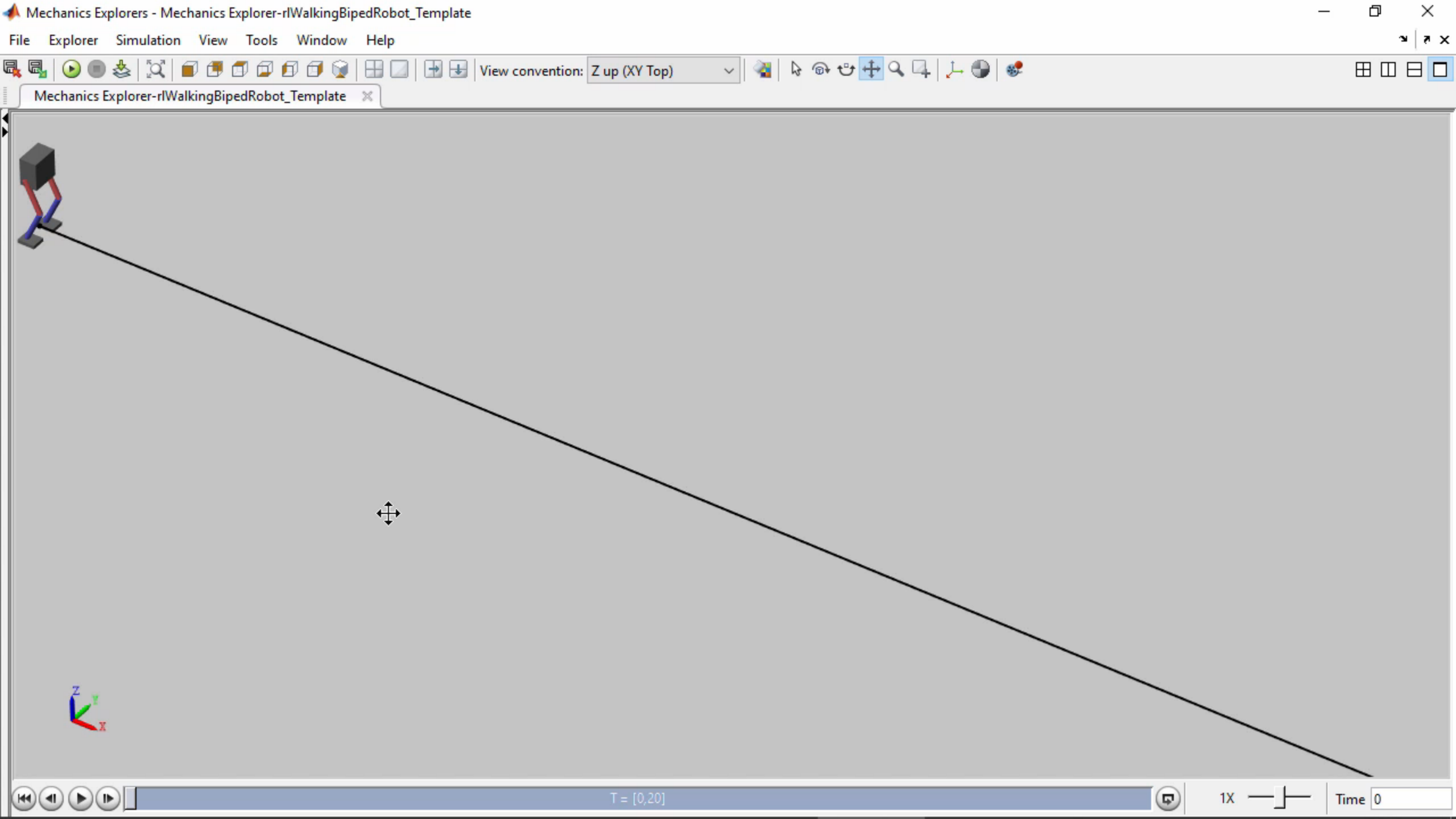
Reinforcement Learning: Leveraging Deep Learning for Controls

Christoph Stockhammer

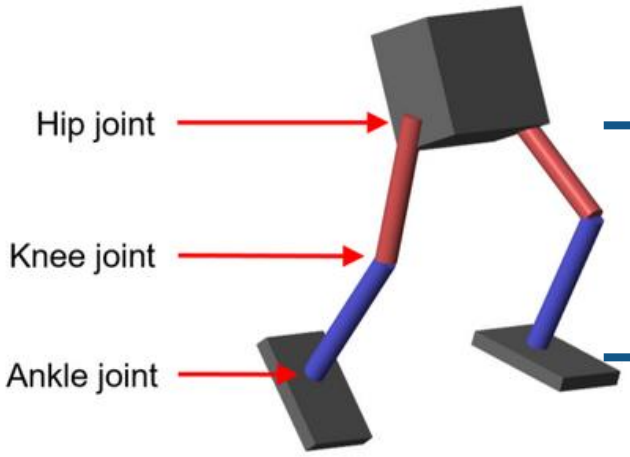
MathWorks Application Engineering


Mechanics Explorers - Mechanics Explorer-rlWalkingBipedRobot_Template
File
Explorer
Simulation
View
Tools
Window
Help
View convention:
Z up (XY Top)
Mechanics Explorer-rlWalkingBipedRobot_Template
T = [0,20]
1X
Time
0

Goal: We hope you walk away knowing the answer to these questions

- What is reinforcement learning and why should I care about it?
- How do I set it up and solve it? [*from an engineer's perspective*]
- What are some benefits and drawbacks?

Let's try to solve this problem the traditional way



Observations

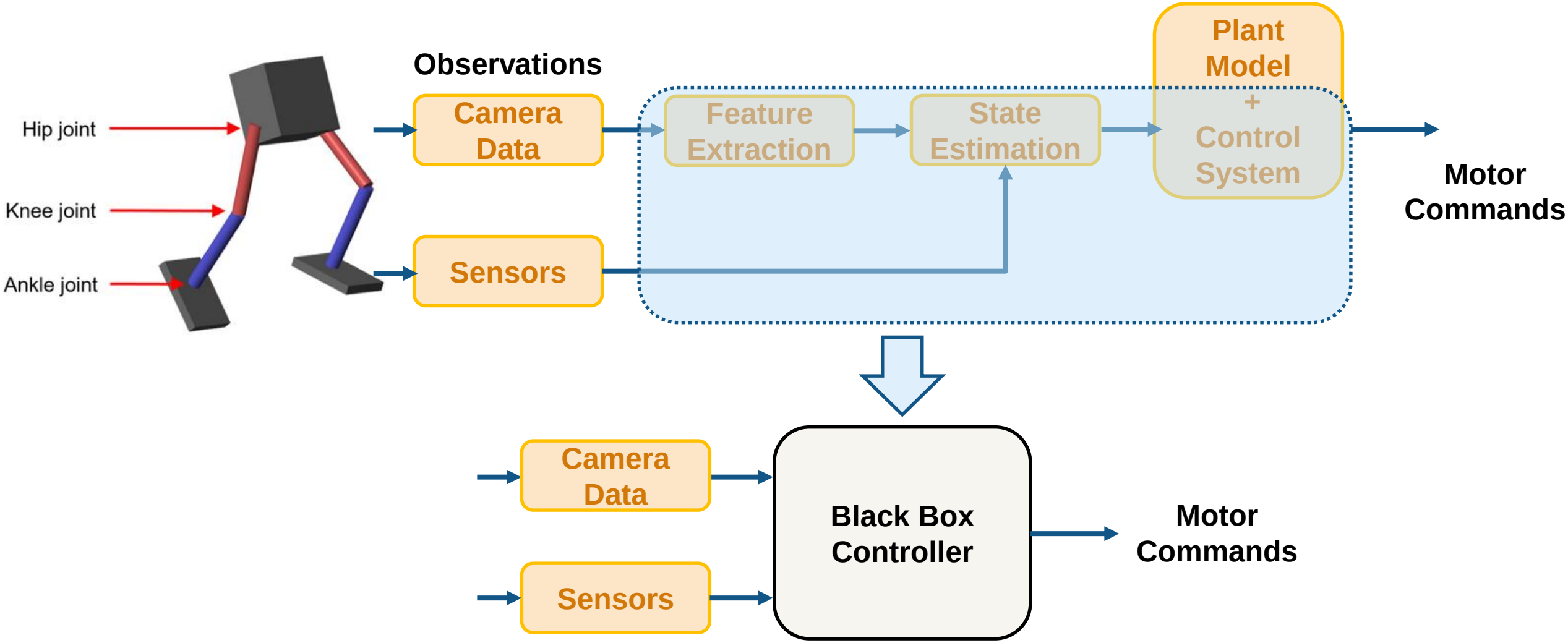
Camera Data → Feature Extraction → State Estimation → Plant Model + Control System → Motor Commands

Sensors → State Estimation

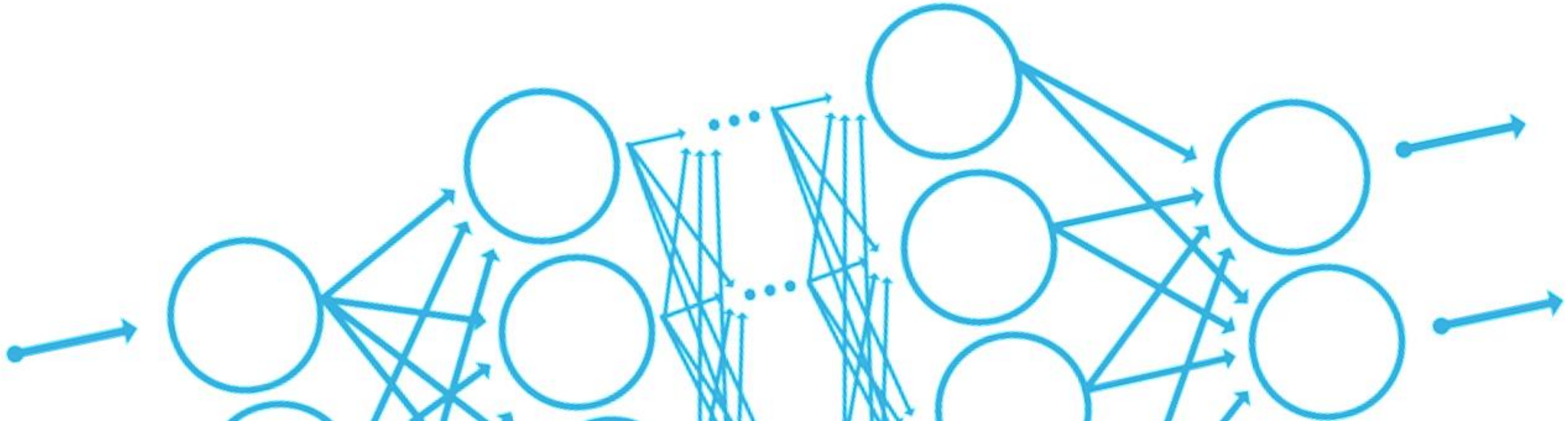
Balance → Leg & Trunk Trajectories → Motor Control → Motor Commands

