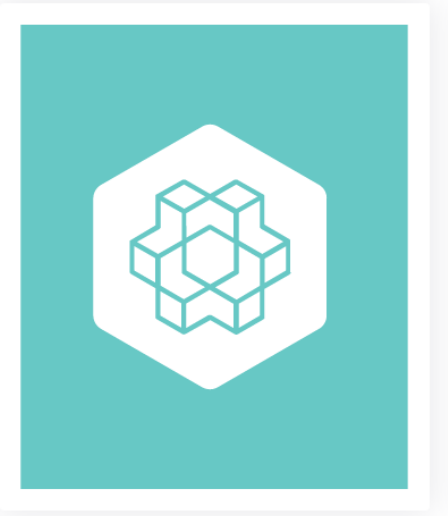
Step 1: Choosing the right framework



Baselines

High-quality implementations of reinforcement learning algorithms.





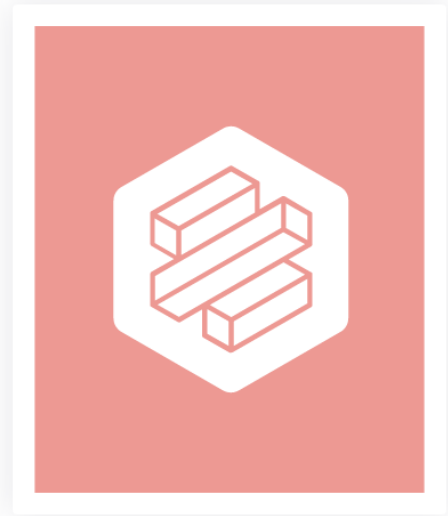
Gym

Toolkit for developing and comparing reinforcement learning algorithms.

OpenAI website: https://openai.com/

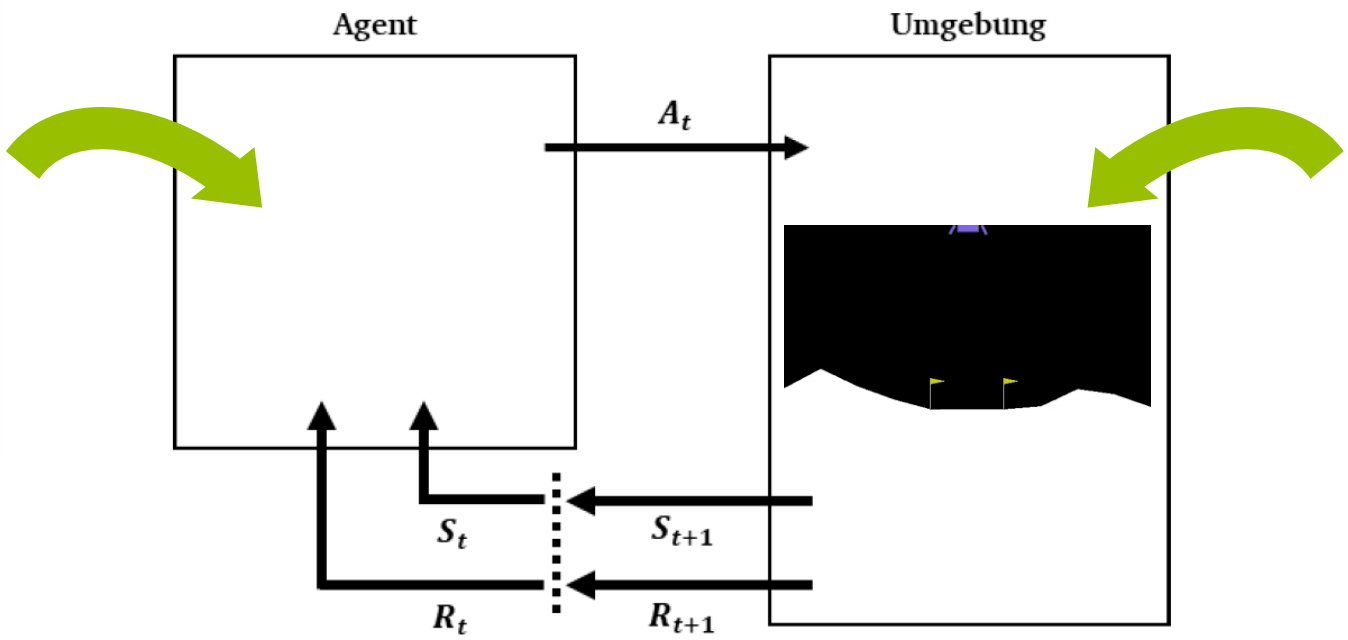
Implementation

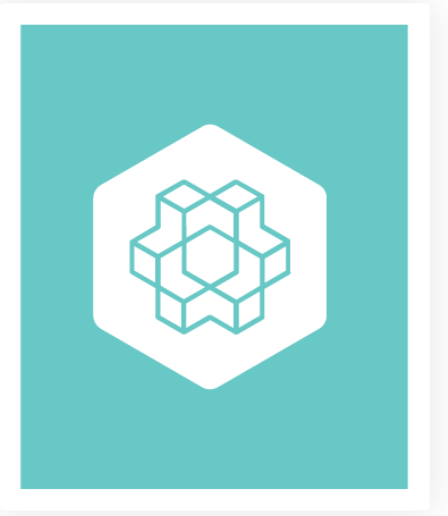
Step 1: Choosing the right framework



Baselines

High-quality implementations of reinforcement learning algorithms.





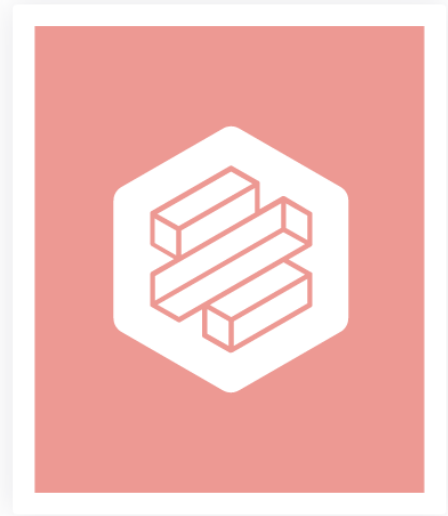
Gym

Toolkit for developing and comparing reinforcement learning algorithms.

OpenAI website: https://openai.com/

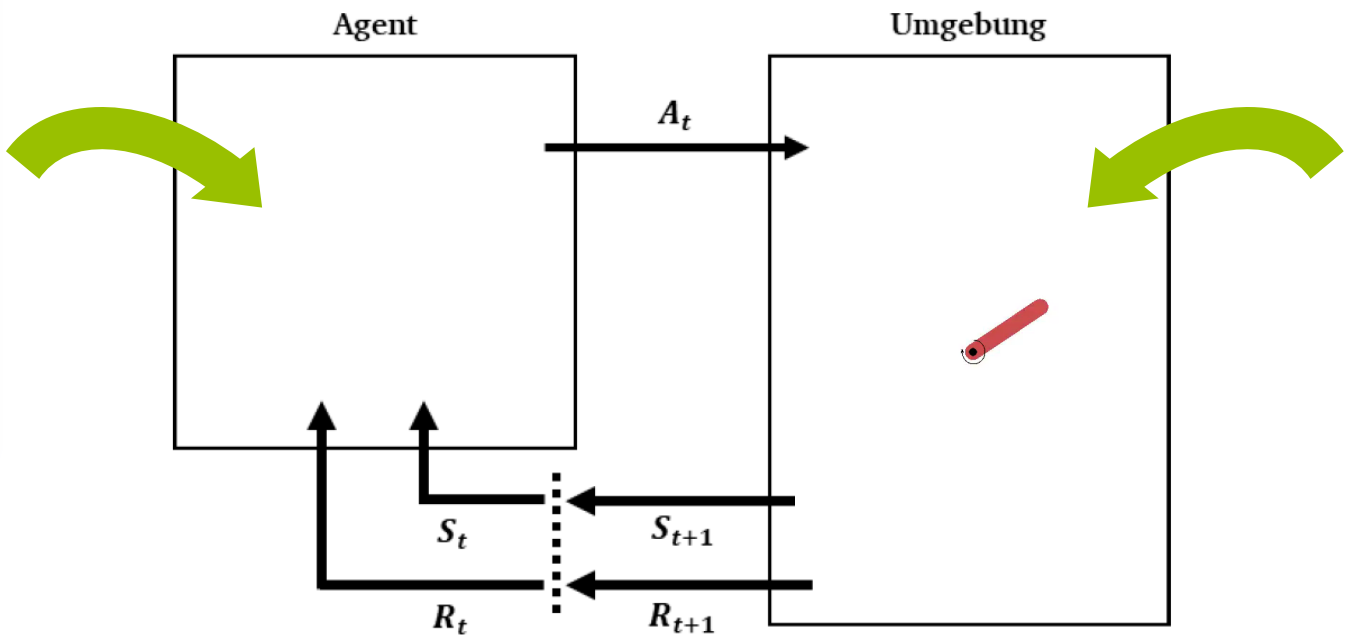
Implementation

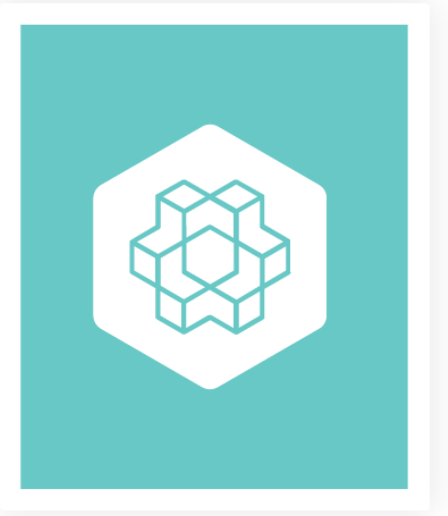
Step 1: Choosing the right framework



Baselines

High-quality implementations of reinforcement learning algorithms.





