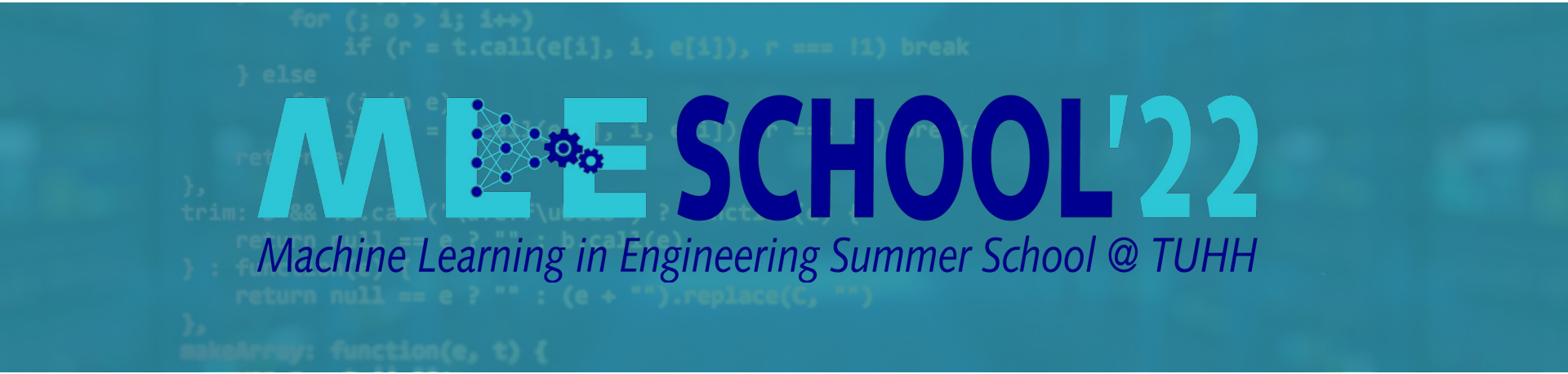
TUHH
Hamburg University of Technology

Machine Learning in Engineering Summer School @ TUHH SCHOOL'22



Reinforcement Learning for the Optimization of Particle Accelerators

Prof. Dr. Annika Eichler & Jan Kaiser

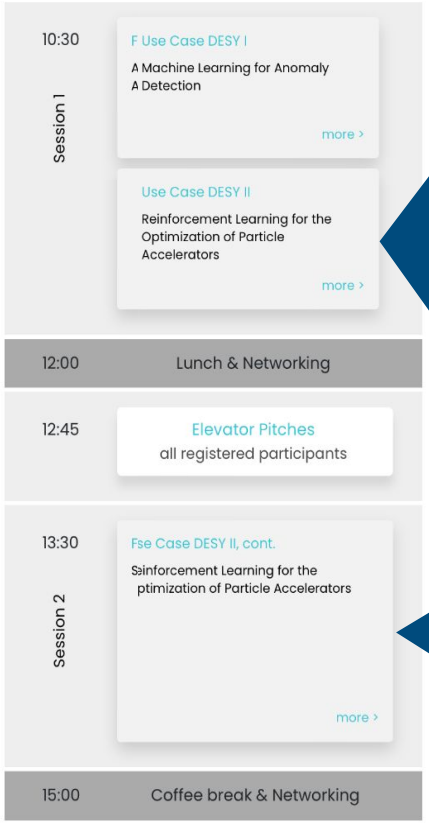
Introduction of Speakers

Prof. Dr. Annika Eichler
DESY Accelerator Beam Controls
TUHH Institute of Control Systems
annika.eichler@desy.de

Jan Kaiser
DESY Accelerator Beam Controls
jan.kaiser@desy.de

Reinforcement Learning for Particle Accelerators

Session Schedule



- Introduction to Reinforcement Learning
- Software Overview
- Lunar Lander Example - An Introduction to OpenAI Gym and Stable Baselines3
- Some accelerator basics
- Why we need RL for particle accelerators
- Practical example of RL on the ARES accelerator

- Hands-On with ARES example
  Requirement: Laptop with Anaconda installed

Reinforcement Learning for Particle Accelerators

Format

Code & Slides on GitHub.

https://github.com/desy-ml/mle-school-2022-use-case-desy-ii

Let’s keep this interactive.

How much do you know already?

Raise your hands

Machine Learning?

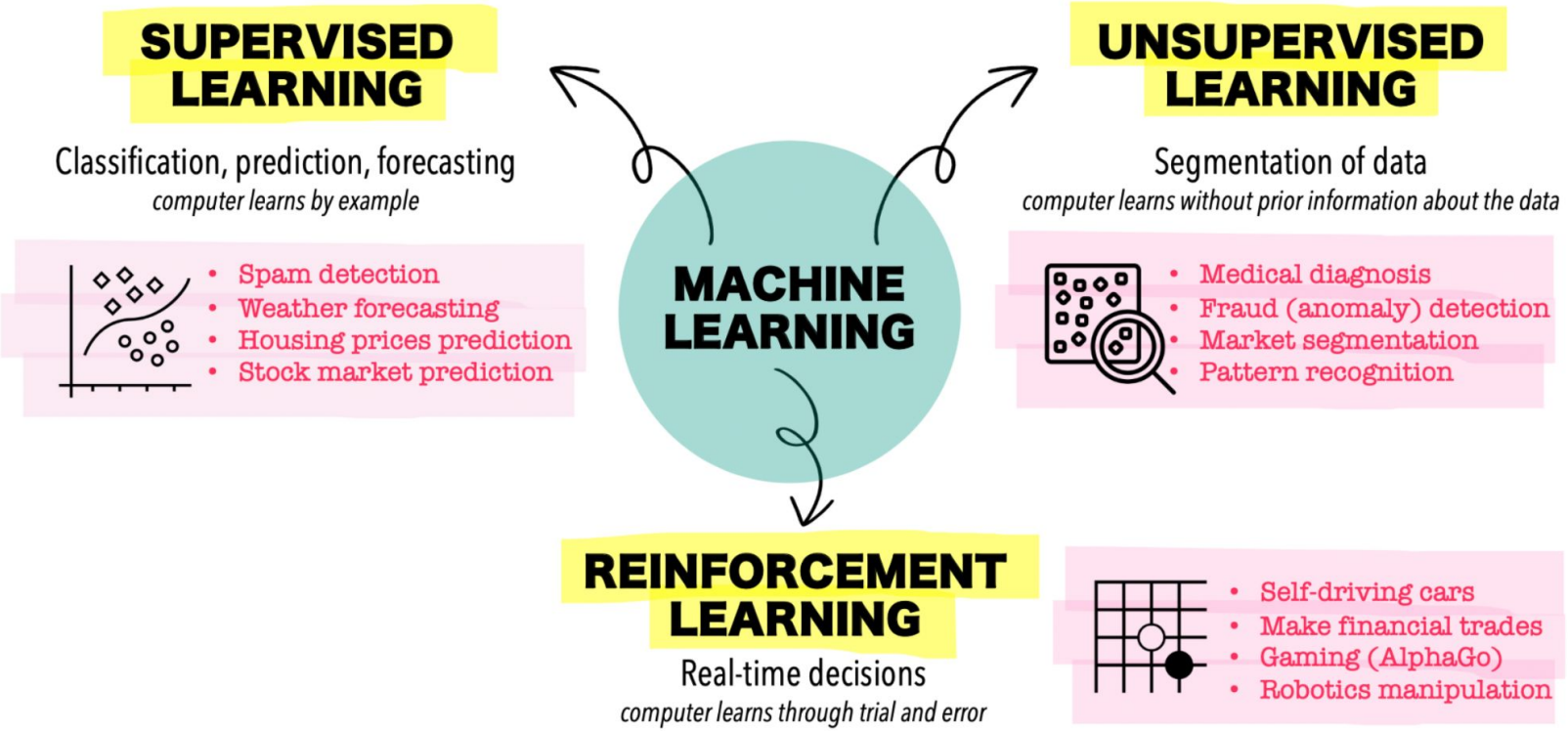
Reinforcement Learning?

Particle Accelerators?

Introduction to Reinforcement Learning

A Taxonomy of Machine Learning

Where does reinforcement learning fit in?

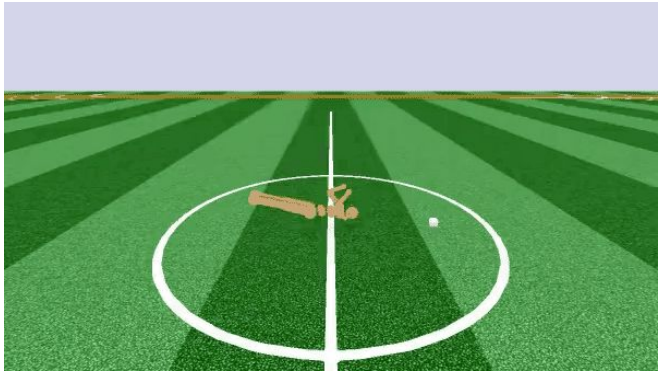


Courtesy Andrea Santamaria Garcia

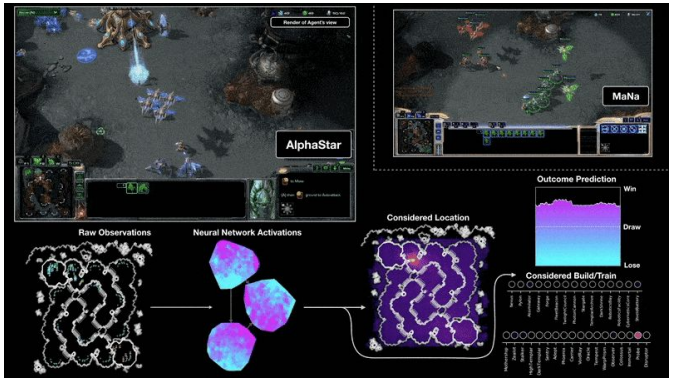
What can RL do?