Observations

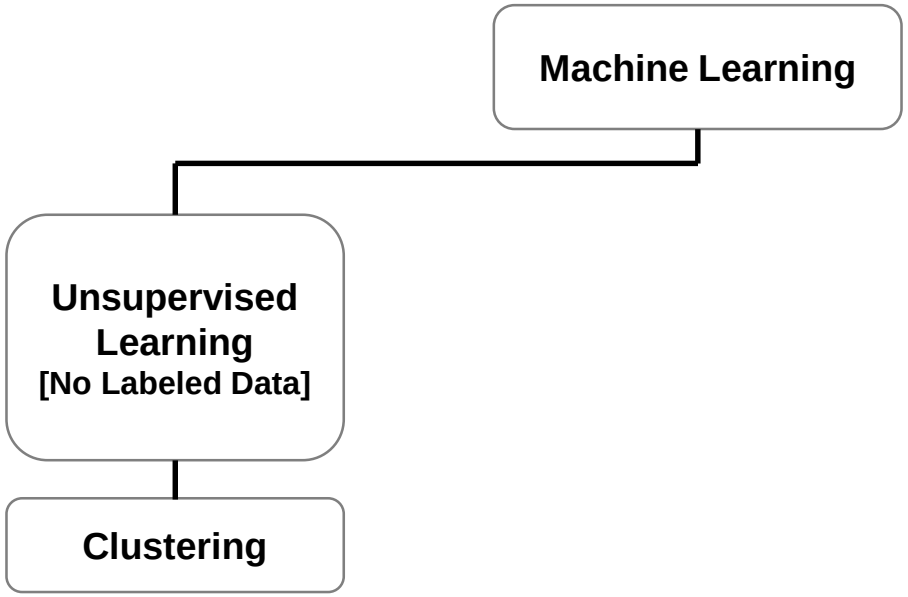
What is the alternative approach?

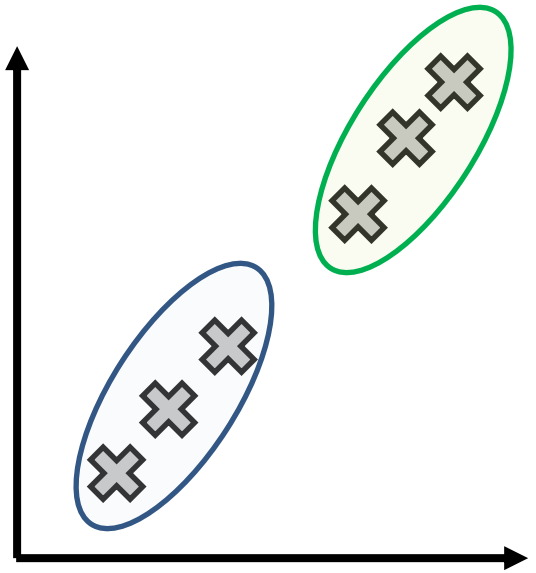


What is reinforcement learning?

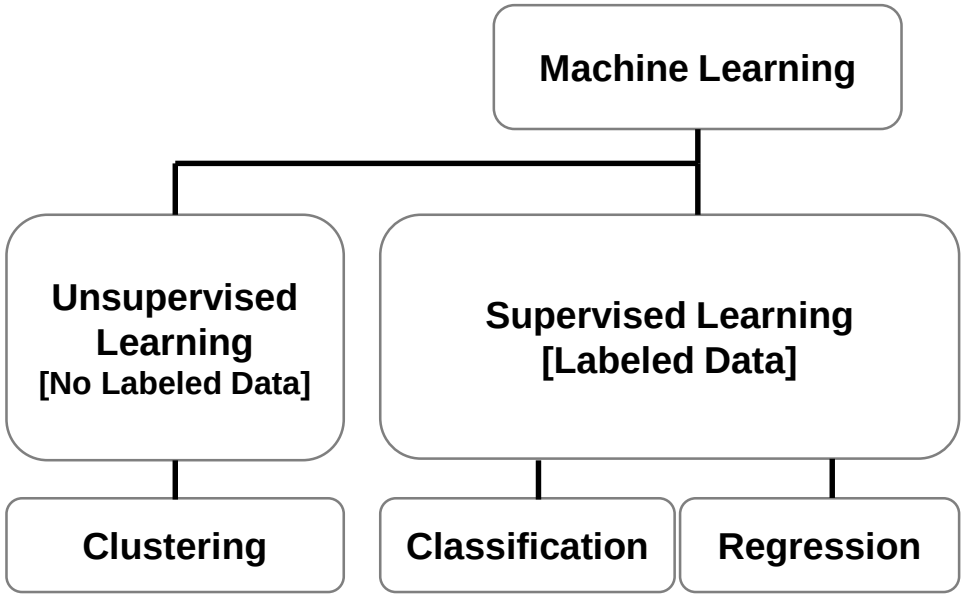


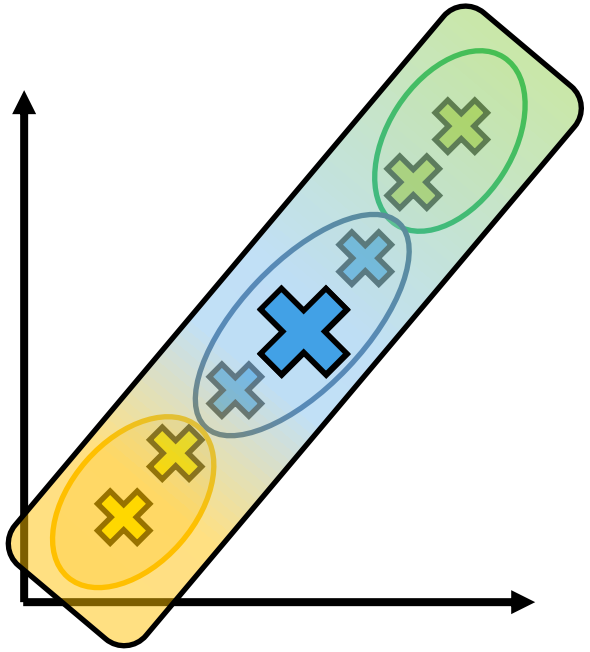
Reinforcement Learning vs Machine Learning vs Deep Learning



