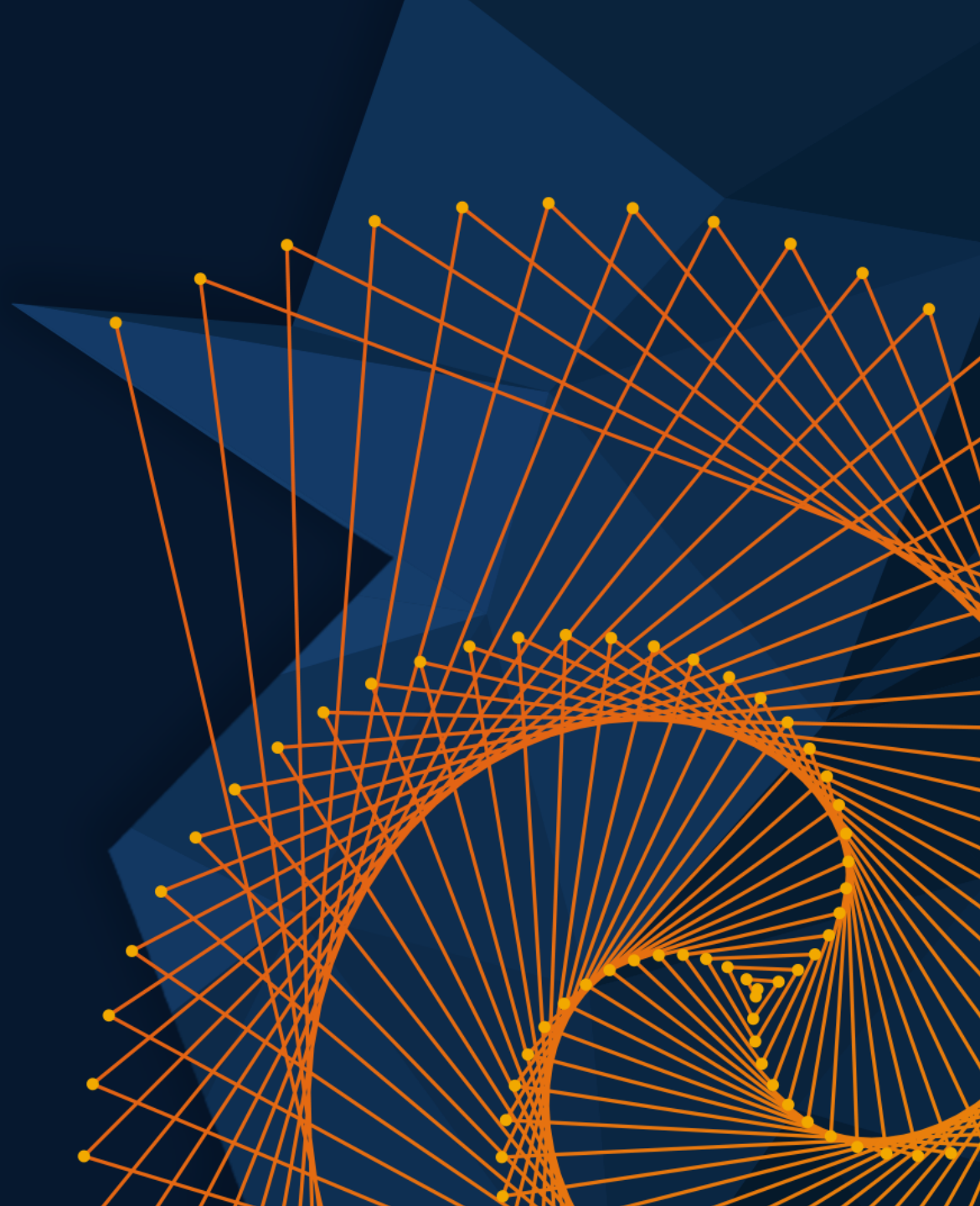
MATLAB EXPO

A Hands-On Introduction to Reinforcement Learning with MATLAB and Simulink

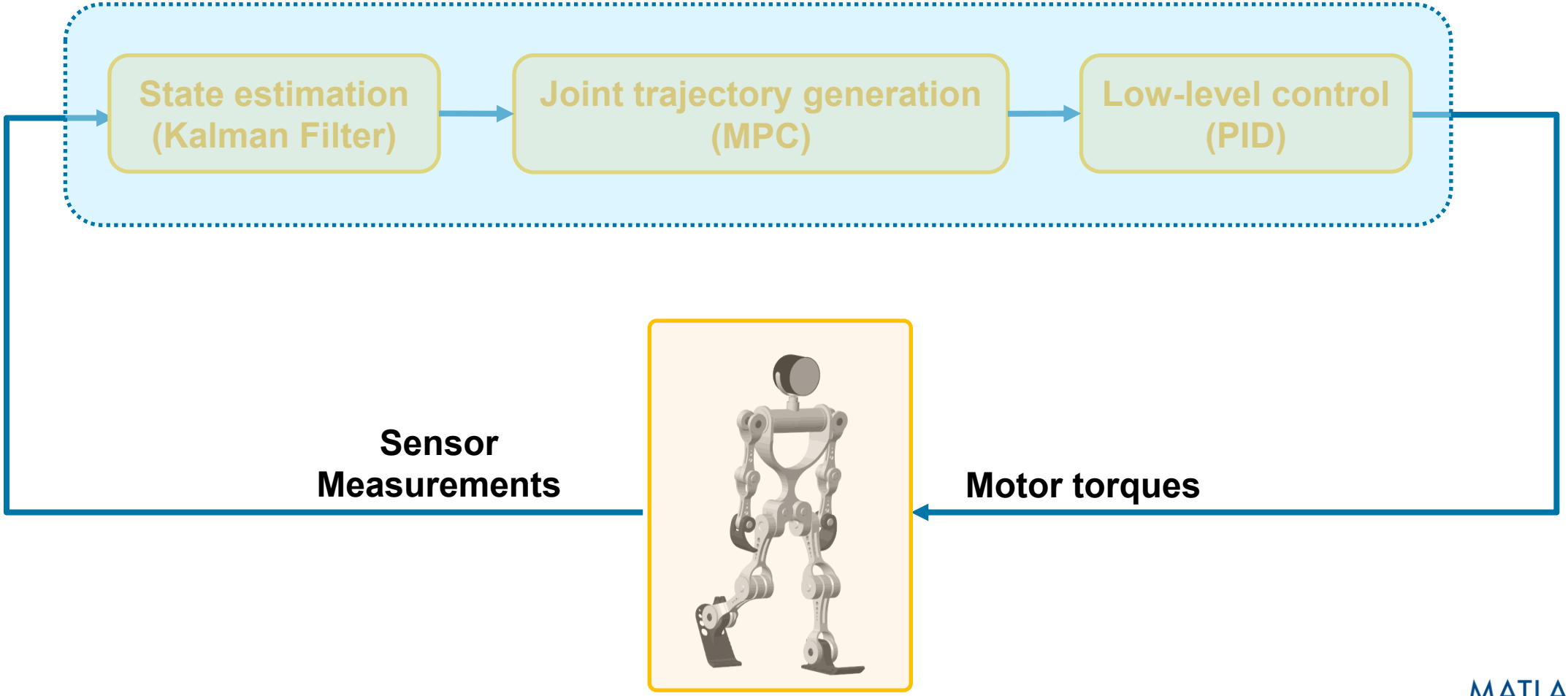
Jordan Olson, MathWorks

MathWorks®

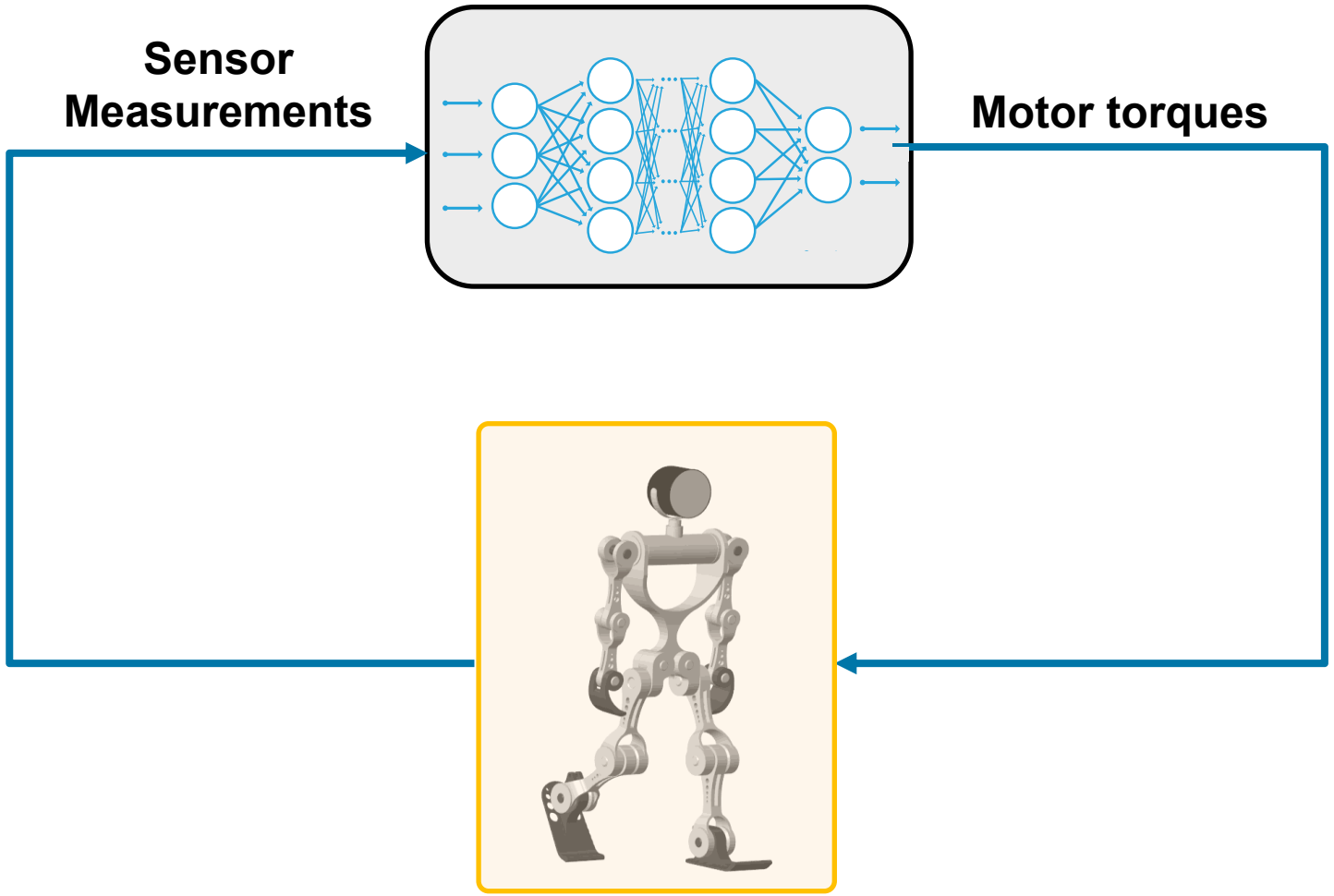


Why use reinforcement learning?

Why use reinforcement learning?



Why use reinforcement learning?



Key takeaways

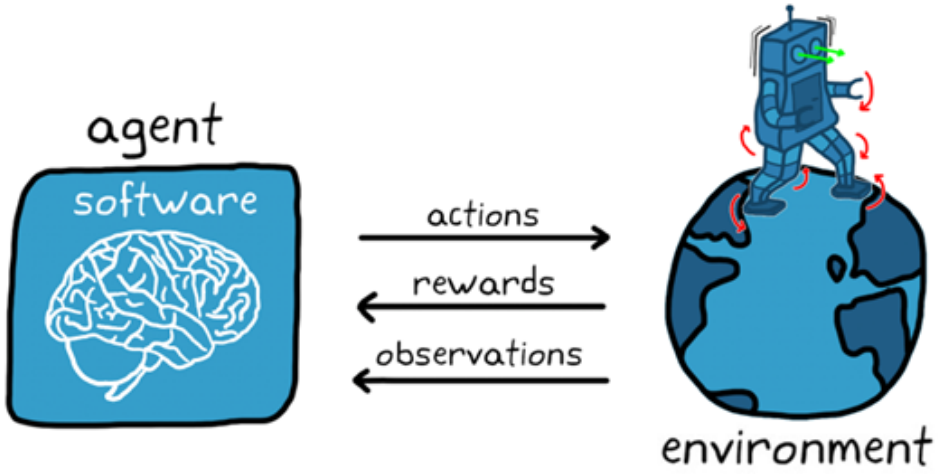
- What is reinforcement learning and why is it useful?
- When can/should I use reinforcement learning?
- How do I set up and solve a reinforcement learning problem?