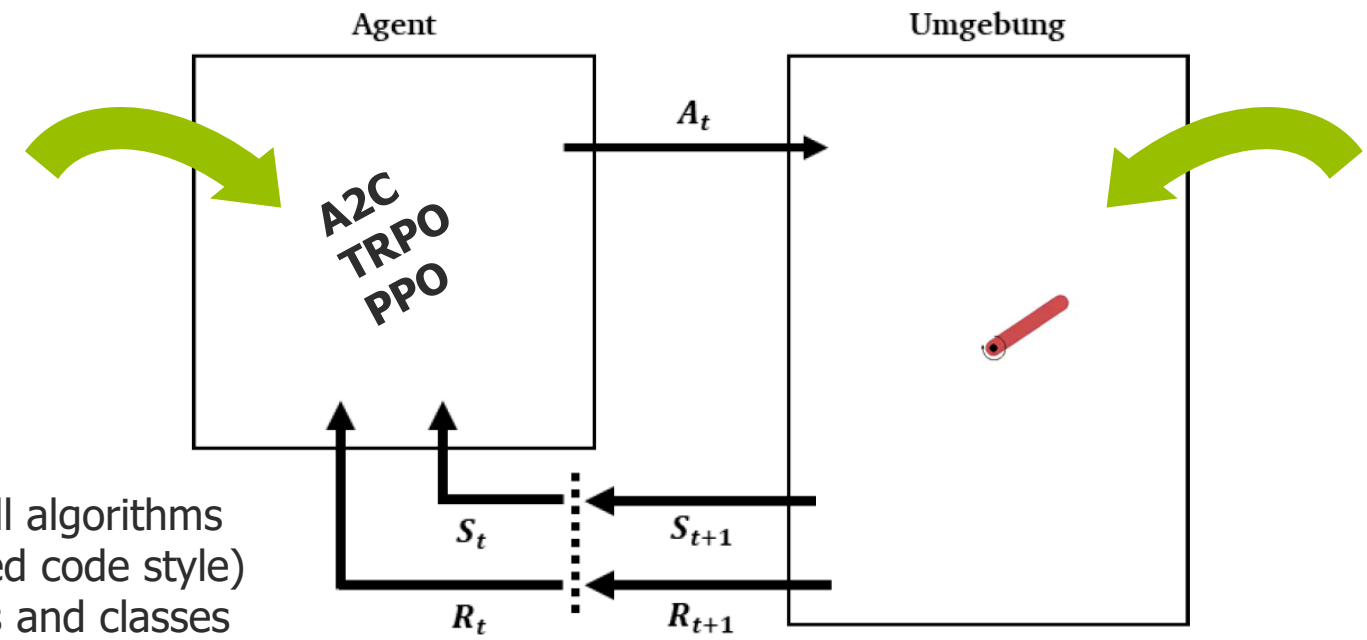
Gym

Toolkit for developing and comparing reinforcement learning algorithms.

OpenAI website: https://openai.com/

Implementation

Step 1: Choosing the right framework



Gym
Toolkit for developing and comparing reinforcement learning algorithms.

Stable-Baselines

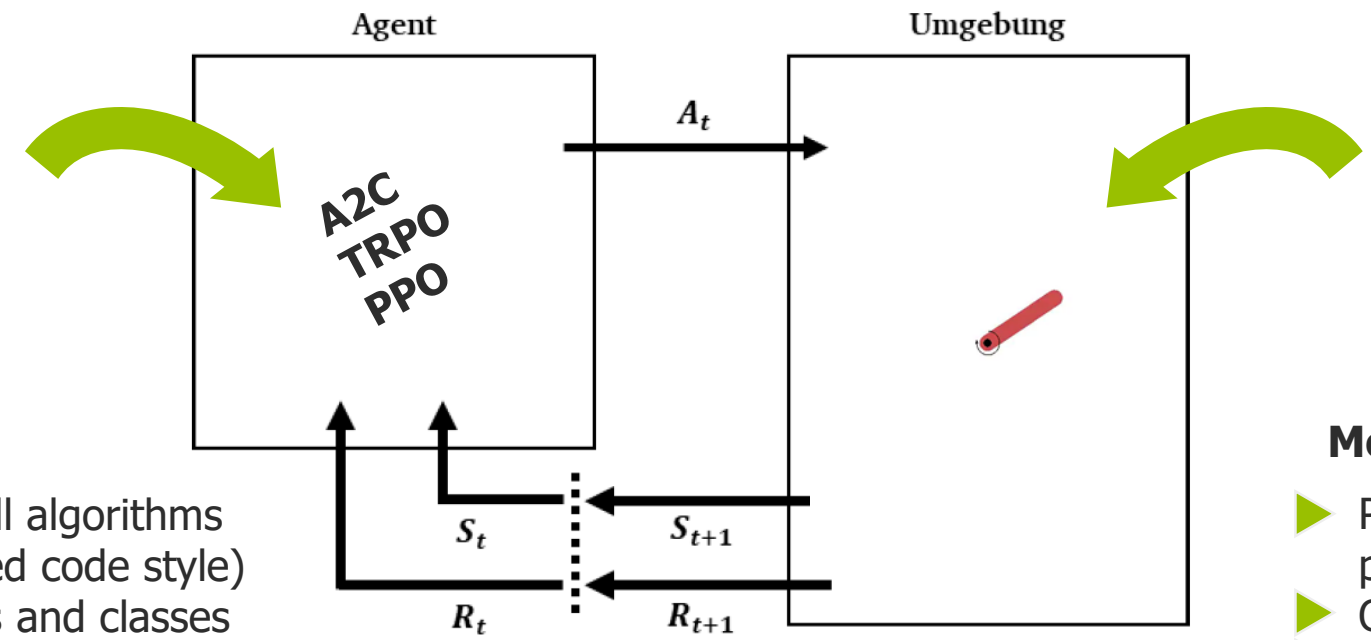
- Unified structure for all algorithms
- PEP8 compliant (unified code style)
- Documented functions and classes
- More tests & more code coverage
- Additional algorithms

Initialize
Step (loop)
Reset

- **Stable-Baselines:** https://stable-baselines.readthedocs.io/en/master/
  https://github.com/hill-a/stable-baselines
- **OpenAI Gym:** http://gym.openai.com/

Implementation

Step 1: Choosing the right framework



Stable-Baselines

- Unified structure for all algorithms
- PEP8 compliant (unified code style)
- Documented functions and classes
- More tests & more code coverage
- Additional algorithms

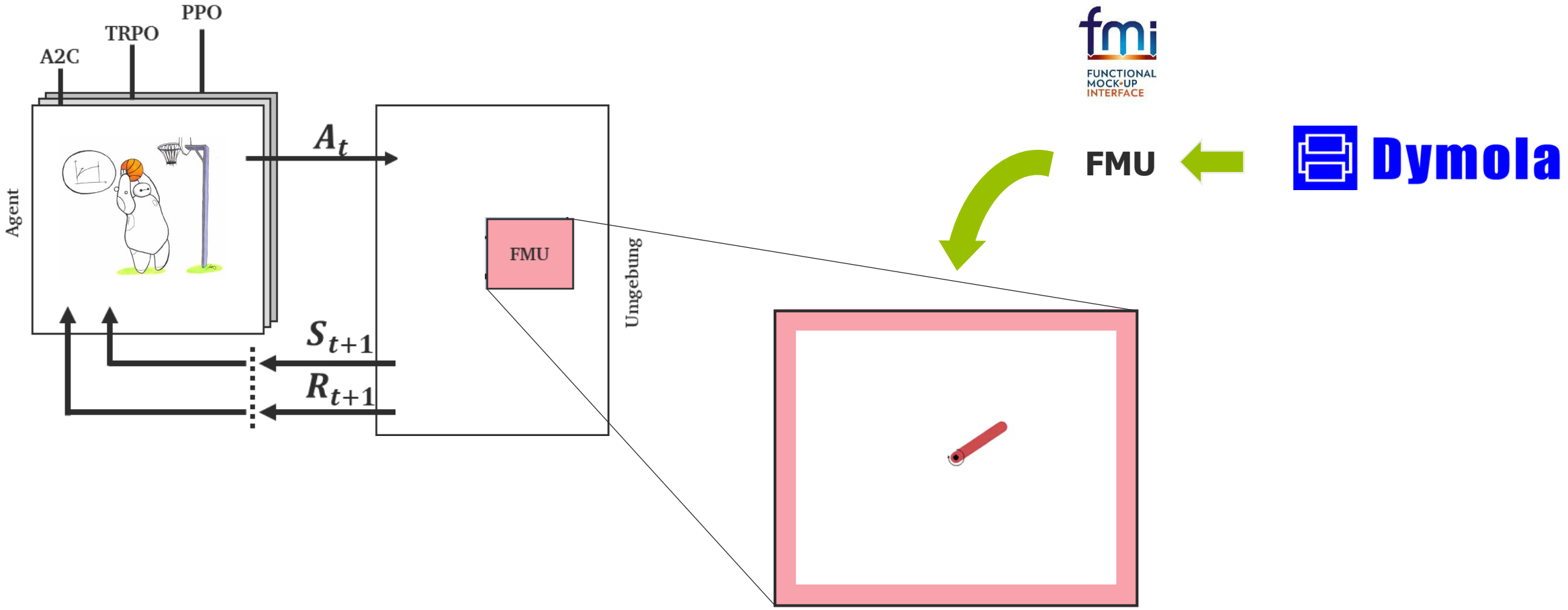
Modelica (Dymola)

- Powerful modeling tool for physical systems
- OpenSource (Modelica)
- Noncausal modelling possible

- **Stable-Baselines:** https://stable-baselines.readthedocs.io/en/master/
- **OpenAI Gym:** http://gym.openai.com/

