Deep Reinforcement Learning for Robot Control

Advanced topic II

Simon Bøgh

Associate Professor

Faculty of Engineering, Department of Materials and Production

In collaboration with Prof. Nestor Arana Arexolaleiba, Mondragon University, Spain

Spring 2020

AALBORG UNIVERSITY
DENMARK

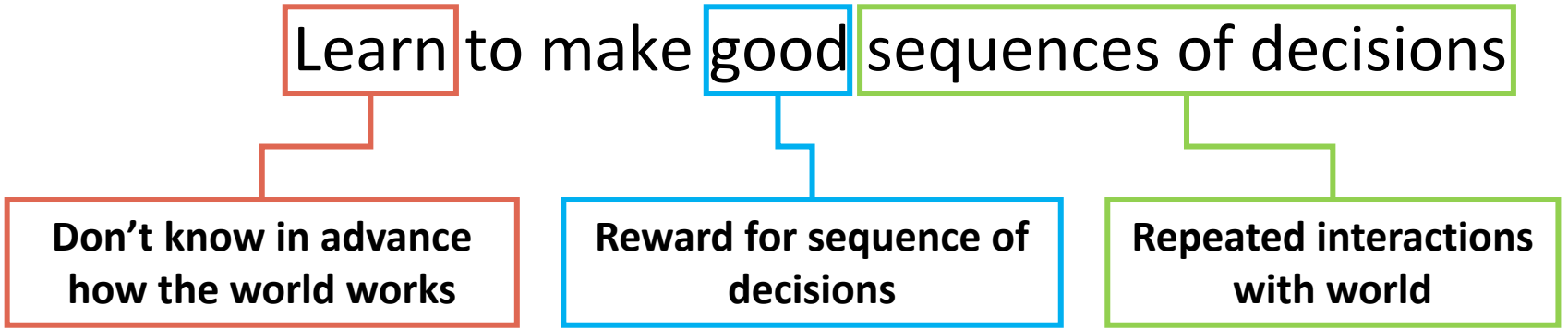
Outline

- Introduction to Reinforcement Learning
- Deep Reinforcement Learning practical applications in Robotics
- Important RL/DRL literature
- OpenAI Gym and Stable-Baselines
  - Deep Reinforcement Learning implementations
- Assignment
  - CartPole environment: installation and training
  - OMTP Factory – example on training the robot to reach an object

Deep Reinforcement Learning

Building intelligent robot systems

What is reinforcement learning

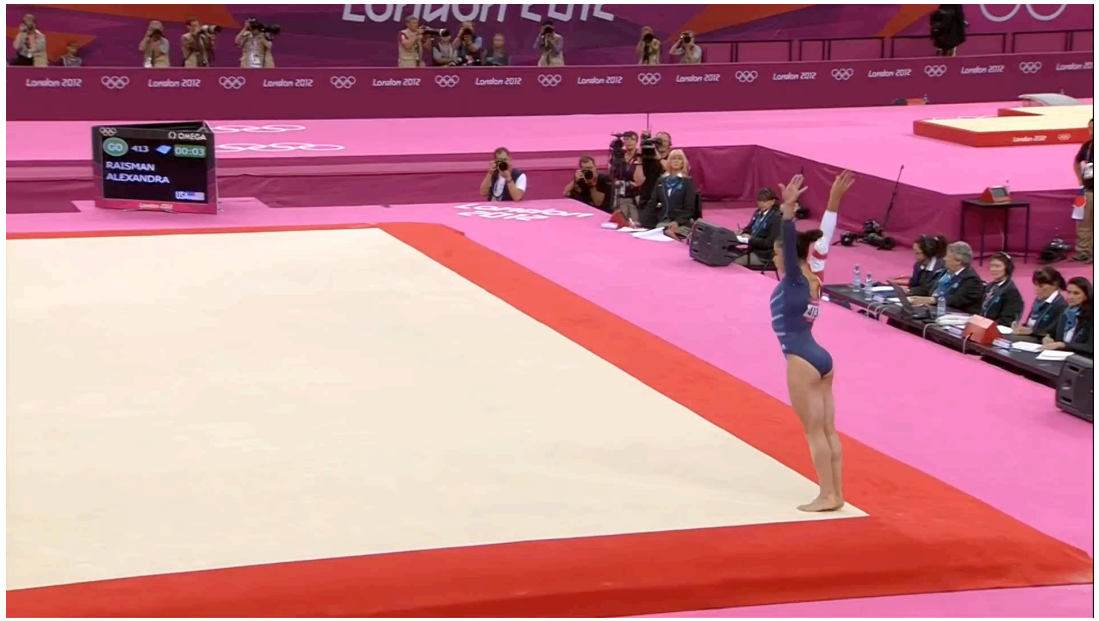


Reinforcement Learning

- Agent receiving reward based on actions performed in an environment
- Adaptability to changes in environment
- Inspired by human learning ability

Reinforcement Learning

- Agent receiving reward based on actions performed in an environment
- Adaptability to changes in environment
- Inspired by human learning ability



Types of Machine Learning

SUPERVISED LEARNING

- Machine learns explicitly
- Data with clearly defined output is given (A to B mapping)
- Predicts outcome / future
- Classification and regression problems

Example #	X	Y
0	X0	Y0
1	X1	Y1
2	X2	Y2
3	X3	Y3
4	X4	Y4

UNSUPERVISED LEARNING

- Machine identifies patterns / structures
- Segmentation, or outlier detection
- Does not predict / find anything specific
- Clustering, Anomaly Detection

REINFORCEMENT LEARNING

- Machine (agent) learns how to act in a certain environment
- Machine learns by trial and error
- Reward based learning
- Think how you train your dog; “that’s a good dog” vs. “bad dog”

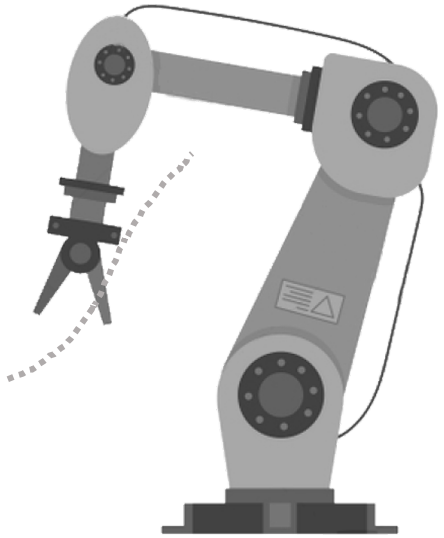
Example #	State	Action	Reward
0	S0	A0	-1
1	S1	A1	0
2	S2	A2	0
3	S3	A3	0
4	S4	A4	1

The idea

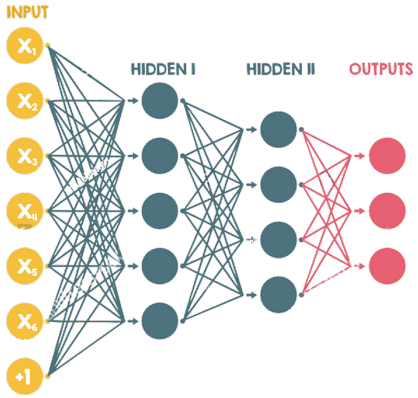
Define an environment for the agent to act in

Learn policy from training through trial and error interaction in the environment

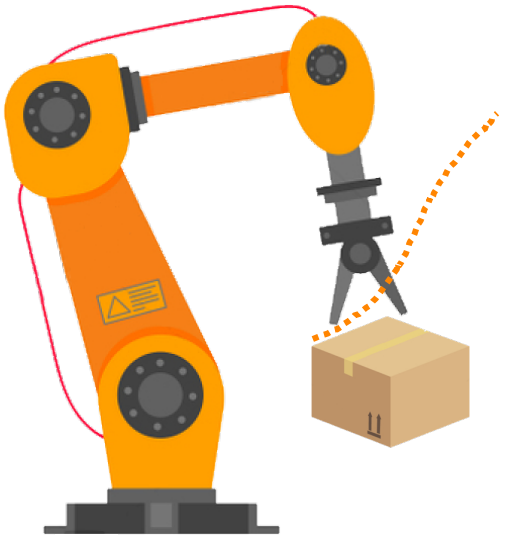
Execute learned optimal Policy



sensors

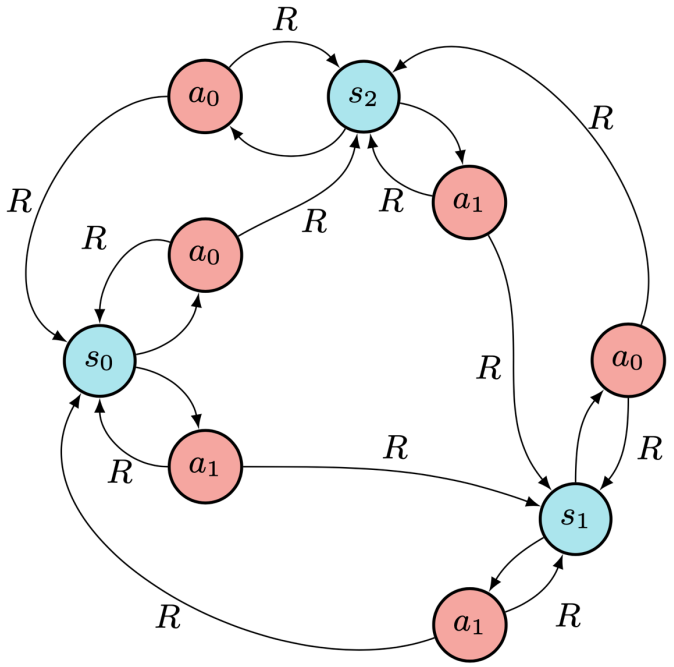


actions

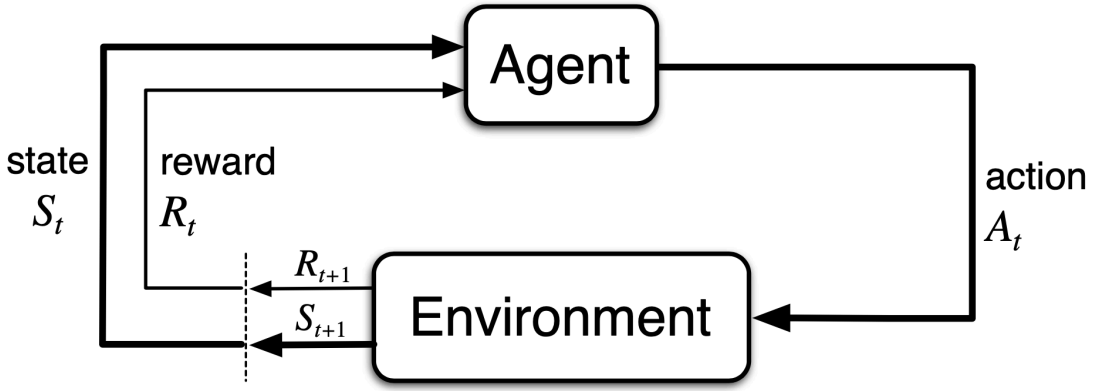


Markov Decision Process

- MDPs are a classical formalization of **sequential decision making**
- **Actions influence not just immediate rewards**, but also subsequent situations, or states, and through those future rewards
- Thus MDPs involve delayed reward and the need to **tradeoff immediate and delayed reward**