Agenda

- Introduction to reinforcement learning
- Deep dive: Teach a robot to walk! (with hands-on exercises)
  - Defining environments and reward functions
  - Creating policies and agents
  - Training, testing, and deploying policies
- Wrap-Up and Additional Resources

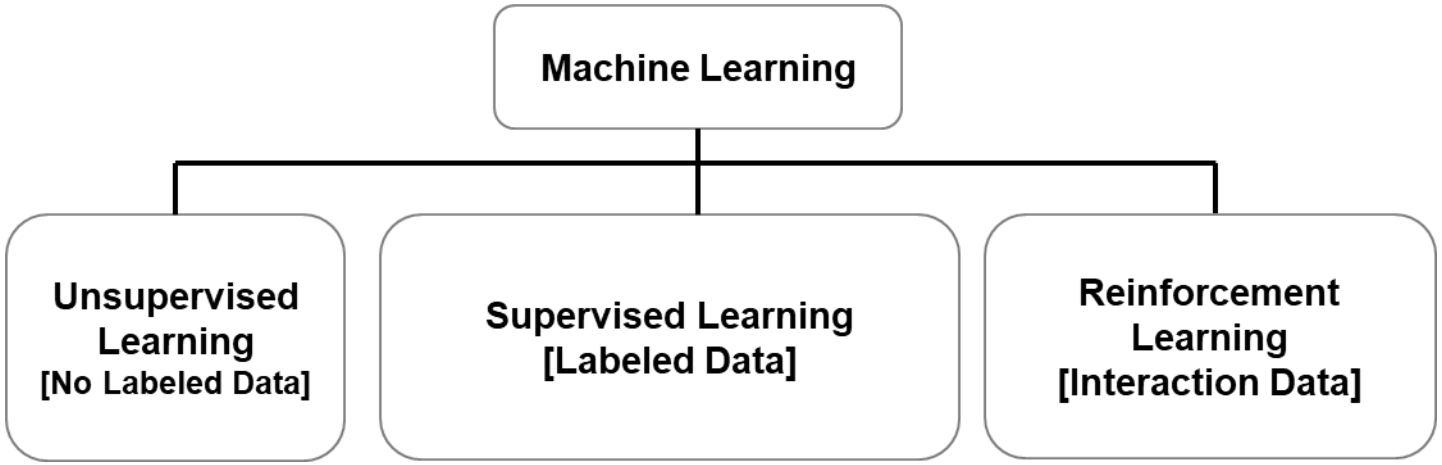
What is reinforcement learning?

Type of machine learning that trains an **agent** through trial & error interactions with an **environment**

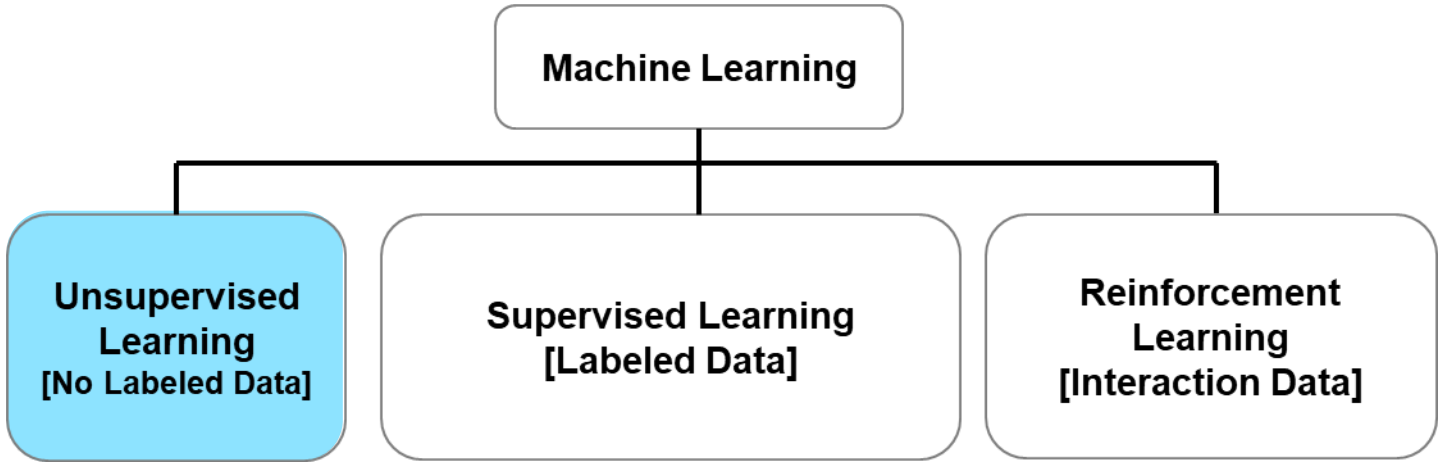


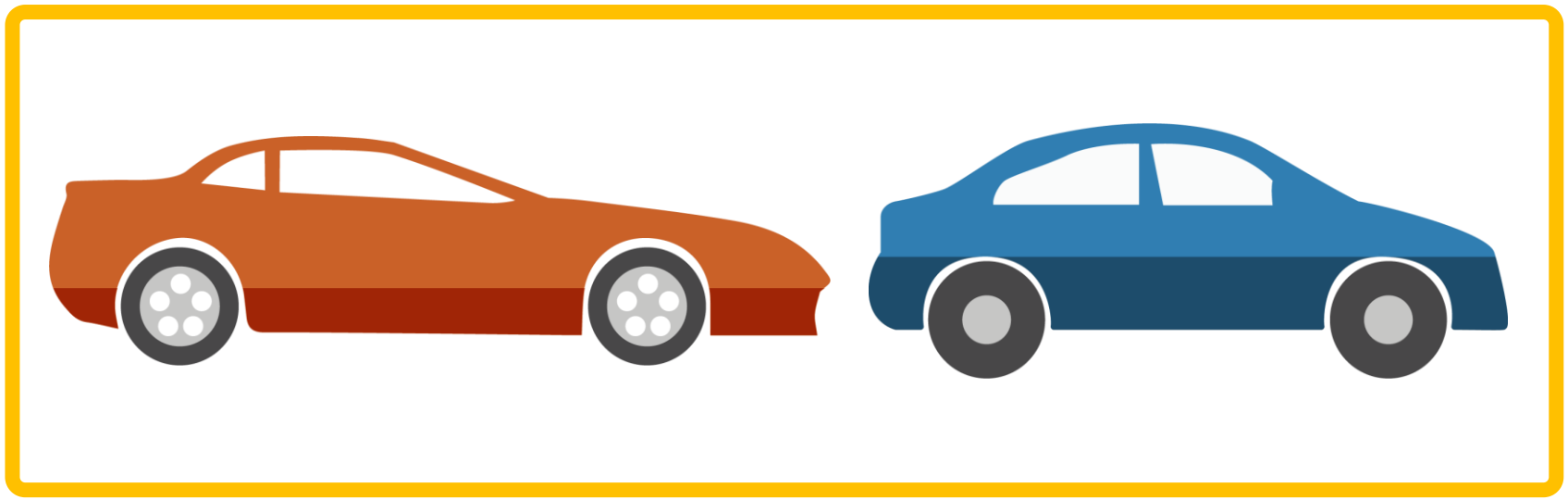
“AI for decision-making”

Reinforcement learning vs. machine learning

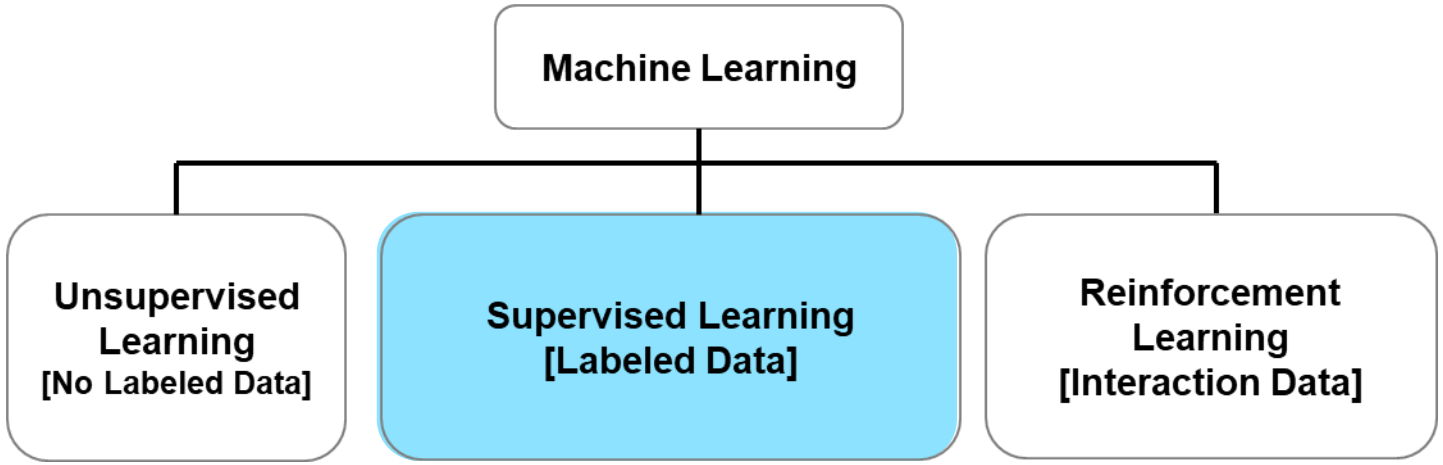


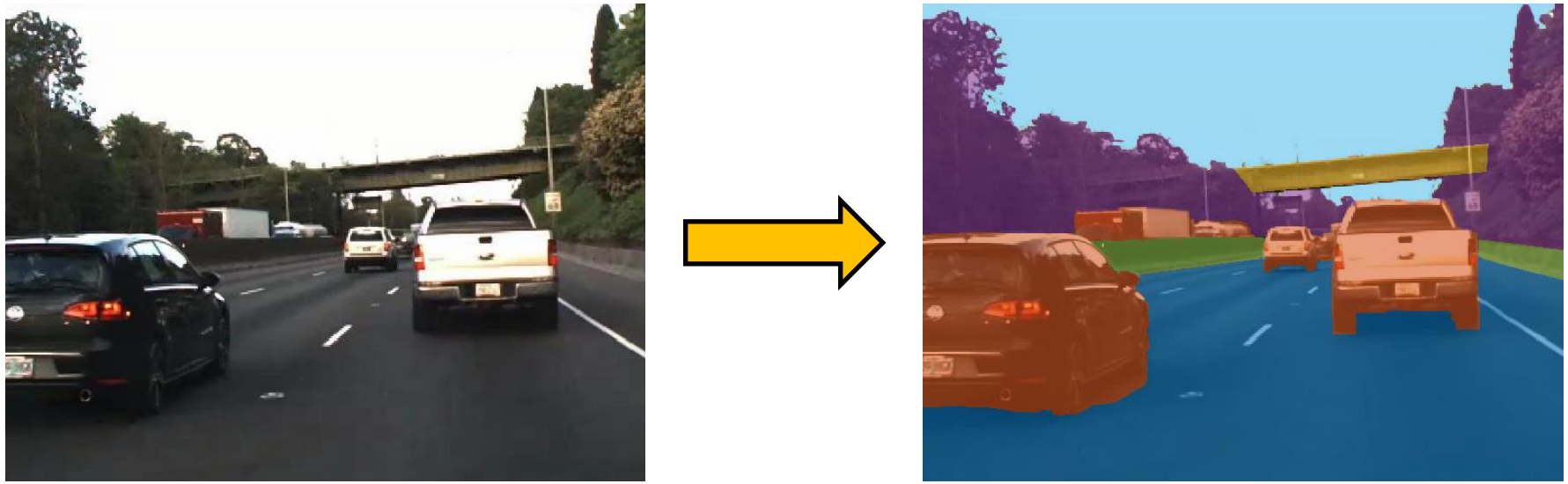
Reinforcement learning vs. machine learning