Some examples

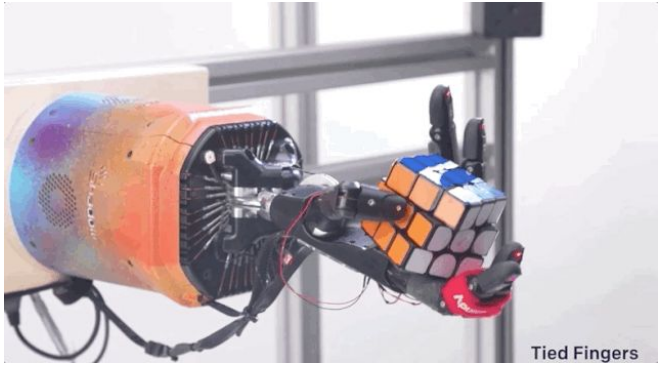
Control humanoid in simulation



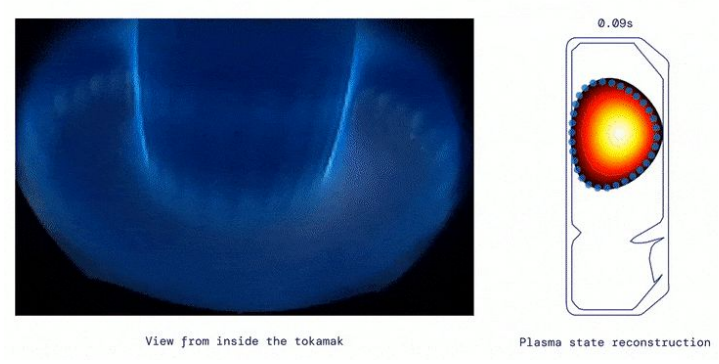
Play games with imperfect information and develop long-term strategies



Control robot hands in the real world

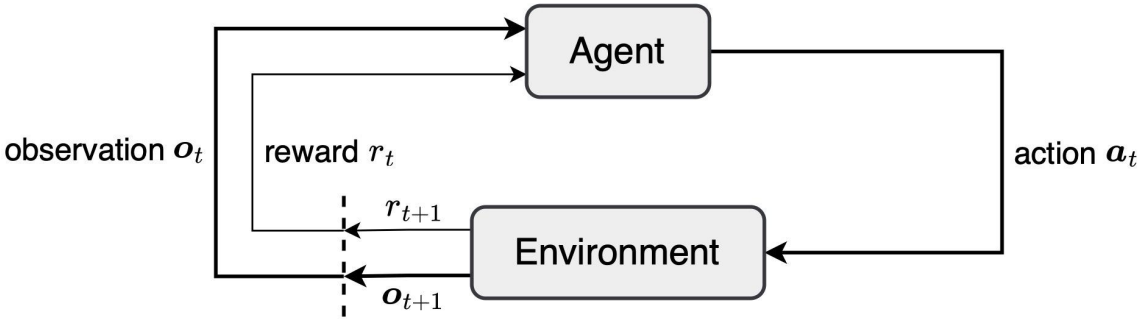


Control the plasma in a tokamak fusion reactor



Concepts of Reinforcement Learning

A high-level overview



- The **agent** (or **policy**) is the function we are trying to learn and tells us what to do to solve the task.
- The **environment** is the world that the RL agent lives in and defines the task.
- **Actions** are how the RL agent interacts with the environment.
- **Observations** are what the agent sees of the environment.
- The **reward** is returned by the environment after each action and describes the goodness of that action.
- The **return** is the cumulative reward over time. The goal of RL is to maximise the return.

Reinforcement Learning Algorithms

Taxonomy of RL

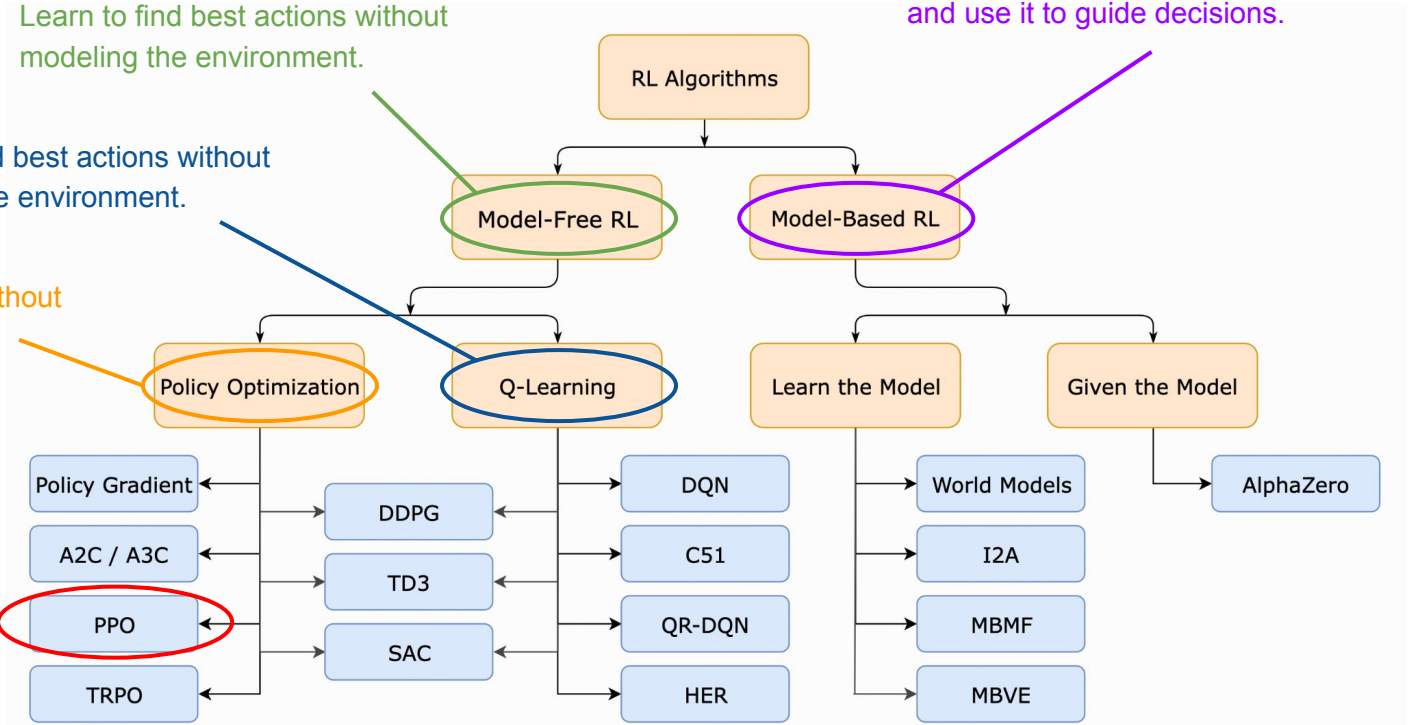
Use existing model or learn a model and use it to guide decisions.

Learn to find best actions without modeling the environment.

Learn to find best actions without modeling the environment.

Learn to find best actions without modeling the environment.

