Reinforcement Learning

Reinforcement Learning (RL)

Explains

- Introduction to RL
- Model-based RL
- Model-free RL
- Open problems in RL

Prerequisites

- Introduction to Machine Learning

Credits

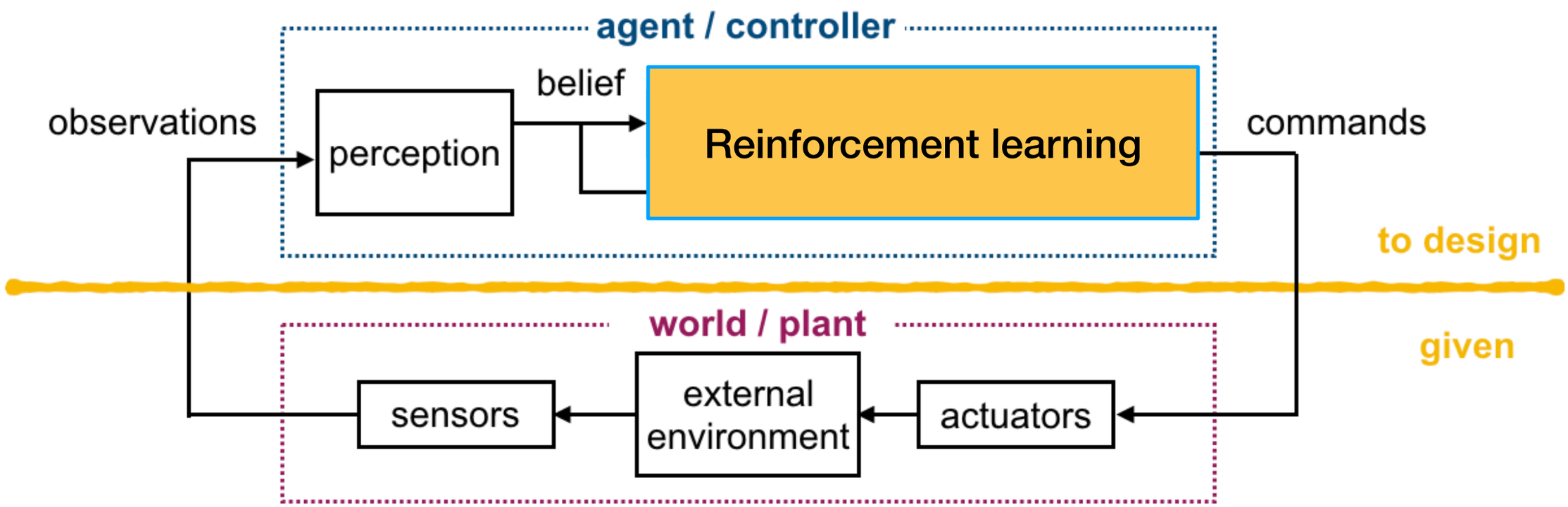
- Julian Zilly, 20.11.19, 07.12.20

These slides are part of the Duckietown project.
For more information about Duckietown, see the website http://duckietown.org