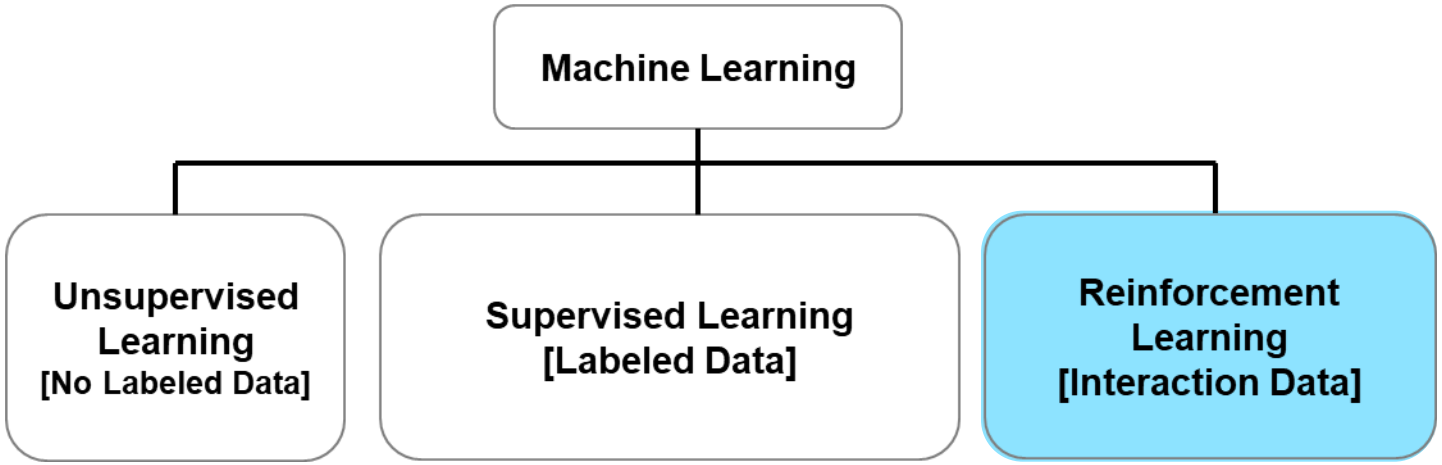


Reinforcement learning vs. machine learning

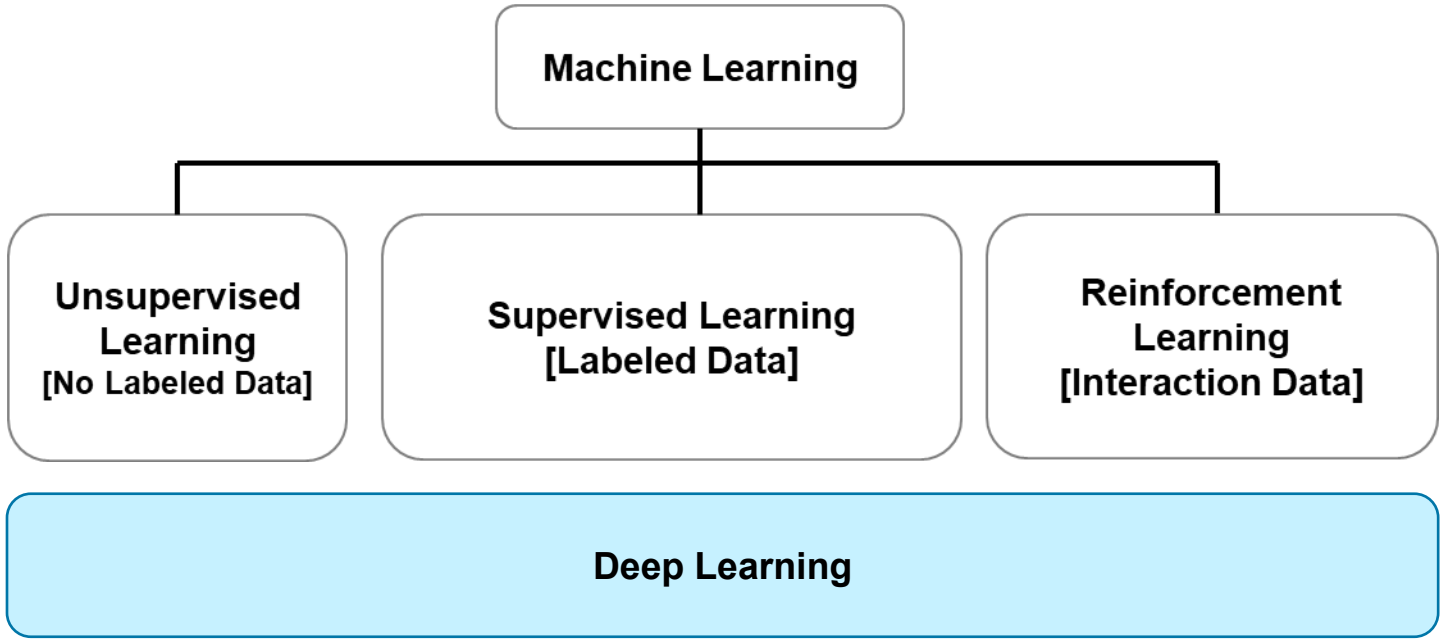




Reinforcement learning vs. machine learning



Reinforcement learning vs. machine learning

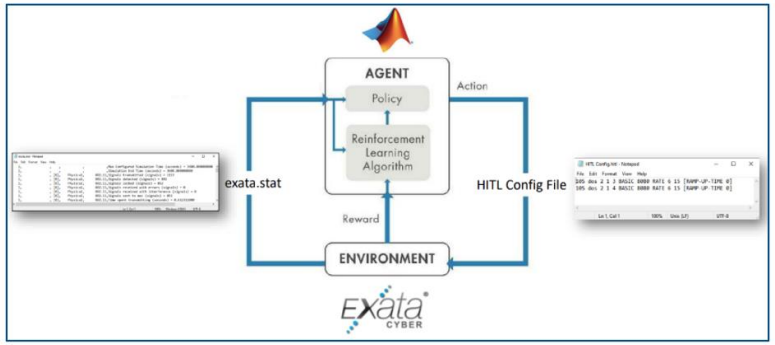


What about deep learning?

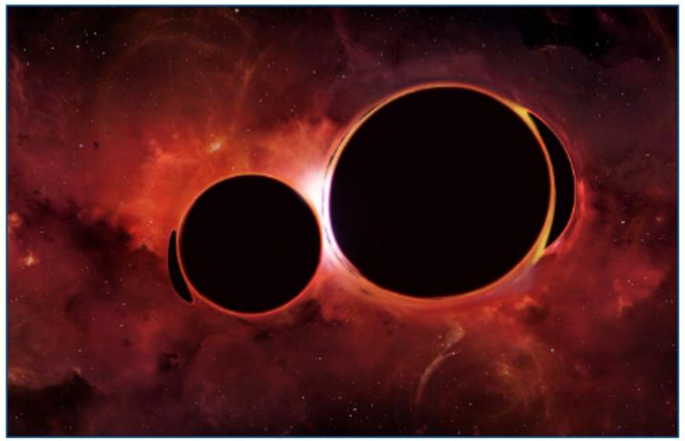
Complex reinforcement learning problems typically need deep neural networks
[Deep Reinforcement Learning]

What can reinforcement learning be used for?

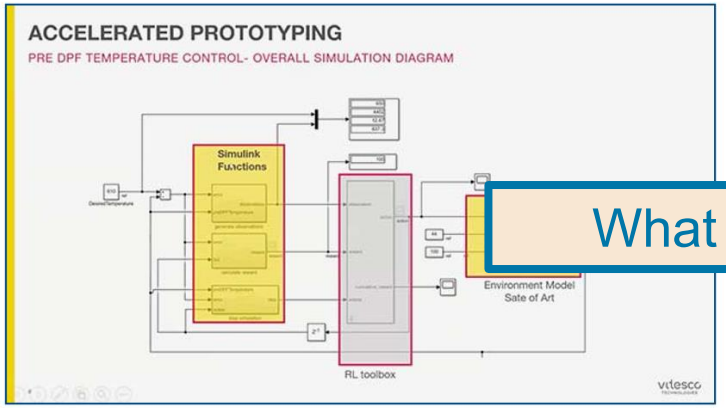
Lockheed Martin Assesses 5G Network Vulnerabilities with Reinforcement Learning Toolbox



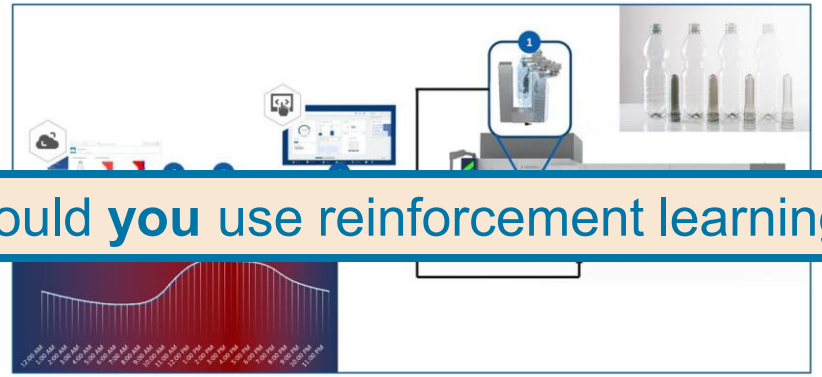
Max Planck Institute Develops Deep Learning System to Detect Gravitational Waves



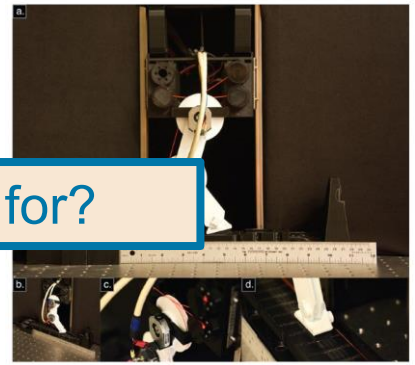
Vitesco Technologies Applies Deep Reinforcement Learning in Powertrain Control



Krones AG Builds Reinforcement Learning–Based Process Control in the Blow Molder Contiloop AI for PET and rPET Bottles



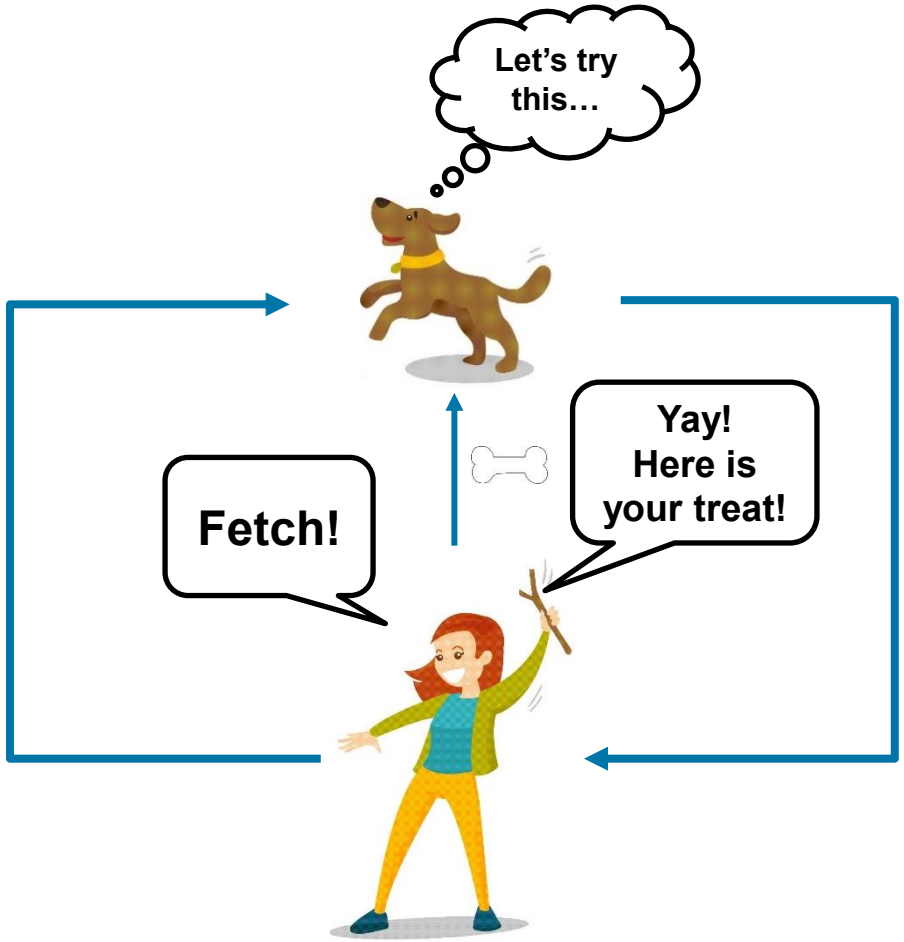
USC Researchers Create a Tendon-Driven Robot That Teaches Itself to Walk with Reinforcement Learning

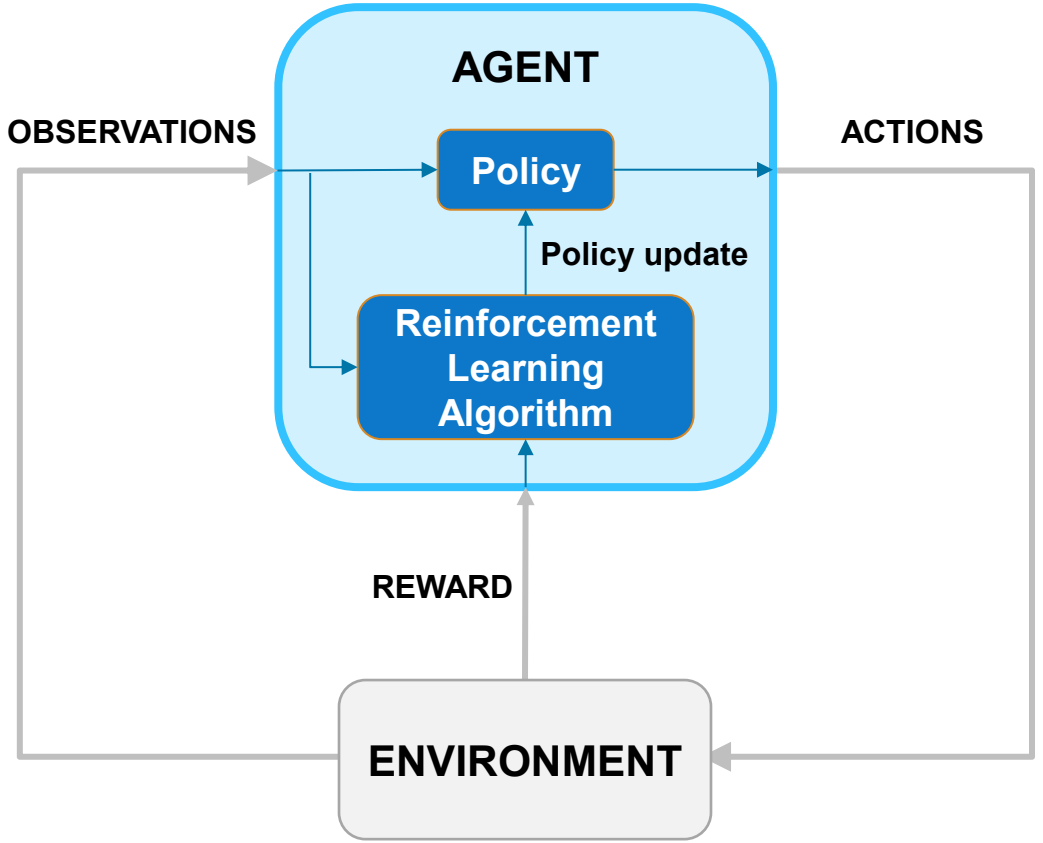


What could **you** use reinforcement learning for?

How does training work?

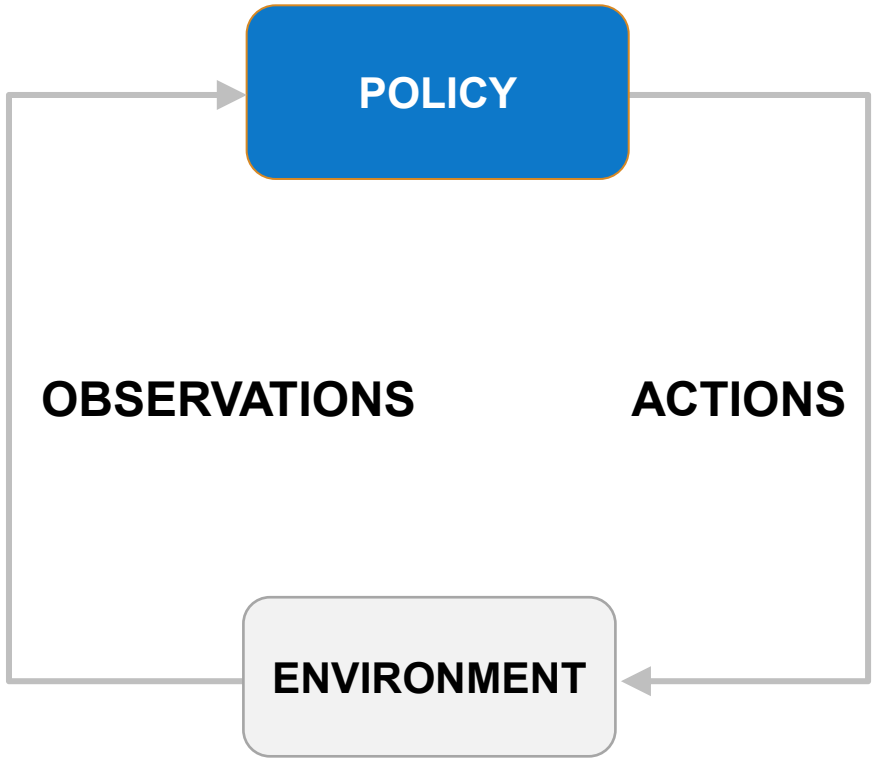
Reinforcement learning works through **trial and error**, like **training a pet.**





How does training work?