Proximal Policy Optimisation



Software for Reinforcement Learning

A brief overview

Algorithms

Chainer RL

OpenAI Baselines

Keras RL2

TensorForce Library

RLlib

Stable Baselines3

Environments

Gym

Google DeepMind

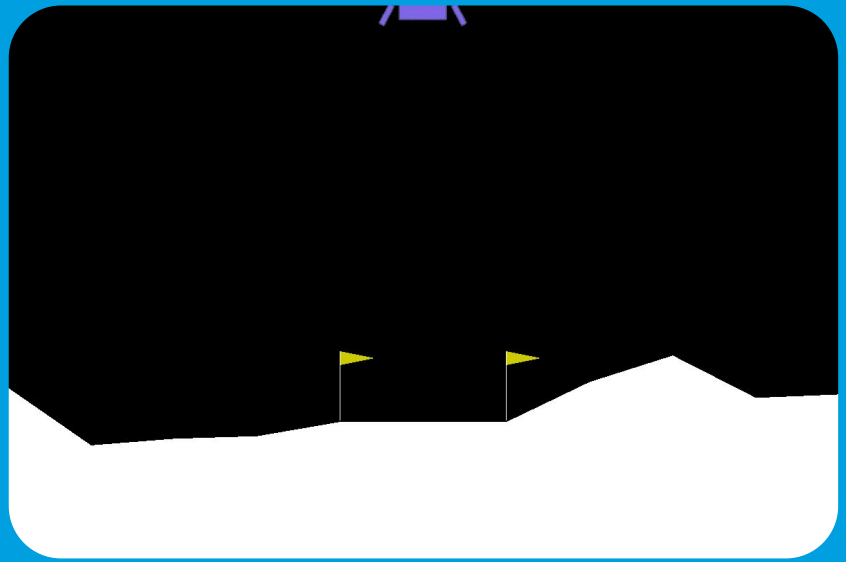
DeepMind Control Suite

MuJoCo

Advanced physics simulation

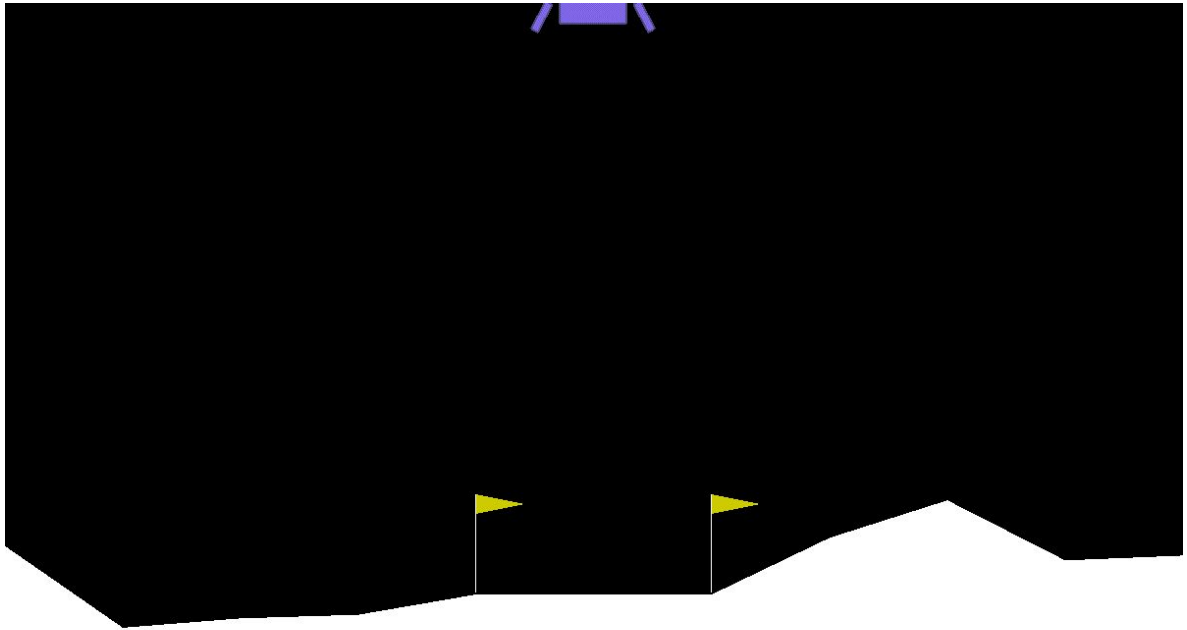
python™

Lunar Lander Example



Lunar Lander Example

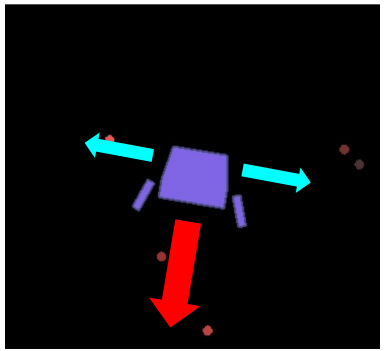
Introduction



Lunar Lander Example

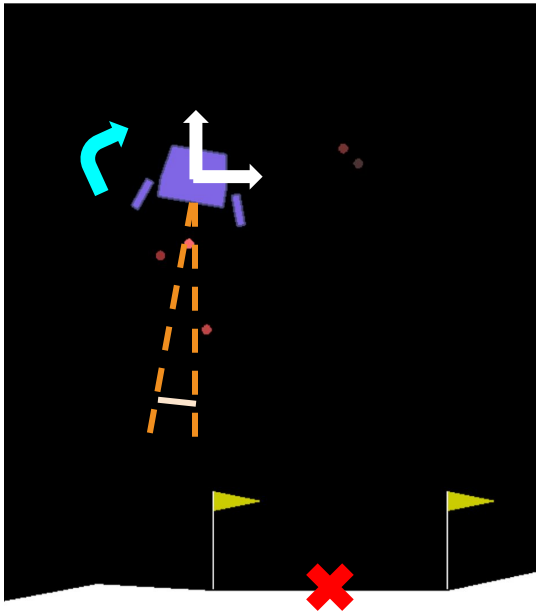
Actions and observations

Action



- Main engine
- Left side thruster
- Right side thruster

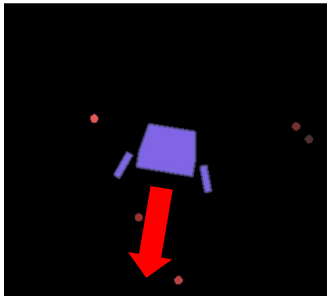
Observation



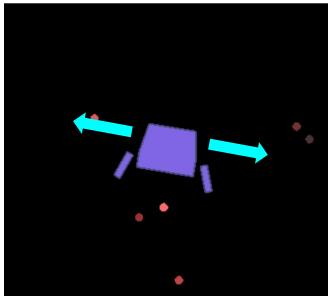
- Horizontal distance to target
- Vertical distance to target
- Horizontal velocity
- Vertical velocity
- Rotation
- Angular velocity
- Left leg touchdown
- Right leg touchdown

Lunar Lander Example

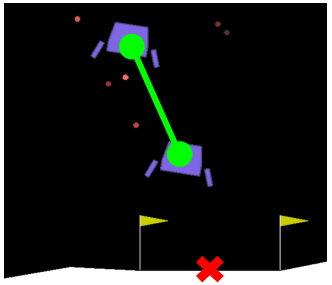
Rewards



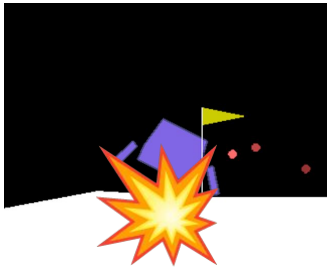
-0.3



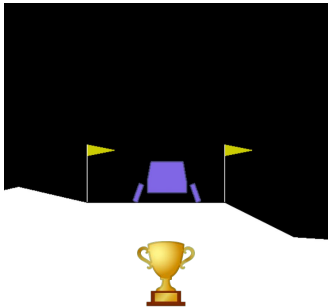
-0.03



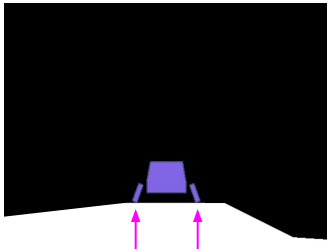
+/- d



-100



+200



+10 each

Lunar Lander Example

Running the lunar lander enviornment

```python
import gym

env = gym.make("LunarLander-v2")
done = False
observation = env.reset()
while not done:
   action = policy(observation)  # User-defined policy function
   observation, reward, done, info = env.step(action)
env.close()
```

Lunar Lander Example

Training a lunar lander agent and running it

Training in Stable Baselines3

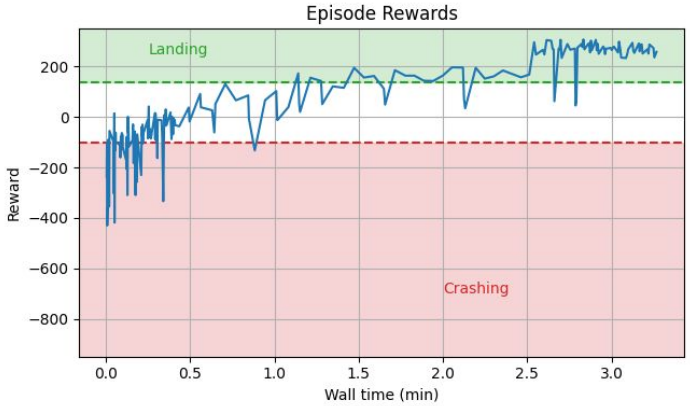
```python
import gym

from stable_baselines3 import PPO

env = gym.make("LunarLander-v2")

model = PPO("MlpPolicy", env, verbose=1)
model.learn(total_timesteps=10000)
```