The agent-environment interaction in a Markov Decision Process

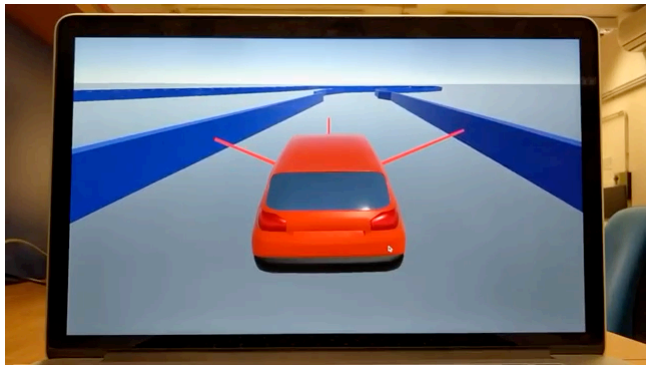


- The ***agent*** interacts with the ***environment***
- The agent selects ***actions*** in the environment and ends om in new ***states***
- The environment gives rise to ***rewards*** (numerical values)
- The agent seeks to ***maximize reward*** over time

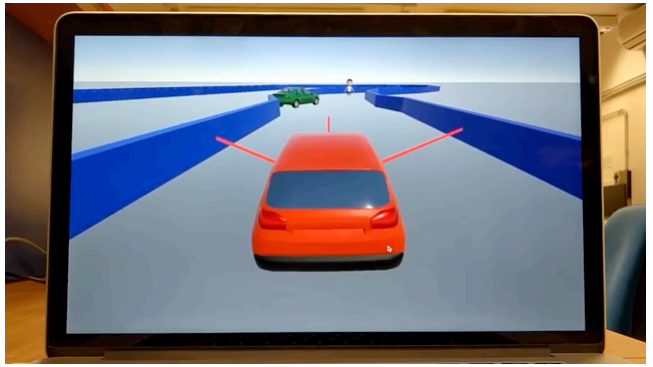
Exploration/exploitation dilemma

- **Exploitation** is about using what you know, whereas **exploration** is about gathering more data/information so that you can learn
- Restaurant Selection
  - **Exploitation**: Go to your favorite restaurant (you know the reward)
  - **Exploration**: Try a new restaurant (the reward may be high or low)

Exploration

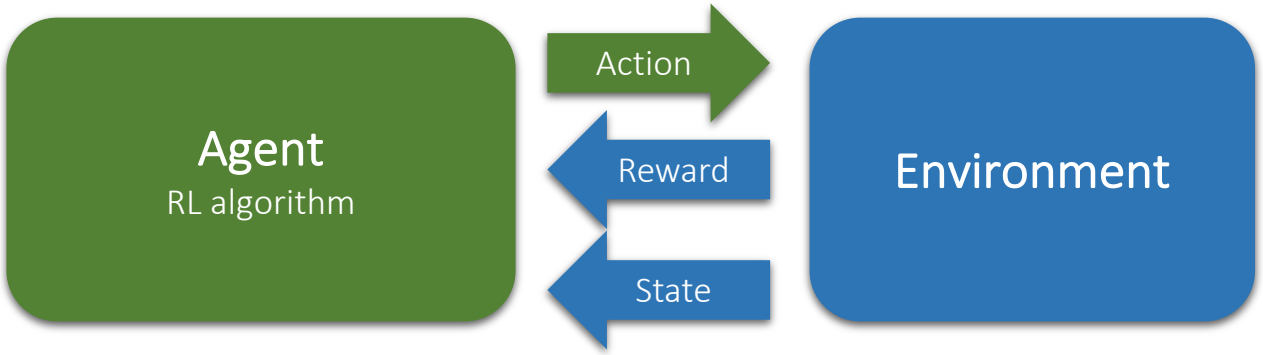


Exploitation



http://www.primarydigit.com/blog/multi-arm-bandits-

Elements of Reinforcement Learning

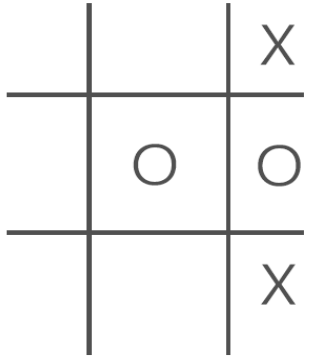


- **Agent**
  - Learns to achieve goals by interacting with environment
  - Basic loop:
    - Senses: State, Reward
    - Effect: selects an Action
- **Environment**
  - Defines the world that the agent interacts with. Basic loop:
    - Produces: States, Rewards for the agent to sense and process
    - Accepts: Actions from the agent

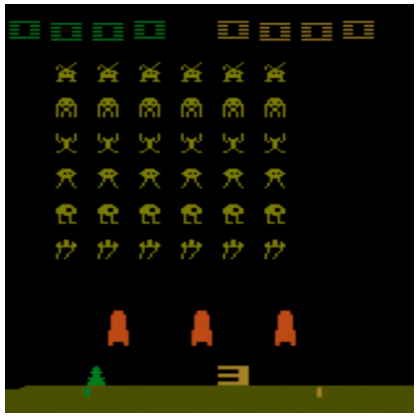
State

- Notation s

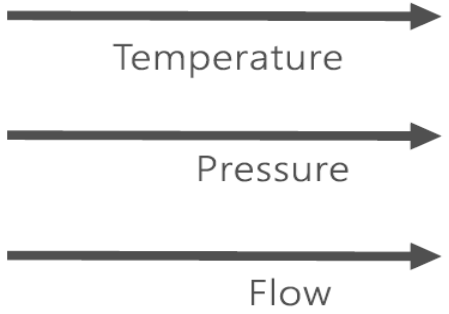
Low dimensional discrete state



High-dimensional discrete state



Continuous state



t	s	Action	Reward
0	S0	A0	-1
1	S1	A1	0
2	S2	A2	0
3	S3	A3	0
4	S4	A4	0
5	S5	A5	0
6	S6	A6	1

Reward

- The scalar value (floating point number) returned by the environment when the agent selects an action
  - Represents the goal, or goals
  - Determined by the RL problem designer
  - Usual r_t for reward at time t

t	s	Action	r_t
0	S0	A0	-1
1	S1	A1	0
2	S2	A2	0
3	S3	A3	0
4	S4	A4	1

Action

- Notation action a
- **Discrete** (1 of N actions)
  - Robot
    - Tic-Tac-Toe
    - Go up, down, right, left, forward, backward 5 mm
- **Continuous** (action as scalar/vector of real values)
  - Car
    - Steering angle or gas pedal

t	s	Action	r_t
0	S0	A0	-1
1	S1	A1	0
2	S2	A2	0
3	S3	A3	0
4	S4	A4	1

Policy

- Notation: π
- A mapping from states to actions
  - Deterministic
  - Stochastic
    - 60% you take action 1
    - 40% you take action 2
- Can be represented by a table, function, or neural network

Value function

- **Value Function**
  - Expected long-term accumulation of reward starting from s, following policy π
  - Human analogy:
    - Reward = immediate pleasure/pain
    - Value function = farsighted judgement for the state
- **Two flavours of the value function**
  - **Value Function** $V^{\pi}(s)$ – how good is the current state given the policy
  - **Value-State Function** $Q^{\pi}(s, a)$ – a mapping from each state/action pair to the expected long-term accumulation of reward starting from state s and taking action a, and thereafter following policy π

Applications of Reinforcement Learning

Chess and Go, onwards to Atari

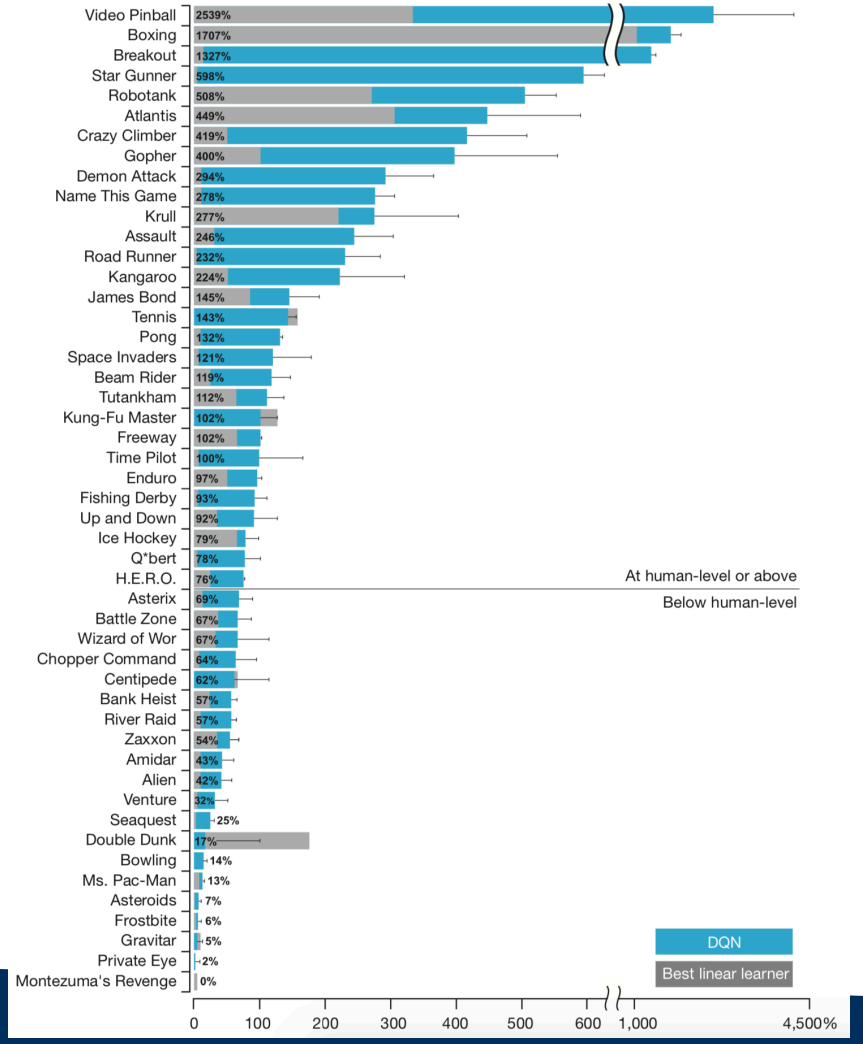
- AlphaGo is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion, and is arguably the strongest Go player in history (2015)
- Reinforcement Learning and tree-search
- Played against itself from scratch

https://youtu.be/WXuK6gekU1Y

ATARI - Google DeepMind at it again

Breakout

Google DeepMind: Mnih, V. et al, 2015. Human-level control through deep reinforcement learning. *Nature*