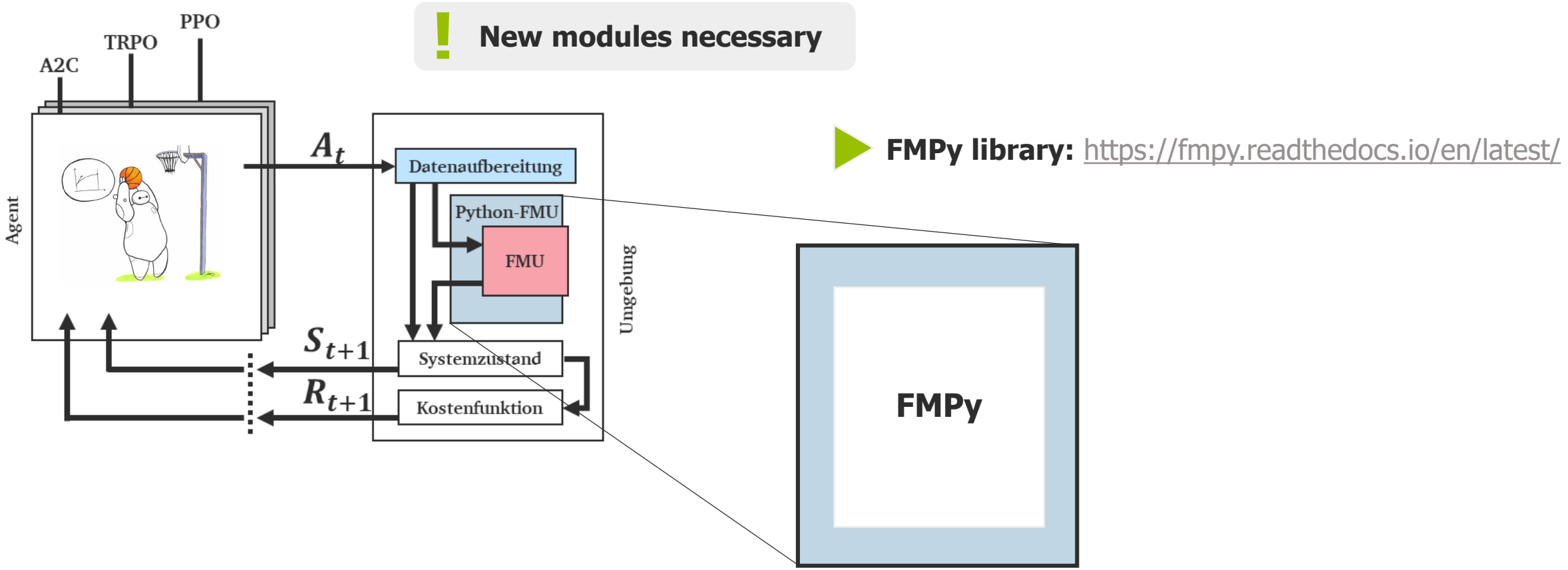
Implementation

Step 2: Creating a very simple Digital Twin and implementing it as an Environment



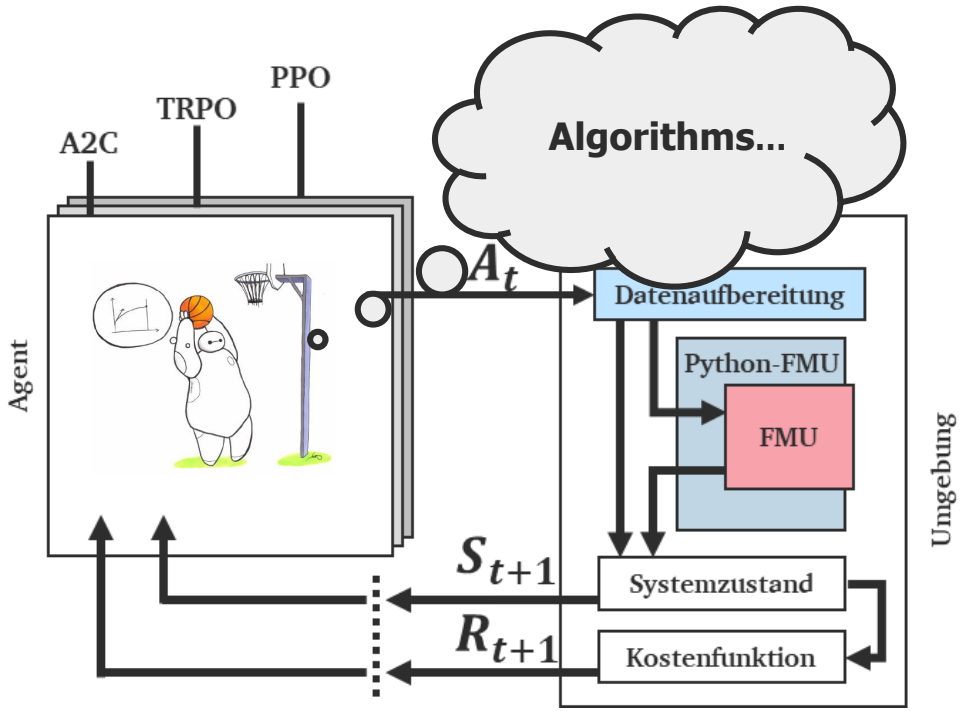
Implementation

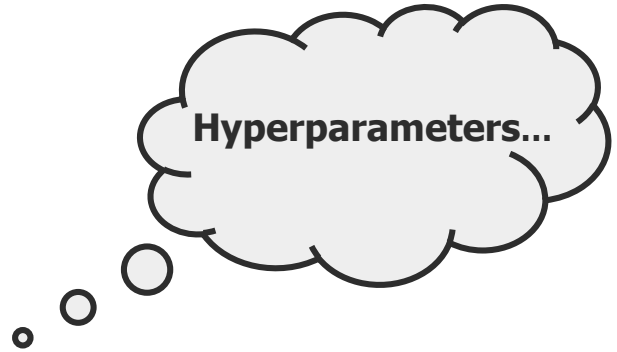
Step 3: Creating an Python-FMU interface that is compatible to the Gym-Structure



Implementation

Influences on the performance



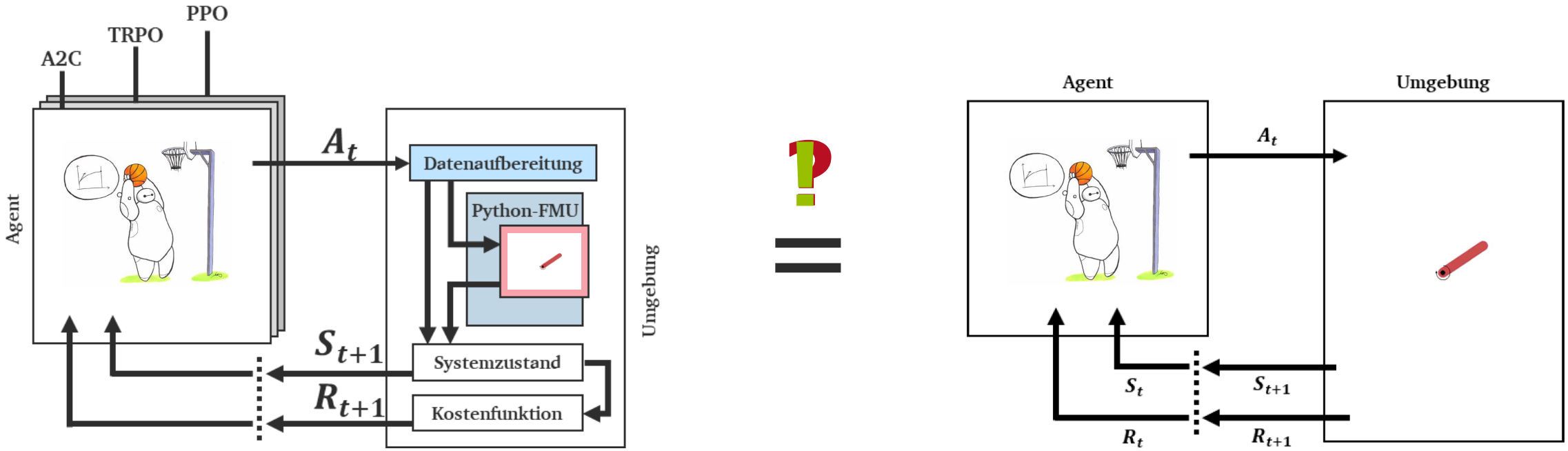


- Choosing the right **observations** for the **state** is crucial!
- Choosing the right **cost function** is crucial!