Big picture

Reinforcement Learning

Reinforcement Learning: Today's lecture

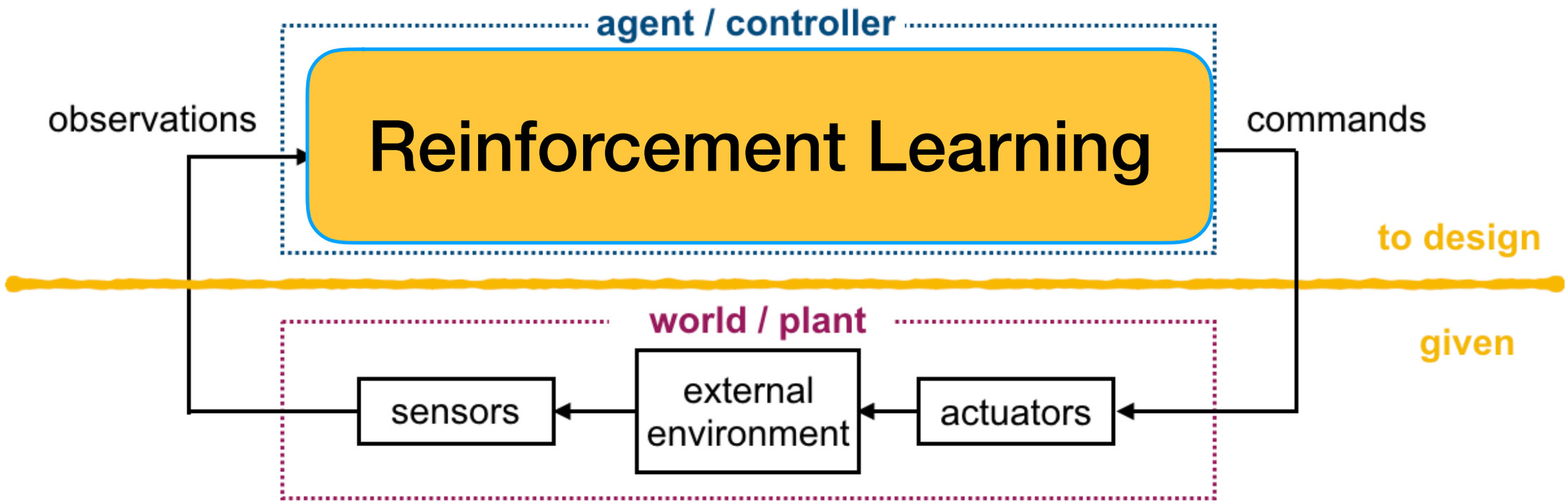


Sutton and Barto: http://incompleteideas.net/book/bookdraft2017nov5.pdf
Good overview: https://arxiv.org/pdf/1806.09460.pdf

Big picture

Reinforcement Learning

Reinforcement Learning: Today's lecture



Sutton and Barto: http://incompleteideas.net/book/bookdraft2017nov5.pdf
Good overview: https://arxiv.org/pdf/1806.09460.pdf

Example of Reinforcement Learning

Example: Learning to walk

You move muscles, observe outcome and try to improve the movements as you gain more experience.

https://en.wikipedia.org/wiki/Carrot_and_stick#/media/File:Carrot_and_stick.svg

$$\max_{\pi_\theta} \mathbb{E}[\sum_t \gamma^t \underbrace{r(s_t, a_t)}_{\text{reward given state s, action a}}]$$

The learning cake analogy by LeCun

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Slide by Yann Lecun, 2016 NIPS Keynote Presentation,