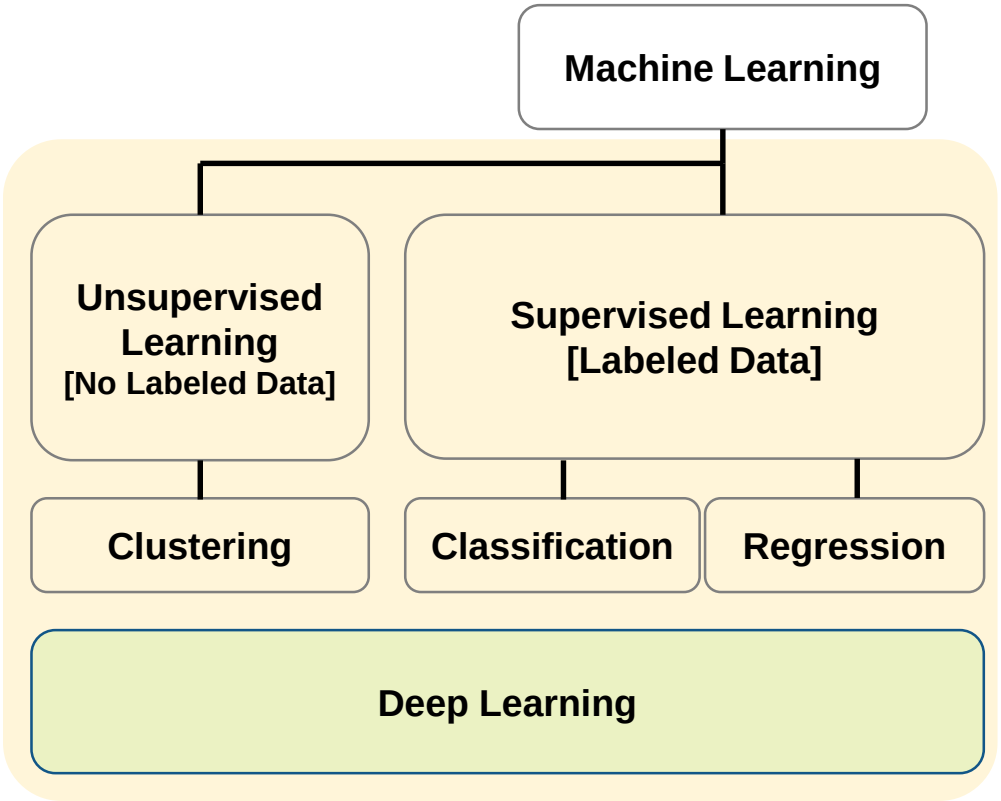


Reinforcement Learning vs Machine Learning vs Deep Learning





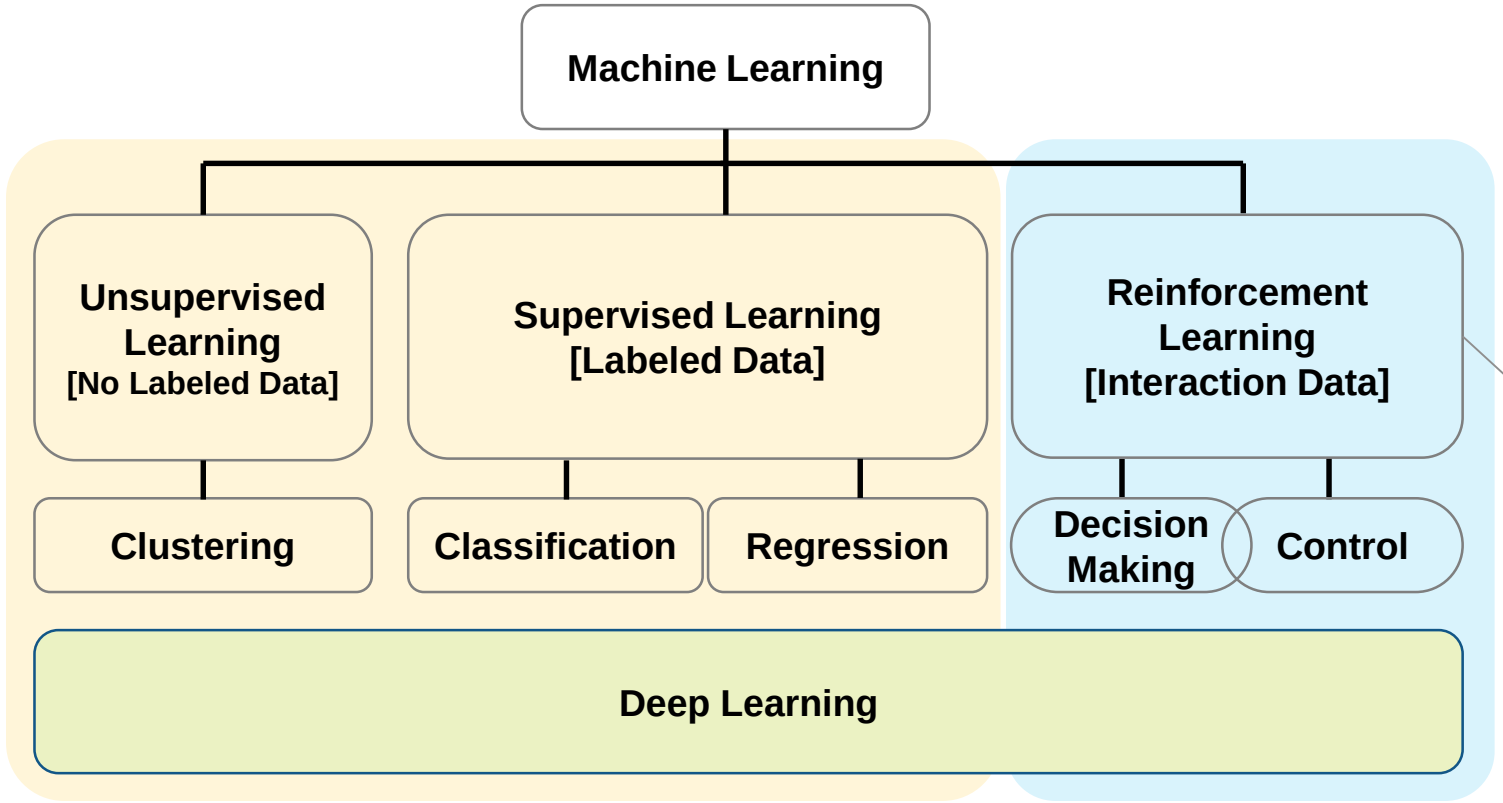
Reinforcement Learning vs Machine Learning vs Deep Learning



Supervised learning typically involves feature extraction

Deep learning typically simplifies feature extraction

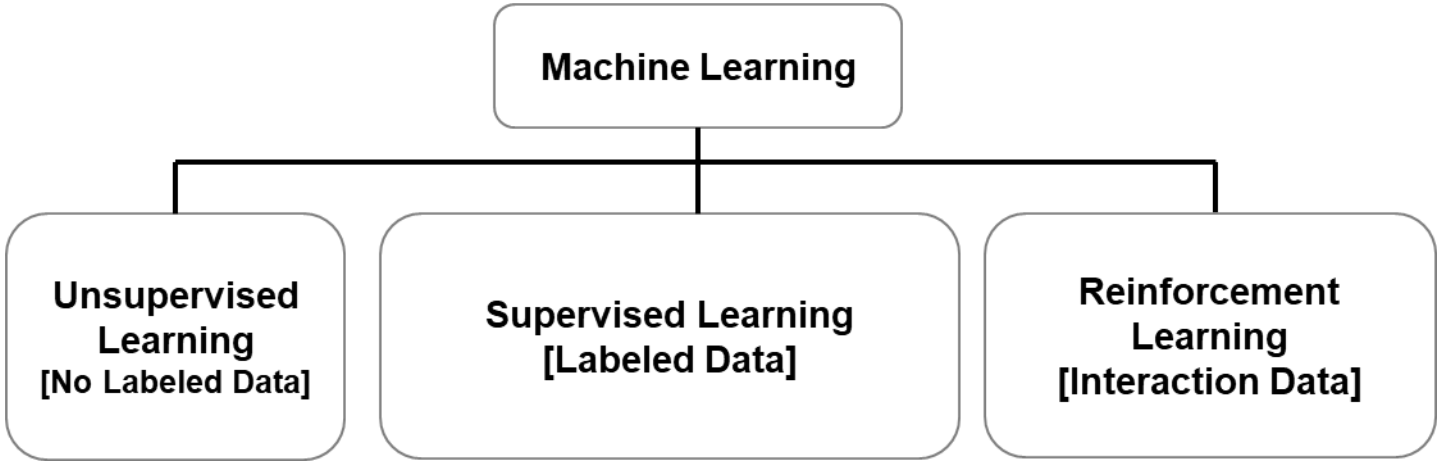
Reinforcement Learning vs Machine Learning vs Deep Learning



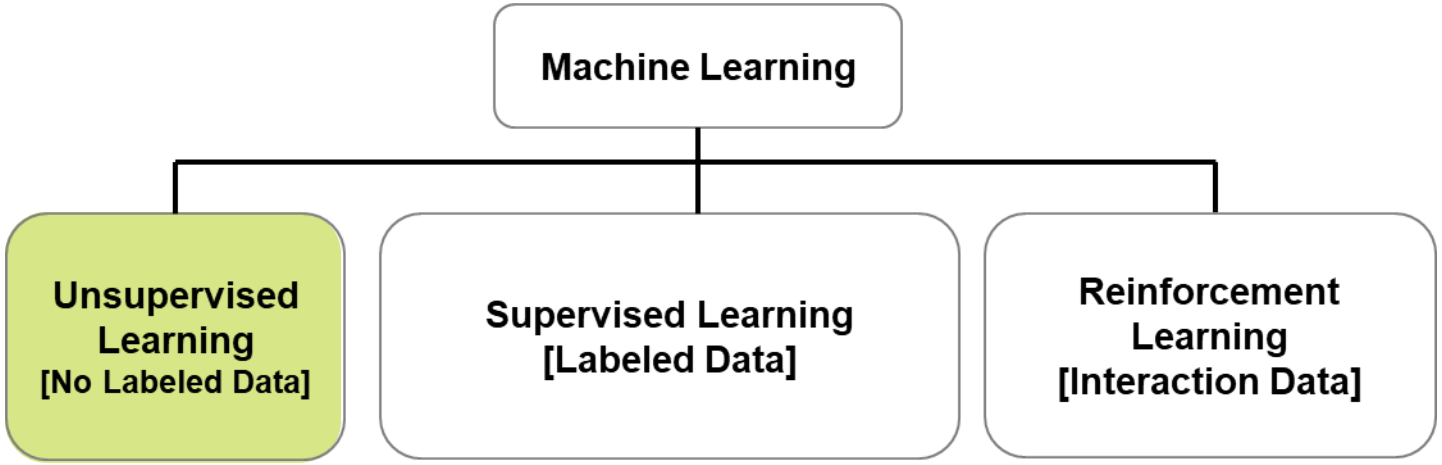
Reinforcement learning:

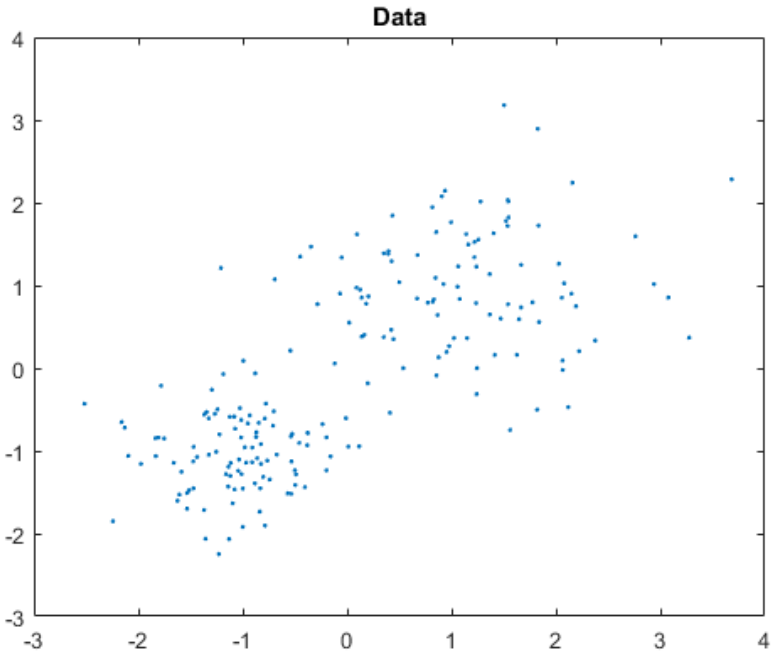
- Learning through trial & error [*interaction*]
- Complex problems typically need deep learning [*Deep Reinforcement Learning*]
- It's about learning a **behavior** or accomplishing a **task**

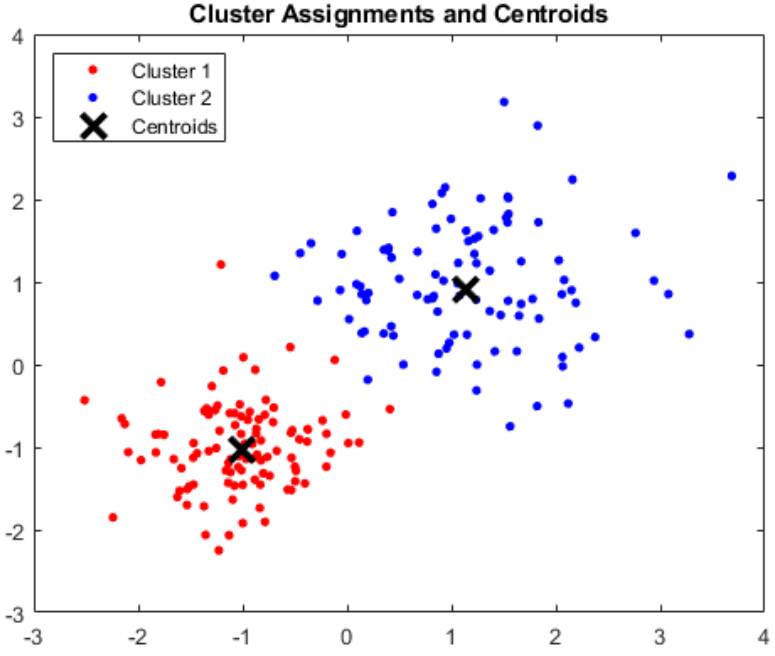
Reinforcement Learning vs Machine Learning vs Deep Learning