Playing Atari with Deep Reinforcement Learning (Mnih, 2013)
- Paper: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
- Video: https://www.youtube.com/watch?v=TmPfTpjtdgg

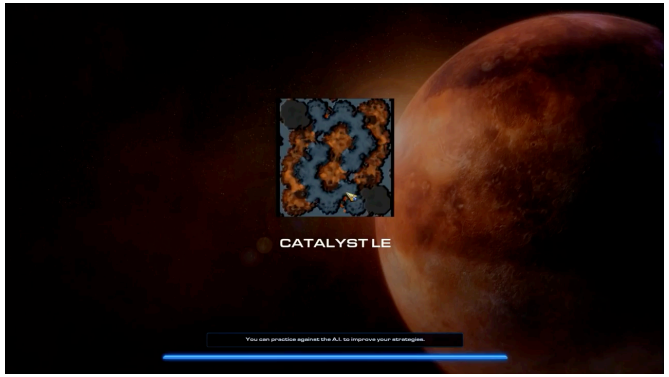
Beating Human World Champions in Computer Games

- Agents trained entirely from self-play
  - **DOTA by OpenAI:** https://youtu.be/yBEidvm_tZQ
  - **Starcraft II by DeepMind:** https://youtu.be/cUTMhmVh1qs

  - More than 20.000 actions
  - High-dimensional state space

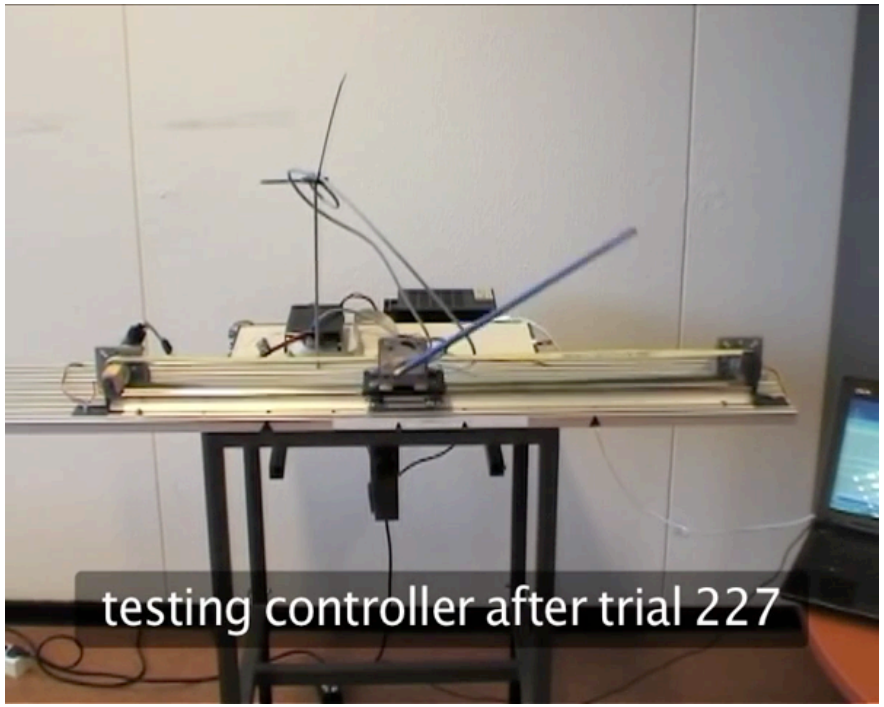
DOTA 5v5

StarCraft II



Automatic Control

Inverted pendulum

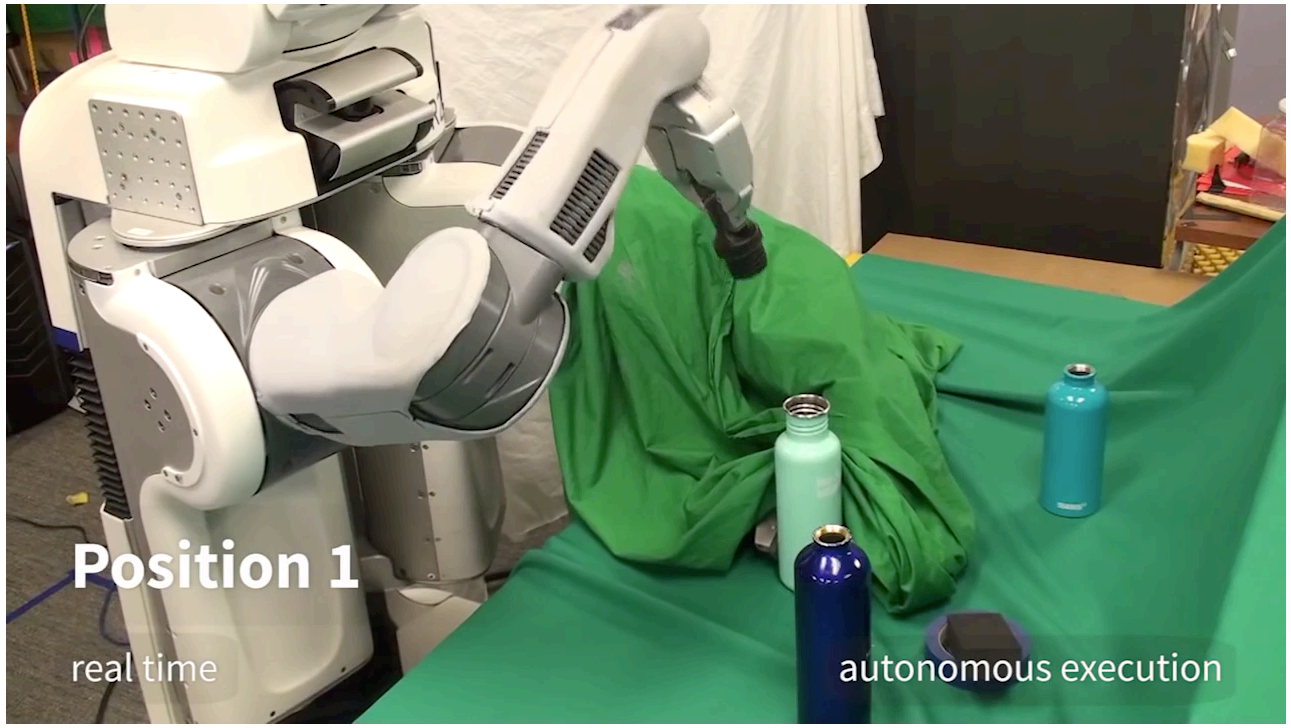


https://youtu.be/Lt-KLtkDlh8

Goalkeeper

https://youtu.be/CIF2SBVY-J0

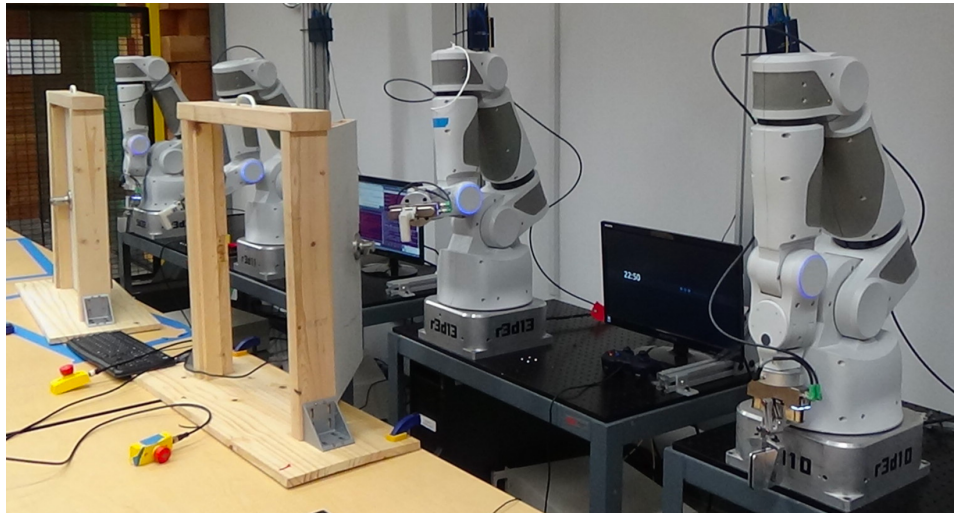
Robotics

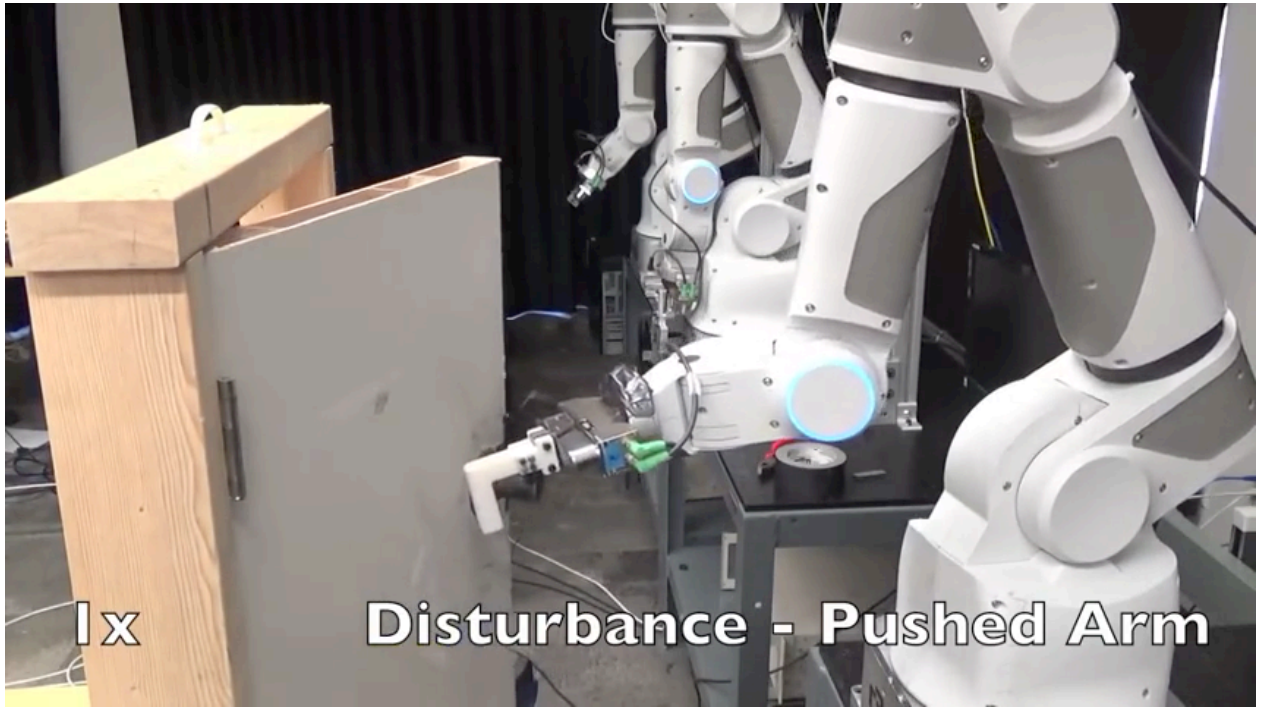


Berkeley: Levine, S., Finn, C., Darrell, T. and Abbeel, P., 2016. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research