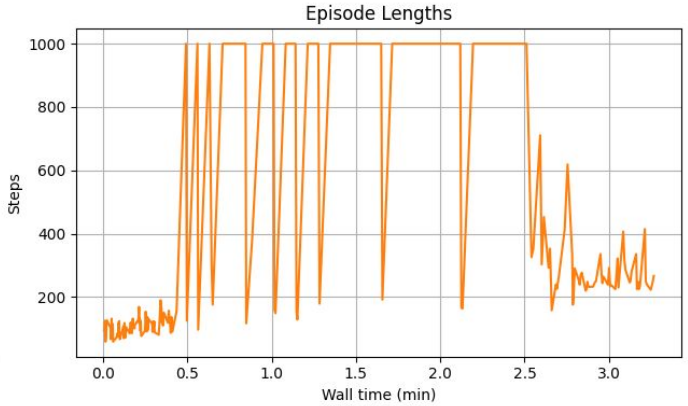
Running the trained agent

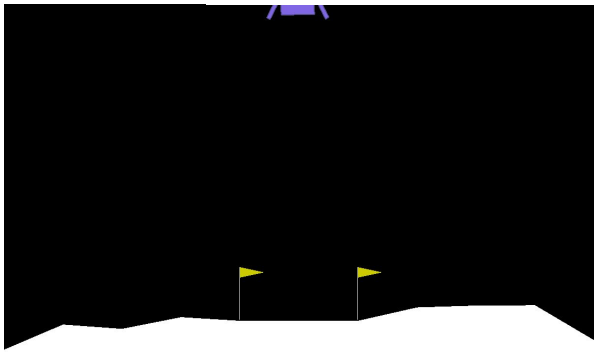
```python
done = False
observation = env.reset()
while not done:
    action, _state = model.predict(obs, deterministic=True)
    observation, reward, done, info = env.step(action)

env.close()
```

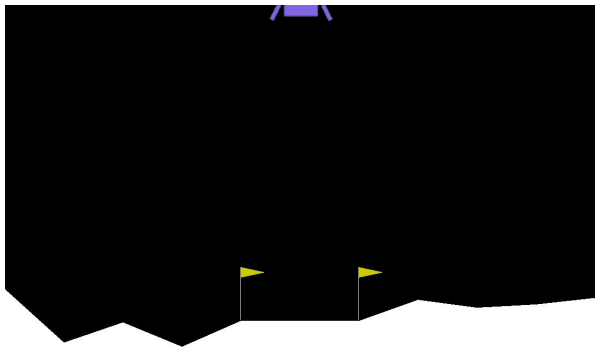
Lunar Lander Example

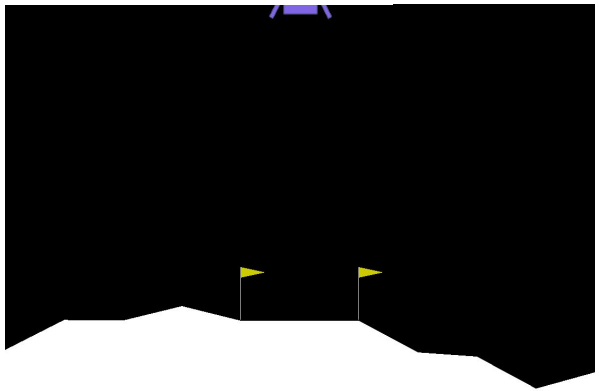
Training results









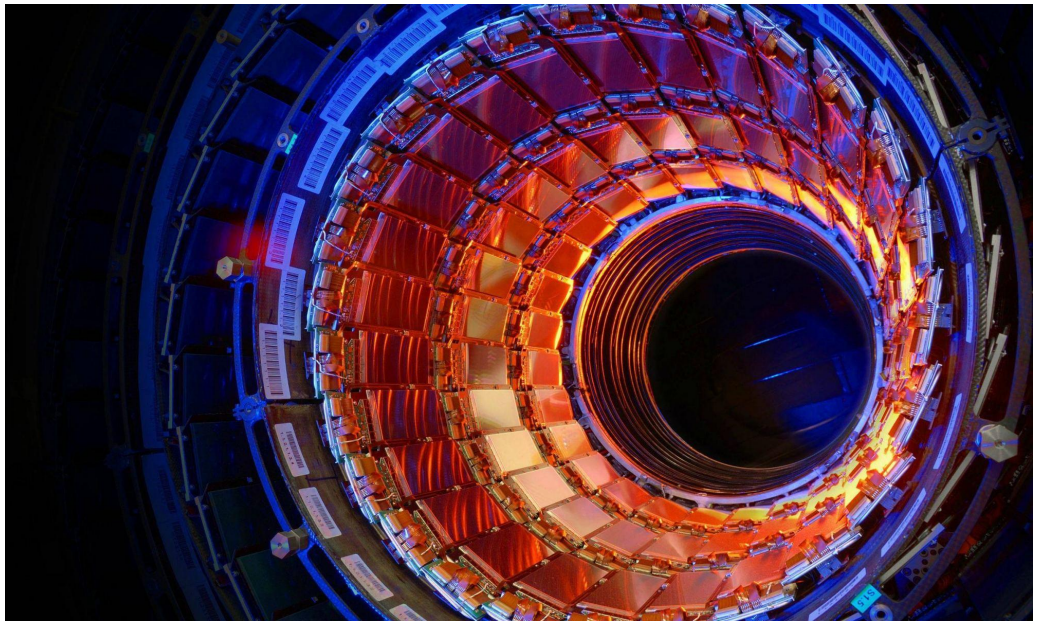


Particle Accelerators

Particle Accelerators

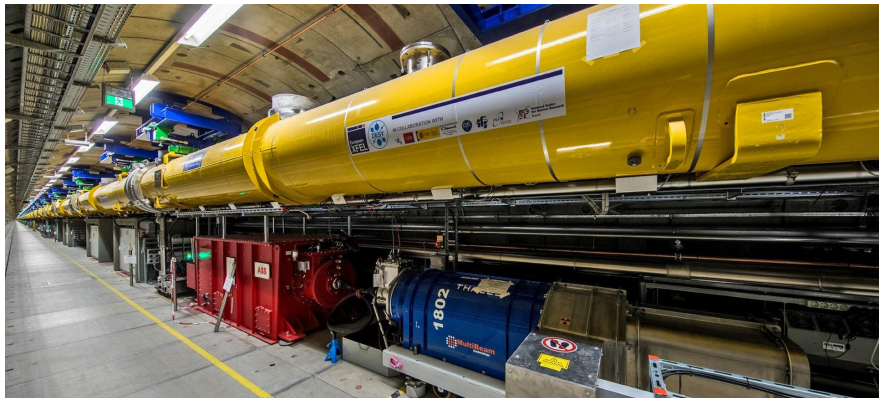
What are they?

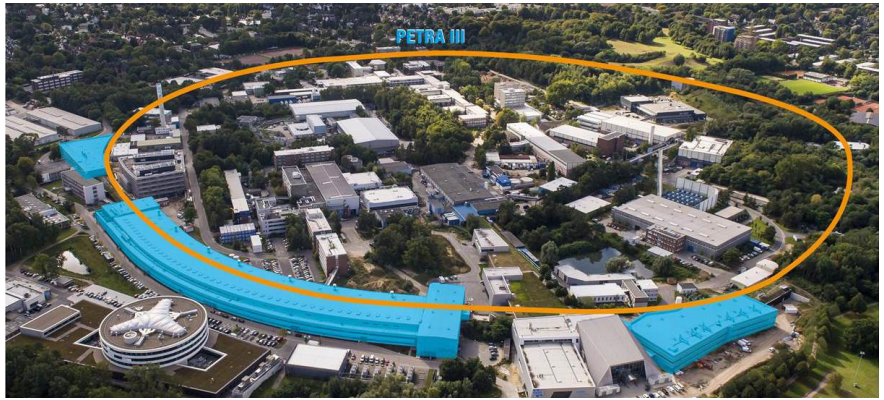
What do you know about particle accelerators?



Particle Accelerators?

Some examples

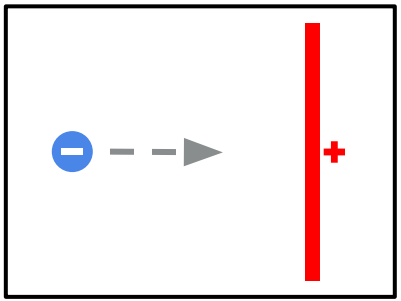




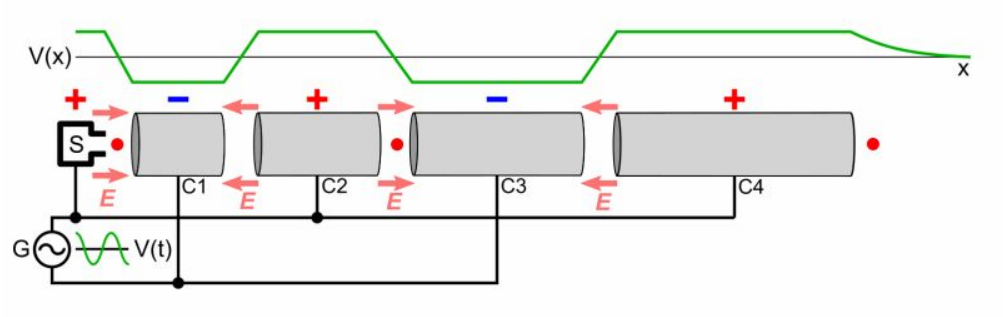
The Charged Particle Beam

What and how do we accelerate

- **Charged particles** such as electrons or protons can be **accelerated using electric fields**.
- Charged particles can also be **manipulated by magnetic fields**.
- Create particles at **particle source**.
  - E.g. electrons using the photoelectric effect by shining a laser on a cathode.