Pendulum-Environment-Example (Structure and Domain randomization)

Implementation

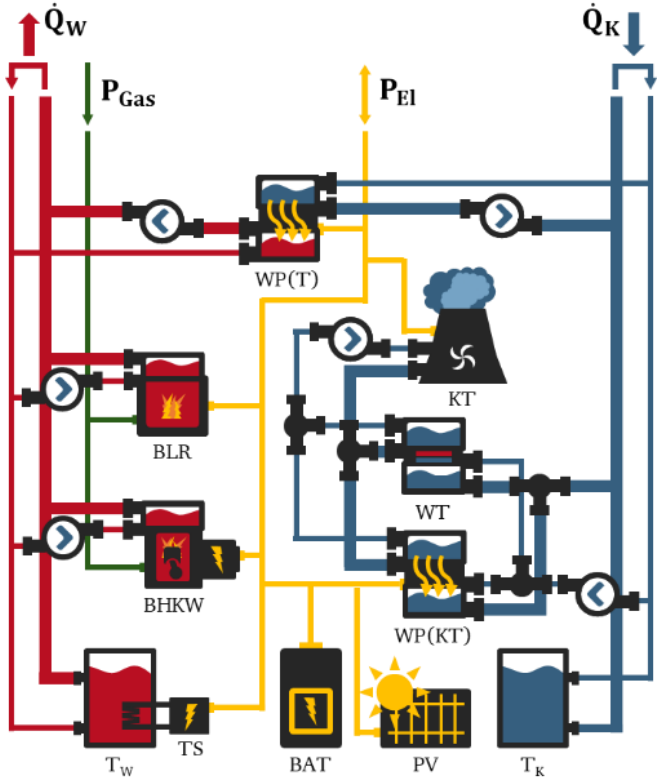
Step 4: Validate the FMU-implementation with existing system

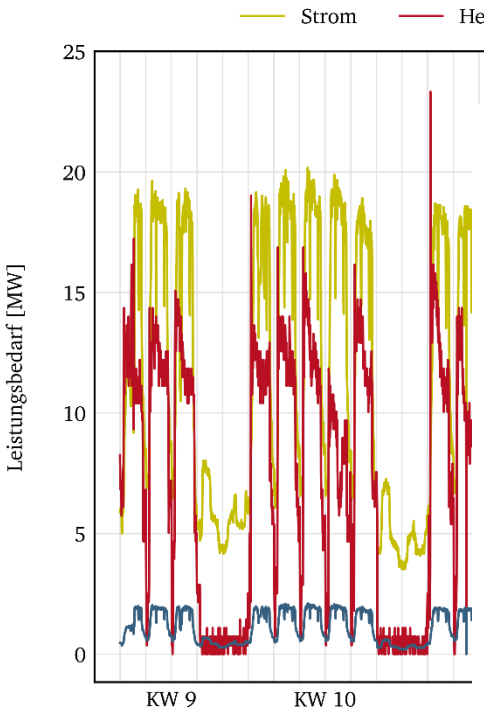


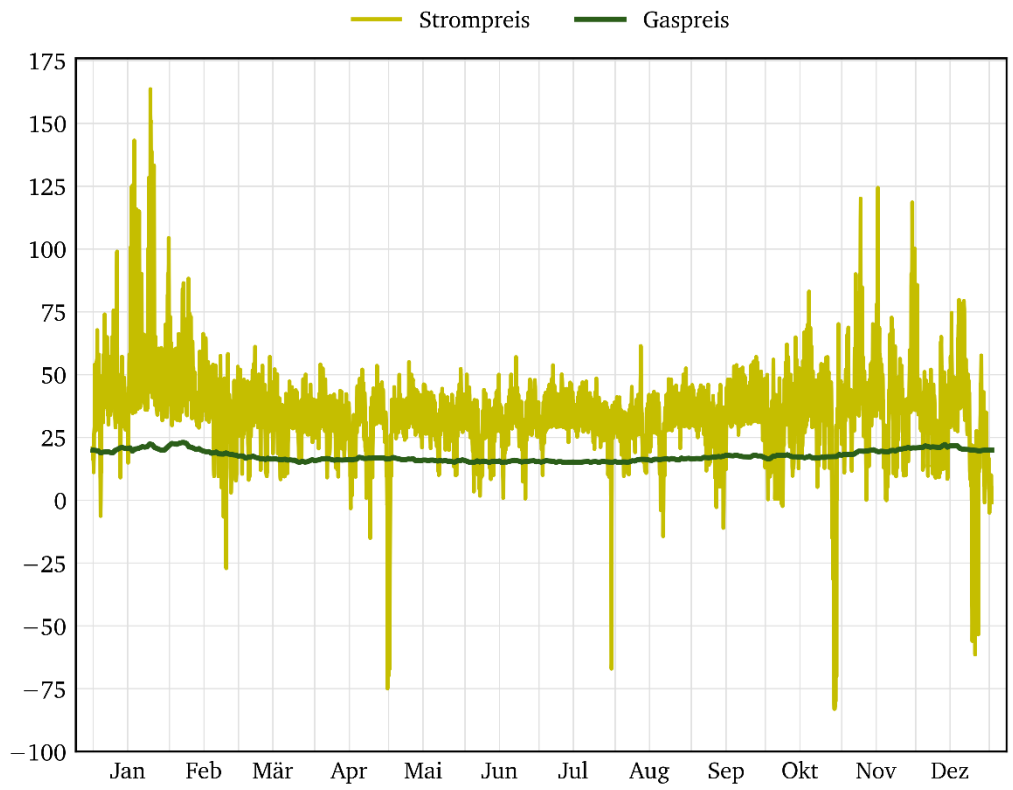
Pendulum-Environment-Example (Structure and Domain randomization)