Robotic Manipulation

Multiple robots simultaneously pooling their experiences



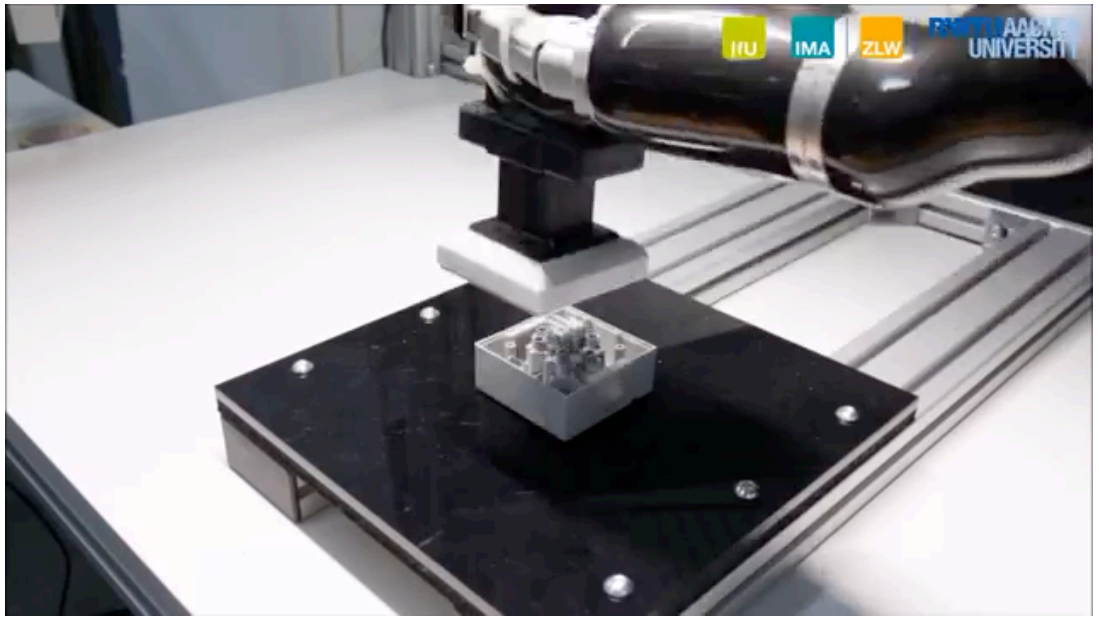


https://youtu.be/ZhsEKTo7V04

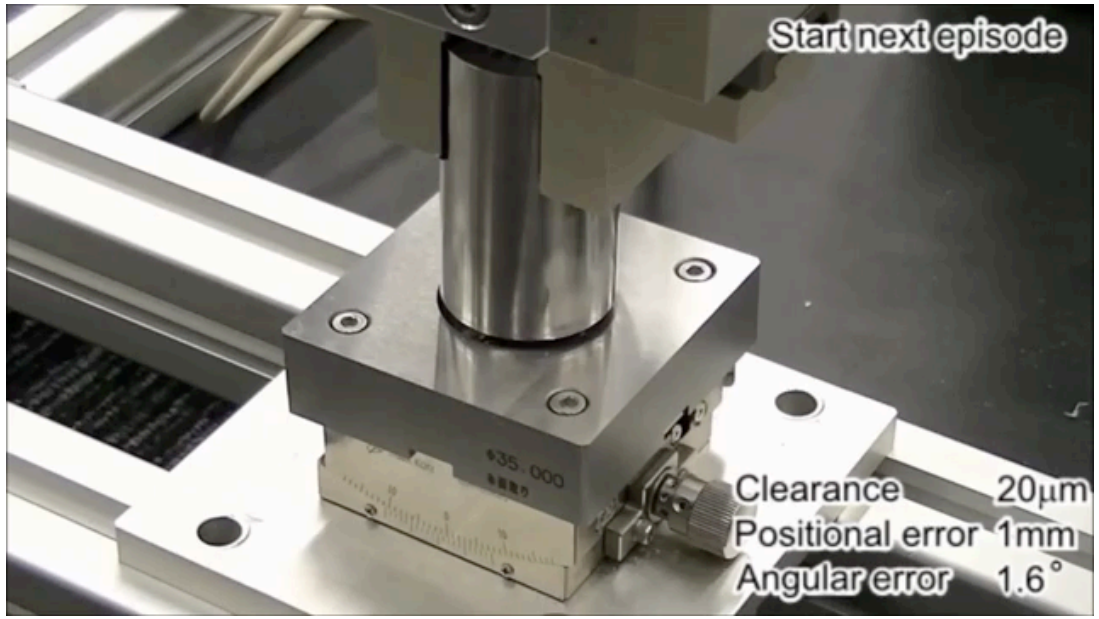
Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017, May). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 3389-3396). https://arxiv.org/pdf/1610.00633.pdf