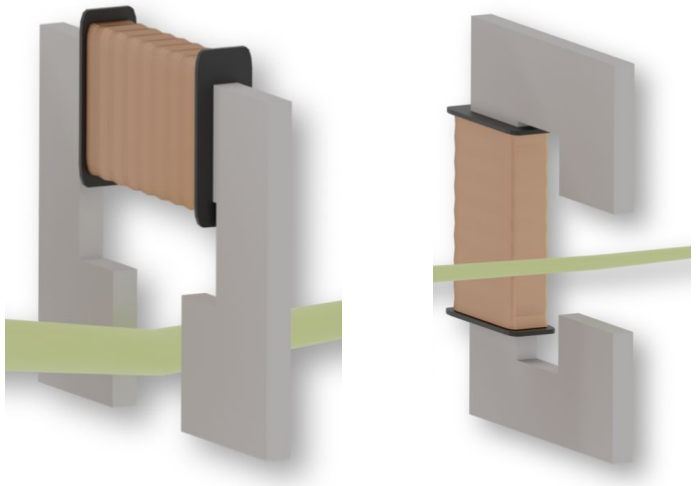


- Place a **positive charge** in front of the electron to accelerate the electron towards it.
- If there is a hole in the charged object, the electron can pass through. If it encounters another charge further ahead and the charge of the previous object switches, we can accelerate further.
- We usually produce **bunches of 10^6 to 10^9 particles**.



Dipole Magnets

Bending the beam

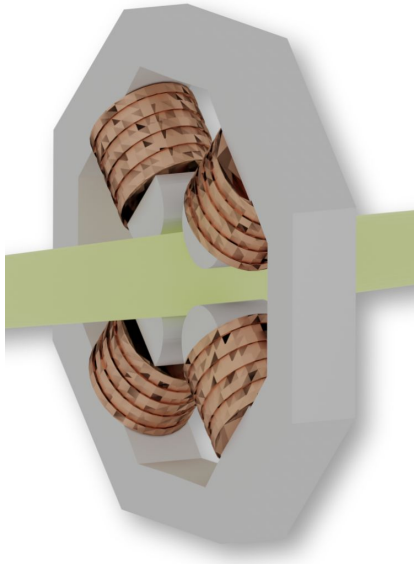


- To **steer the beam** through the beam pipe, we can use magnets that deflect charged particles as a result of **Lorentz force**.
- **Electromagnets** allow for control of deflection angle.

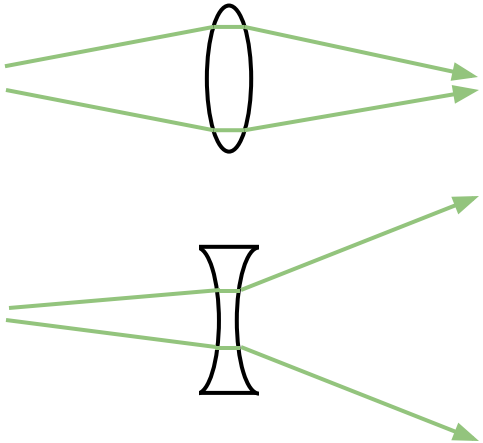
Quadrupole Magnets

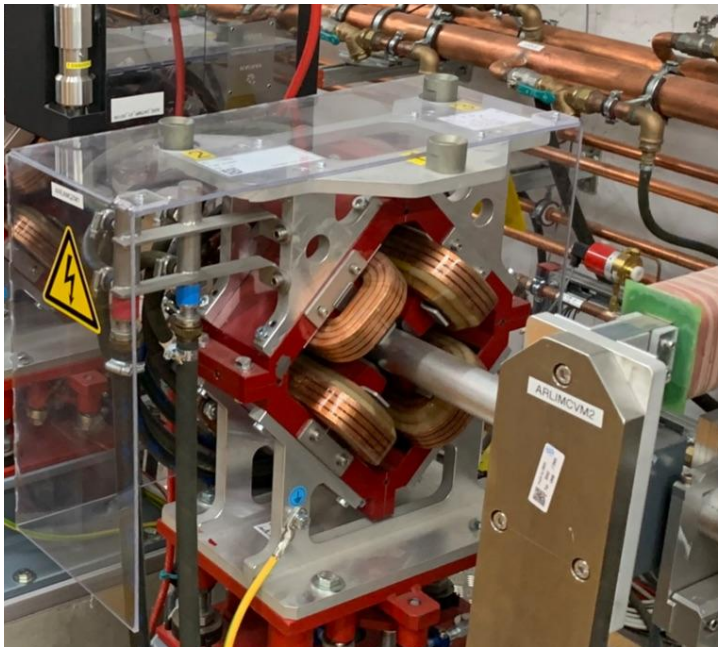
Focusing the beam

Particles of the same charge repel each other. So how can we keep our electrons as one beam?



Lens analogy

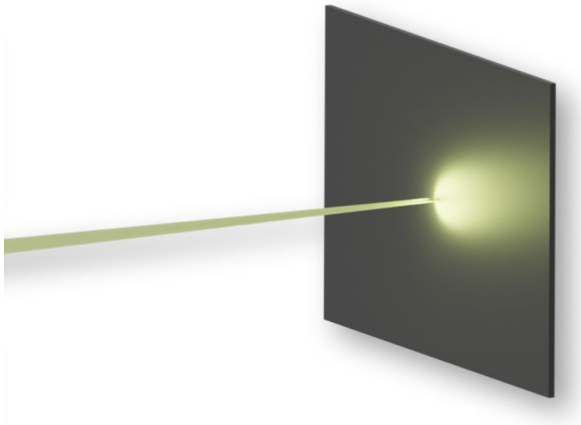


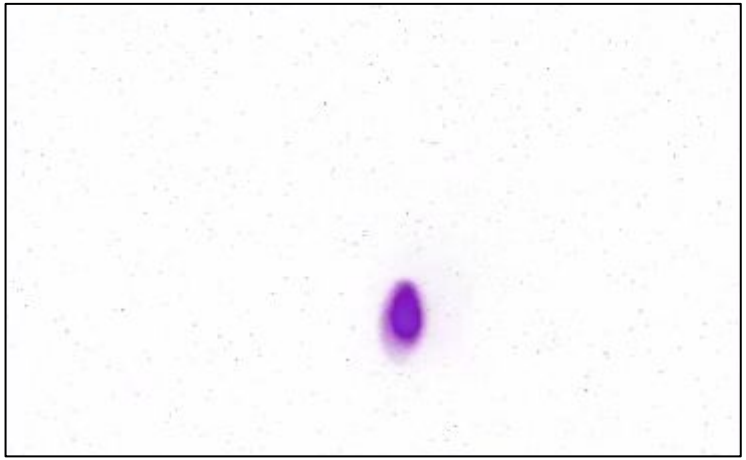


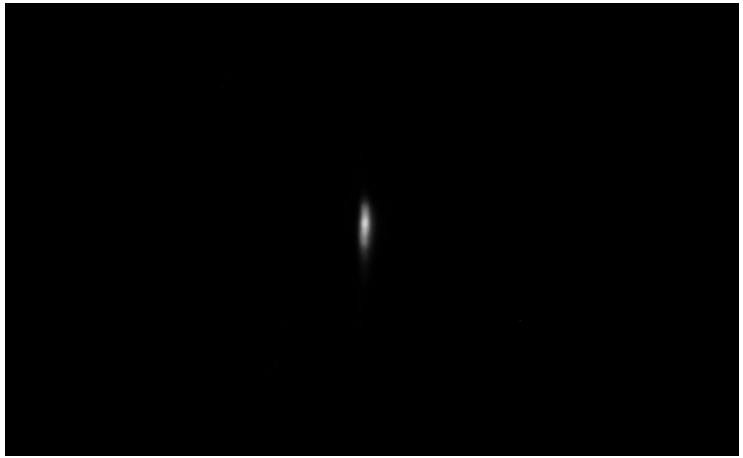
Diagnostic Screens

Seeing the beam

- **Screen from scintillating material** glows when hit by electrons.
- **Camera** films screen.
- Methods from computer vision allow us to **calculate beam parameters** from image.



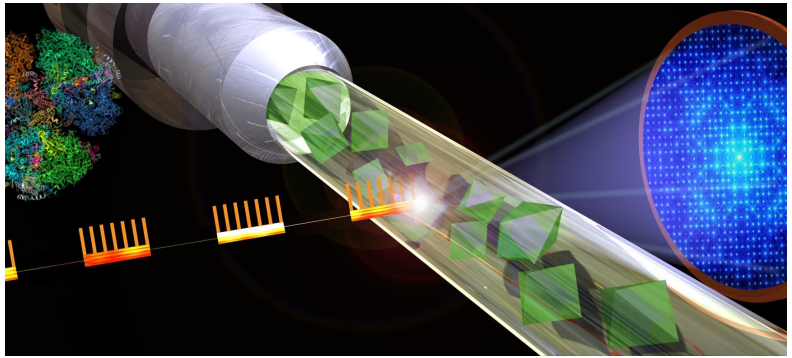




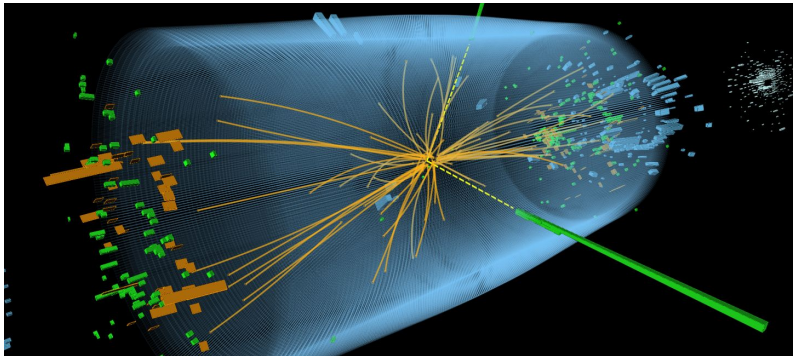
What do we use Particle Accelerators for?