Implementation

Step 5: Build a Digital Twin of the supply systems and export it as an FMU

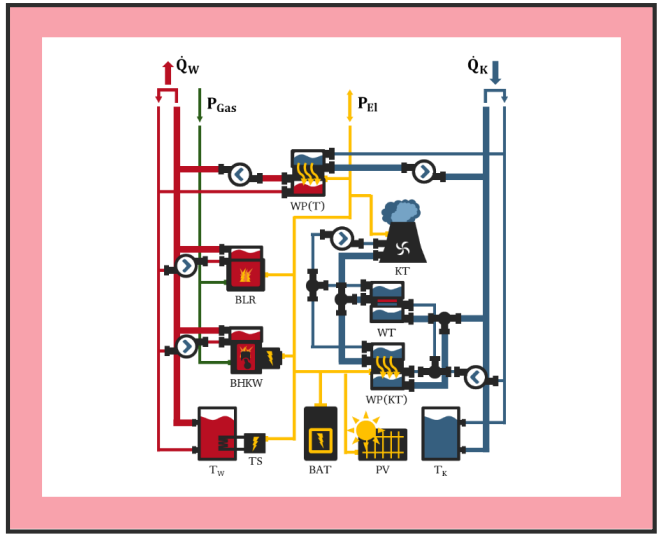




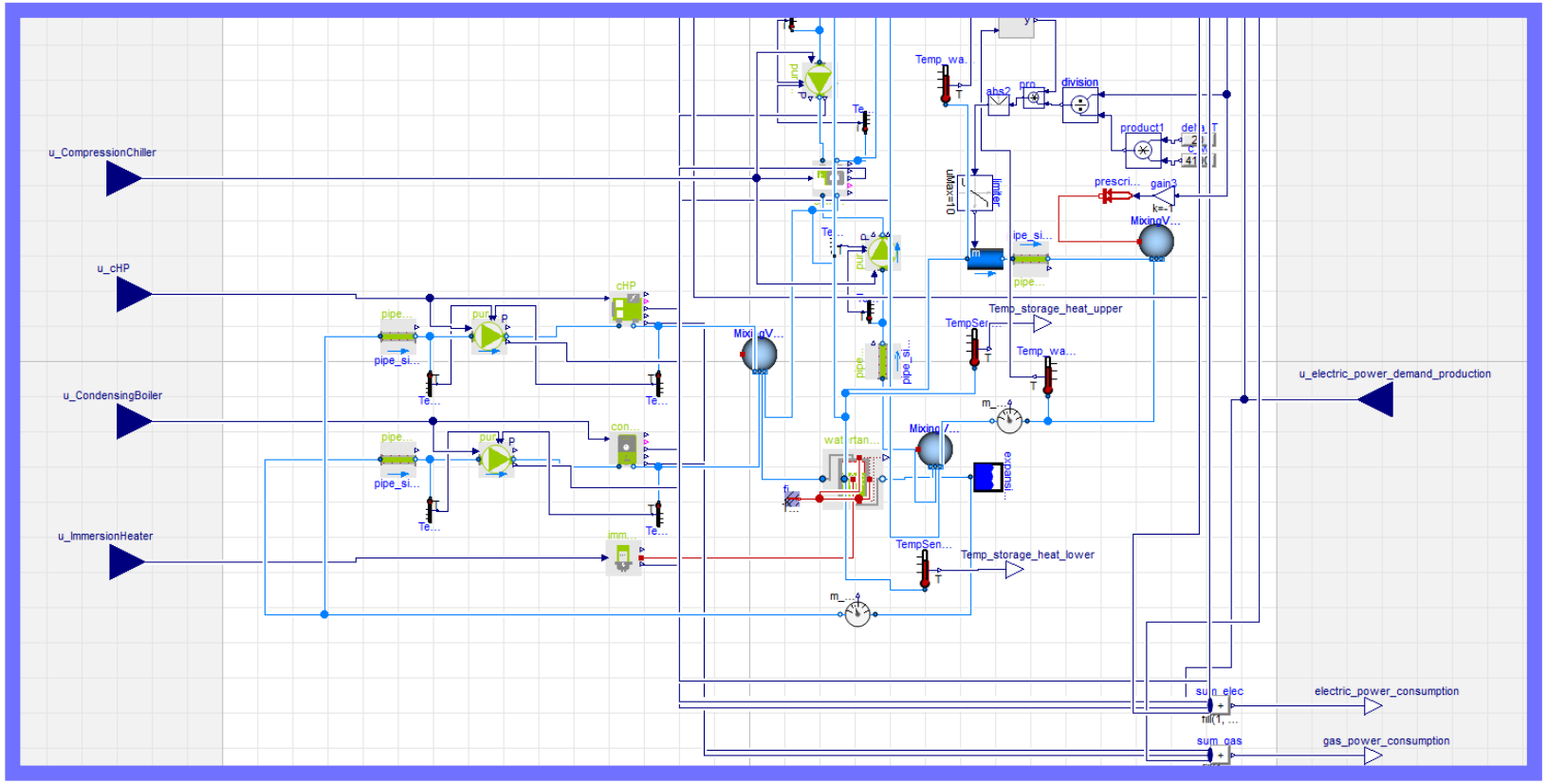


Implementation

Step 5: Build a Digital Twin of the supply systems and export it as an FMU

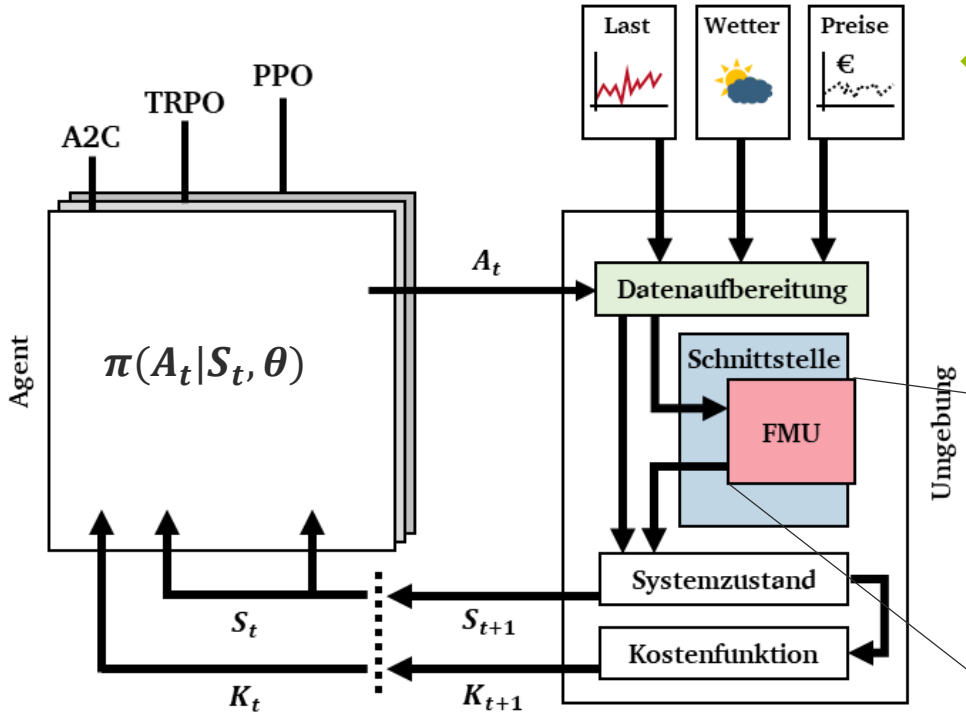


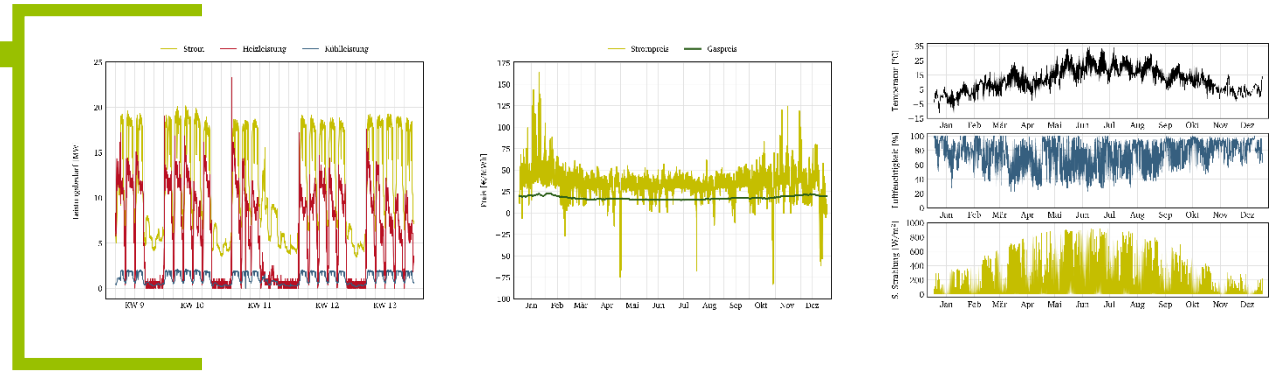
Dymola

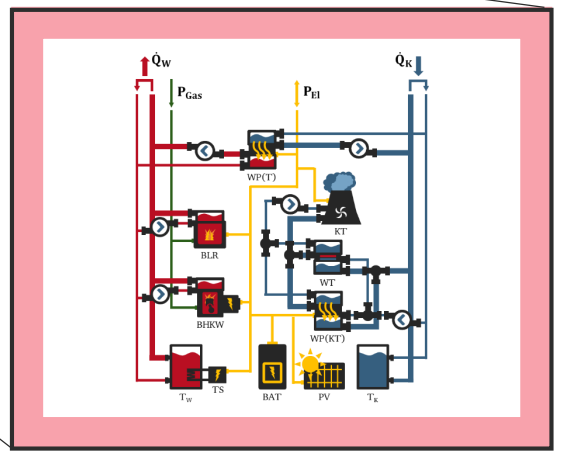


Implementation

Step 6: Create advanced data-preparation module

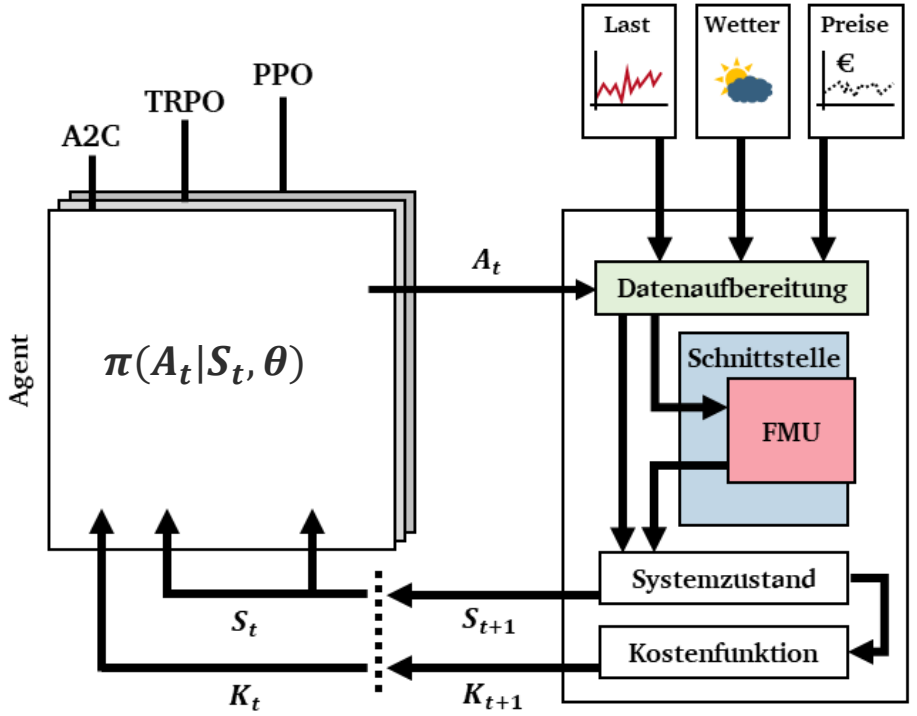


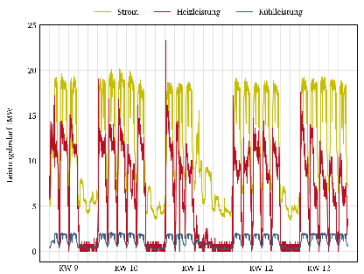


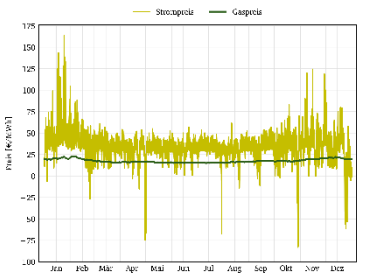