Robotics – Picking, shared learning

Robotics - Assembly



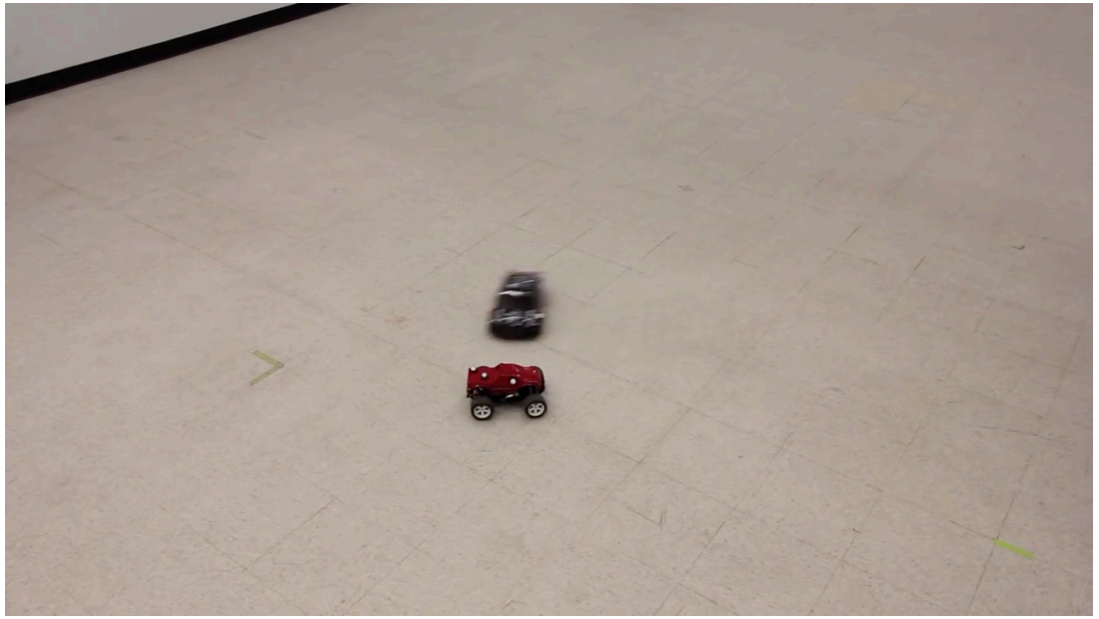
https://youtu.be/Nh08oBb30-c



https://youtu.be/b2pC78rBGH4

Autonomous Vehicles

RC car drifting

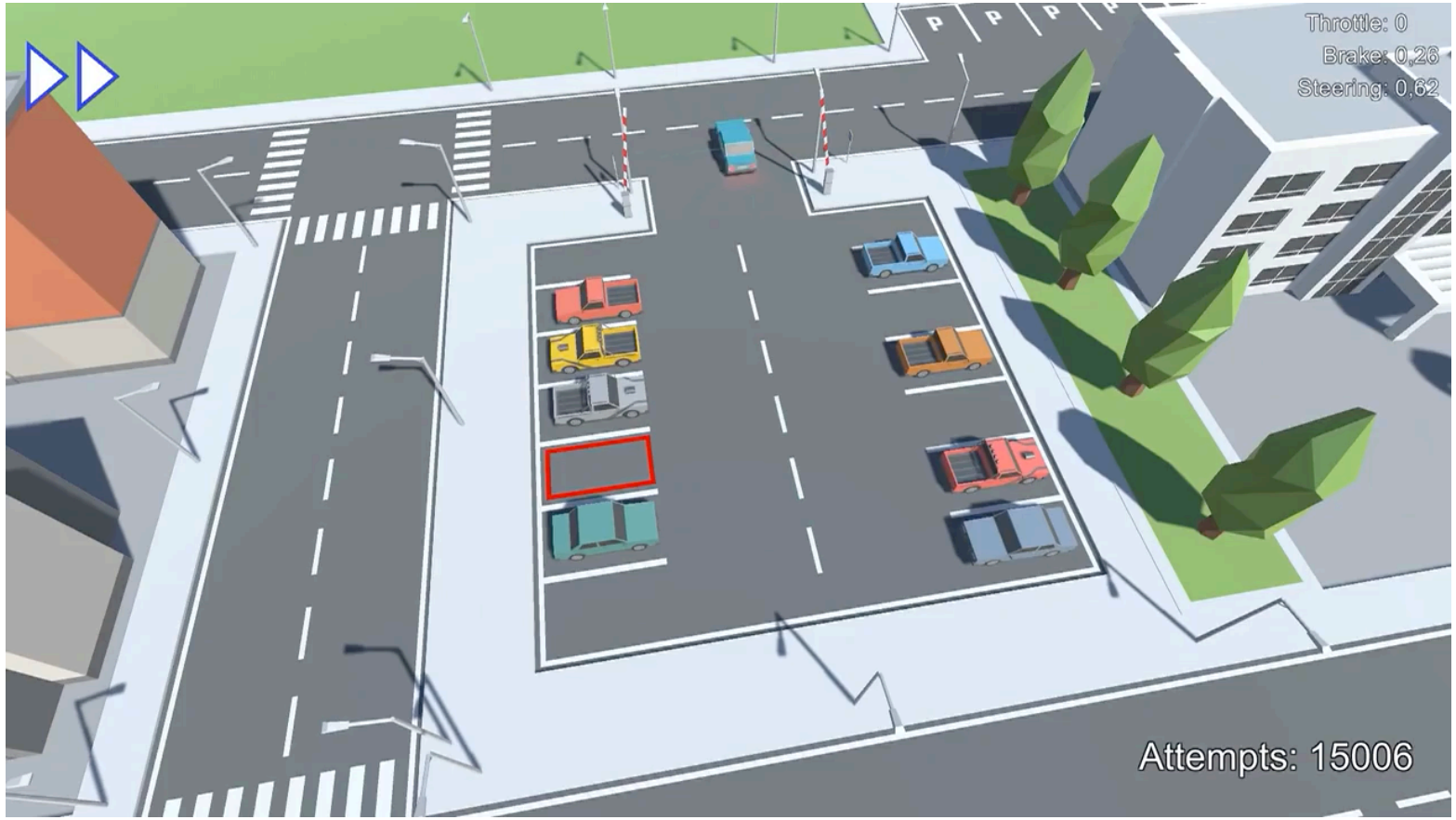


https://youtu.be/opsmd5yuBF0

Acrobatic helicopter flight

https://youtu.be/VCdxqn0fcnE

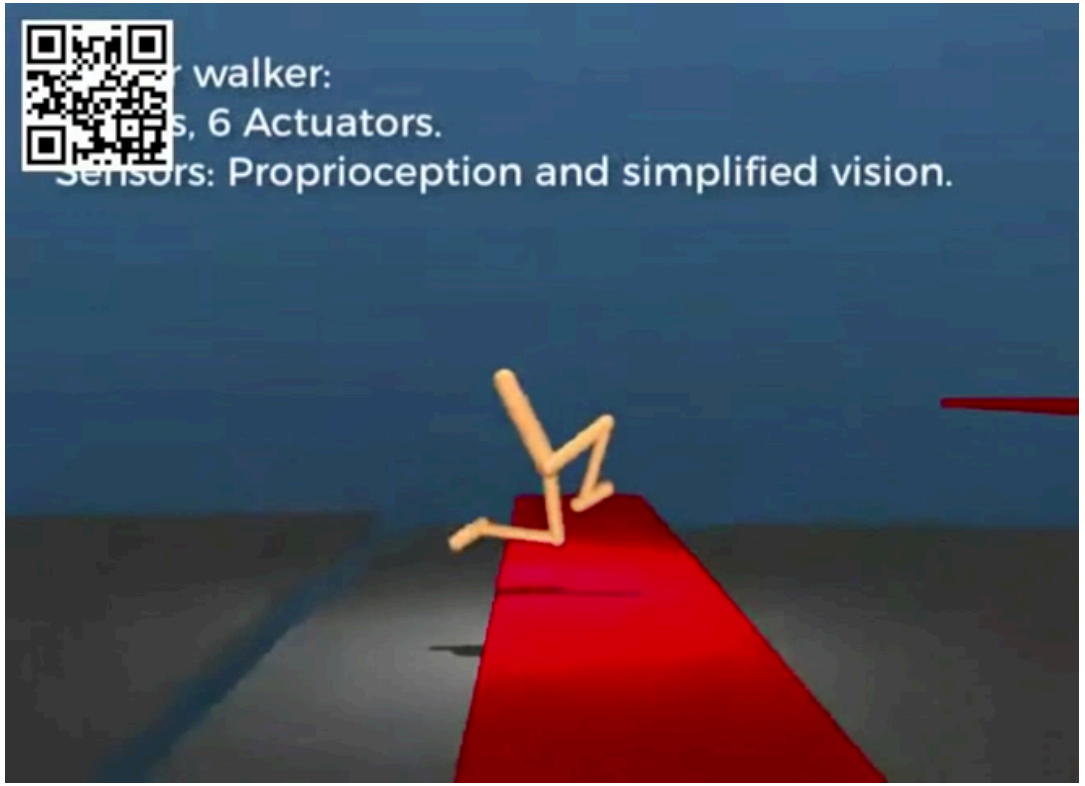
Self-Parking Car



https://youtu.be/VMp6pq6_QjI

Training a Local Planner for Mobile Robots

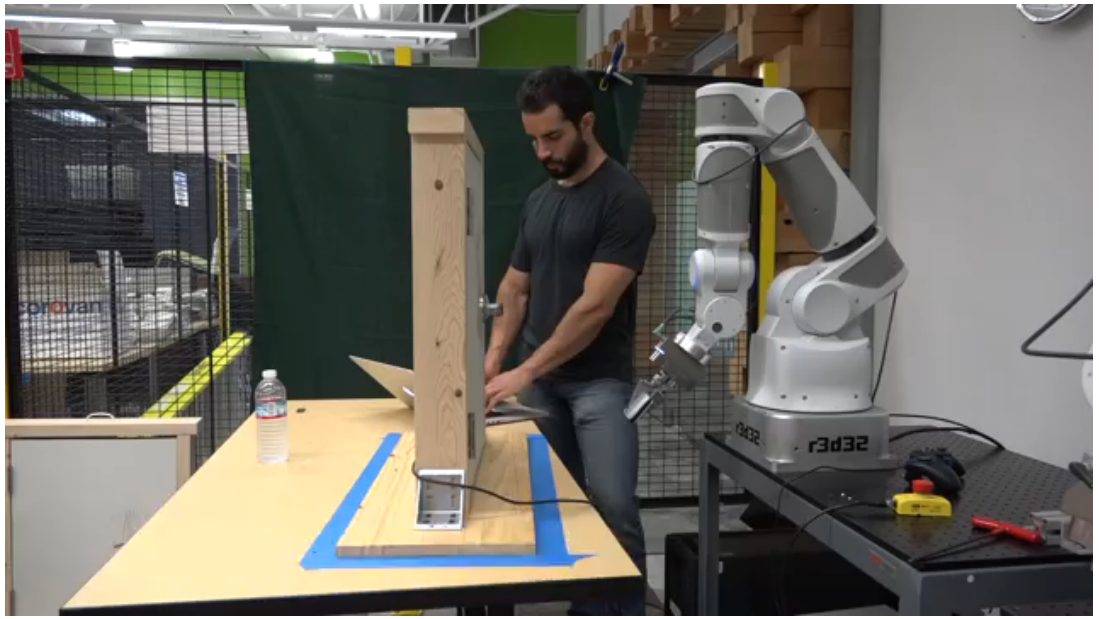
Robot learning to walk, run, and jump



https://youtu.be/hx_bgoTF7bs

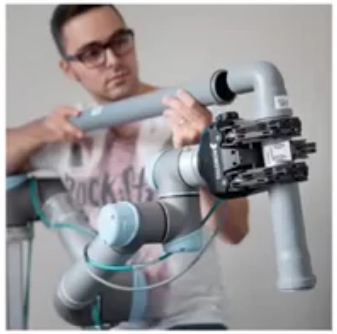
Emergence of Locomotion Behaviors in Rich Environments (Heess, 2017)
Paper: https://arxiv.org/pdf/1707.02286.pdf

Learning From Humans with Imitation Learning



TECHNISCHE UNIVERSITÄT BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

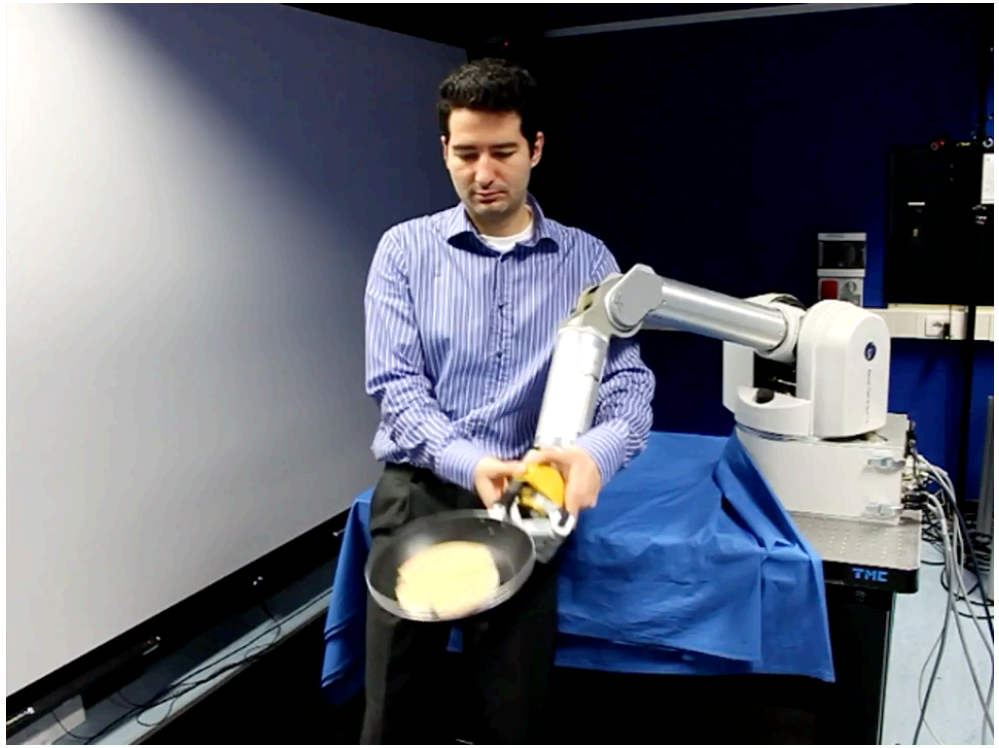
IRA A. FULTON SCHOOLS OF engineering
ARIZONA STATE UNIVERSITY



Learning Continuous Human-Robot Interactions from Human-Human Demonstrations

David Vogt, Simon Stepputtis, Richard Weinhold, Bernhard Jung, Heni Ben Amor

Flipping pancakes



https://youtu.be/bxtPyJqVrmk

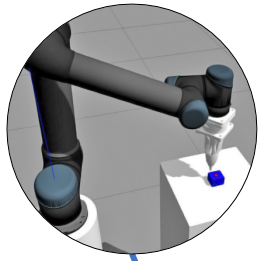
Playing ping pong

https://youtu.be/SH3bADiB7uQ

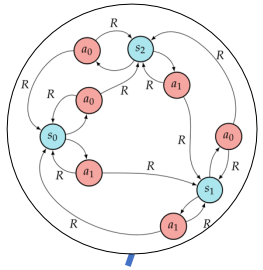
Reinforcement Learning at AAU

Brine Injection Optimization with DDPG

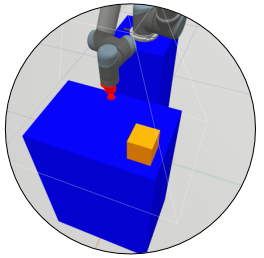
RL-Unscrew RL Simulation Framework



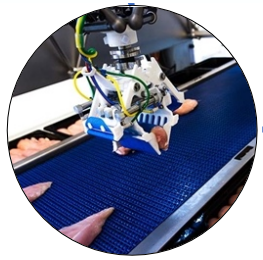
Learning Human Behavior to Teach Robots with RL