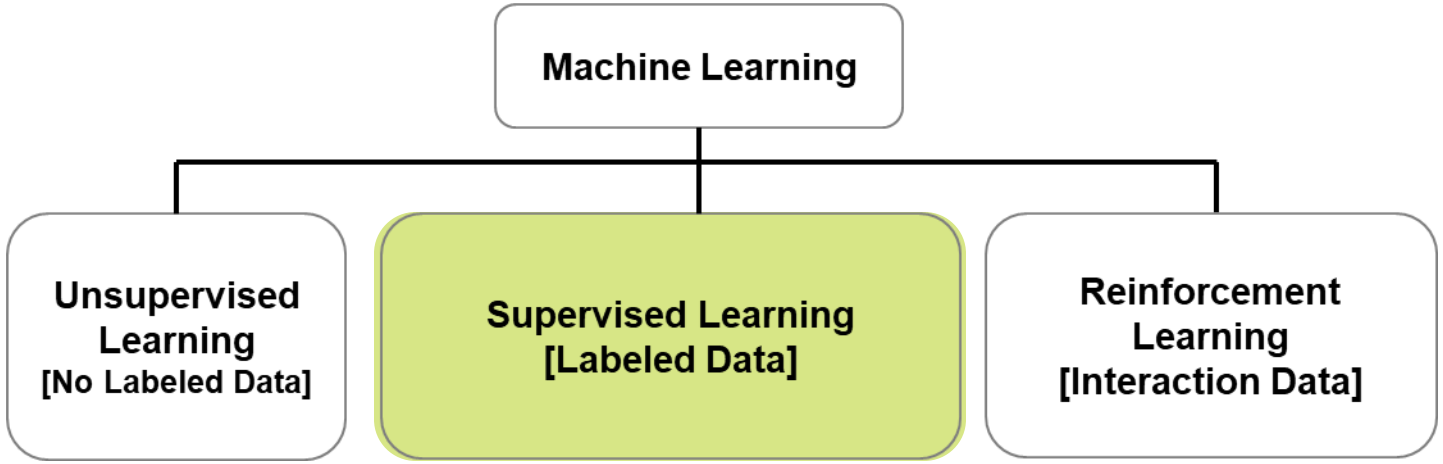
Reinforcement Learning vs Machine Learning vs Deep Learning