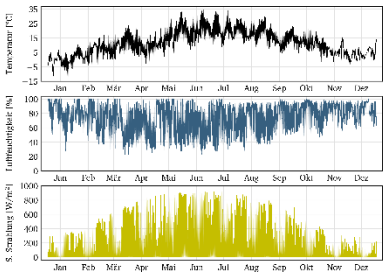
Implementation

Step 7: Create cost function





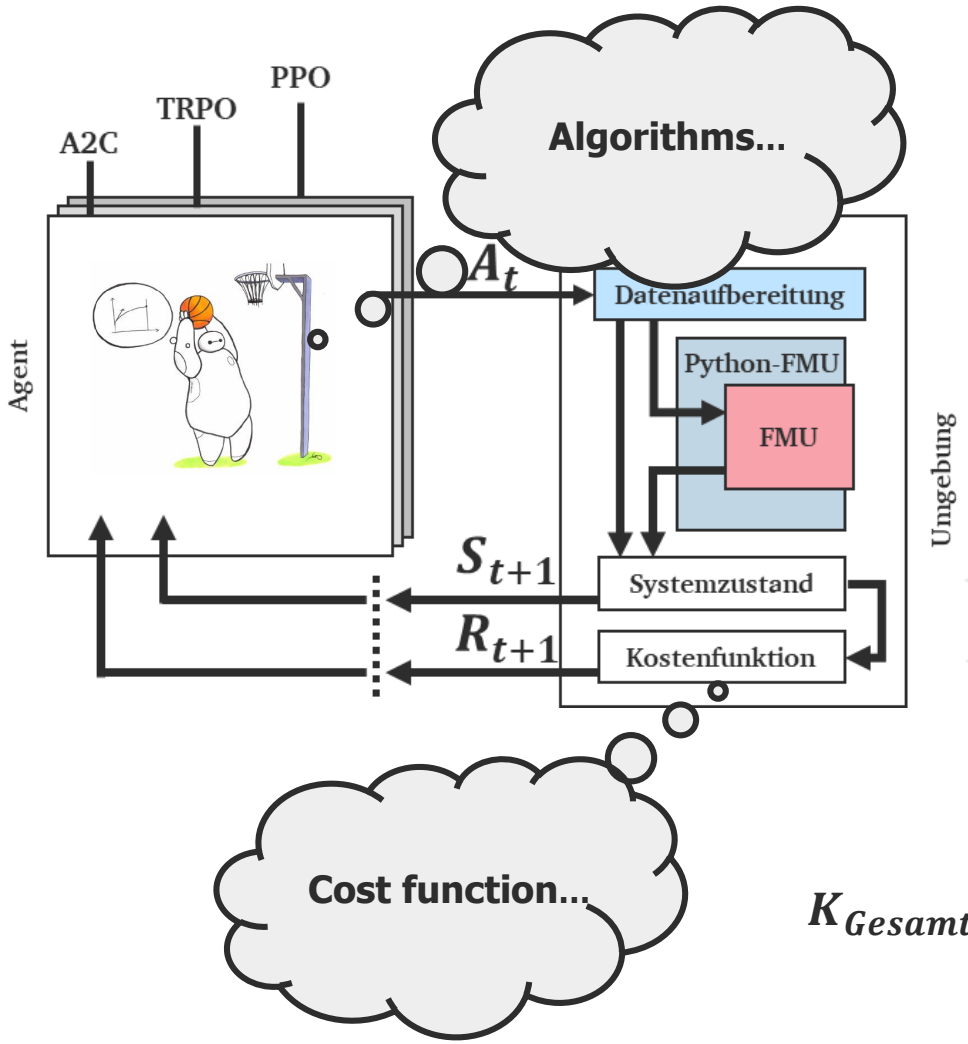


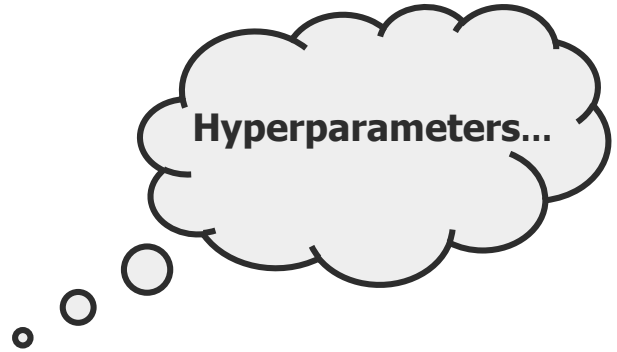


$$K_{Gesamt} = g_T K_T + g_E K_E + g_A K_A$$

Implementation

Influences on the performance



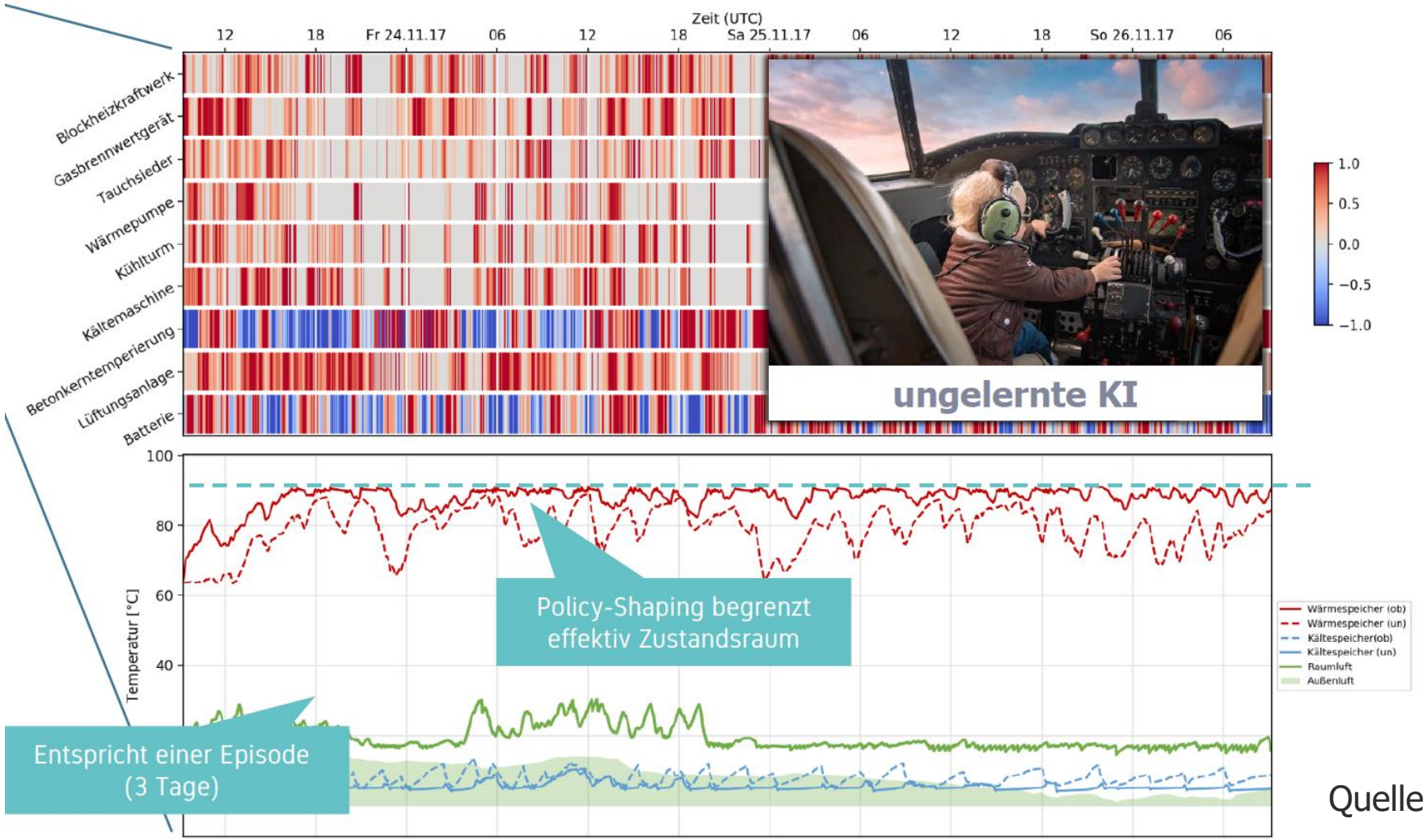


- Choosing the right **observations** for the **state** is crucial!
- Choosing the right **cost function** is crucial!

$$K_{Gesamt} = g_T K_T + g_E K_E + g_A K_A$$

Implementation

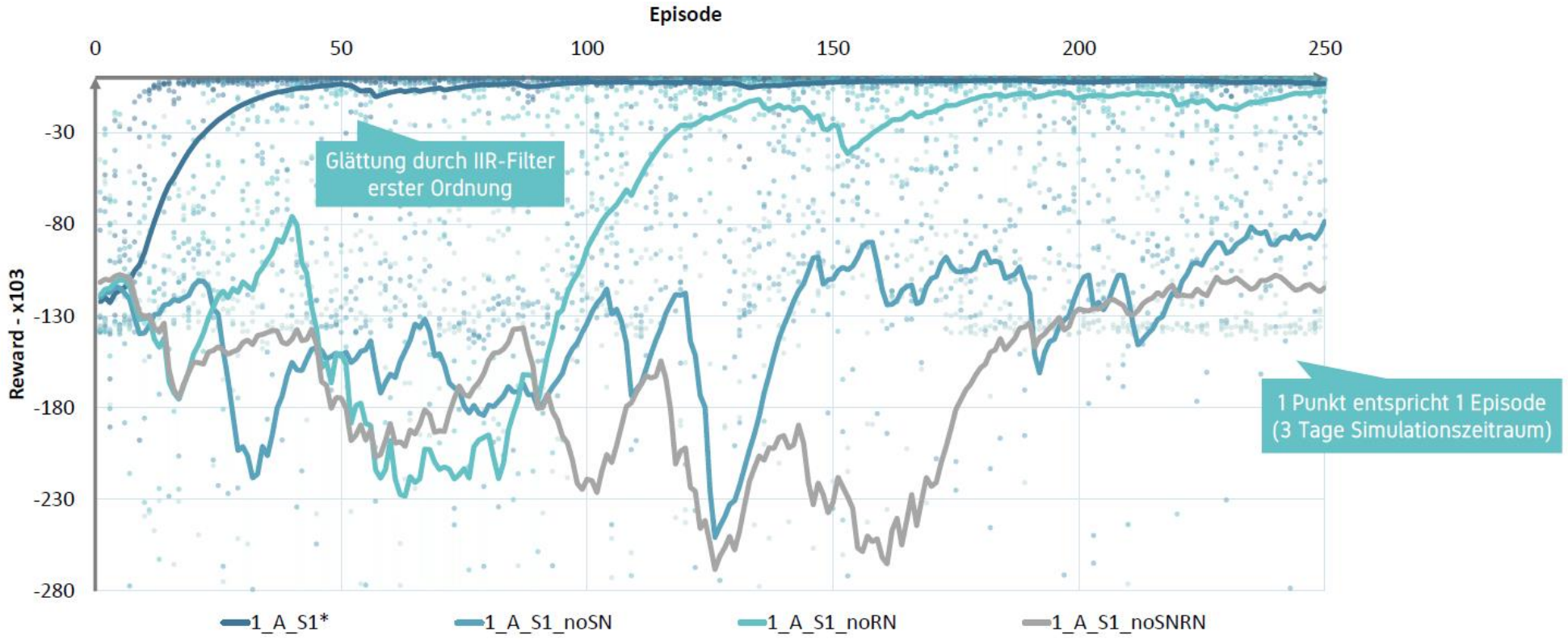
Step 8: Hyperparameter-tuning and result generation