Why do we need them?

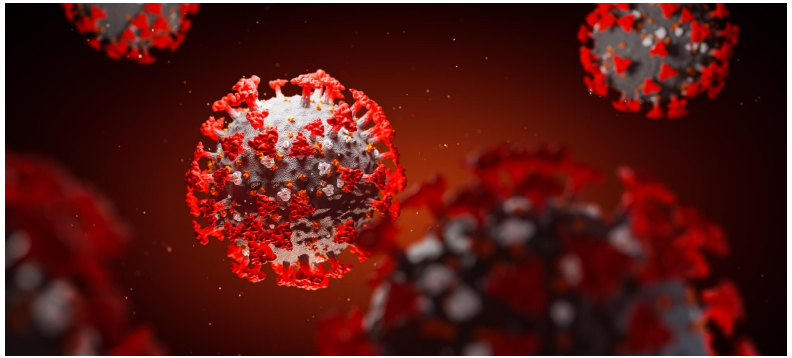
Image and video proteins



Study and discover physics (Higgs boson)



Study viruses (to develop vaccines)

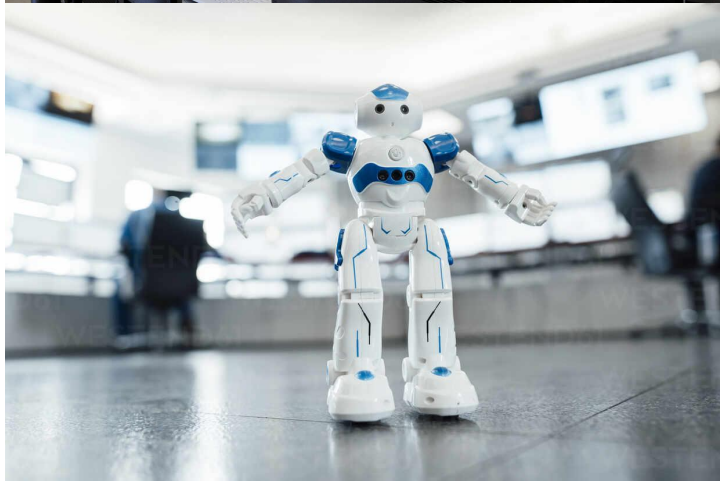


Treat cancer (proton therapy)

Why RL for Accelerators?

The challenges of today

- Large **number of actuators** (e.g. over 2,000 magnets in LHC) and complex **underlying dynamics**.
- Manual tuning requires expert knowledge and experience, and takes a long time.
- Particle accelerators are **expensive to operate** (e.g. ca. XFEL 500,000€ / day), so “wasting” time tuning is very costly.
- Providing more time to experiments **increases scientific output**.
- Some tunes **only attainable by one experienced operator**.
  - Operators had to be recalled from retirement.
- **Reproducibility** of experiments.