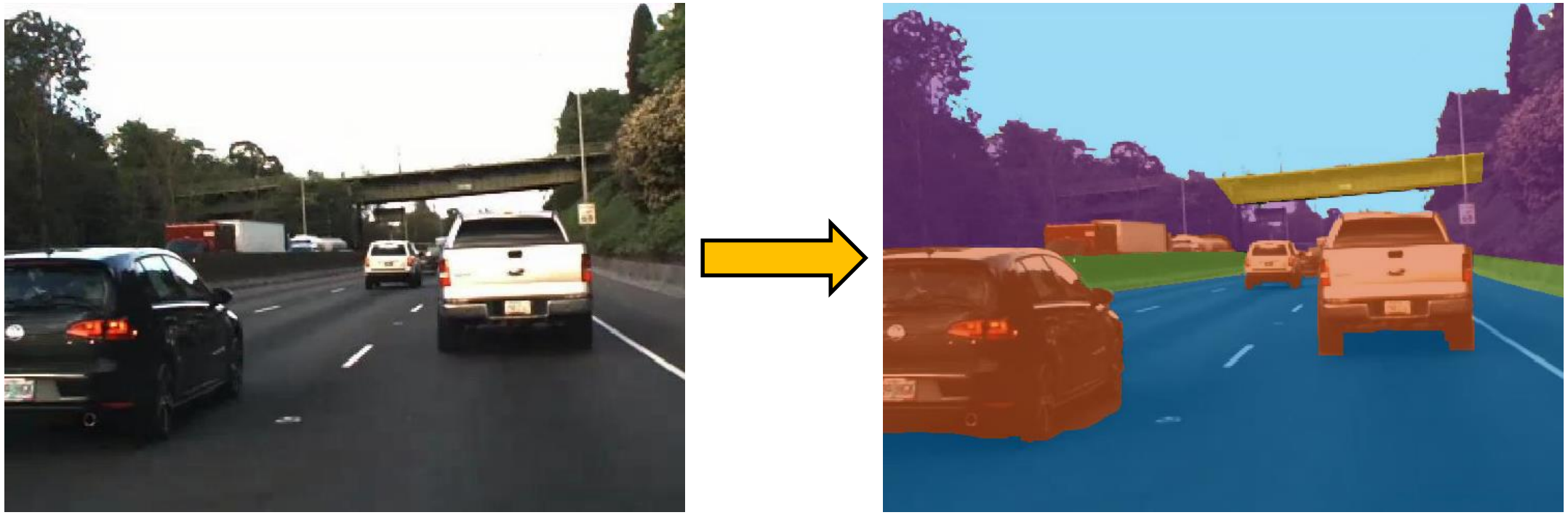




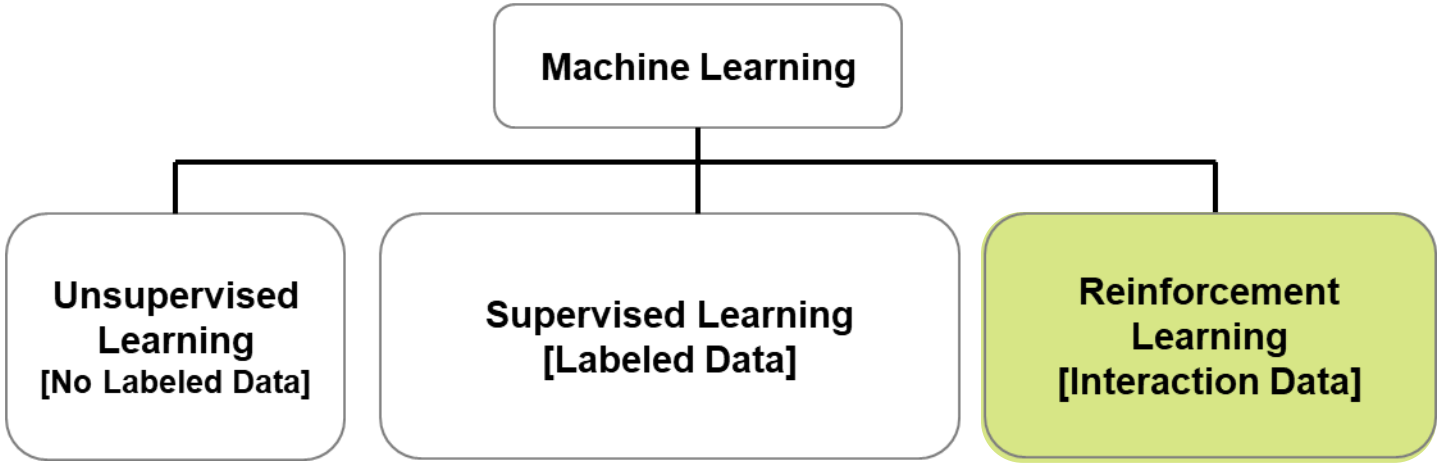


Reinforcement Learning vs Machine Learning vs Deep Learning



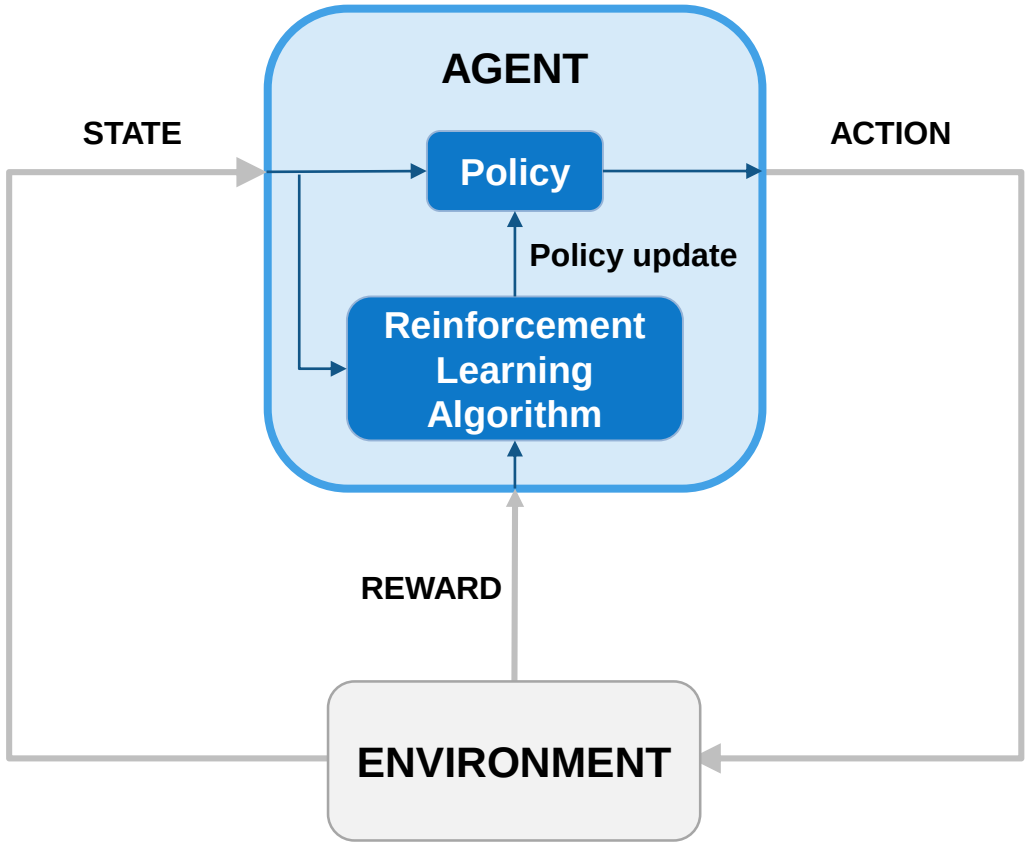


Reinforcement Learning vs Machine Learning vs Deep Learning



A Practical Example of Reinforcement Learning

Training a Self-Driving Car

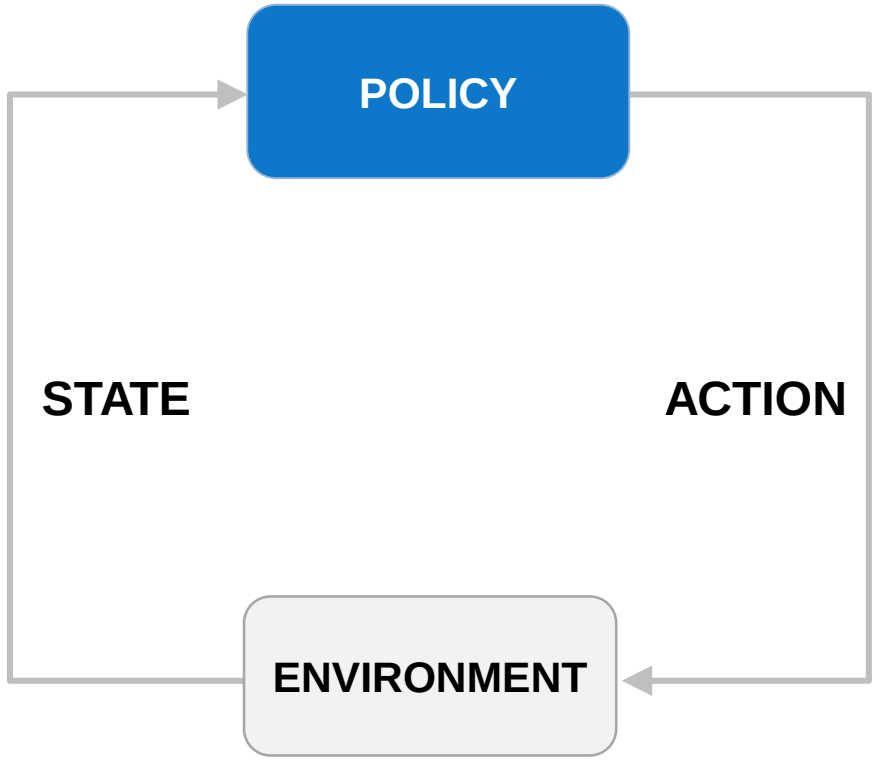


- Vehicle's computer learns how to drive… (**agent**)
- using sensor readings from LIDAR, cameras,… (**state**)
- that represent road conditions, vehicle position,… (**environment**)
- by generating steering, braking, throttle commands,… (**action**)
- based on an internal state-to-action mapping… (**policy**)
- that tries to get you from A to B without an accident while possibly optimizing driver comfort & fuel efficiency… (**reward**).

- The policy is updated through repeated trial-and-error by a **reinforcement learning algorithm**

A Practical Example of Reinforcement Learning

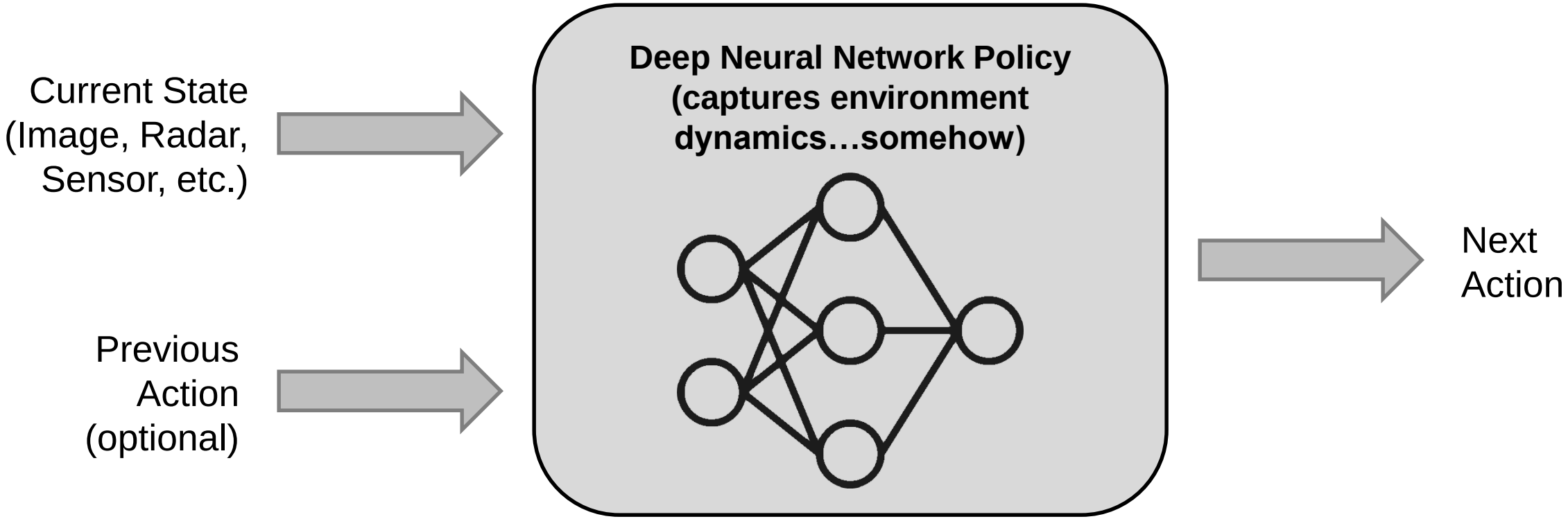
A Trained Self-Driving Car Only Needs A Policy To Operate



- Vehicle's computer uses the final state-to-action mapping… (**policy**)
- to generate steering, braking, throttle commands,… (**action**)
- based on sensor readings from LIDAR, cameras,… (**state**)
- that represent road conditions, vehicle position,… (**environment**)

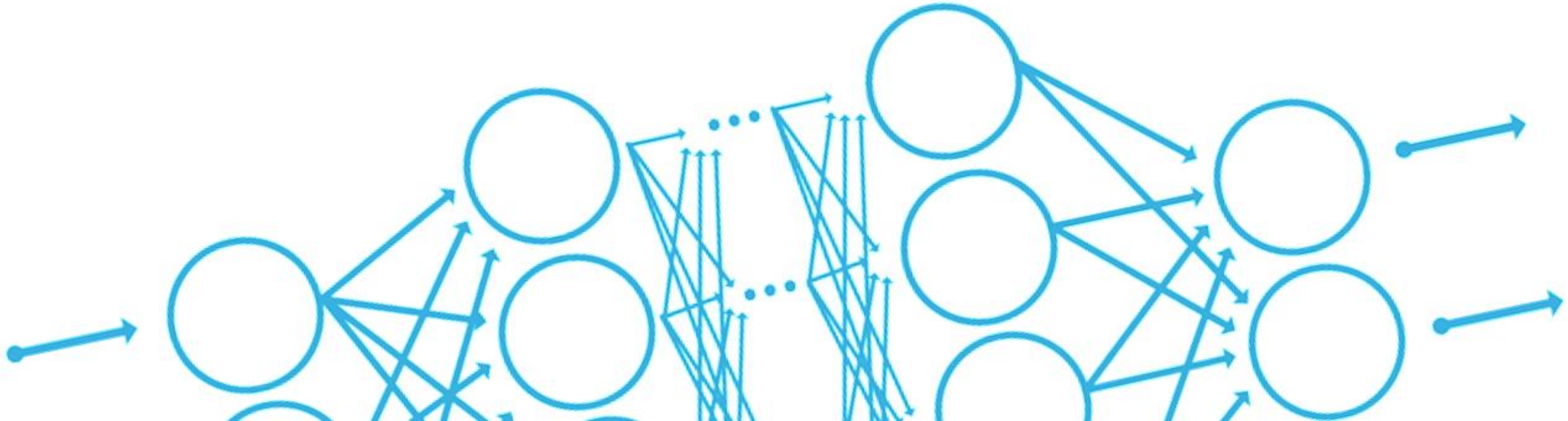
By definition, this trained policy puts into practice what is in the reward function

A deep neural network trained using reinforcement learning is a black-box model that determines the best possible action

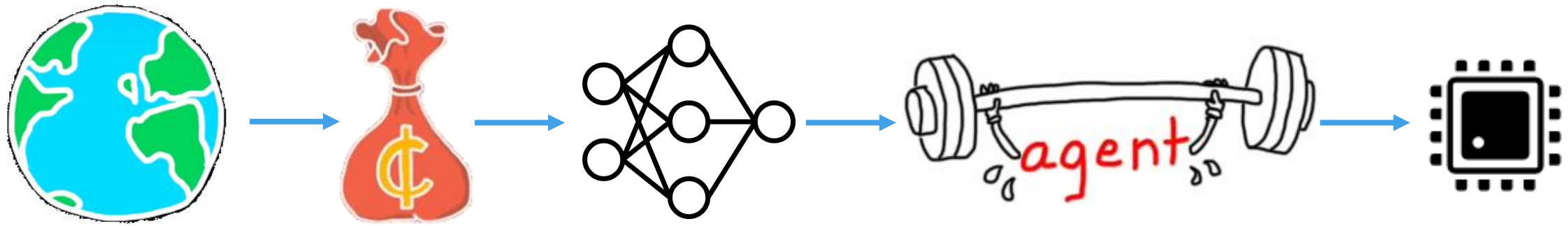


By representing policies using deep neural networks, we can solve problems for **complex, non-linear systems** (continuous or discrete) by directly using **data that traditional approaches cannot use easily**

How do I set it up and solve it?