Quelle: Panten, N: Disputation

Implementation

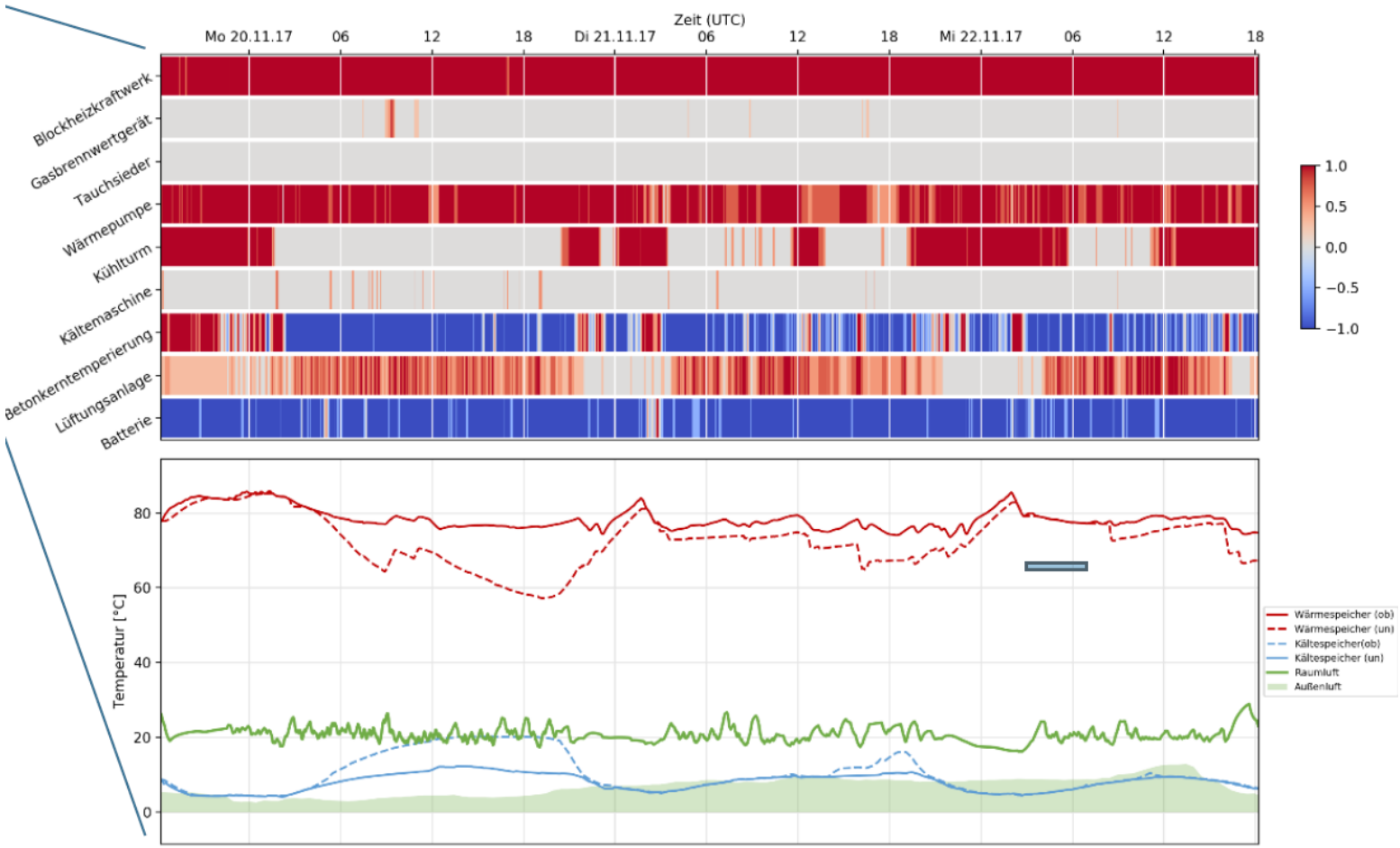
Step 8: Hyperparameter-tuning and result generation



Quelle: Panten, N: Disputation

Implementation

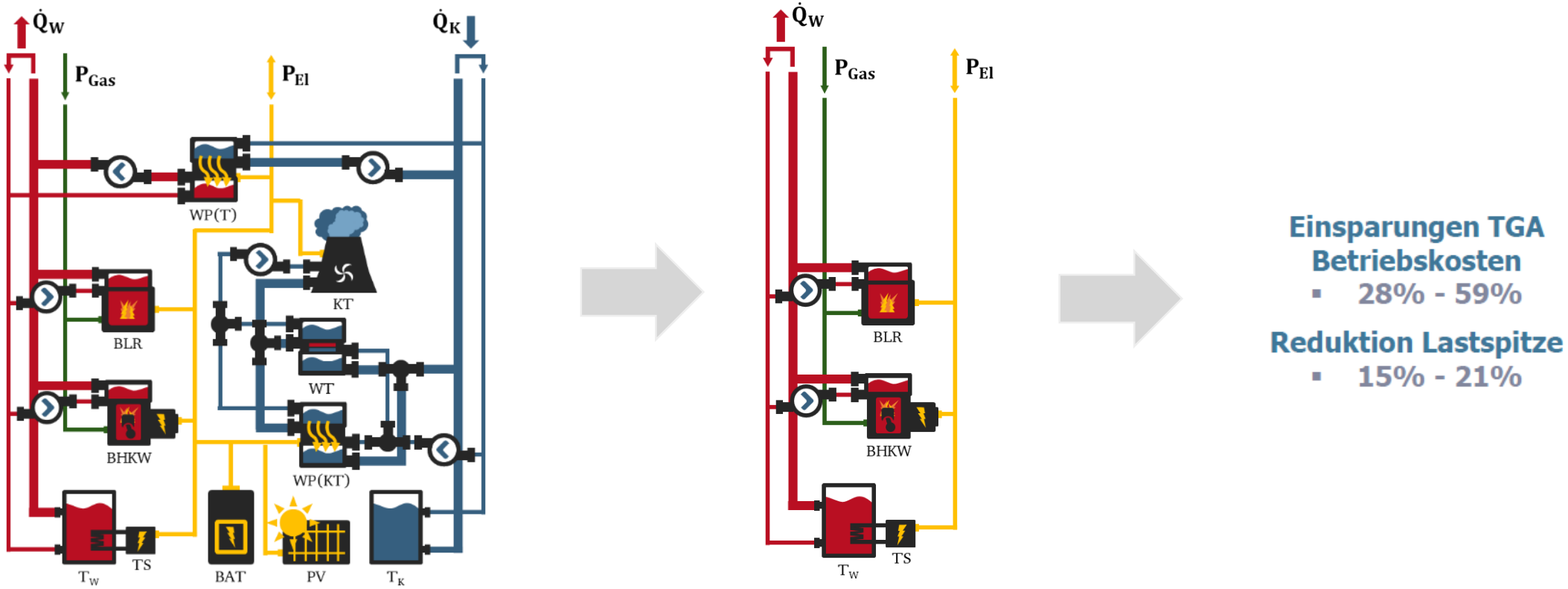
Step 8: Hyperparameter-tuning and result generation



Quelle: Panten, N: Disputation

Implementation

Step 8: Hyperparameter-tuning and result generation



Einsparungen TGA Betriebskosten

- 28% - 59%

Reduktion Lastspitze

- 15% - 21%

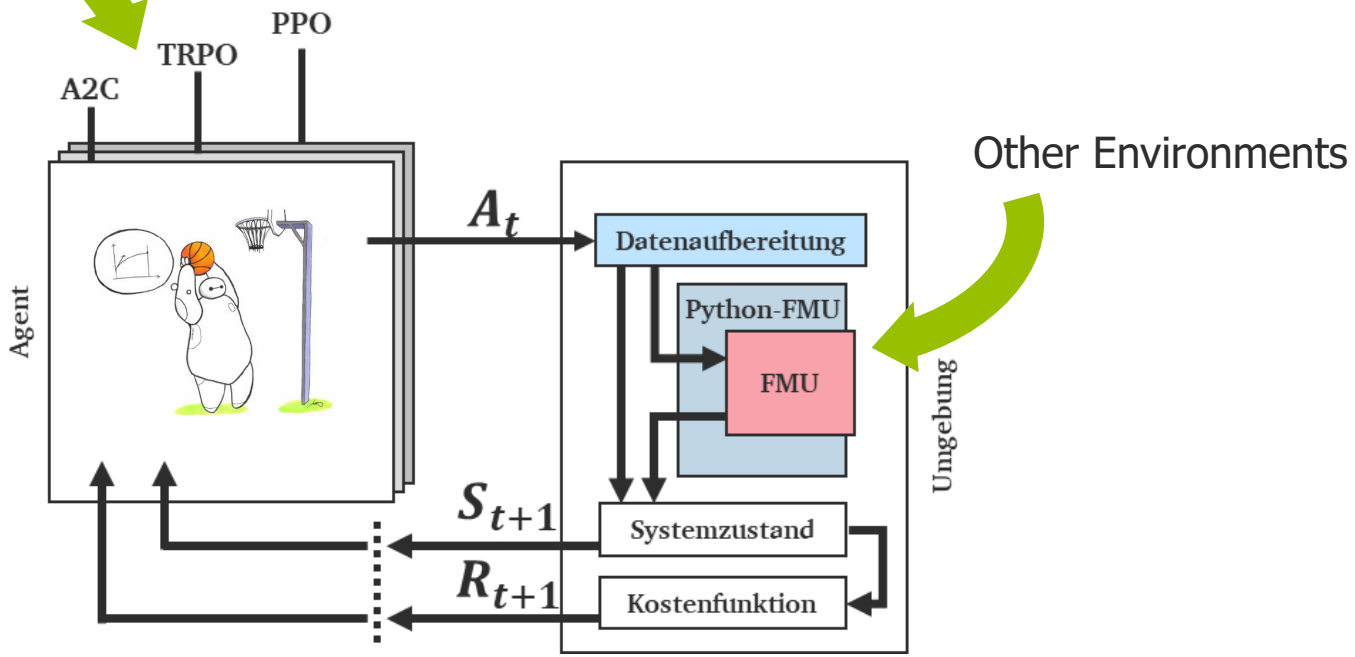
Quelle: Panten, N: Disputation

Implementation

ETAi Framework

MPC (MILP)

Conventional Controllers



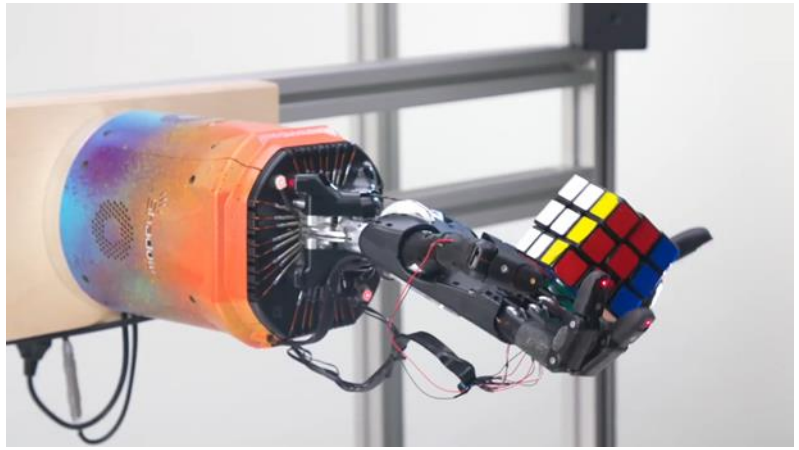
Other Environments

Agenda

Insights and future research

Future research: How do we get from simulated environments to real environments?