After training, only trained policy is needed

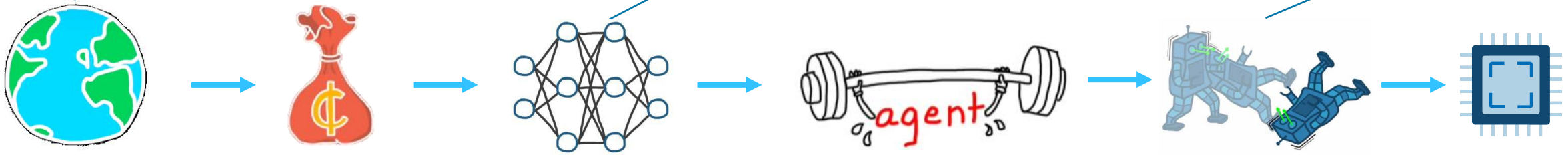


Reinforcement learning workflow

- Simulation models or real hardware
- Virtual models are safer and cheaper

- Select training algorithm
- Policy architecture
- Tune hyperparameters

- Simulation, SIL, PIL, HIL…



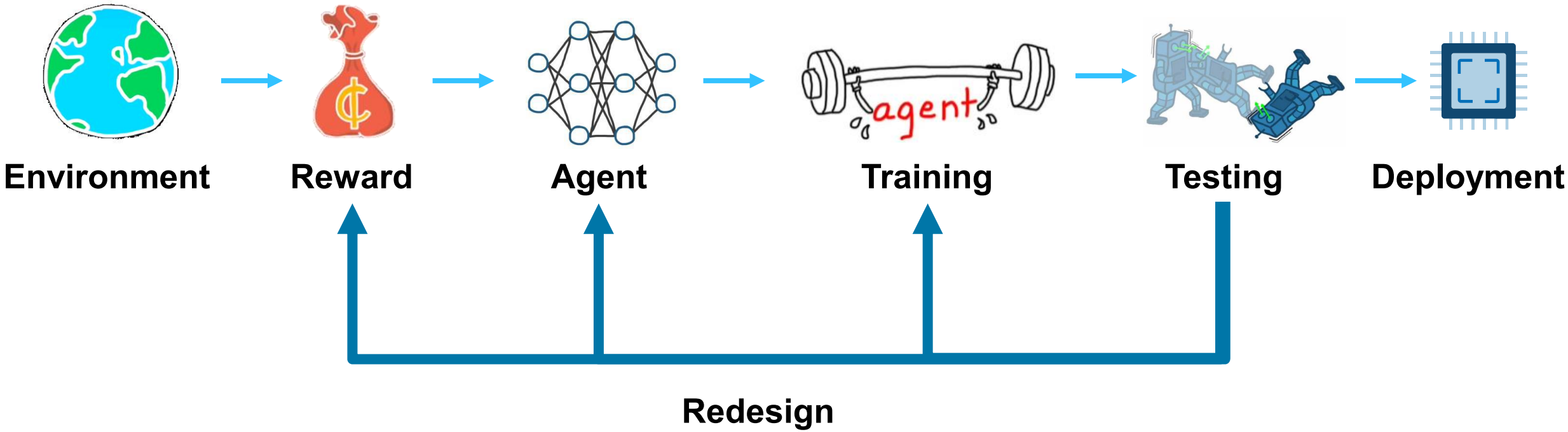
Environment **Reward** **Agent** **Training** **Testing** **Deployment**

- Numerical value that evaluates goodness policy
- Reward shaping can be challenging

- Large number of simulations needed
- Parallel & GPU computing can speed up training

- Trained policy is a standalone function

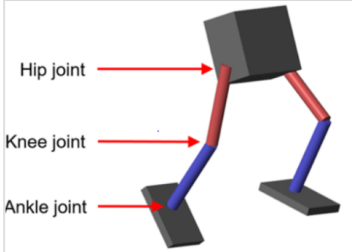
Reinforcement learning workflow



Reinforcement Learning in MATLAB

Reinforcement Learning Toolbox provides

- Built-in and custom reinforcement learning **algorithms**
- Seamless **integration with Simulink**
- Visual interactive workflow with **Reinforcement Learning Designer**
- **Deploy** trained policies to embedded and production systems
- Reference **examples** for getting started

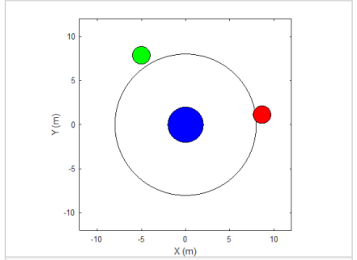


Train Biped Robot to Walk Using Reinforcement Learning Agents

Train a reinforcement learning agent to control a biped walking robot modeled in Simscape Multibody.

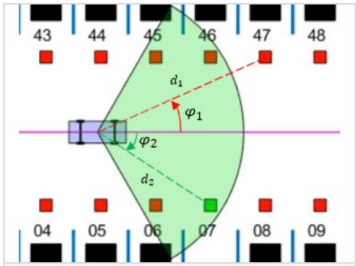
Automatic Parking Valet with Unreal Engine Simulation

Use a reinforcement learning agent with an MPC controller to perform a parking maneuver.



Train Multiple Agents to Perform Collaborative Task

Train two PPO agents to collaboratively move an object.



Train PPO Agent for Automatic Parking Valet

Train a reinforcement learning agent to park a car in an open parking space.

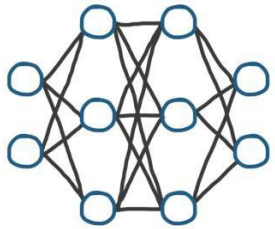
Agenda

- Introduction to reinforcement learning
- Deep dive: Teach a robot to walk! (with hands-on exercises)
  - Defining environments and reward functions
  - Creating policies and agents
  - Training, testing, and deploying policies
- Wrap-Up and Additional Resources

Reinforcement learning workflow

Environment

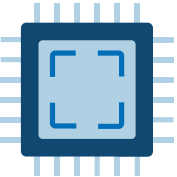
Reward



Agent

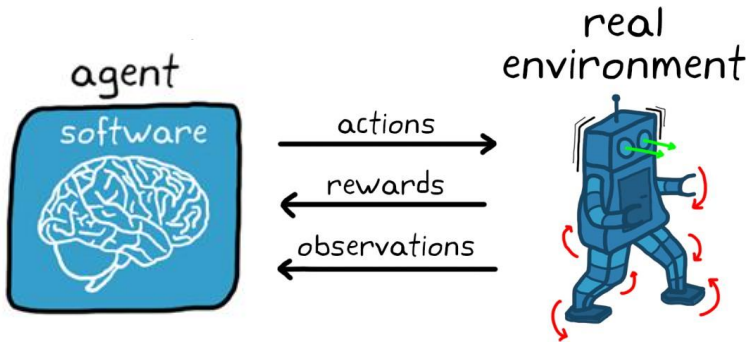
Training

Testing



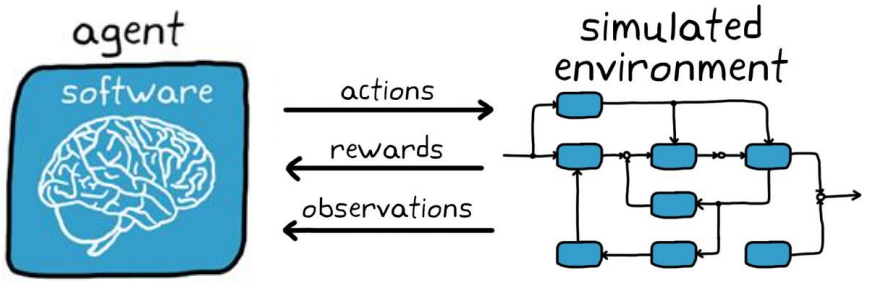
Deployment

Real vs. simulated environments



☺ Accuracy

☹ Risk



☺ Training speed

☺ Flexible simulated conditions

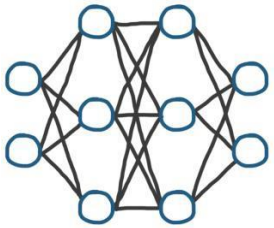
☺ Safety

☹ Model inaccuracies

Reinforcement learning workflow

Environment

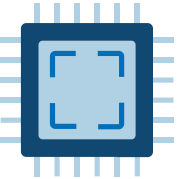
Reward



Agent

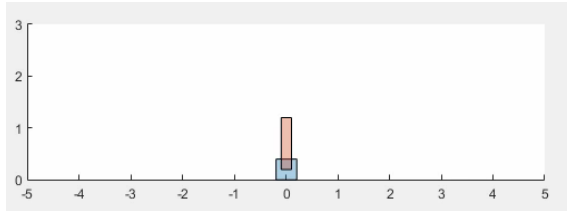
Training

Testing



Deployment

$$r_t = -\left(\theta_t^2 + 0.1\dot{\theta}_t^2 + 0.001{u_{t-1}}^2\right)$$



Maximize score

Maximize profit

Reward functions

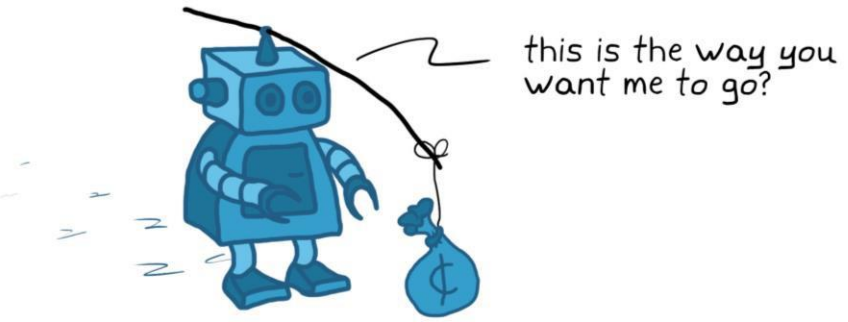
Reward is a **function** that produces a **scalar number** that represents the "**goodness**" of an agent being in a particular state and taking a particular action.

reward = function (state, action)

scalar representing "goodness"

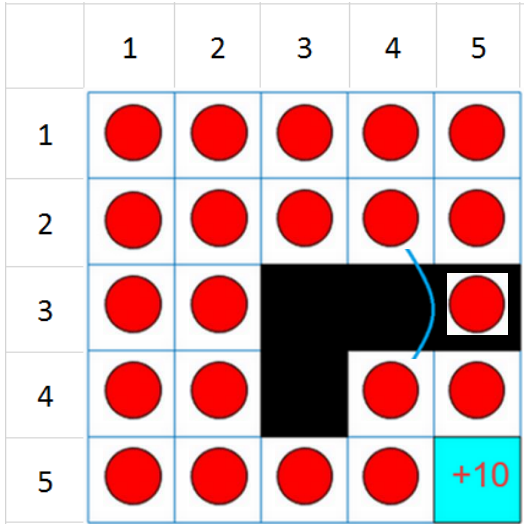
There are **no restrictions** on creating a reward function.
They could be

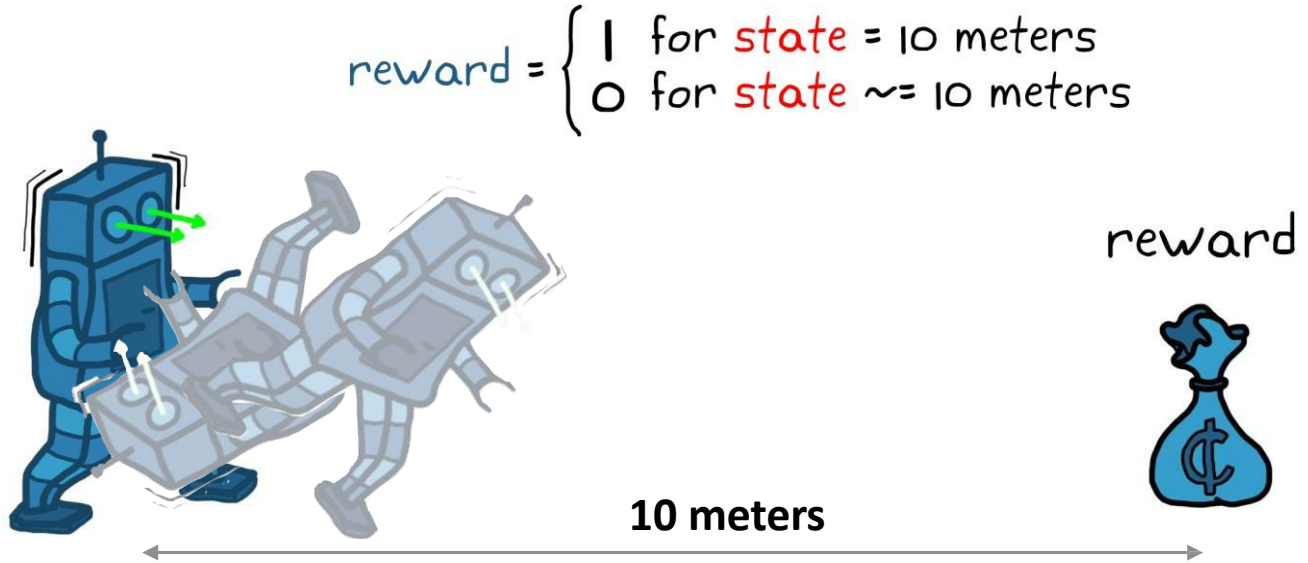
- provided at each time step
- provided at the end of an episode or after long periods of time (sparse rewards)
- calculated based on some logic or a function
- …