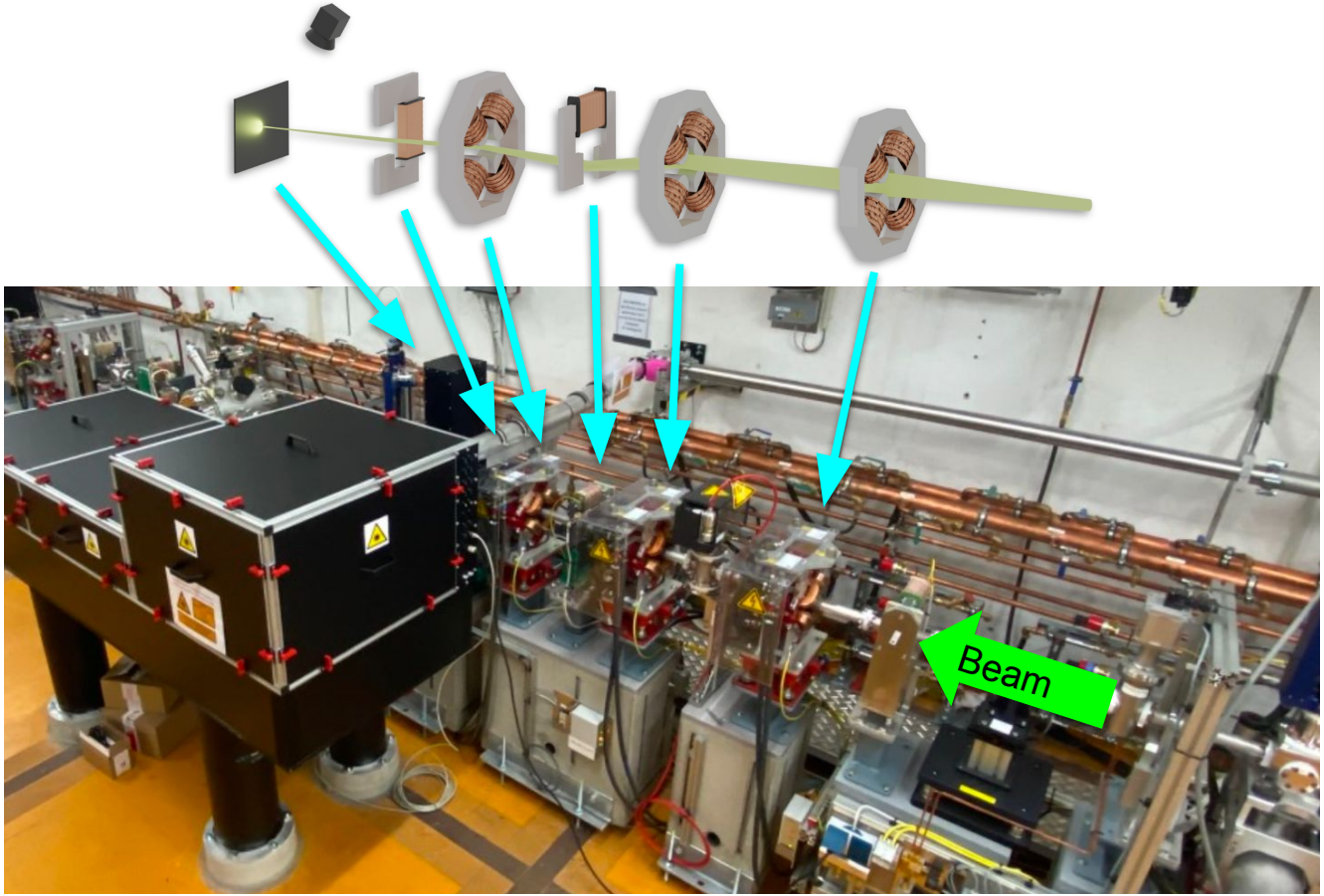


Reinforcement Learning for Particle Accelerators

ARES Experimental Area

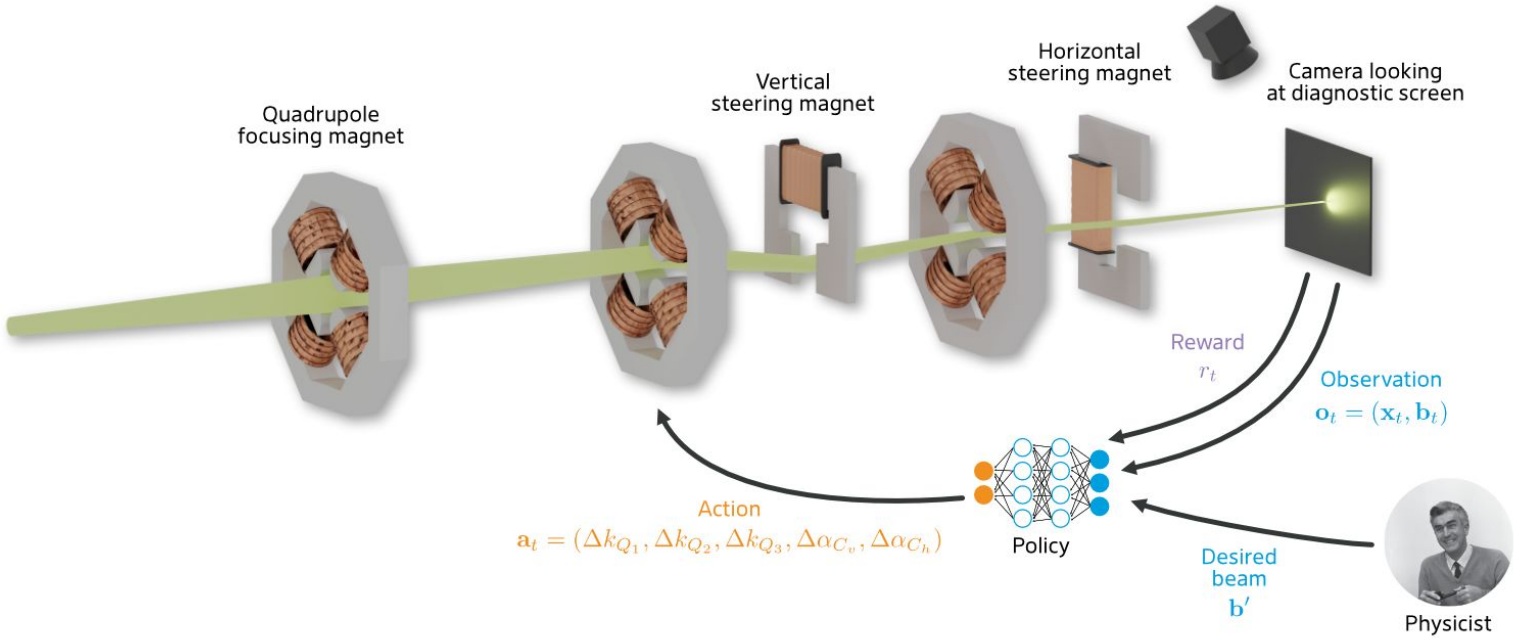
Positioning and focusing

- Operator wants to see **desired beam position and size on screen**.
- Reinforcement learning agent should **tune five magnets** to achieve beam parameters.



Reinforcement Learning for Accelerators Example

Positioning and focusing in the ARES Experimental Area



Difficulties

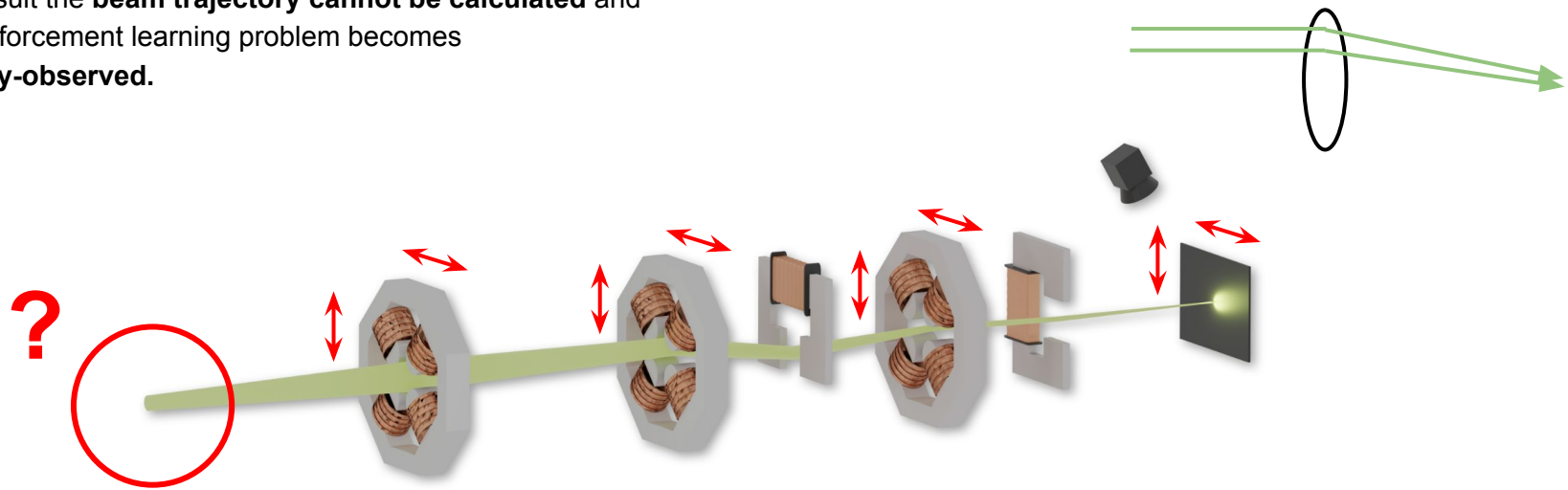
The real world is never simple

Unknown incoming beam

- Parameters of the beam entering the Experimental Area are unknown.
- Incoming beam may be different from day-to-day.
- As a result the **beam trajectory cannot be calculated** and the reinforcement learning problem becomes **partially-observed.**

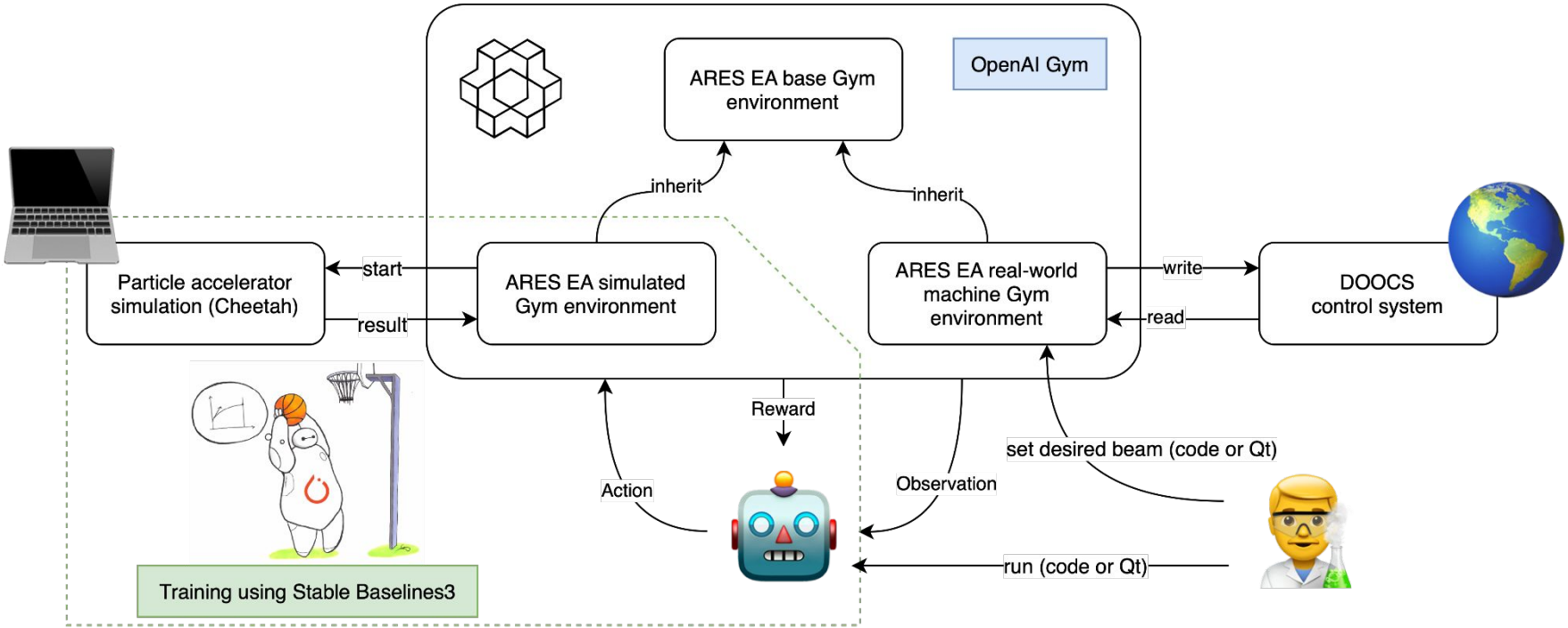
Misaligned quadrupoles and screen

- The quadrupoles and screen are **not perfectly aligned** with the beam.
- When entering a quadrupole off-center, the **beam is deflected**.