Simulated environments

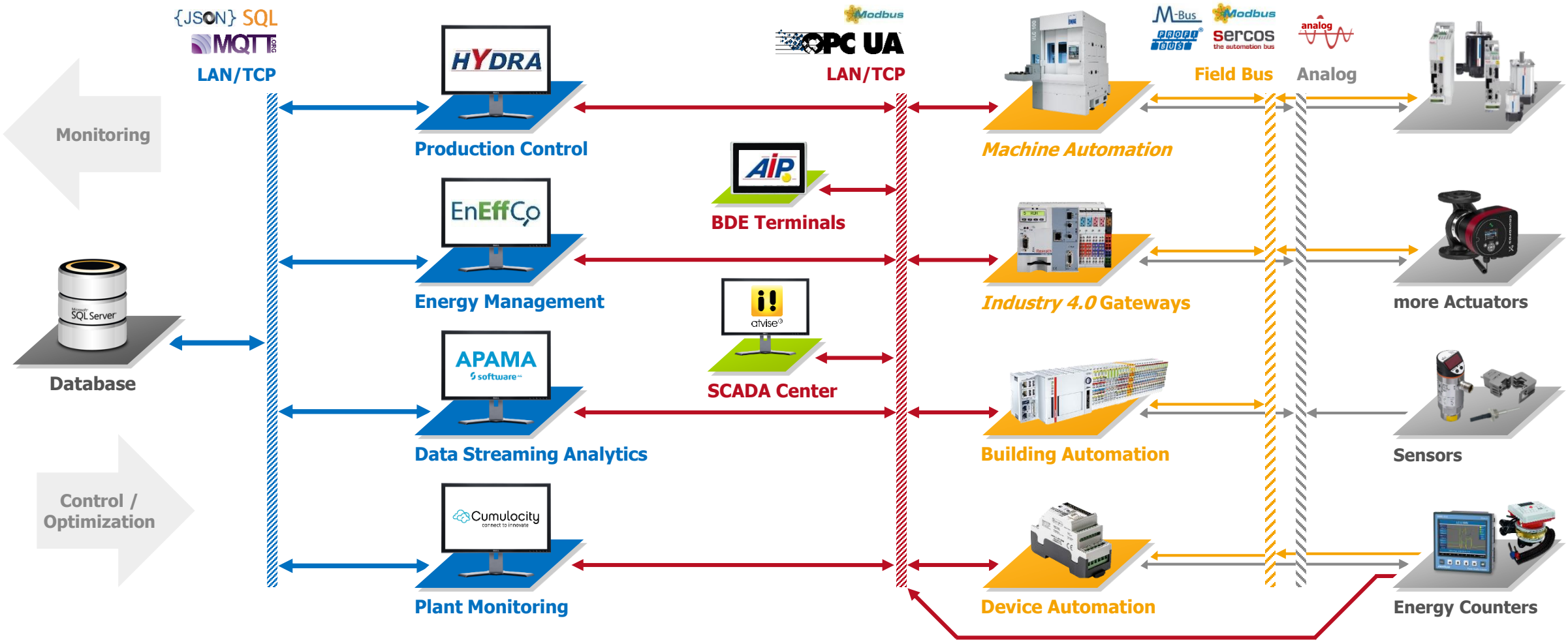


Real environments

- **Automatic Domain randomization (ADR)**
- **Direct Agent-Machine Interaction (OPC-UA)**

Insights and future research

Future research: How do we get from simulated environments to real environments?



Insights and future research

Future research: How do we get from simulated environments to real environments?

Research fields: Energy efficiency// Energy flexibility// Resource efficiency

Software oriented

Artificial Intelligence for energy systems

- optimum, adaptive control
- optimal scheduling
- energy assistance systems
- pattern recognition and classification
- autonomous measuring systems

interaction

Simulation of energy systems

- object-oriented modeling
- system identification
- digital twin
- virtual commissioning
- digital planning tools

interaction

Energy management & -monitoring

- energy indicators
- measuring systems
- ICT & data acquisition concepts
- condition & quality monitoring
- prognosis systems

transfer // teaching

decision making support | operating strategy | method pool for analysis & optimization | data | field knowledge

Hardware oriented

Energy-optimized production machines

machining // cleaning
heat treatment // process engineering

- energy recovery
- project planning for new and existing machines
- cross-sectional technology benchmarks
- energy analyses

interaction

Energy optimized supply systems

cold // heat // compressed air
air conditioning // ventilation

- energy recovery, storage & grids
- topologies & dimensioning
- technology benchmarks
- energy system analyses

Insights and future research

Künstliche Intelligenz: EU-Parlament will nachvollziehbare Algorithmen

Die Abgeordneten verlangen eine stärkere Kontrolle über automatisierte Entscheidungsfindungen mit voller Nachprüfbarkeit von KI-Systemen.

Lesezeit: 1 Min. In Pocket speichern 39

Das EU-Parlament will, dass maschinell getroffene Entscheidungen transparent, kontrollier- und revidierbar sind. (Bild: Tatiana Shepeleva/Shutterstock.com)

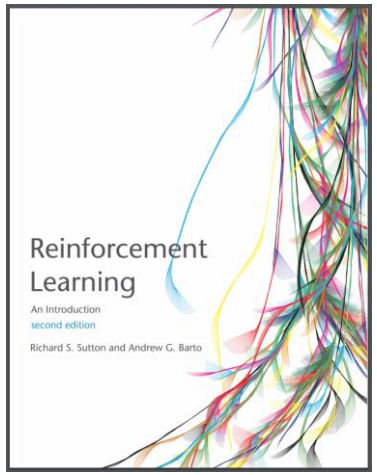
EU-Resolution from the 12.02.2020

„Die Systeme sollten nur […] „**nachvollziehbare und tendenzfreie Algorithmen**" verwenden […]. Es müssen **Kontrollmechanismen** eingerichtet werden, um mögliche Fehler automatisierter Entscheidungen korrigieren zu können. […] Das Parlament fordert ein **Risikobewertungsschema** für KI und automatisierte Entscheidungsfindung."

Quelle: https://www.europarl.europa.eu/news/de/press-room/20200206IPR72015/kunstliche-intelligenz-parlament-will-faire-und-sichere-nutzung-fur-verbraucher

- **Explainable AI**
- **Performance KPIs**
- **Fallback Mechanisms**

More information



Sutton, R., Barto, A. (2018): **Reinforcement Learning. An Introduction.** Second Edition

Free PDF available online

Panten, N. (2019): **Deep Reinforcement Learning zur Betriebsoptimierung hybrider industrieller Energienetze.**

PDF available: 13 €

- **OpenAI website:** https://openai.com/
- **YouTube Channel TwoMinutePapers:** https://www.youtube.com/user/keeroyz
- **Stable-Baselines:** https://stable-baselines.readthedocs.io/en/master/
- **DL&RL Course (DeepMind):** https://www.youtube.com/playlist?list=PLqYmG7hTraZDNJre23vqCGIVpfZ_K2RZs

Objectives

After today you should know:

Potentials and applications of Deep Reinforcement Learning

Basics of **Reinforcement Learning** (RL) and **Deep Reinforcement Learning** (DRL)

Influencing factors on the **performance** of DRL Algorithms

Tools that enable **interaction** between **current DRL algorithms** and **powerful simulations**

Basics of industrial **supply systems**

An **application case** at the **ETA-Factory** (control of supply systems)

Vielen Dank für Ihre Aufmerksamkeit!

Bei Fragen stehen wir Ihnen gerne zur Verfügung.

Prof. Dr.-Ing. Eberhard Abele
Prof. Dr.-Ing. Joachim Metternich
Prof. Dr.-Ing. Matthias Weigold

Institut für Produktionsmanagement, Technologie und Werkzeugmaschinen
Technische Universität Darmstadt

Otto-Berndt-Straße 2
64287 Darmstadt

Tel.: +49 61 51 | 16 2 00 80
Fax: +49 61 51 | 16 2 00 87
E-Mail: info@ptw.tu-darmstadt.de
Internet: www.ptw.tu-darmstadt.de

DRL Basics

Q-Learning

Let's start with RL before we move to DRL. What does Q stand for?

Immediate Reward Future Reward

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T,$$

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \end{aligned}$$

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+3} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+3} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

$$q_* (s, a) = E \left[R_{t+1} + \gamma \max_{a'} q_* \left(s', a' \right) \right]$$

$$q^{new} (s, a) = (1 - \alpha) \underbrace{q (s, a)}_{\text{old value}} + \alpha \overbrace{\left(R_{t+1} + \gamma \max_{a'} q \left(s', a' \right) \right)}^{\text{learned value}}$$