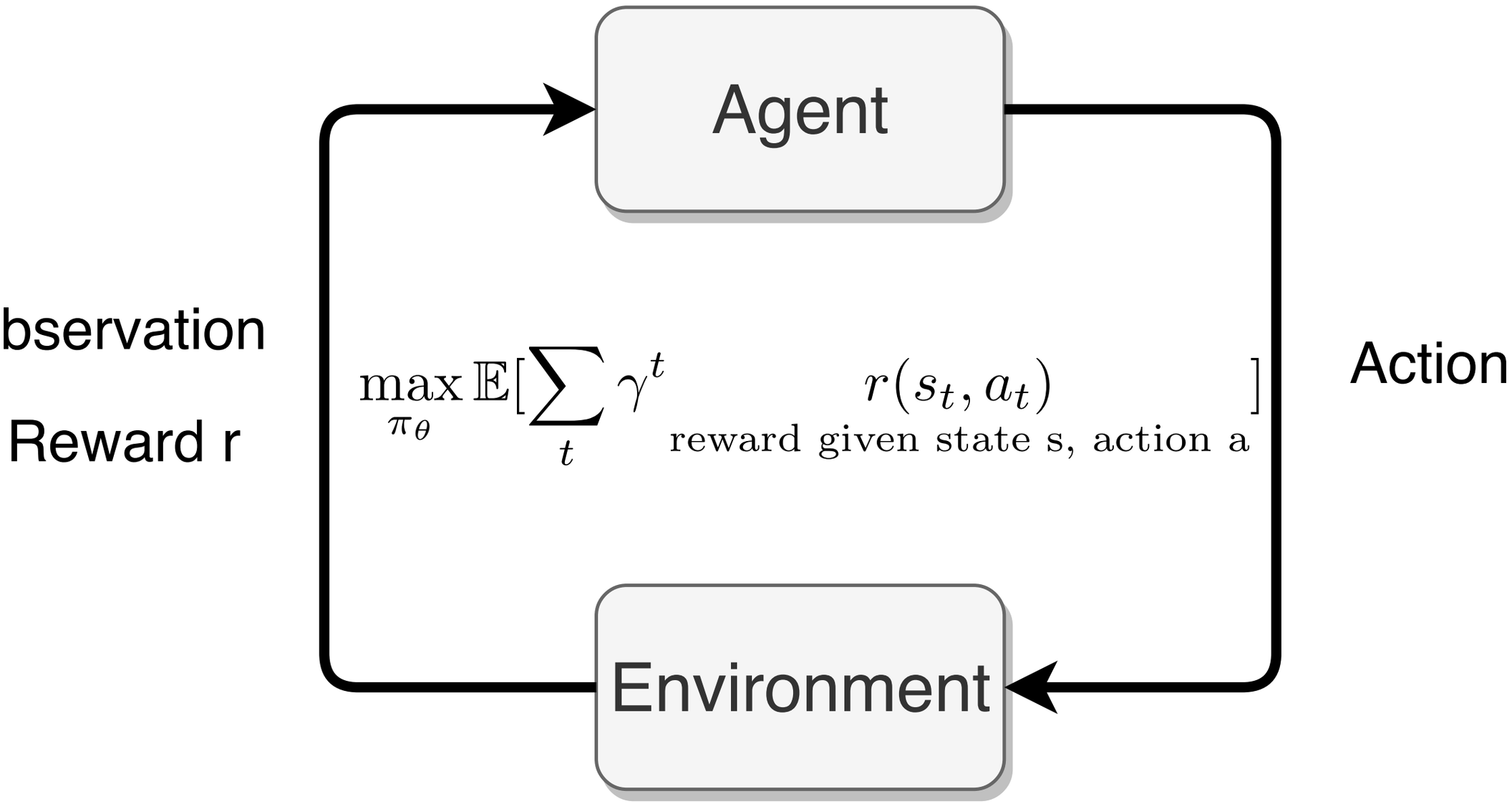
Why reinforcement learning?

- How can **agents learn to interact with the world** by maximizing rewards?
- Possible path to true autonomy
- Subject of a lot of research since the middle of the 20th century

http://people.idsia.ch/~juergen/robbyraus.jpg

https://de.wikipedia.org/wiki/M%C3%BCnchhausen-Trilemma#/media/File:M%C3%BCnchhausen-Sumpf-Hosemann.png

Reinforcement Learning



Sutton and Barto: http://incompleteideas.net/book/the-book-2nd.html
Good overview: https://arxiv.org/pdf/1806.09460.pdf

Elements of Reinforcement Learning

1. *Cost function approximation (value based, policy based)*

2. *Dynamics approximation (model-based vs. model-free)*

3. *Policy (exploration, exploitation)*

4. *Learning (on or off-policy)*

Cost function approximation

- ***Supervised learning:***
  - No feedback
  - Only consider current cost

- ***Reinforcement learning:***
  - Take into account path through new states
  - Planning required

- **Example using object classification setting for supervised learning:**
  - The next image you see depends on the last image class you predicted

What is special?
Cost function

The true cost function is unknown. Only rewards are provided.