Problems with sparse rewards

With sparse rewards, the goal you want to incentivize comes after **a long sequence of actions**

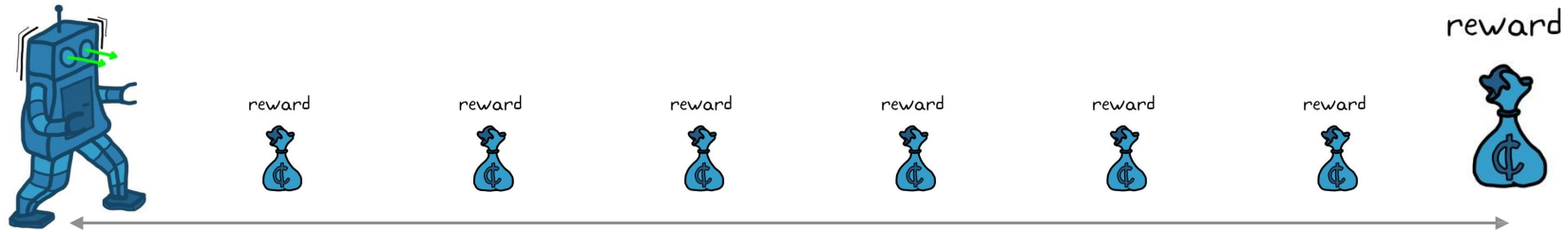




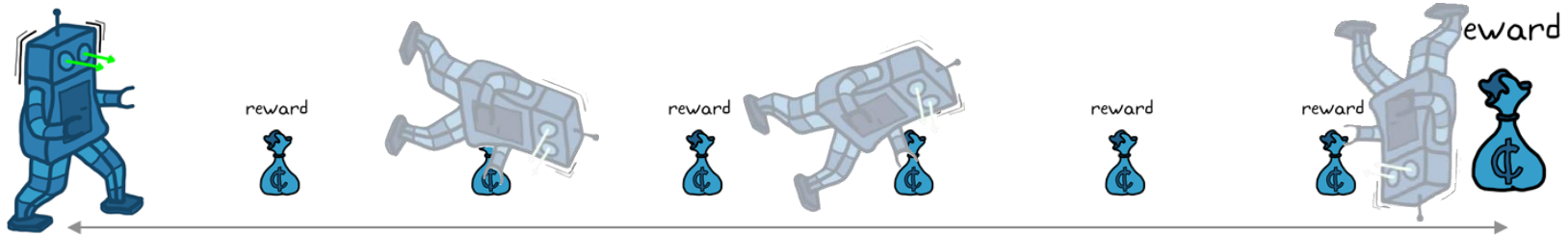
The chance that your agent will randomly stumble on the **exact action sequence** that produces the sparse reward is very **unlikely**.

Reward shaping

You can improve sparse rewards through **reward shaping**. For example, provide **smaller intermediate** rewards that **guide** the agent along the right path.



Reward shaping comes with its own set of **problems**



A **poorly shaped** reward function might cause your agent to converge on a solution that is **not ideal**, even if that solution produces the most rewards for the agent.

Reward shaping

- Unlike a domain-expert, the untrained agent is like a **newborn baby**
- Reward shaping is the perfect opportunity to deliver **domain-specific knowledge** to the agent

$$r_t = v_x - 3y^2 - 50\hat{z}^2 + 25\frac{T_s}{T_f} - 0.02\sum_i u_{t-1}^{i\,2}$$

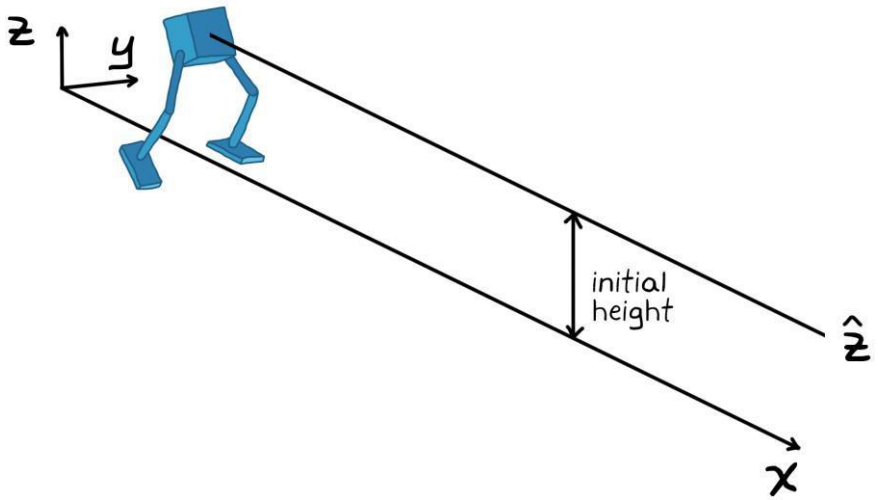
forward velocity

don't stray from path

keep trunk at initial height

walk as long as possible

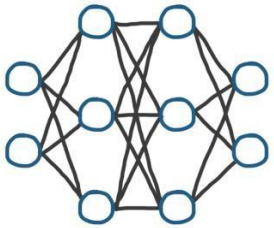
minimize actuator effort



Exercise 1: Defining environments and reward functions

Environment

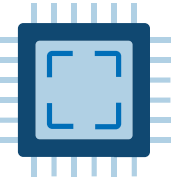
Reward



Agent

Training

Testing

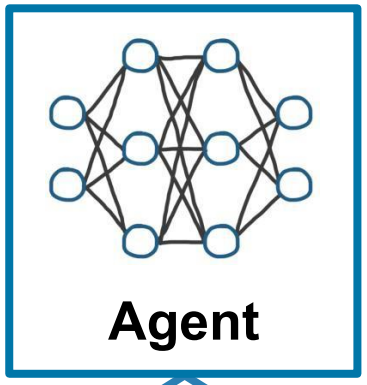


Deployment

Reinforcement learning workflow

Environment

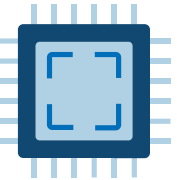
Reward



Agent

Training

Testing



Deployment

Policy

Algorithm

Representing policies

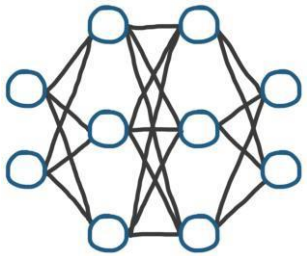
A policy is a **function (mapping)** that takes in **observations** and outputs **actions**.
It represents the **decision-making strategy** of the agent.

policy ⇒ actions = function (state observations)

Tables

	a1	a2	a3
s1	value	value	value
s2	value	value	value
s3	value	value	value

Neural Networks



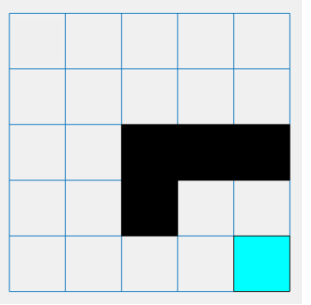
Polynomials

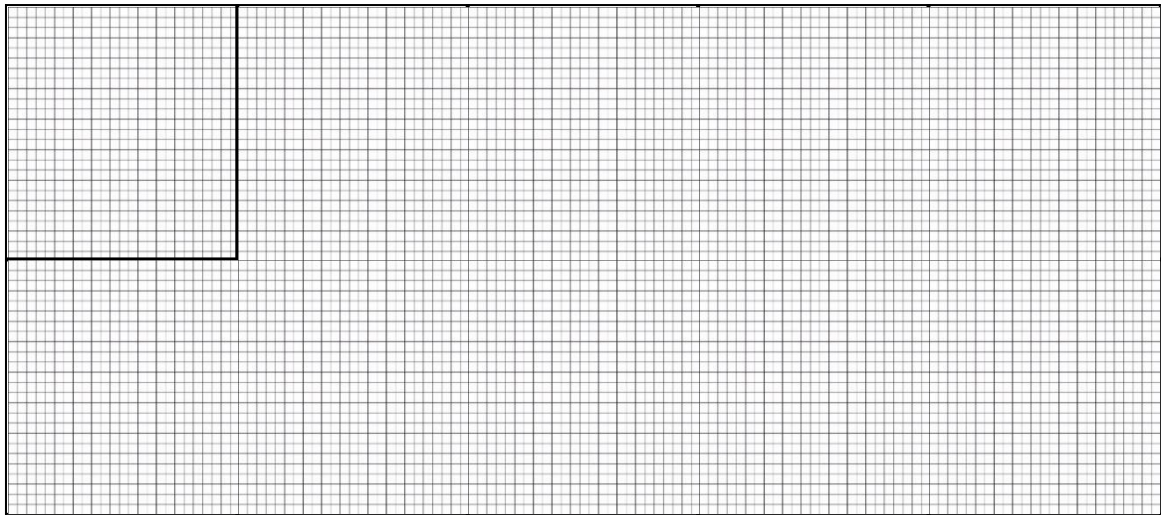
$$value = -(\dot{\theta}^2 + \theta^2) + \tau$$

$$actions = -(\dot{\theta}^2 + \theta^2)$$

Problems with tables

- **Large** state/action spaces
  - **Slow** to learn the value of each state
  - A lot of **memory** required for storage
  - Cannot **generalize** learning to **unvisited** states
- **Continuous** state/action spaces



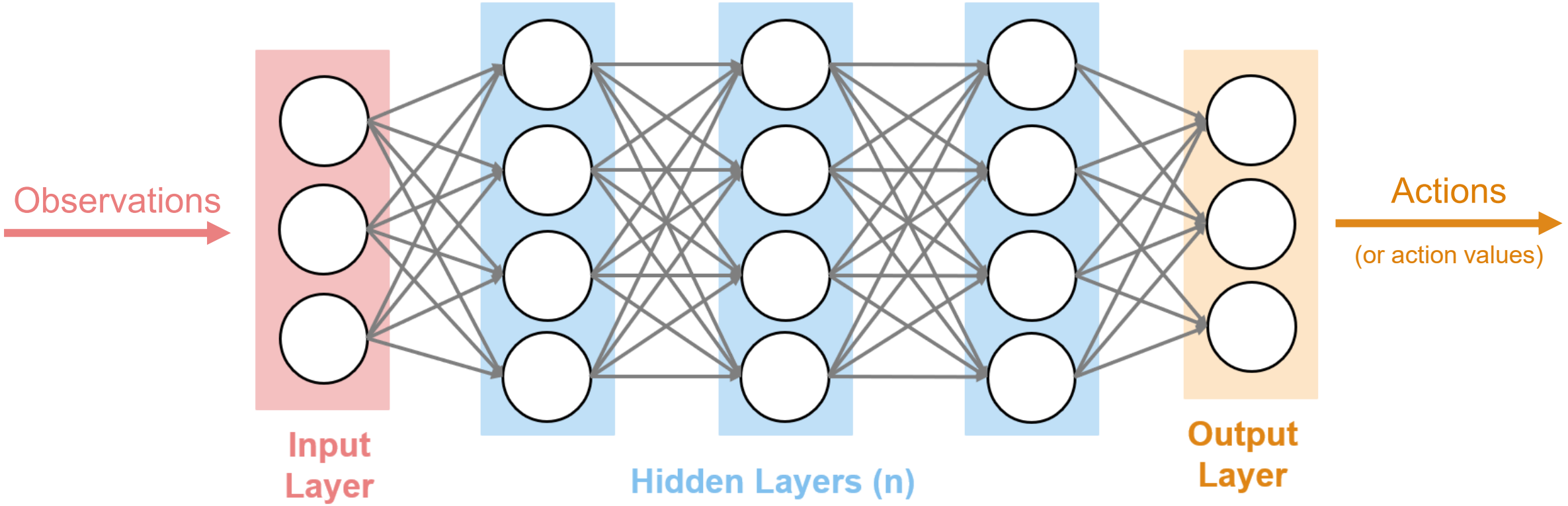


torque

angle rate	angle	-0.01	-0.001	0.02	0.021	...
-0.3	0.124	value	value	value	value	
0.17	0.137	value	value	value	value	
0.175	0.139	value	value	value	value	
0.223	0.204	value	value	value	value	
⋮	⋮					