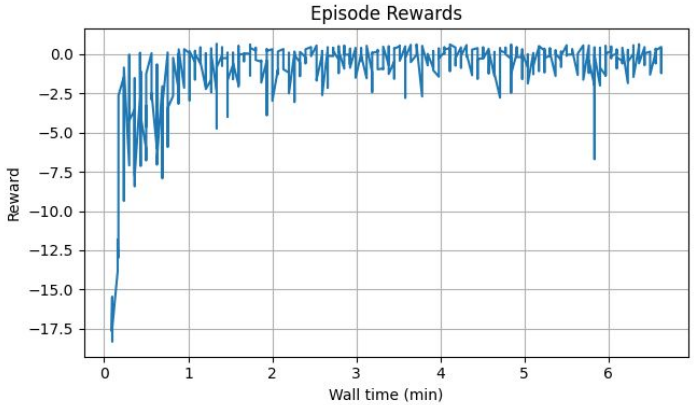
Accelerator Example

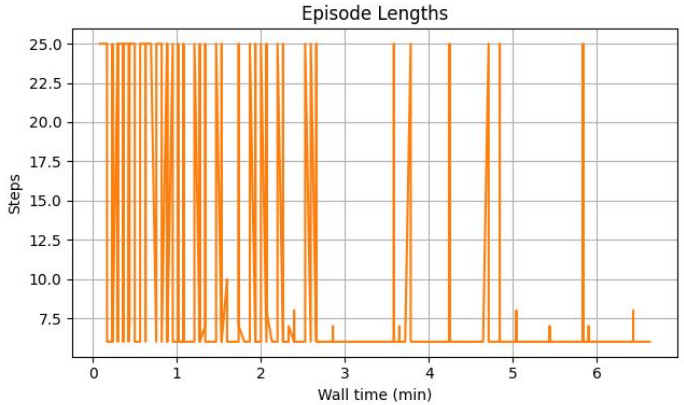
Technical overview

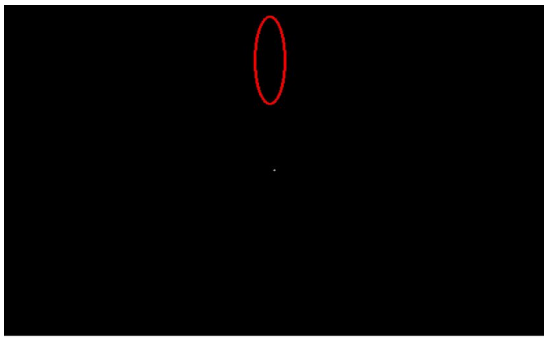


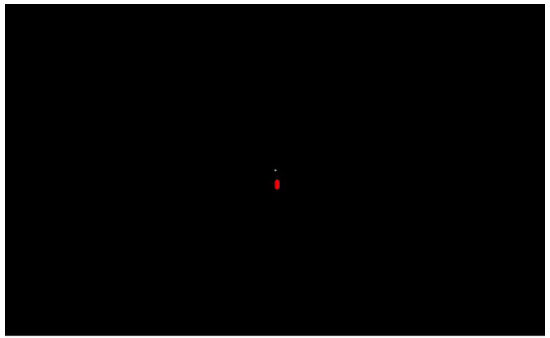
Accelerator Example

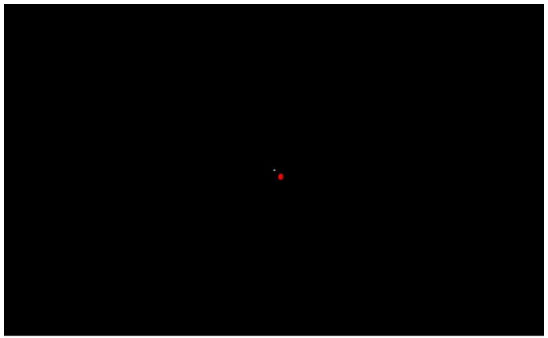
Training results











Accelerator Example

Running on the real accelerator

Hands-On

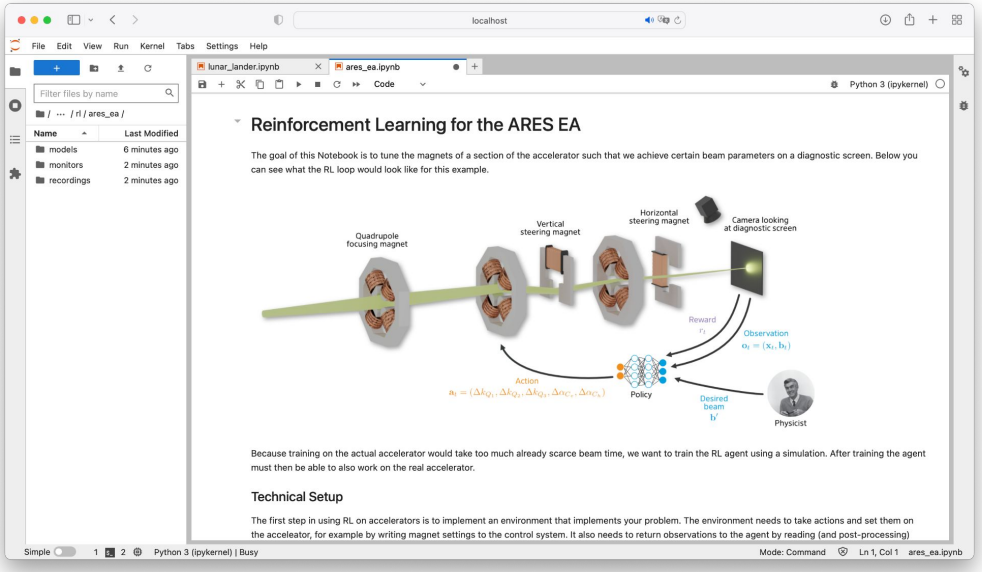
Try Reinforcement Learning Yourself

Jupyter Notebook with code for examples from this presentation

Code & Slides on GitHub.

https://github.com/desy-ml/mle-school-2022-use-case-desy-ii

- Try changing `config` dictionary in `ares_ea.ipynb` and train new agents. Evaluate your models and compare scores.
- If you are interested, train your own lunar lander in `lunar_lander.ipynb`.
- **(VERY ADVANCED)** Check out `utils/rl/ea_train.py` to see training code.



Finish Up

Further Resources

Where to start if you want to get into reinforcement learning

Getting started in RL

- OpenAI Spinning Up - Very understandable explanations on RL and the most popular algorithms accompanied by easy-to-read Python implementations.
- Reinforcement Learning with Stable Baselines 3 - YouTube playlist giving a good introduction on RL using Stable Baselines3.
- Build a Doom AI Model with Python - Detailed 3h tutorial of applying RL using *DOOM* as an example.
- An introduction to Reinforcement Learning - Brief introduction to RL.
- An introduction to Policy Gradient methods - Deep Reinforcement Learning - Brief introduction to PPO.

Papers

- Learning-based optimisation of particle accelerators under partial observability without real-world training - Tuning of electron beam properties on a diagnostic screen using RL.
- Sample-efficient reinforcement learning for CERN accelerator control - Beam trajectory steering using RL with a focus on sample-efficient training.
- Autonomous control of a particle accelerator using deep reinforcement learning - Beam transport through a drift tube linac using RL.
- Basic reinforcement learning techniques to control the intensity of a seeded free-electron laser - RL-based laser alignment and drift recovery.
- Real-time artificial intelligence for accelerator control: A study at the Fermilab Booster - Regulation of a gradient magnet power supply using RL and real-time implementation of the trained agent using field-programmable gate arrays (FPGAs).
- Magnetic control of tokamak plasmas through deep reinforcement learning - Landmark paper on RL for controlling a real-world physical system (plasma in a tokamak fusion reactor).

Literature

- Reinforcement Learning: An Introduction - Standard text book on RL.

Packages

- Gym - Defacto standard for implementing custom environments. Also provides a library of RL tasks widely used for benchmarking.
- Stable Baslines3 - Provides reliable, benchmarked and easy-to-use implementations of the most important RL algorithms.
- Ray RLlib - Part of the *Ray* Python package providing implementations of various RL algorithms with a focus on distributed training.

Thank you!

Contact

DESY. Deutsches
Elektronen-Synchrotron

www.desy.de

Annika Eichler & Jan Kaiser
Accelerator Beam Control (MSK)
annika.eichler@desy.de | jan.kaiser@desy.de

Accelerator Example

Getting it to work

Choosing rewards

- Make beam as small as possible (when reading screen)
  ➡ **Squeeze beam into corner**
- Minimise sum of pixels
  ➡ **Push beam off-screen**
- Get positive reward for each beam parameter while in threshold. After 5 steps in threshold give win bonus and stop.
  - **Briefly jump out of threshold after 4 steps**

  ➡

$$R(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} \hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) & \text{if } \hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) > 0 \\ 2 \cdot \hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) & \text{otherwise.} \end{cases}$$

$$\hat{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) = O(\boldsymbol{x}_t) - O(\boldsymbol{x}_{t+1})$$

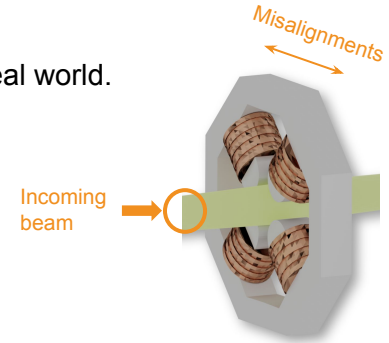
$$O(\boldsymbol{x}_t) = \ln \sum_{p \in \boldsymbol{b}_t, p' \in \boldsymbol{b}'} w_p \left| p - p' \right|$$

Sim2real transfer

Getting RL to run on simulations is easy. Getting it to run on a real accelerator is hard.

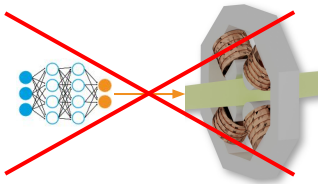
Domain randomisation

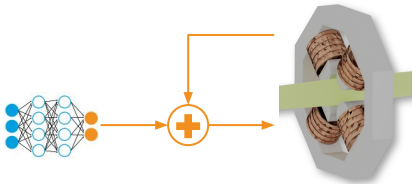
The simulation is never quite like the real world.
-> Add random noise.



Delta actions

Magnet settings may be affected by noise.
-> Policy outputs changes to magnet settings.





Beam Dynamics

Positioning and focusing in the ARES Experimental Area

Formal description of e-beam

Bunch als Parameter (gaussverteilt angenommen)

-> Vector -> mus und sigmas x bis p

Matrix multiplikation zum trackend