- **Supervised learning:**
  - Often relies on gradient descent
  - Assumes that true cost function is known
- **Reinforcement learning:**
  - Unclear how to calculate gradients reliably
  - Need to approximate cost function
  - May have delayed rewards

Cost function approximation

Bellman equation

- Value function determines optimal future cost / rewards

$$\underbrace{V^*(s)}_{\text{value fct.}} = \max_a \Big(\underbrace{r(s,a)}_{\text{intermediate cost L(s,a)}} + \gamma \sum_{s'} P(s'|s,a)\, \underbrace{V^*(s')}_{\text{value fct.}} \Big)$$

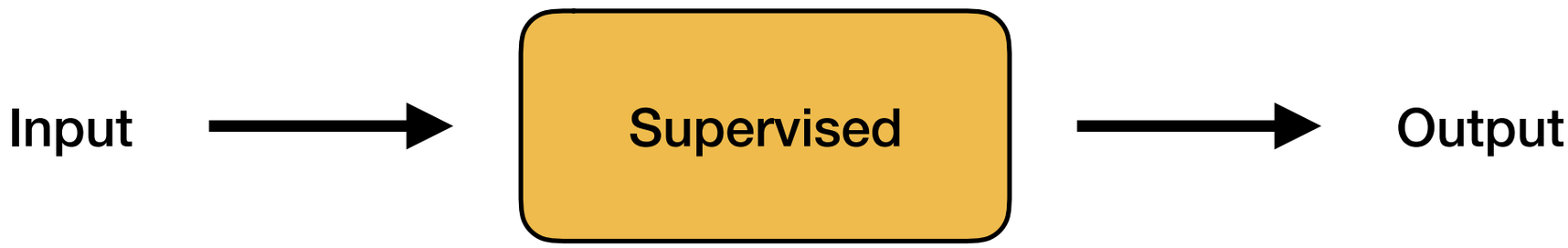
- **Bellman principle:** "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." (Bellman, 1957, Chap. III.3.)

https://medium.com/@jonathan_hui/rl-model-based-reinforcement-learning-3c2b6f0aa323

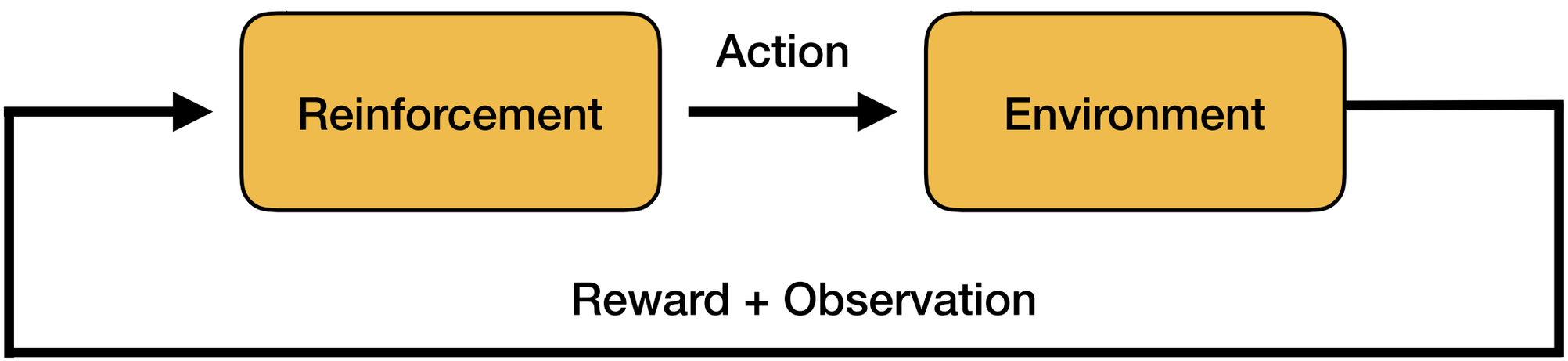
What is special?
Dynamics and Feedback

The effect of actions on states may be unknown.

- **Supervised learning - (Feedforward learning) - no interaction**



- **Reinforcement learning - (Feedback learning) - interactive**



Dynamics approximation

- Modeling of transition probabilities

$$P(s'|s,a)$$

- Becomes important for problems where the greedy-best action is not the overall best action

Policy (exploration / exploitation)

- Based on what has been learned:
  - Do we try unexplored states and actions?
  - Do we follow and exploit what we have learned so far?

- Example:
  Restaurants in a city. Do we go to our favorite place or try a new restaurant?