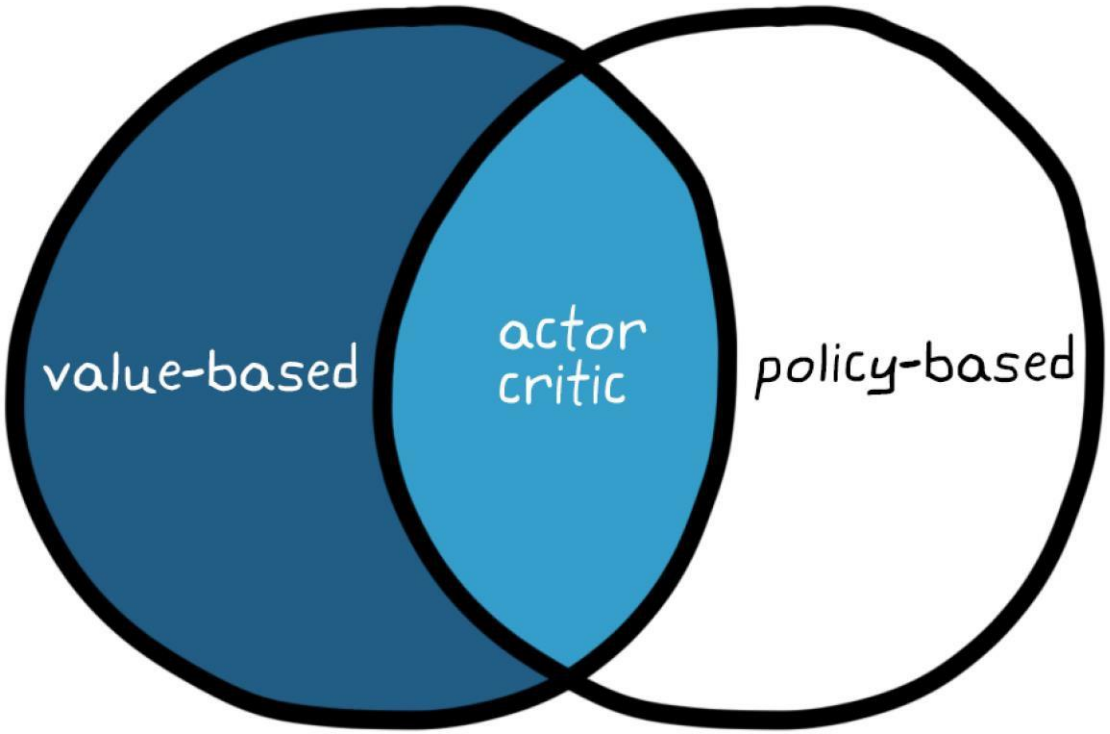
Deep neural networks as policy representations

- Deep neural networks have many layers
- Data is passed through the network, and the layer parameters are updated through **training**



Types of reinforcement learning algorithms

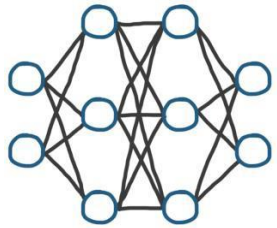
- Value-based
  - **Indirect** policy representation
    - **Learning value** function (critic)
    - Policy is **extracted**
- Policy-based
  - **Direct** policy representation
    - Learning **policy** (actor)
    - **No value** function needed
- Actor-Critic
  - **Combines** above two methods
    - Learning **policy** (actor)
    - Learning **value** function (critic)



Exercise 2: Creating policies and agents

Environment

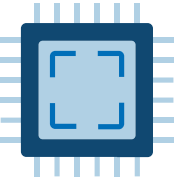
Reward



Agent

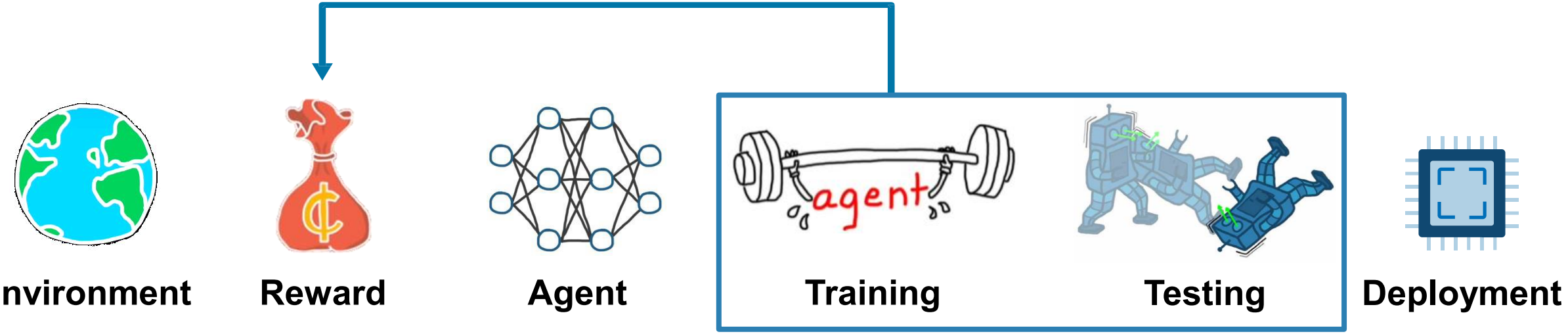
Training

Testing



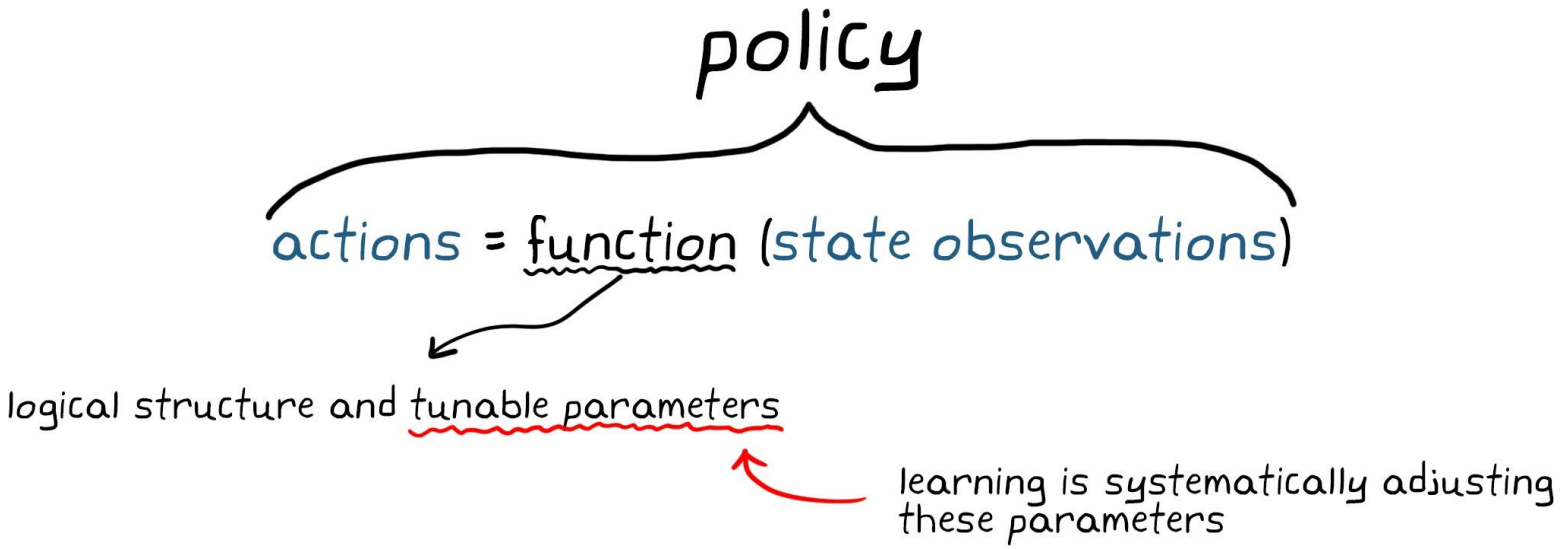
Deployment

Reinforcement learning workflow



Updating policies through learning

- Policy (direct or indirect) is a **function** made up of **tunable** parameters
- There is a **set of parameters** that produces the **optimal** policy



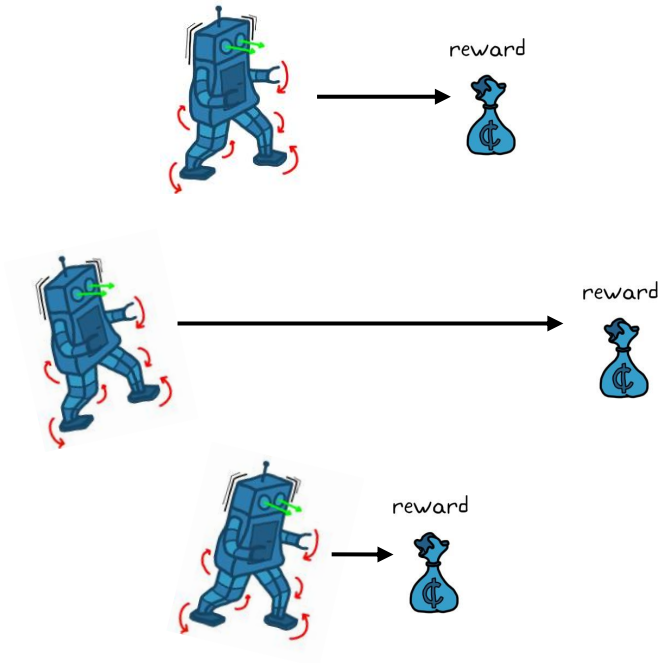
Learning is the process of systematically **adjusting parameters** of the policy (e.g. neural network **weights**)

Improving training performance

Improve policy **robustness** and **performance** by **randomizing environment**…

- initial conditions
- model parameters
- simulation scenarios
- objectives (e.g. tracking signals)
- sensor outputs

…or **modifying reward function**



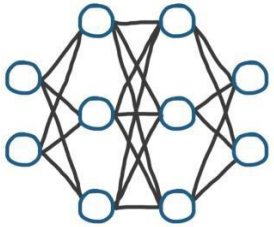
$$r_t = 2v_x - 5y^2 - 50\hat{z}^2 + 25\frac{Ts}{Tf} - 0.02\sum_i u_{t-1}^{i\,2}$$

minimize actuator effort
walk as long as possible
keep trunk at initial height
don't stray from path
forward velocity

Reinforcement learning workflow

Environment

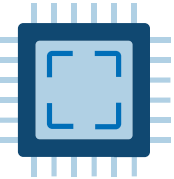
Reward



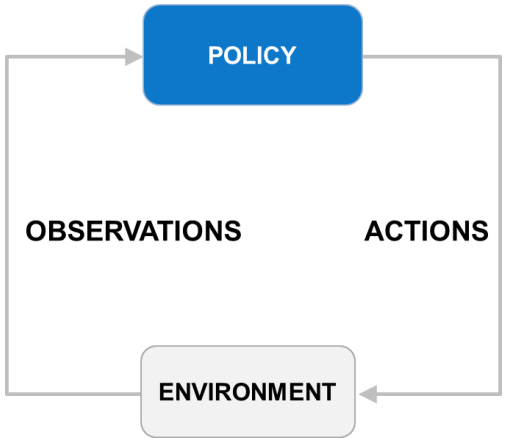
Agent

Training

Testing



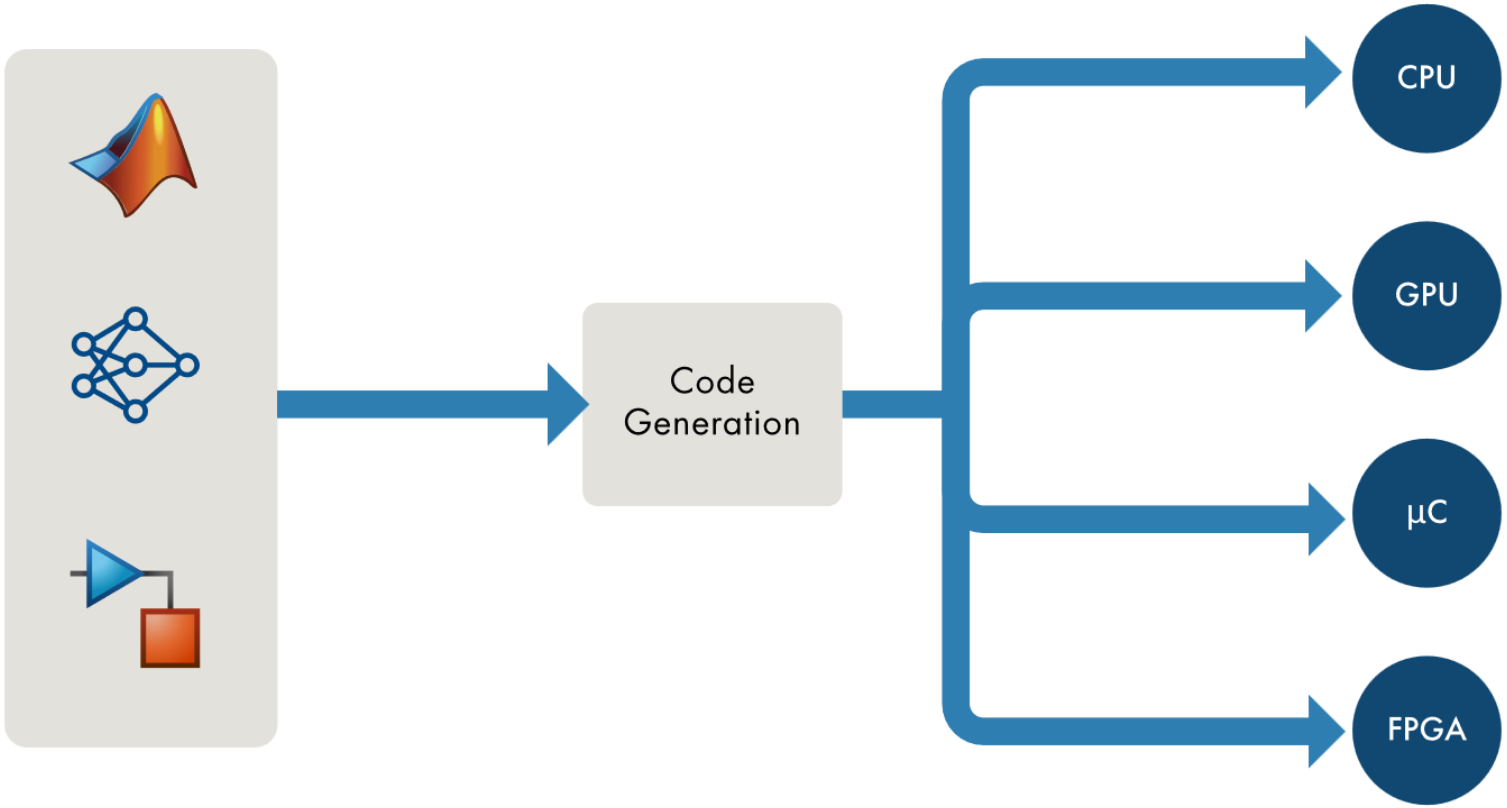
Deployment



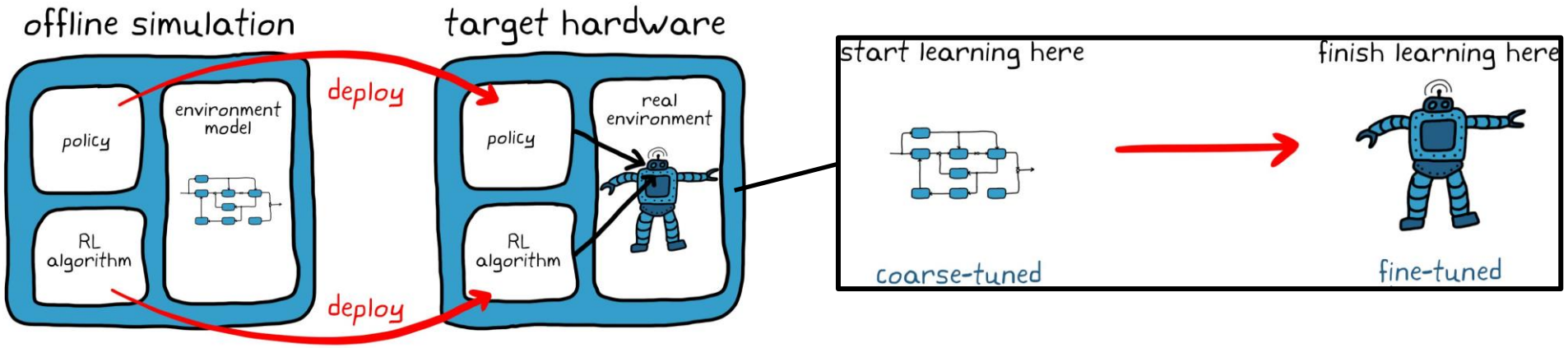
After training, only trained policy is needed