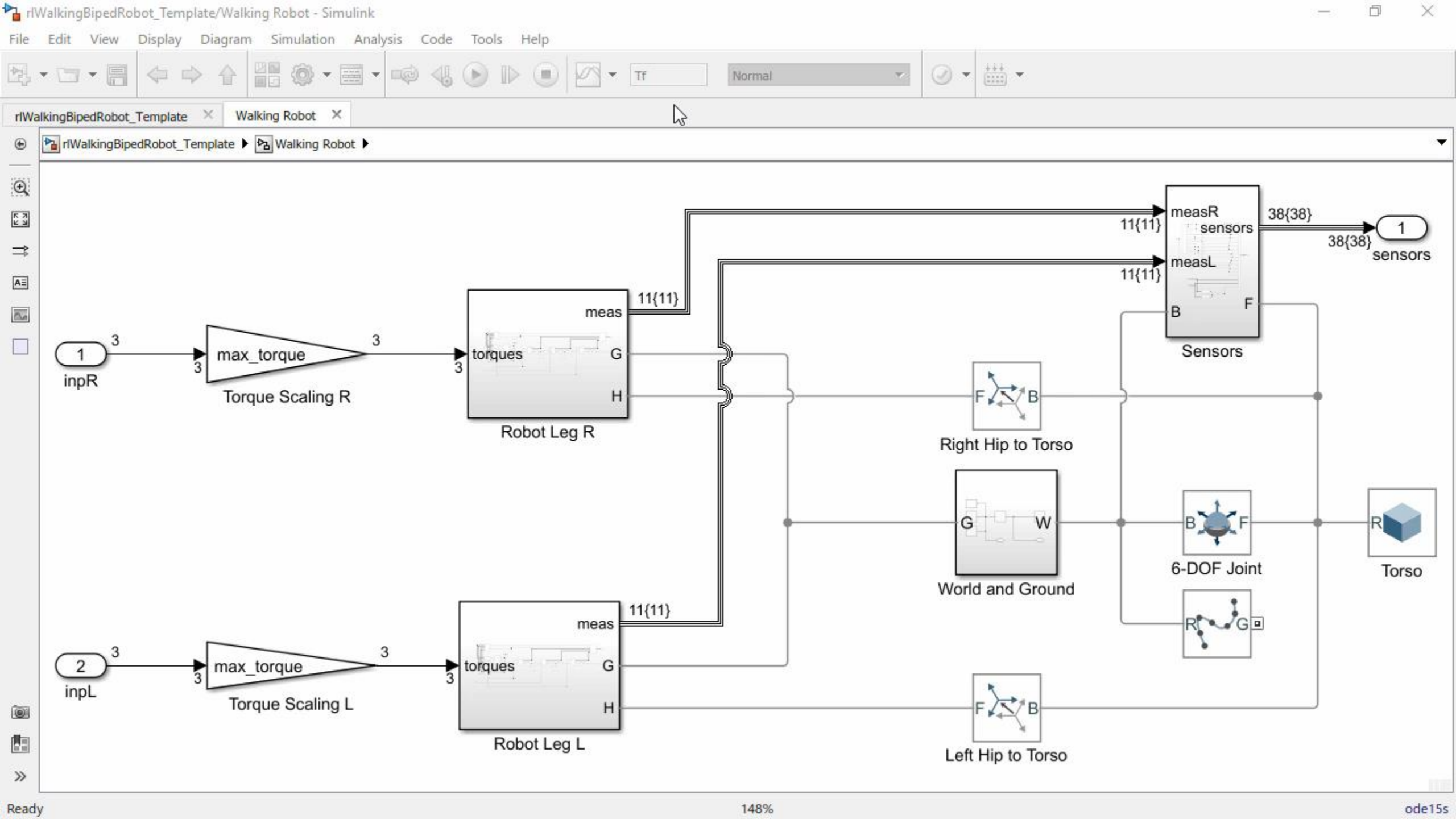


Steps in the Reinforcement Learning Workflow

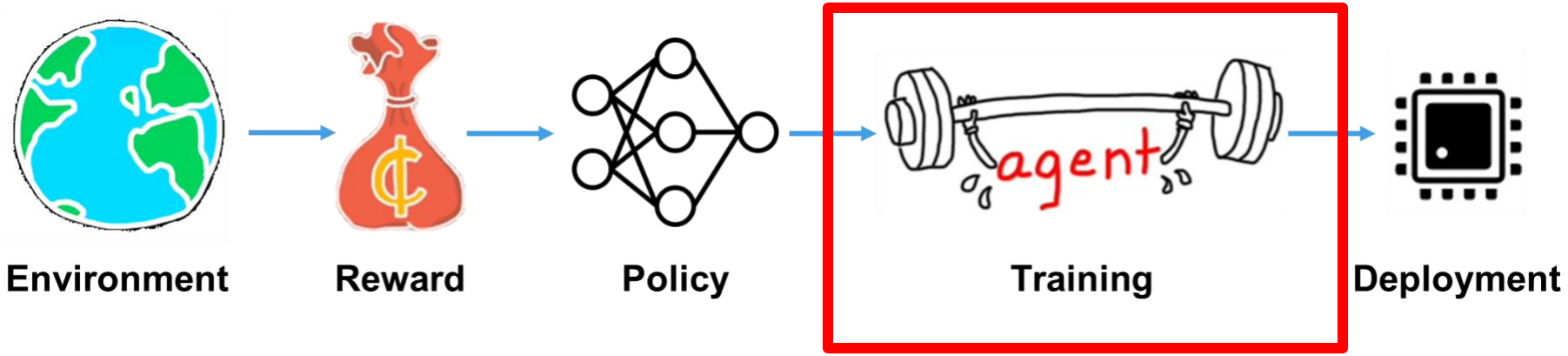


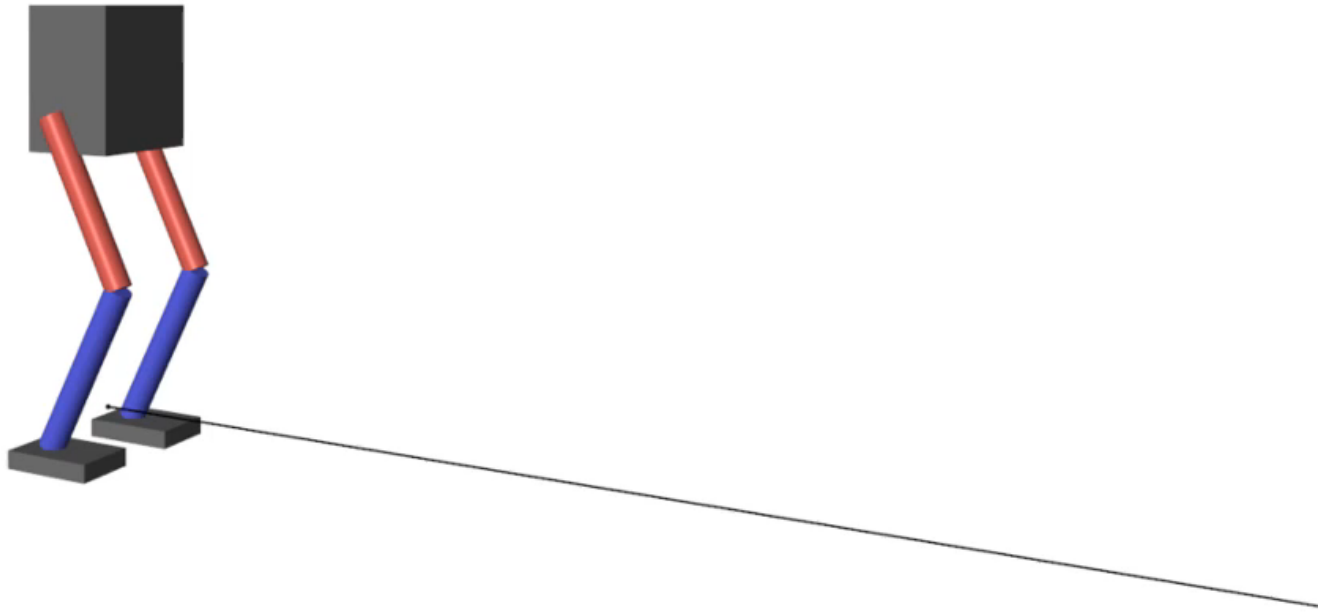
Environment **Reward** **Policy** **Training** **Deployment**

Environment

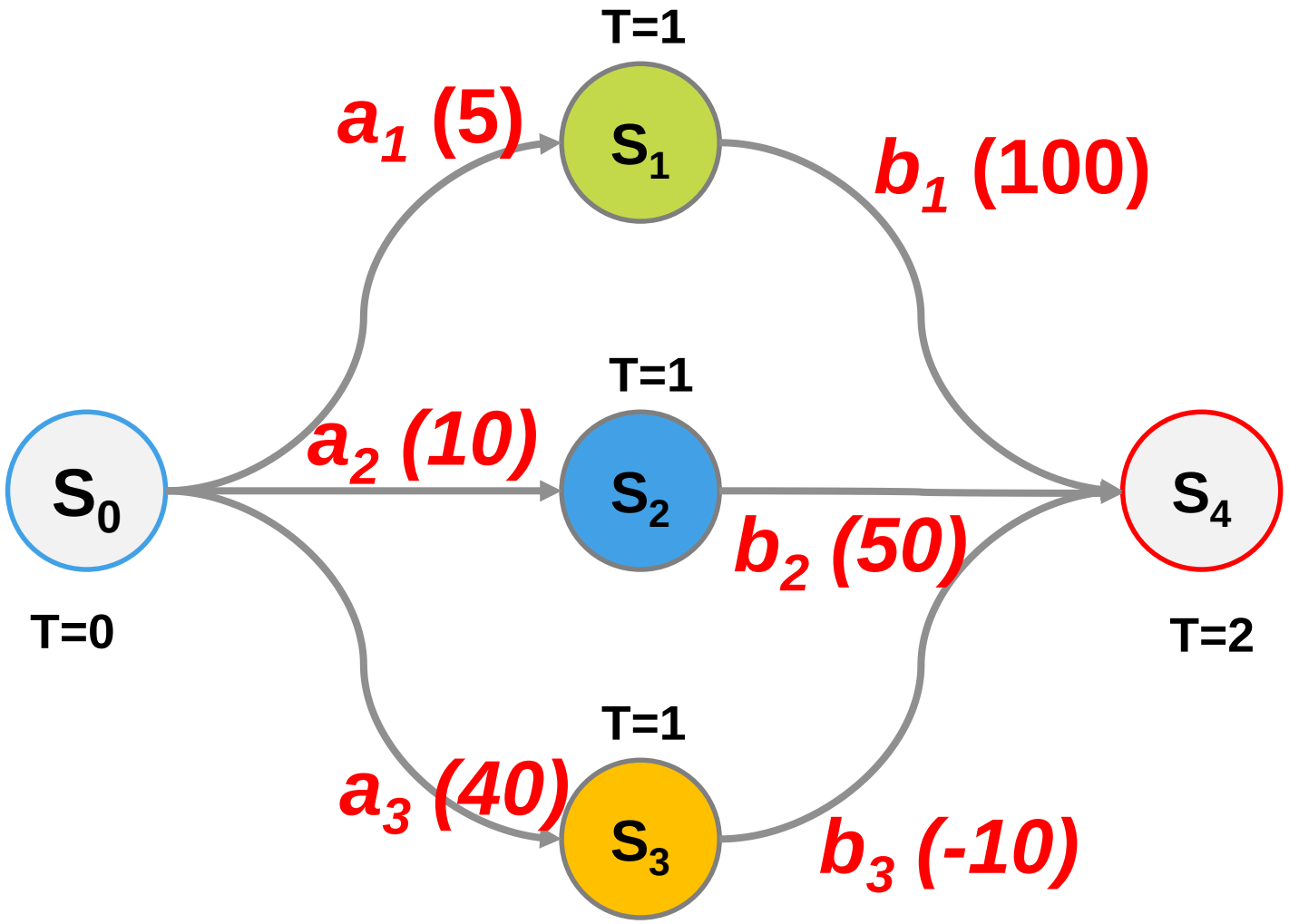

rlWalkingBipedRobot_Template/Walking Robot - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
Tf
Normal
rlWalkingBipedRobot_Template
Walking Robot
rlWalkingBipedRobot_Template
Walking Robot
1
inpR
3
max_torque
Torque Scaling R
meas
torques
G
H
Robot Leg R
11{11}
2
inpL
Torque Scaling L
Robot Leg L
measR
sensors
measL
B
F
Sensors
38{38}
1
sensors
F
B
Right Hip to Torso
G
W
World and Ground
B
F
6-DOF Joint
R
G
R
Torso
Left Hip to Torso
Ready
148%
ode15s