Learning (on or off-policy)

- 1) Changing the policy leads to changes in data distribution.
- 2) But want experience for the policy we are using.

- **On policy:**
  Update the same exploration policy while collecting more experience.

- **Off policy:**
  Use an exploration policy. Collect experience. Learn an exploitation policy based on collected experience. Exploitation policy is not used to collect experience.

Recap: Contrast to supervised learning

Supervised learning

1. Cost function known - can compute gradients easily
2. We know (state, action)-dynamics (usually static)
3. Updates to learned function do not affect observed data afterwards
4. Exploration is not needed. Gradients provide necessary update information

Reinforcement learning

1. Cost function unknown - very random gradients if any
2. We might not know (state, action)-dynamics
3. Updates to model affect which data is observed
4. Exploration is often random and random exploration is unlikely to work in high dimensions

Model-based vs. model-free reinforcement learning

- Model

$$p(s_{t+1}|s_t, a_t)$$

- Policy / controller

$$\pi_\theta(a_t|s_t)$$

- Objective

$$\max_{\pi_\theta} \mathbb{E}[\sum_t \gamma^t r(s_t, a_t)]$$

https://medium.com/@jonathan_hui/rl-model-based-reinforcement-learning-3c2b6f0aa323

Model-based reinforcement learning

Focus on system dynamics

- Model

$$p(s_{t+1}|s_t, a_t)$$

- Policy / controller

$$\cancel{\pi_\theta(a_t|s_t)}$$

- Objective

$$\max_{\pi_\theta} \mathbb{E}[\sum_t \gamma^t r(s_t, a_t)]$$

https://medium.com/@jonathan_hui/rl-model-based-reinforcement-learning-3c2b6f0aa323

Model-based reinforcement learning

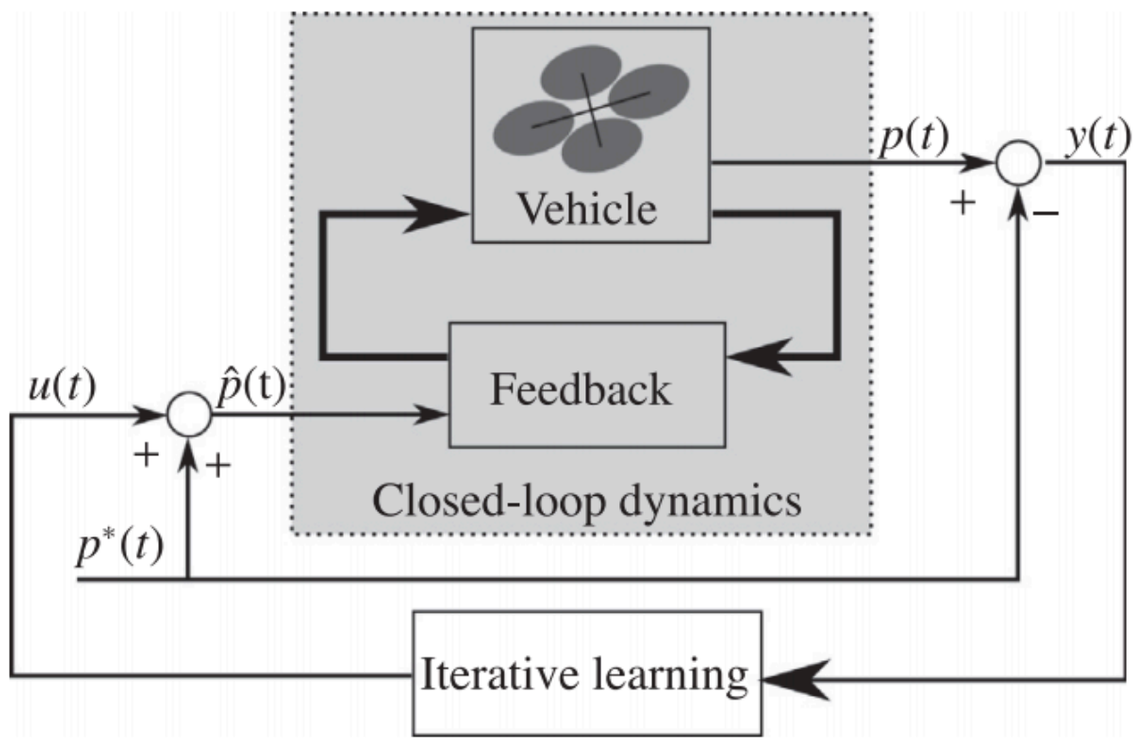
- Uses prior knowledge on **system dynamics** and **cost function**
- **Usual workflow**:
  - 1. Collect data and fit model
  - 2. Compute actions based on model and cost
  - 3. Run controller and collect data -> Repeat steps