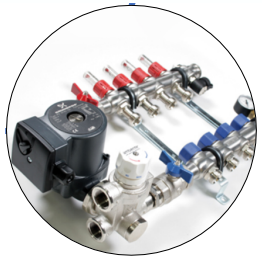
Learning to pick up and throw objects



RoboBatching for Multi-Robot Systems with RL

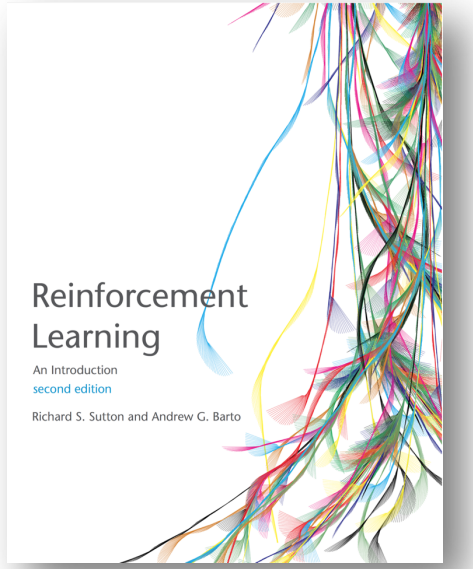


RL-based Control of Underfloor Heating Systems



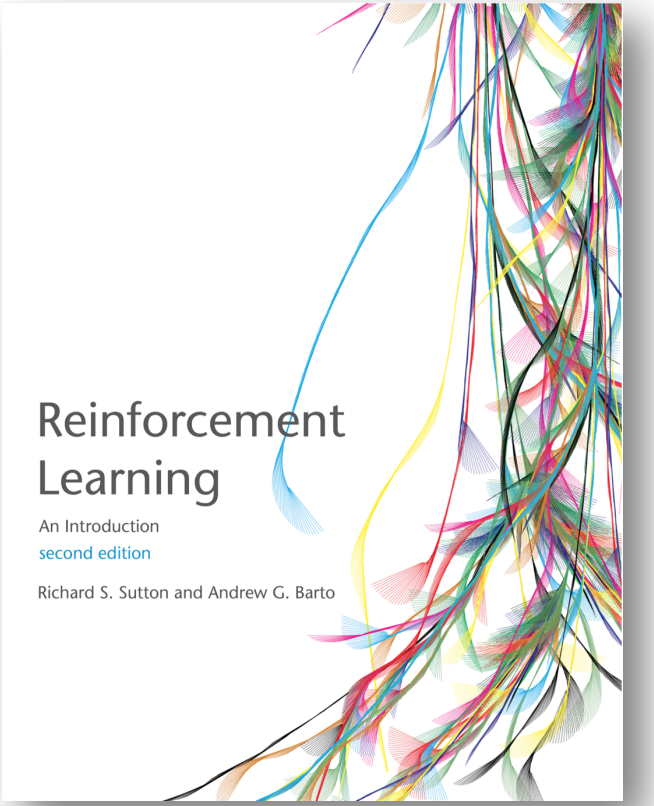
Important literature

- Book:
  - ***Reinforcement Learning – An Introduction 2nd edition***, 2018, Sutton & Barto
  - http://incompleteideas.net/book/the-book-2nd.html
- Research papers:
  - DQN – Deep Q-Networks
  - PPO – Proximal Policy Optimization
  - DDPG – Deep Deterministic Policy Gradient



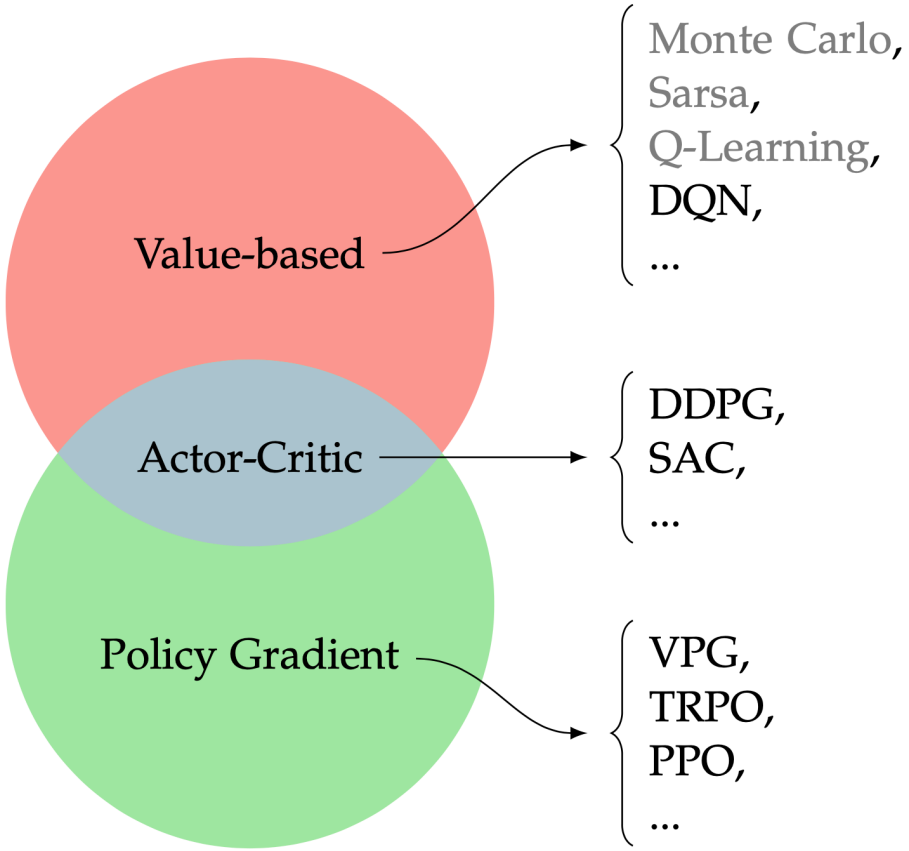
Reinforcement Learning – An Introduction *2nd edition*, 2018, Sutton & Barto

- Cited 35.000+ times
- Content:
  - Introduction to classic RL (not DL) and framing of the problem
  - Markov Decision Processes
  - Q-learning / TD-learning (tabular approach)
  - … and much more
- Download for free!
  - http://incompleteideas.net/book/the-book-2nd.html



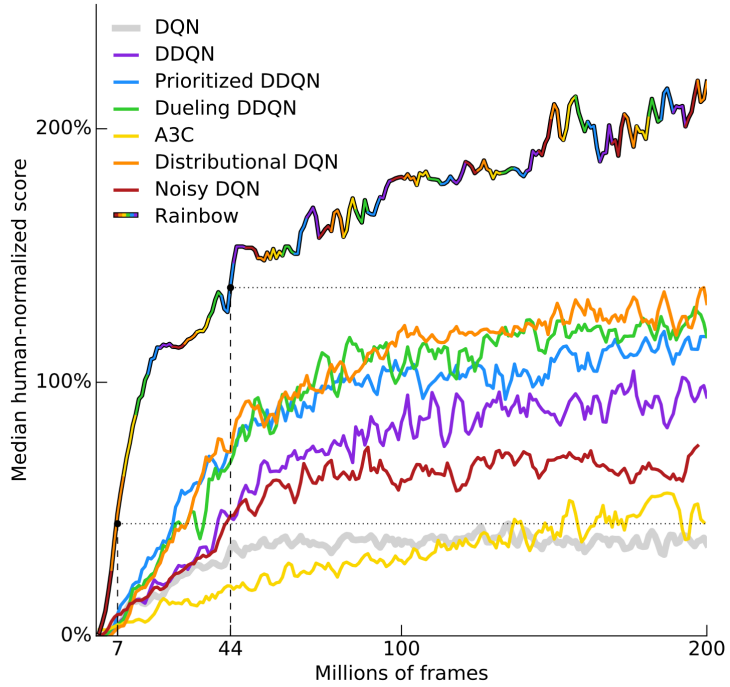
Three main papers

- DQN (value-based)
- PPO (policy-based)
- DDPG (actor-critic)



DQN
– Deep Q-Network

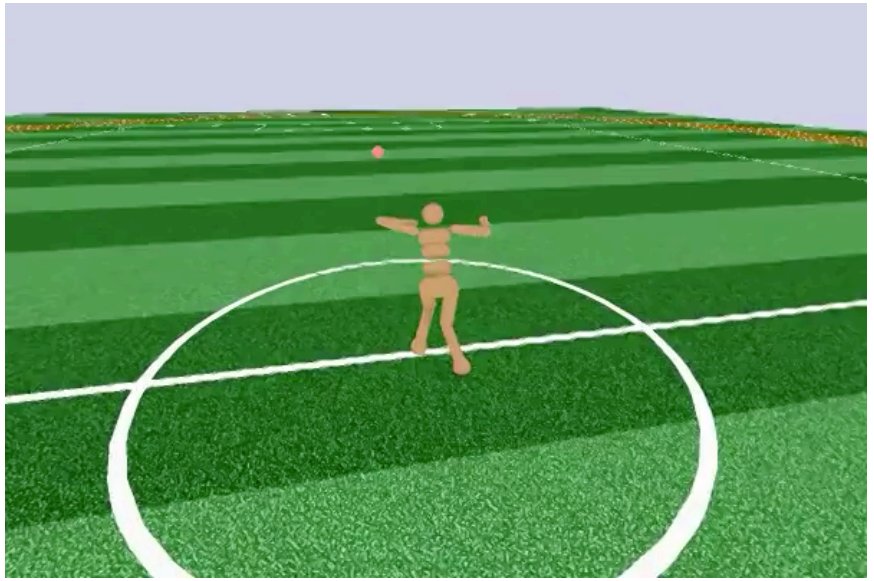
- Where it all started (in recent years)
- Deep Q-Network (2015)
  - Going **from tabular** approach (Q-learning) **to a function approximation** approach with **deep neural network**
- Main challenge before
  - Training could get stuck in one part of the NN
  - Instability when using the same policy we are updating at the same time
- Two main novelties
  - Experience replay memory
  - Target network updates
- More novelties followed:
  - Double-DQN, Dueling DQN, Prioritized experience replay, Noisy DQN → **Rainbow** algorithm



Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529-533.