Deploying policies

Take trained policies to production with **automatic code generation**



Closing the “sim to real” gap



Additional training may be required **after** deployment due to:

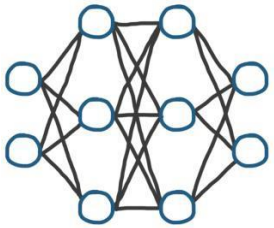
- Model inaccuracies
- Sensor noise
- Environment changes not accounted for in training

Domain randomization may limit the amount of training needed **after** deployment.

Exercise 3: Training, testing, and deploying policies

Environment

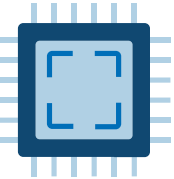
Reward



Agent

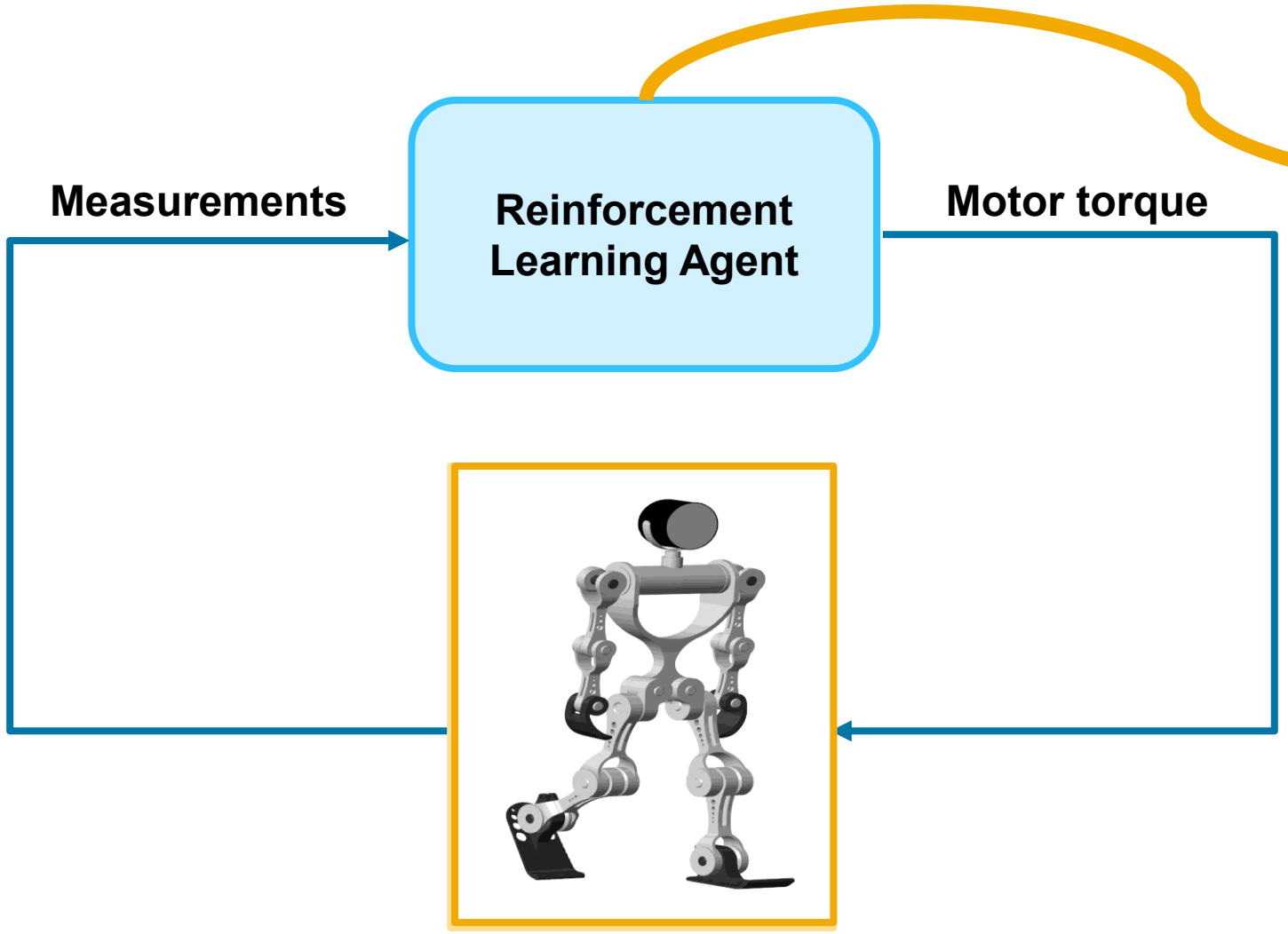
Training

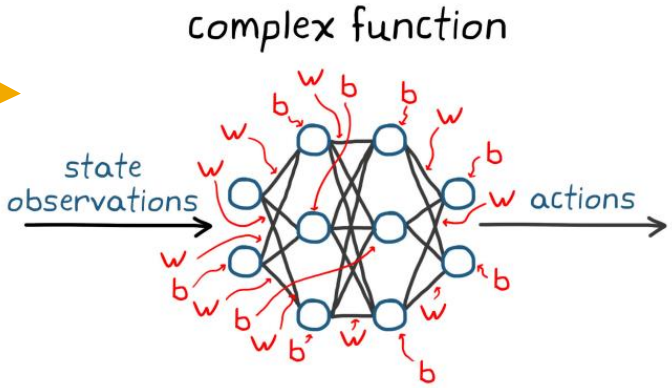
Testing



Deployment

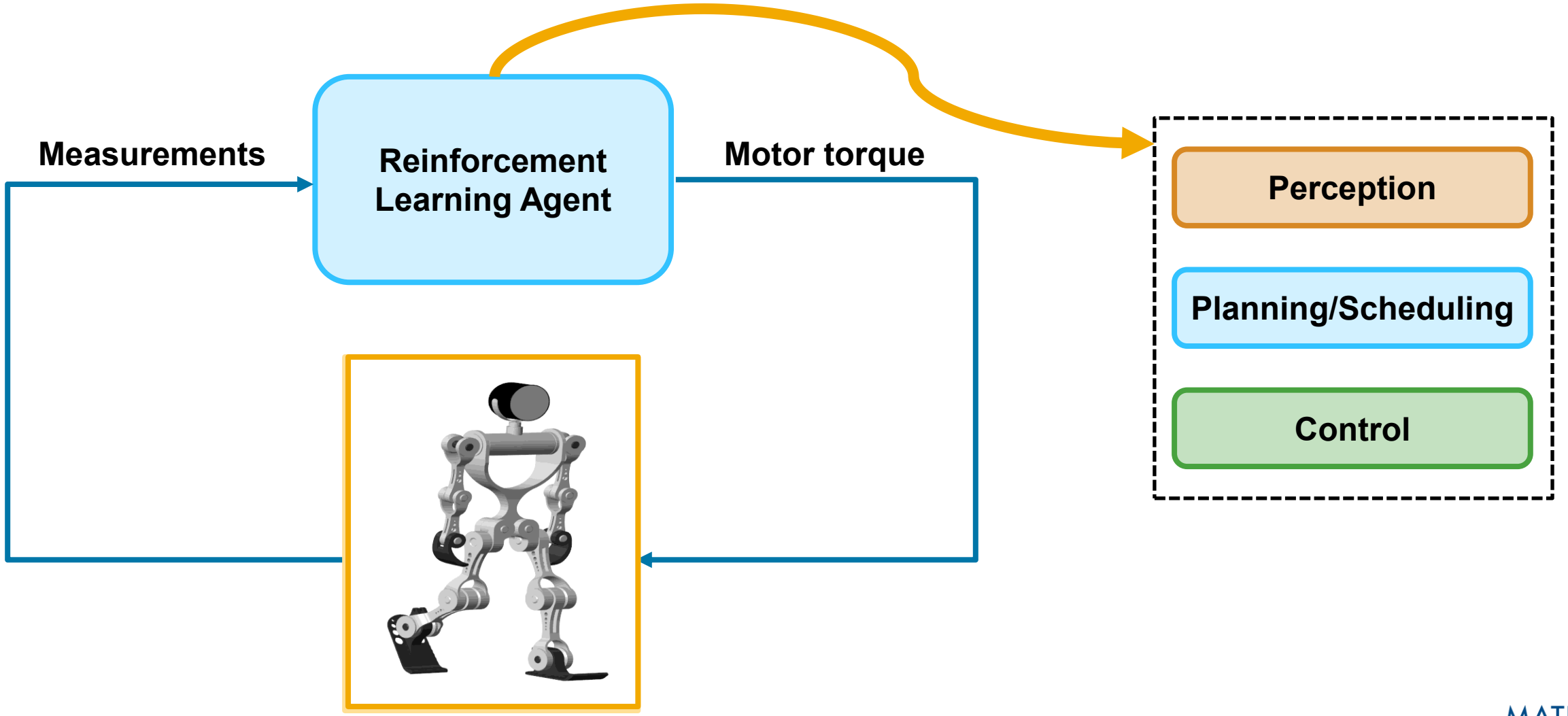
Challenges with end-to-end reinforcement learning



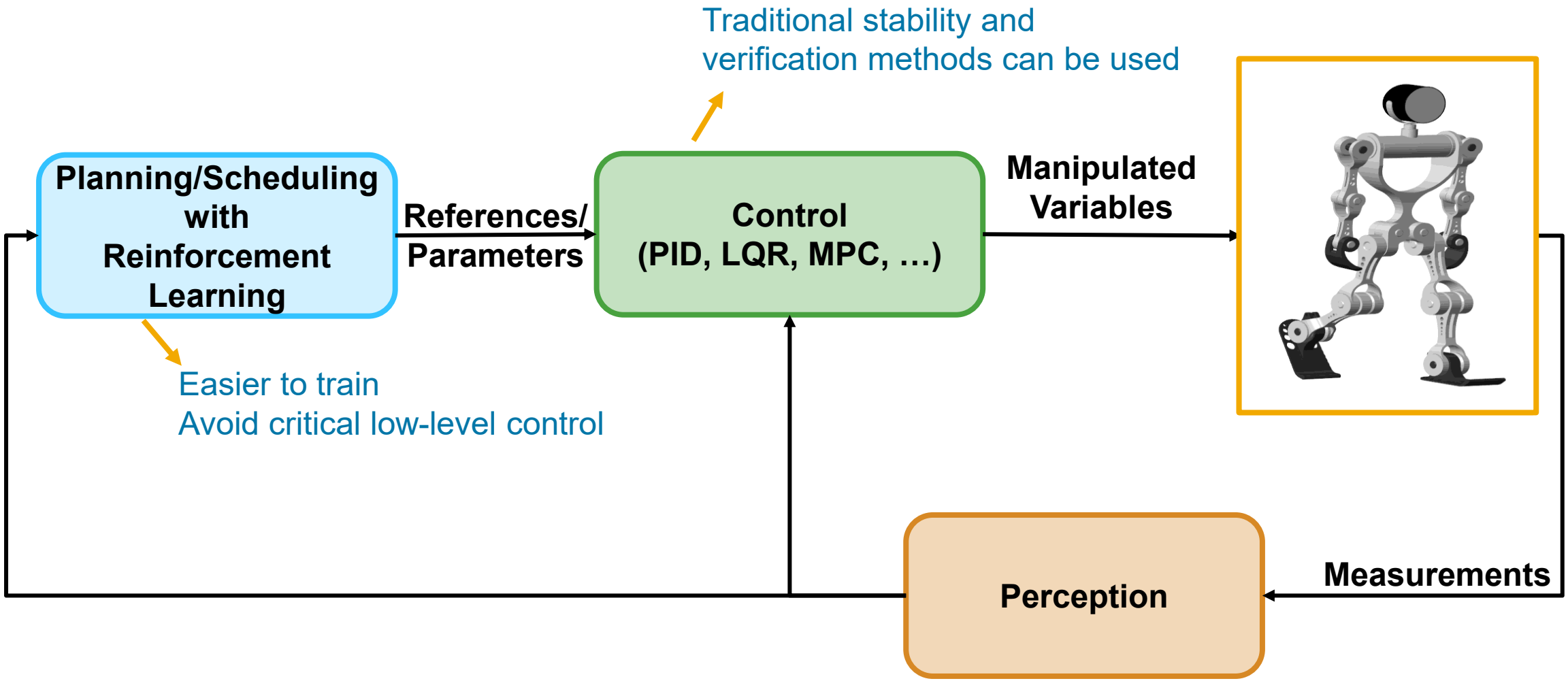


- Black box controller
- Lack of performance guarantees
- Harder to train

Alternatives to end-to-end reinforcement learning



Alternatives to end-to-end reinforcement learning



Agenda

- Introduction to reinforcement learning
- Deep dive: Teach a robot to walk! (with hands-on exercises)
  - Defining environments and reward functions
  - Creating policies and agents
  - Training, testing, and deploying policies
- Wrap-Up and Additional Resources

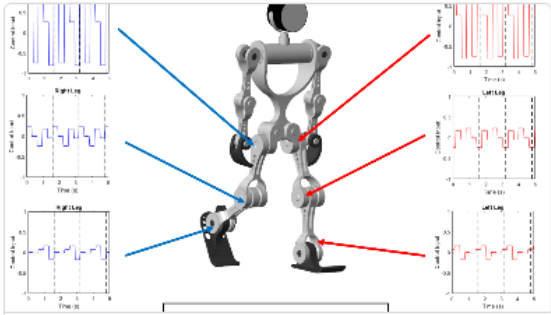
Key takeaways

- What is reinforcement learning and why is it useful?
- When can/should I use reinforcement learning?
- How do I set up and solve a reinforcement learning problem?