Steps in the Reinforcement Learning Workflow





Reward Function

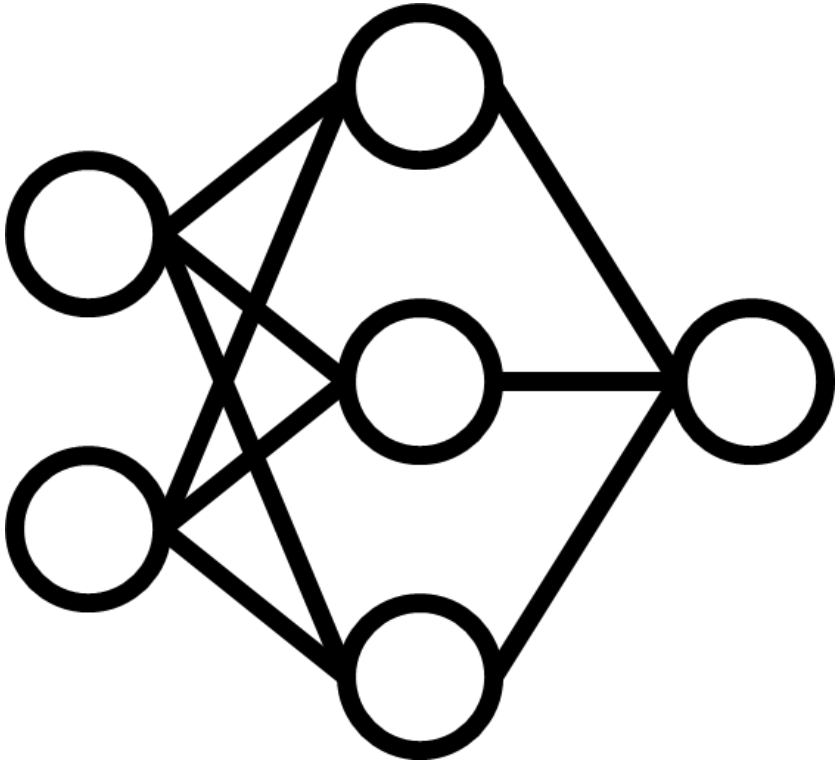


Long-Term Reward (Q)

- $Q(S_0,a_1)$ = **+105 (optimal)**
- $Q(S_0,a_2)$ = **+60**
- $Q(S_0,a_3)$ = **+30**

The logic you used right now to decide you should take action a_1 is a policy

Policy & Agent



Live Editor - \\central-muc\home\cstockha\Documents\MATLAB\Examples\R2019a\rl\RLWalkingBipedRobotExample\RLWalkingBipedRobotExample.mlx

LIVE EDITOR | INSERT | VIEW

New | Open | Save | Find Files | Compare | Print | Go To | Find | Text | Normal | Code | Control | Refactor | Section Break | Run Section | Run and Advance | Run to End | Run | Step | Stop

RLWalkingBipedRobotExample.mlx | createNetworks.m

- T_s is the sample time of the environment.
- T_f is the final simulation time of the environment.

Create Environment Interface

Create the observation specification.

```
numObs = 29;
obsInfo = rlNumericSpec([numObs 1]);
obsInfo.Name = 'observations';
```

Create the action specification.

```
numAct = 6;
actInfo = rlNumericSpec([numAct 1],'LowerLimit',-1,'UpperLimit', 1);
actInfo.Name = 'foot_torque';
```

Create the environment interface for the walking robot model.

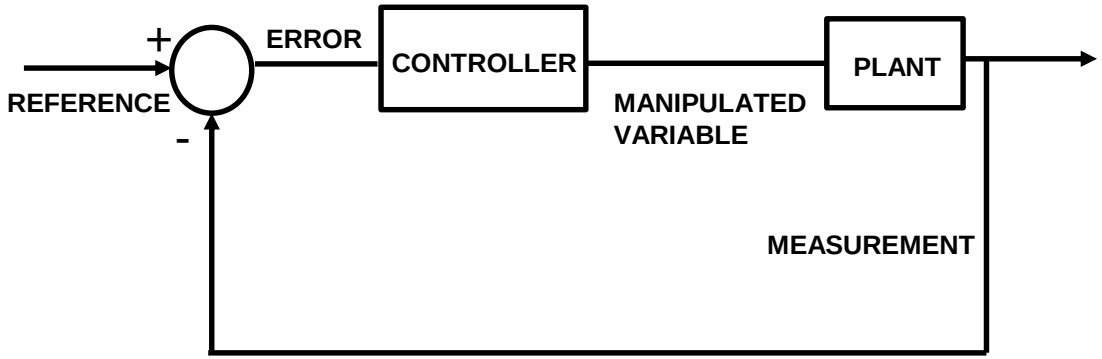
```
blk = [mdl,'/RL Agent'];
env = rlSimulinkEnv(mdl,blk,obsInfo,actInfo);
env.ResetFcn = @(in) walkerResetFcn(in,upper_leg_length/100,lower_leg_length/100,h/100);
```

Create DDPG agent

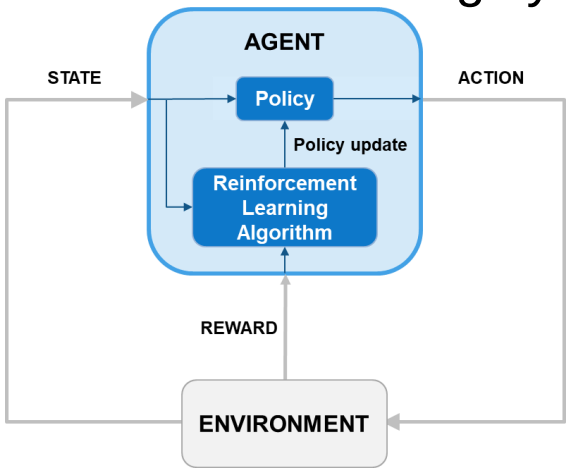
A DDPG agent approximates the long-term reward given observations and actions using a critic value function representation. A DDPG agent decides which action to take given observations using an actor representation. The actor and critic networks for this example are inspired by [2].

Reinforcement Learning vs Controls

Control system



Reinforcement learning system



Adaptation mechanism	RL Algorithm
Error/Cost function	Reward
Manipulated variable	Action
Measurement	Observation
Plant	Environment
Controller	Policy

Reinforcement learning has parallels to **control system design**

When would you use Reinforcement Learning?

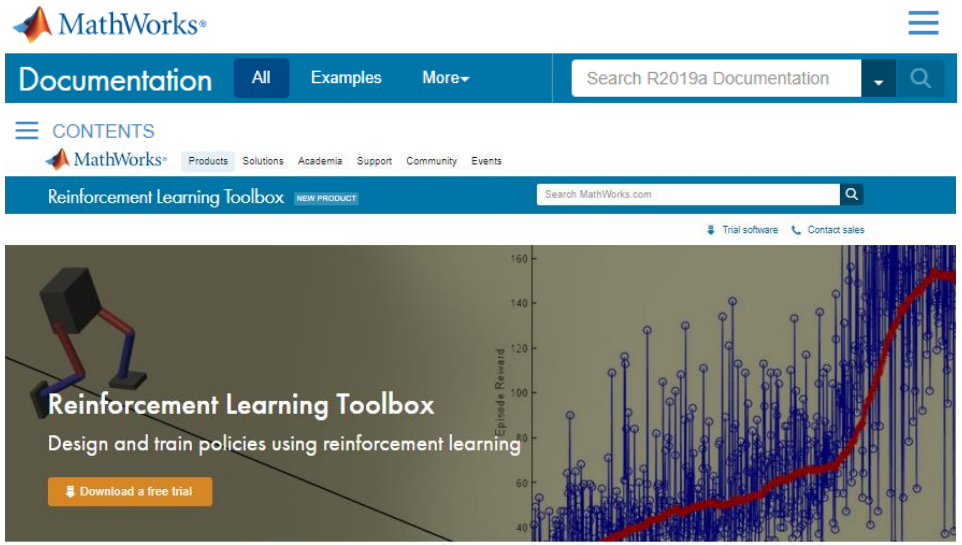
	Controller Capability	Computational Cost in Training/Tuning	Computational Cost in Deployment
PID	Low	Low	Low
Model Pred Control	High	Low	High
Reinforcement Learning	Very High	High	Medium

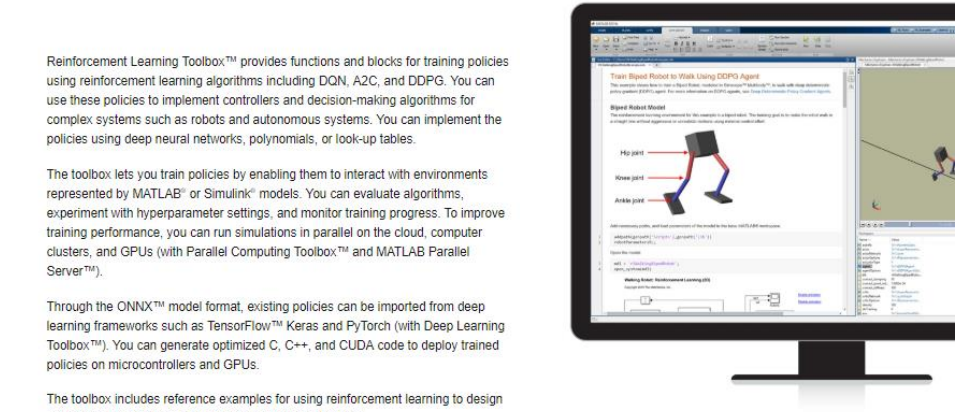
Reinforcement learning might be a good fit if

- An environment model is available (trial & error on hardware can be expensive), and
- Training/tuning time is **not** critical for the application, and
- Uncertain environments or nonlinear environments

Reinforcement Learning Toolbox
New in R2019a

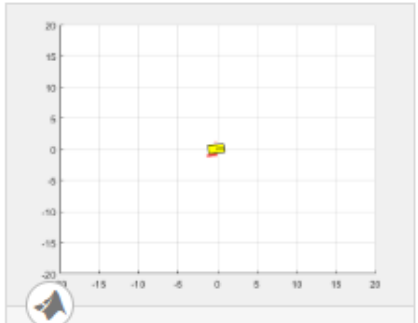
- Built-in and custom algorithms for reinforcement learning
- Environment modeling in MATLAB and Simulink
- Deep Learning Toolbox support for designing policies
- Training acceleration through GPUs and cloud resources
- Deployment to embedded devices and production systems
- Reference examples for getting started





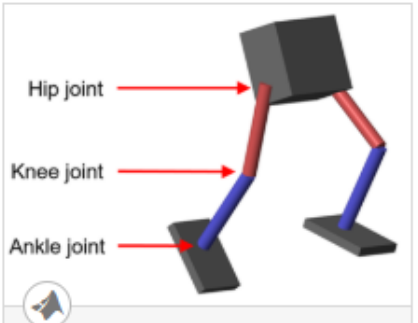
Automotive Applications

- Controller Design
- Lane Keep Assist
- Adaptive Cruise Control
- Path Following Control
- Trajectory Planning



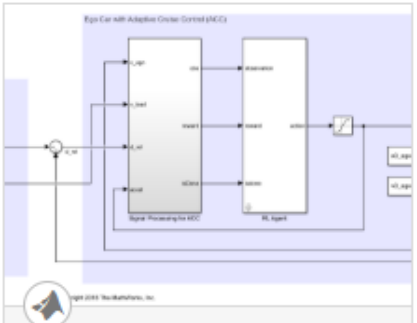
Train DDPG Agent to Control Flying Robot

Train a reinforcement learning agent to control a flying robot model.