PPO
– Proximal Policy Optimization

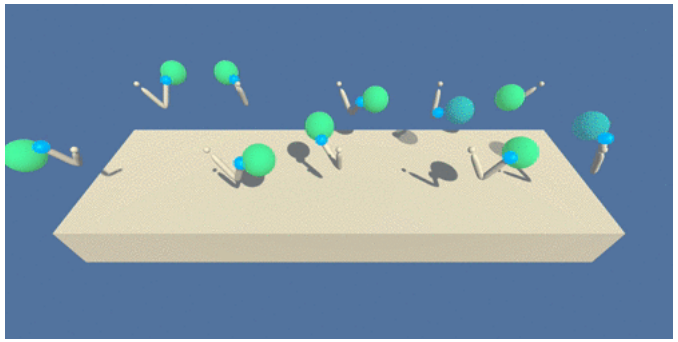
- What if we could estimate the policy (or actions) directly instead of q-values?
- Policy-based methods **predicts the action** to take
- Means we can output **continuous actions** instead of discrete actions
- Simple to implement, general, and have good sample complexity

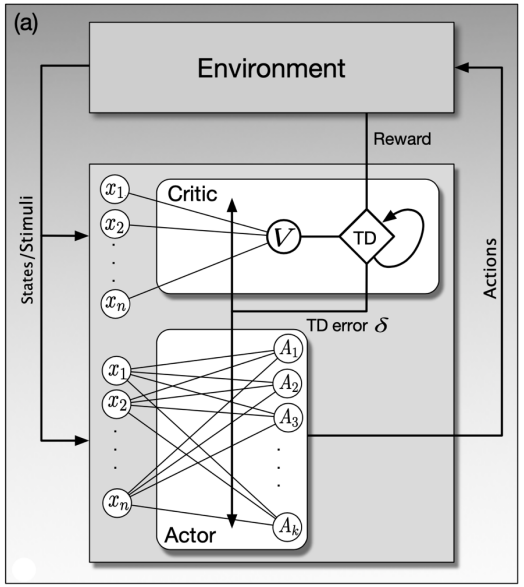


Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

DDPG
– Deep Deterministic Policy Gradient

- Transfer the success from deep Q-learning achieved in **discrete action domain to a continuous action domain**
- Actor–critic algorithms learn both policies and value functions
- Consists of two NN: **actor and critic**
  - **Actor**
    - Learns the policy to follow
    - Adjusts a policy based on the TD error
    - Actor does not have direct access to the reward signal
  - **Critic**
    - Criticizes the actor's action choices
    - Adjusts state-value parameters using the same TD error
    - Produces a TD error from the reward signal, R





Sutton & Barto, 2018

Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971(2015).

Deep Reinforcement Learning

Frameworks,
Environments and Algorithms

Backends

TensorFlow

Keras

PyTorch

theano

Frontends

OpenAI Gym

Toolkit for developing and comparing reinforcement learning algorithms
https://gym.openai.com

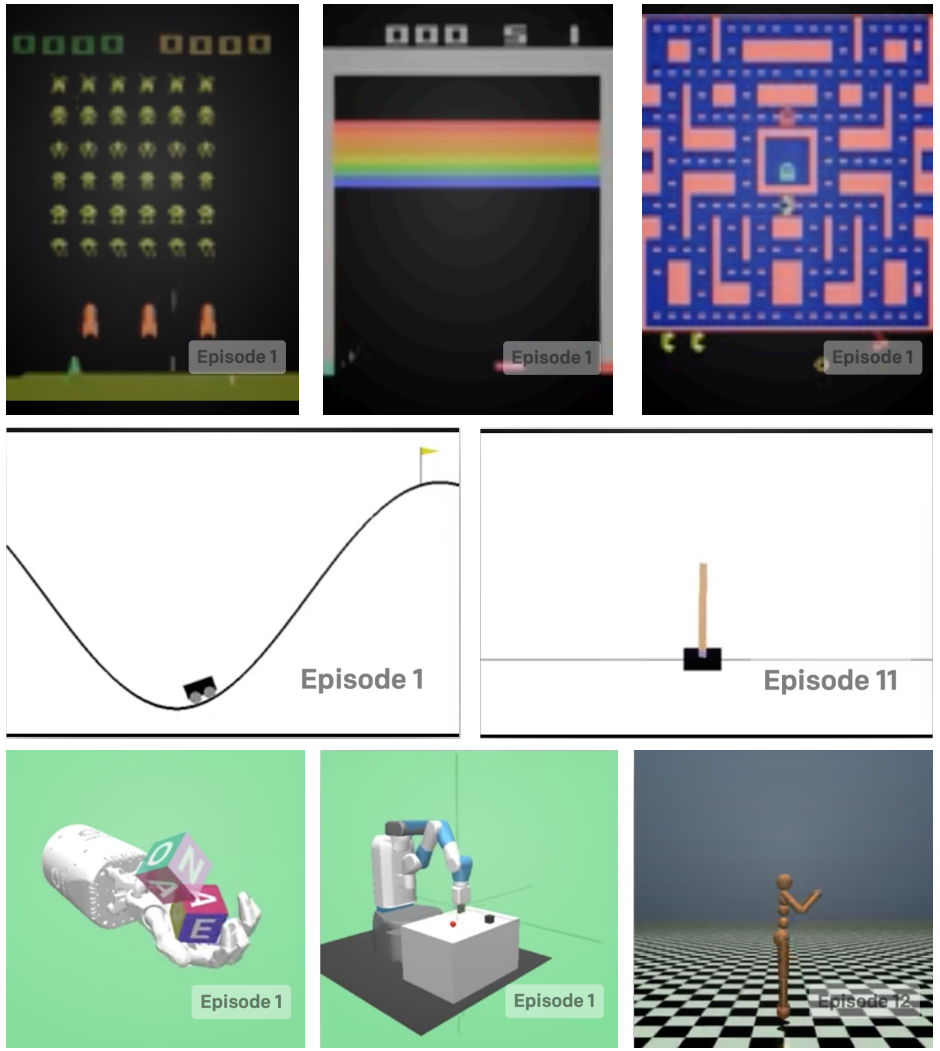
Stable Baselines

A set of improved implementations of reinforcement learning algorithms based on OpenAI Baselines
https://github.com/hill-a/stable-baselines

OpenAI Gym

- Provides environments
- You can test your own algorithm
- Gym interface
  - obs, reward, done, info = step(action)
  - observation = reset()
  - render()

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000): env.render()
        action = env.action_space.sample() # your agent here (this takes random actions)
        observation, reward, done, info = env.step(action)