Reinforcement learning is used in a variety of applications

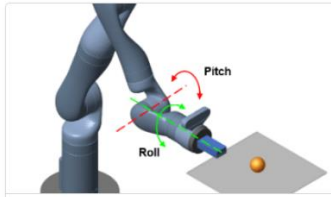
- Decision-making problems:
  - Controls, scheduling, resource allocation, calibration
- Application areas:
  - Wireless communications, flight controls, robotics, automated driving, cybersecurity, ...

Documentation provides reference examples to help you get started.



Train Humanoid Walker

Model a humanoid robot using Simscape Multibody™ and train it using either a genetic algorithm (which requires a Global Optimizatio...

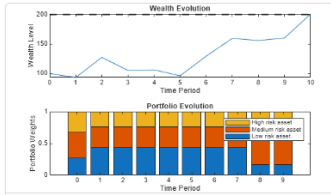


Train SAC Agent for Ball Balance Control

Train a SAC agent to balance a ball on a flat surface using a robot arm.

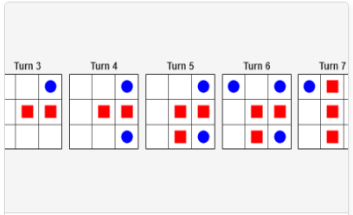
Automatic Parking Valet with Unreal Engine Simulation

Use a TD3 agent with an MPC controller to perform a parking maneuver.



Multiperiod Goal-Based Wealth Management Using Reinforcement Learning

A reinforcement learning (RL) approach to maximize the probability of obtaining an investor's wealth goal at the end of the investment horizon...



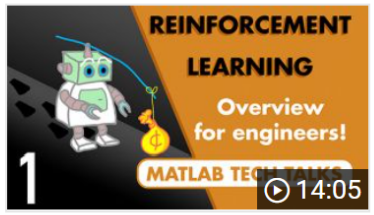
Train Agent to Play Turn-Based Game

Train a DQN agent to play a turn-based game.

[Link to RL reference examples]

[Link to demo]

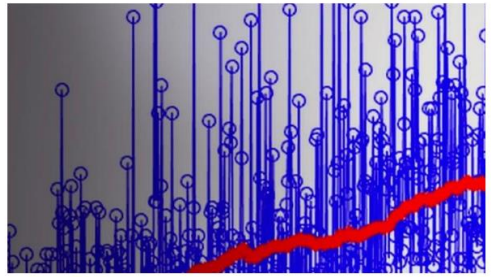
Learn more about reinforcement learning



Part 1: What Is Reinforcement Learning?

Get an overview of reinforcement learning from the perspective of an engineer. Reinforcement learning is a type of machine learning that has the potential to solve some really hard control problems.

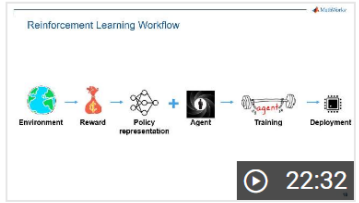
[MATLAB Tech Talk on RL]



Section 1: Understanding the Basics and Setting Up the Environment

Learn the basics of reinforcement learning and how it compares with traditional control design. See the difference between supervised, unsupervised, and reinforcement learning, and see how to set up a learning environment in MATLAB and Simulink.

[Ebook: "Guide to Understanding Reinforcement Learning"]



Reinforcement Learning: Leveraging Deep Learning for Controls

In this session, you will learn how to do reinforcement learning using MathWorks products, including how to set up environment models, define the policy structure and scale training through parallel computing...

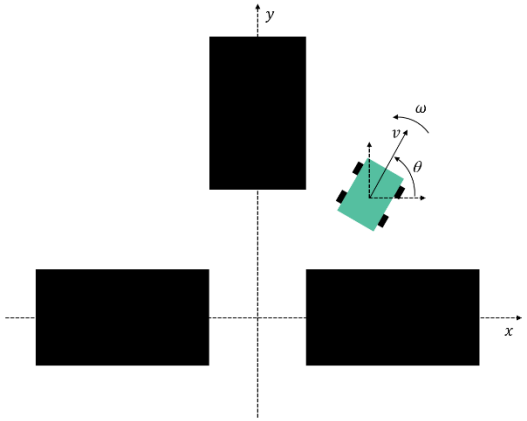
Date: 19 Mar 2020

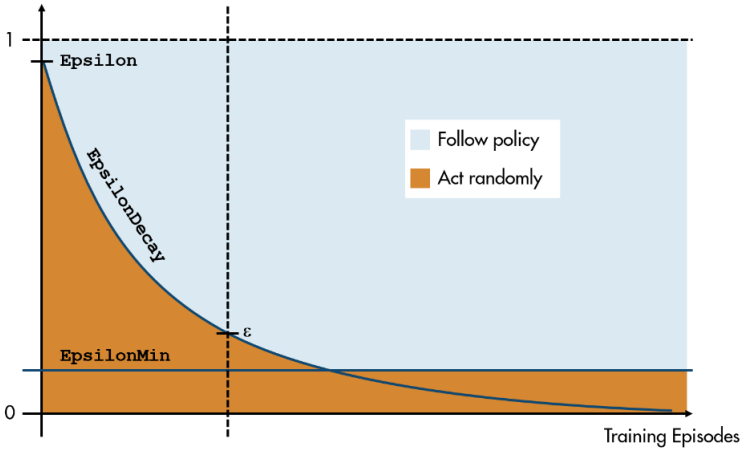
Audio: English

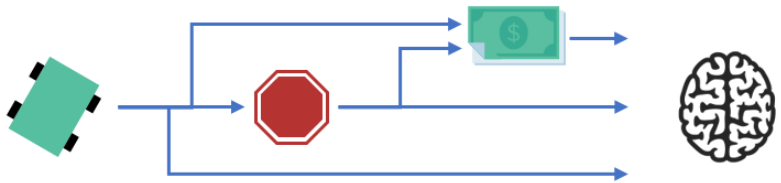
[Videos and recorded webinars]

Reinforcement Learning Onramp

- Free, short course on reinforcement learning methods for control problems
- Hands-on exercises and short video demonstrations
- Learn by working through an example of navigating a robot through a warehouse
- Topics include:
  - Simulating with a pretrained agent
  - Defining environments and agents
  - Creating neural networks
  - Training agents







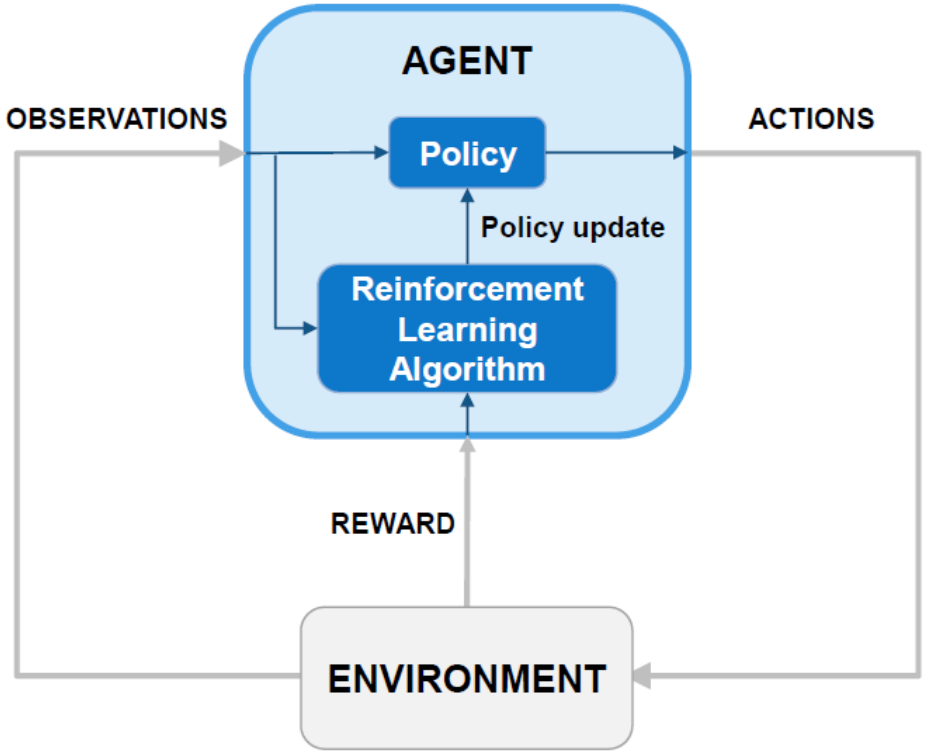
[Link to Reinforcement Learning Onramp]

Training: Reinforcement Learning in MATLAB and Simulink

Topic included in this 1-day training:

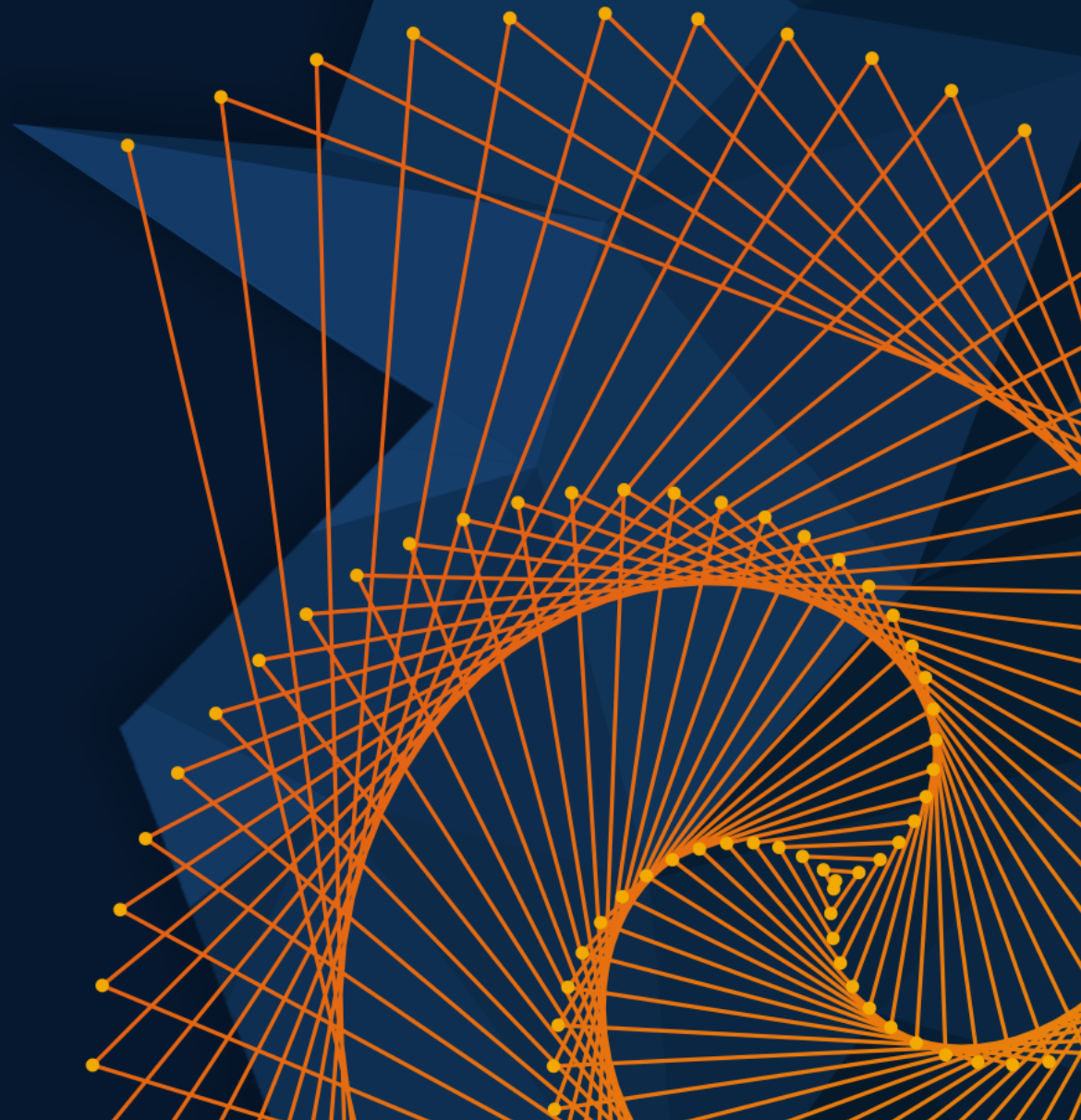
- Set up environment and rewards
- Represent a policy and create agent
- Training neural networks
- Generate code

[See detailed course outline]



MATLAB EXPO

Thank you



MathWorks®