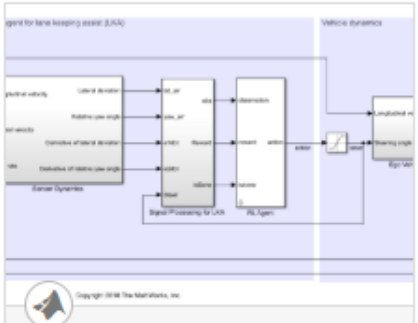
Train Biped Robot to Walk Using DDPG Agent

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



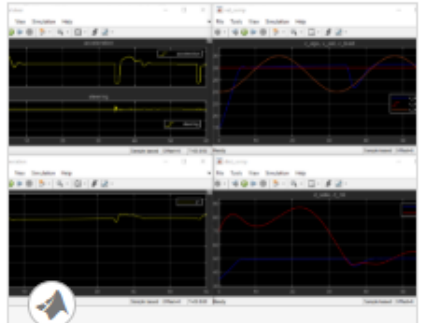
Train DDPG Agent for Adaptive Cruise Control

Train a reinforcement learning agent for an adaptive cruise control application.



Train DQN Agent for Lane Keeping Assist

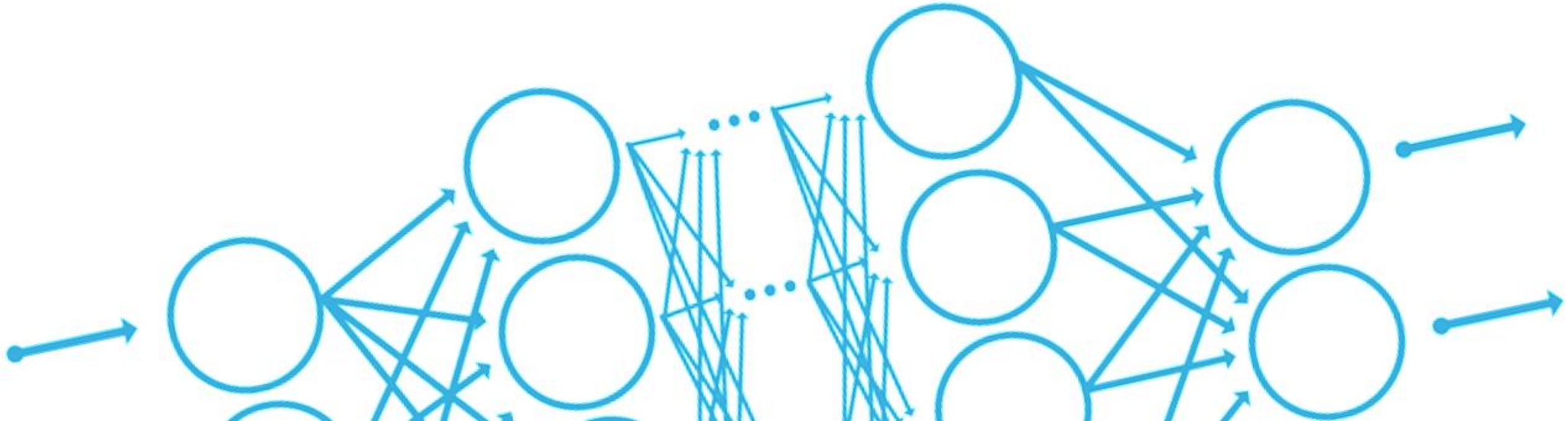
Train a reinforcement learning agent for a lane keeping assist application.



Train DDPG Agent for Path Following Control

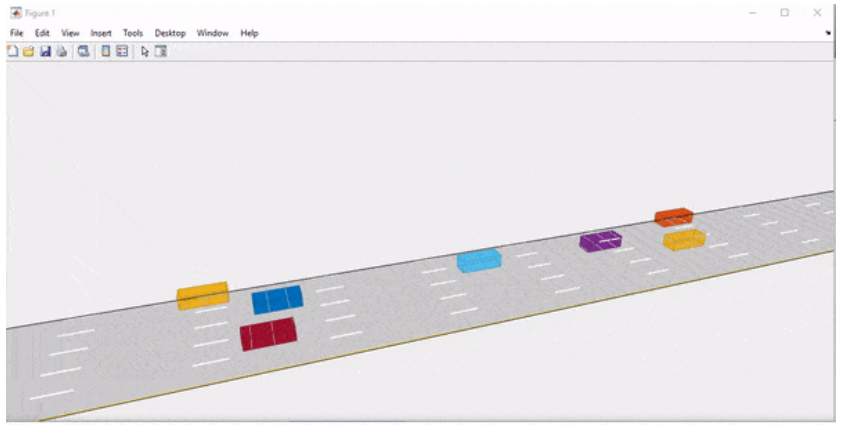
Train a reinforcement learning agent for a lane following application.

Takeaways

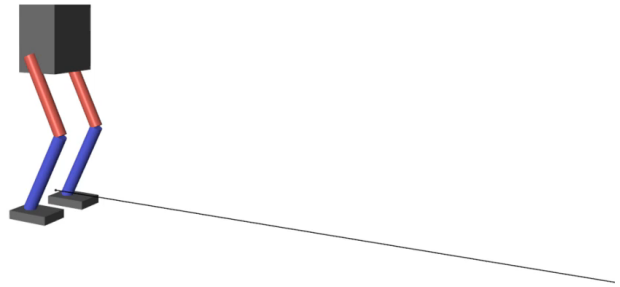


Simulation and virtual models are a key aspect of reinforcement learning

- Reinforcement learning needs **a lot** of data (*sample inefficient*)
  - Training on hardware can be prohibitively expensive and dangerous



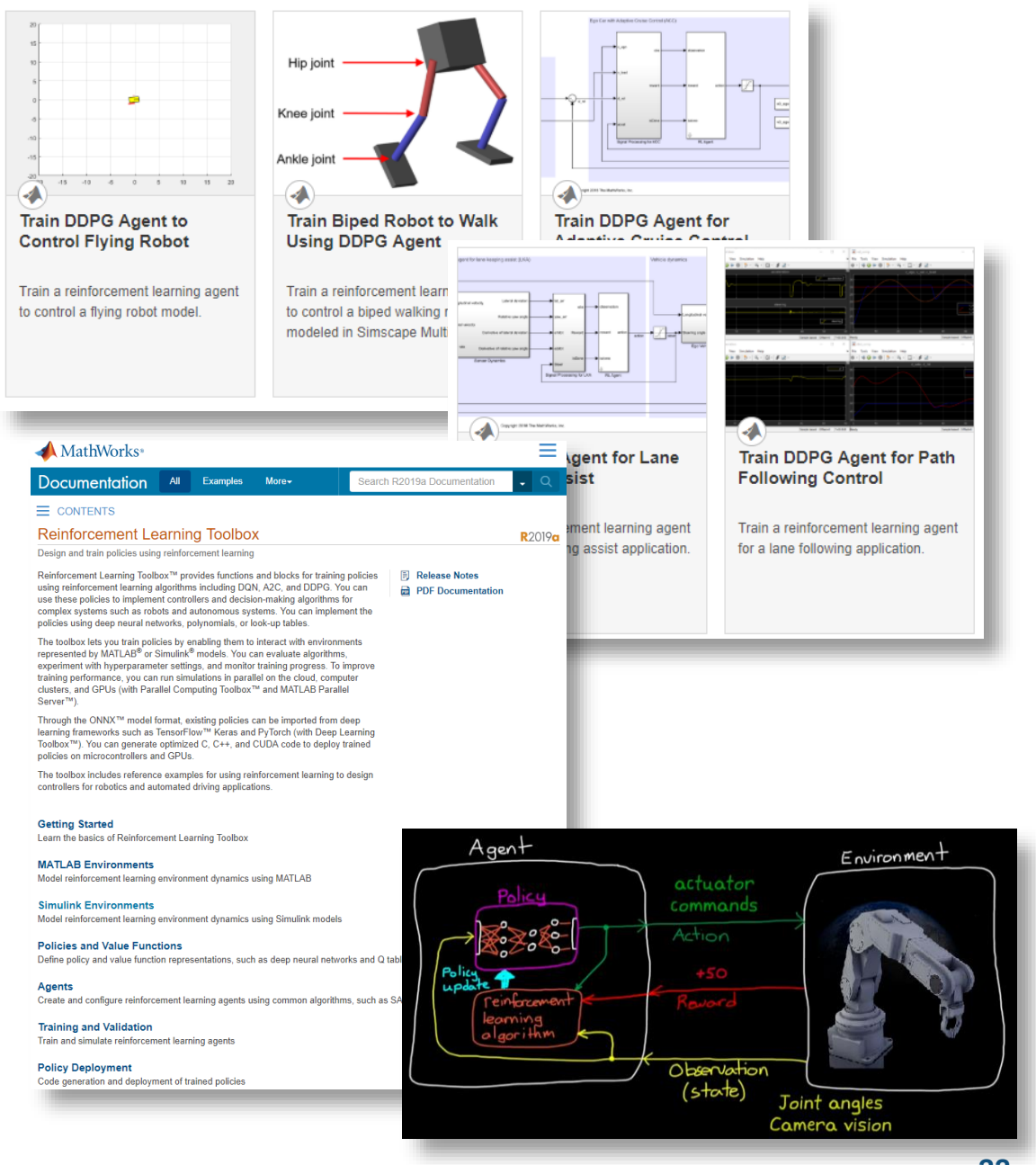
- Virtual models allow you to simulate conditions hard to emulate in the real world
  - This can help develop a more robust solution



- Many of you have already developed MATLAB and Simulink models that can be reused

Resources

- Examples for automotive and autonomous system applications
- Documentation written for engineers and domain experts
- Tech Talk video series on reinforcement learning concepts for engineers



Thank You!