        if done:
                observation = env.reset()
env.close()
```

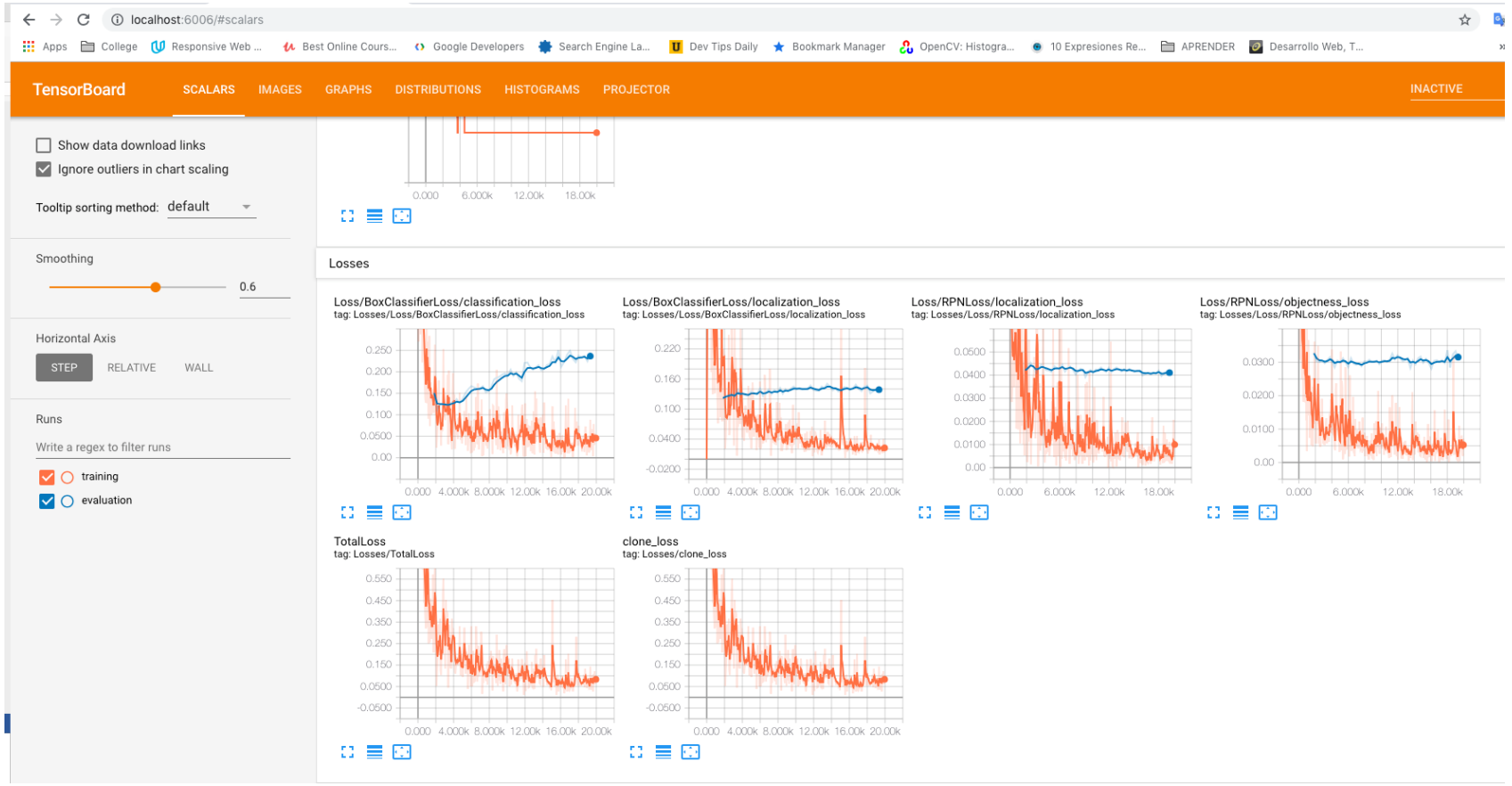


Stable-Baselines

- A set of improved implementations of Reinforcement Learning (RL) algorithms
- https://stable-baselines.readthedocs.io

Name	Refactored [1]	Recurrent	Box	Discrete	Multi Processing
A2C	✓	✓	✓	✓	✓
ACER	✓	✓	✗ [4]	✓	✓
ACKTR	✓	✓	✓	✓	✓
DDPG	✓	✗	✓	✗	✓ [3]
DQN	✓	✗	✗	✓	✗
HER	✓	✗	✓	✓	✗
GAIL [2]	✓	✓	✓	✓	✓ [3]
PPO1	✓	✗	✓	✓	✓ [3]
PPO2	✓	✓	✓	✓	✓
SAC	✓	✗	✓	✗	✗
TD3	✓	✗	✓	✗	✗
TRPO	✓	✗	✓	✔	✓ [3]

Tensorboard (comes with tensorflow)



Where is the Technology in 10 years?

- Reinforcement Learning has existed for more than 30 years
- It was not until 2015 we saw a lot of progress because of successes within Deep Neural Networks
- Currently, most research on RL algorithms is applied on computer games (95%) because of good simulation environments
- Maybe only 5% is done on real-world applications e.g. robots and process control
- Our work focus on the application of Deep Reinforcement Learning in robotics, industrial manufacturing and process optimzation, and getting it to work
- Doing DRL in the real world is much more difficult than simulation
- Training experience is expensive to acquire e.g. crashing your robot

Useful literature

- ***Reinforcement Learning – An Introduction, 2nd edition***, 2018, Sutton & Barto
- **Value-based: DQN**
  - Mnih, Volodymyr, et al. "**Human-level control through deep reinforcement learning.**" *Nature* 518.7540 (2015): 529-533.
  - Van Hasselt, Hado, Arthur Guez, and David Silver. "**Deep reinforcement learning with double q-learning.**" *Thirtieth AAAI conference on artificial intelligence*. 2016.
  - Wang, Ziyu, et al. "**Dueling network architectures for deep reinforcement learning.**" arXiv preprint arXiv:1511.06581 (2015).
  - Hessel, Matteo, et al. "**Rainbow: Combining improvements in deep reinforcement learning.**" *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- **Policy-based:**
  - PPO: Schulman, John, et al. "**Proximal policy optimization algorithms.**" *arXiv preprint arXiv:1707.06347* (2017).
- **Actor-Critic**
  - DDPG: Lillicrap, Timothy P., et al. "**Continuous control with deep reinforcement learning.**" *arXiv preprint arXiv:1509.02971*(2015).
  - SAC: Haarnoja, Tuomas, et al. "**Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.**" *arXiv preprint arXiv:1801.01290* (2018).

The Complete Reinforcement Learning Dictionary

- The Reinforcement Learning Terminology, A to Z
  - https://towardsdatascience.com/the-complete-reinforcement-learning-dictionary-e16230b7d24e

Online courses

- Udacity.com
  - Reinforcement Learning by Georgia Tech (course, free): https://www.udacity.com/course/reinforcement-learning--ud600
  - Deep Reinforcement Learning Nanodegree (nanodegree, not free) https://www.udacity.com/course/deep-reinforcement-learning-nanodegree--nd893
- Coursera.org, EDX.org

Recommendations to get started

- **Take a course** and/or **read a book/paper**
- Then **stop reading** and **start implementing – it's the best way to learn RL**

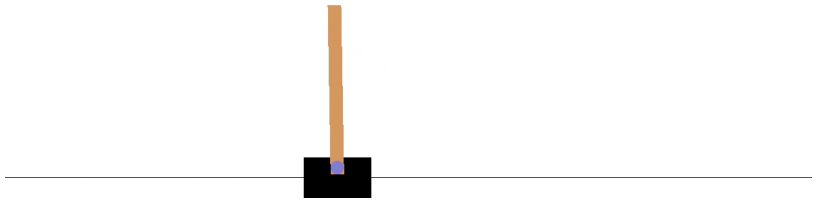
- Follow experts in the field
  - Pieter Abbeel, Berkeley
  - Sergey Levine, Berkeley
  - Chelsey Finn, Berkeley
  - David Silver, Google DeepMind
  - Jan Peters, TU Darmstadt

Let's train a DQN agent

Assignment

Train a Deep Q-Network agent

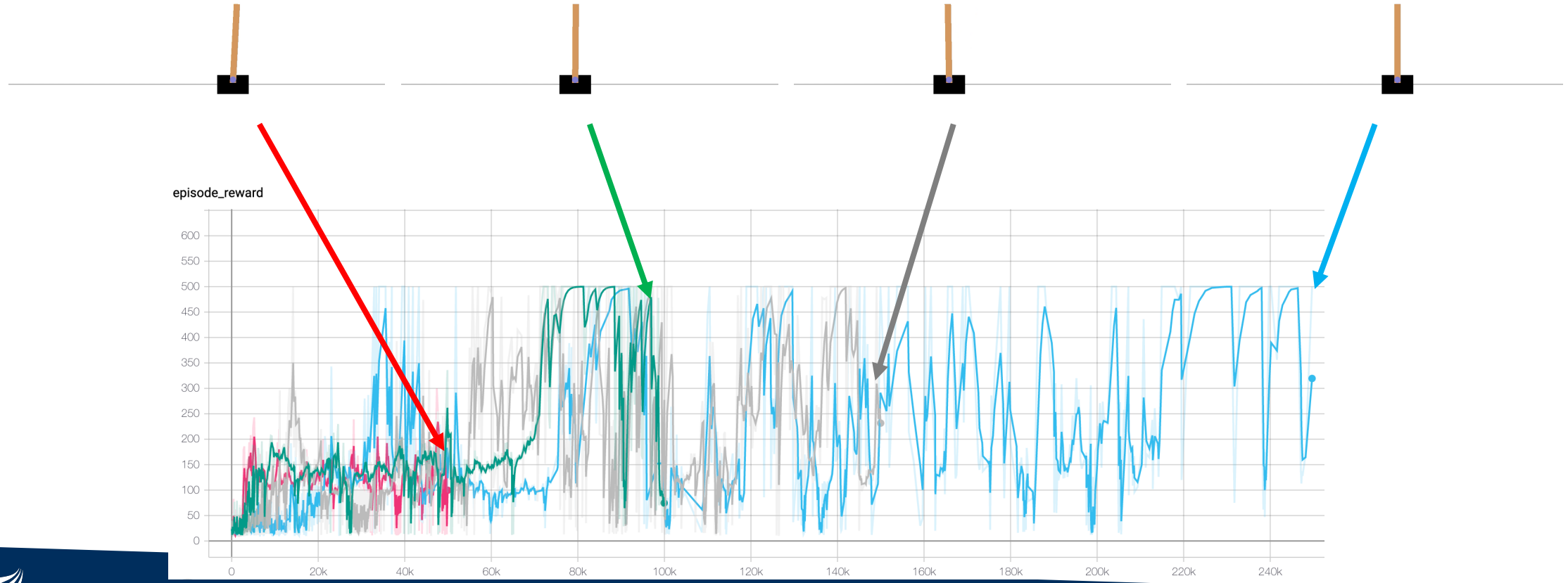
- You will train a **DQN** agent to learn how to balance a pole on a cart
- **Environment**: ***CartPole-v1***
- **Observation**: type: Box(4) = continuous
  - Cart Position -4.8 4.8
  - Cart Velocity -Inf Inf
  - Pole Angle -24 deg 24 deg
  - Pole Velocity At Tip -Inf Inf
- **Actions**: type: Discrete(2)
  - Push cart to the left
  - Push cart to the right



Assignment setup CartPole

1. Make Python >=3.5 virtual environment e.g. with Anaconda/virtualenv
2. Install OpenAI gym:
   - https://github.com/openai/gym#installation
3. Install tensorflow+tensorboard<=1.15.0
4. Install Stable-Baselines:
   - https://stable-baselines.readthedocs.io/en/master/guide/install.html
5. Train the agent
   - Find DQN example on stable-baselines and run the code to train the agent
6. Observe the training with TensorBoard

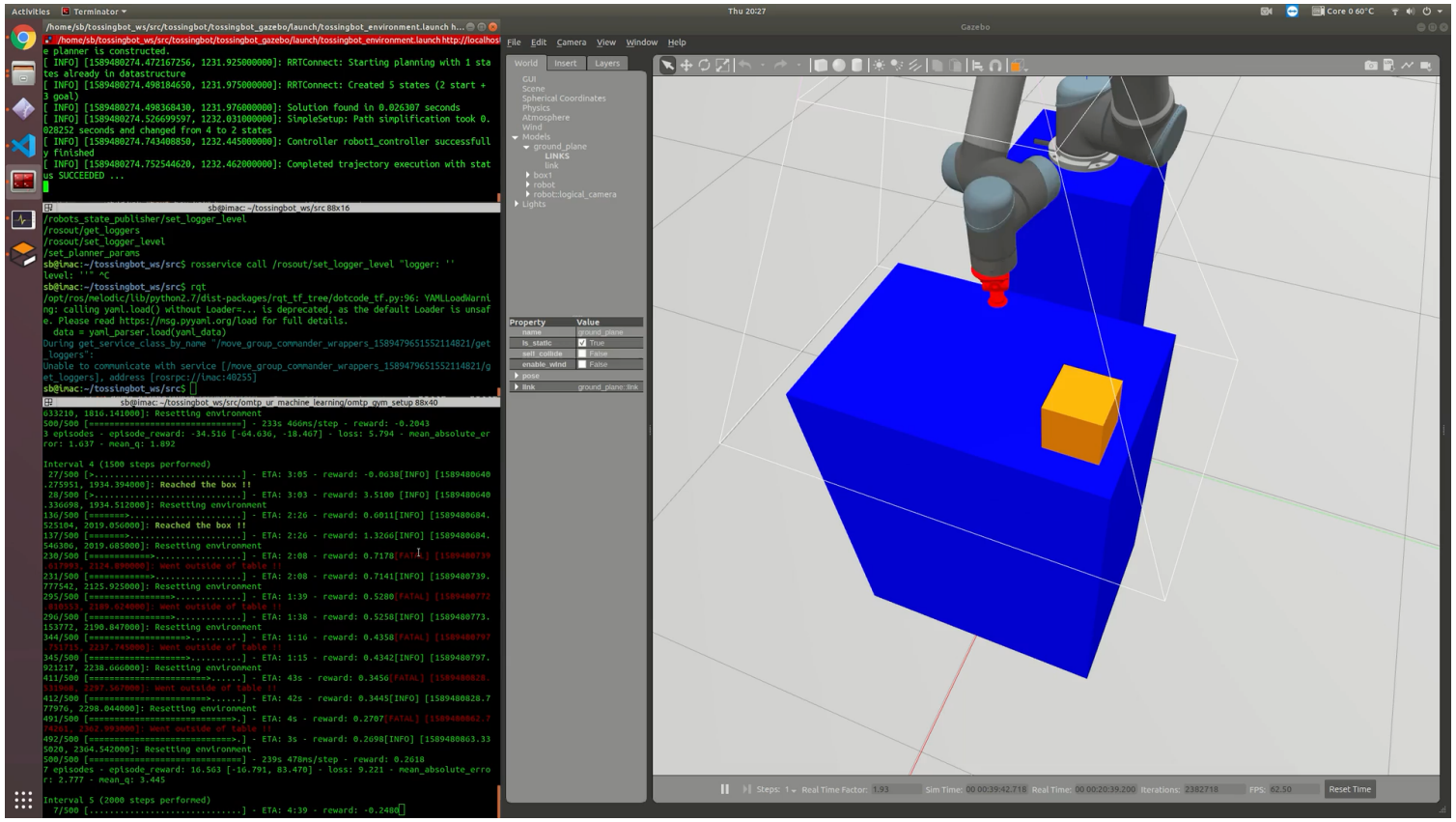
CartPole: trained agent with DQN



DQN implementation in Stable-Baselines

```
import gym

from stable_baselines.common.vec_env import DummyVecEnv
from stable_baselines.deepq.policies import MlpPolicy
from stable_baselines import DQN

# CartPole-v1
env = gym.make('CartPole-v1')
model = DQN(MlpPolicy, env, verbose=1, tensorboard_log='./logs')

# Set whether to train or test
train_model = False
model_filename = 'dqn_cartpole_100000'

# Train model
if train_model:
      model.learn(total_timesteps=100000)
      model.save(model_filename)

# Test model
else:
      # del model # remove to demonstrate saving and loading
      model = DQN.load(model_filename)

      # Enjoy trained agent
      obs = env.reset()
      while True:
            action, _states = model.predict(obs)
            obs, rewards, dones, info = env.step(action)
            env.render()
```

OMTP Factory – reaching for an object



https://youtu.be/1nsNMHdZGF0