- MPC, LQR, iLQC, ...

https://medium.com/@jonathan_hui/rl-model-based-reinforcement-learning-3c2b6f0aa323

Model-based reinforcement learning

- Often applied in robotics because of sample efficiency
  - Hehn (IDSC), Schöllig (IDSC), Trajectory optimization 1., 2.
  - Jan Peters (MPI, Darmstadt), Ball in the cup



Hehn, Markus, and Raffaello D'Andrea. "A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers." *Mechatronics* 24.8 (2014): 954-965.

Model-based reinforcement learning

- Discrete Time Linear Quadratic Regulator (LQR)

- Linear

$$x_{n+1} = A_n x_n + B_n u_n$$

- Regulator

$$\delta x_n =: x_n$$
$$\delta u_n =: u_n$$

Optimal Learning Control, Lecture 5: http://www.adrlab.org/doku.php/adrl:education:lecture:fs2015

Model-based reinforcement learning

- **Control:** Linear state feedback

$$\underset{\text{action}}{u} = -K \underset{\text{state}}{x}$$

- **Cost:** Quadratic

$$\mathcal{J} = \sum_{n=0}^{\infty} \frac{1}{2} x_n^T Q x_n + \frac{1}{2} u_n^T R u_n + u_n^T P x_n$$

$$\mathcal{J} = \sum_{n=0}^{\infty} \underset{\text{reward}}{r(s,a)} = \sum_{n=0}^{\infty} \underset{\text{intermediate cost}}{L(x,u)}$$

Optimal Learning Control, Lecture 5: http://www.adrlab.org/doku.php/adrl:education:lecture:fs2015

Model-based reinforcement learning

- **Ansatz:**

$$V^*(x_n) = \frac{1}{2} x_n^T S_n x_n$$

$$S_n = S_{n+1} =: S$$

- **Next steps:**

$$V(x_n) = \min_{u_n} (L(x_n, u_n) + V(x_{n+1}))$$

$$x_{n+1} = Ax_n + Bu_n$$

- u* is found by taking derivative of right hand side and setting to 0

Optimal Learning Control, Lecture 5: http://www.adrlab.org/doku.php/adrl:education:lecture:fs2015

Model-based reinforcement learning

- Discrete time algebraic Riccati equation:

$$S = Q + A^T S A - (P^T + A^T S B)(R + B^T S B)^{-1}(P + B^T S A)$$

- Optimal control

$$u^* = -Kx = -(R + B^T S B)^{-1}(P + B^T S A)x$$

Optimal Learning Control, Lecture 5: http://www.adrlab.org/doku.php/adrl:education:lecture:fs2015

Break

- What questions do you have?

- Robotic ping pong: https://www.youtube.com/watch?v=Fhb26WdqVuE

- Learning to throw a pendulum: https://robohub.org/video-throwing-and-catching-an-inverted-pendulum-with-quadrocopters/

Summary - first section

- Reinforcement learning (RL) asks an agent to maximize rewards
- Different to supervised learning:
  - State dynamics unknown
  - Feedback dynamics
  - No cost function
  - Exploration vs. exploitation
- Model-based RL uses data to estimate a dynamics model, then optimizes behavior directly with dynamics model

Model-free reinforcement learning

Focus on policy

- Model

$$\cancel{p(s_{t+1}|s_t, a_t)}$$

- Policy / controller

$$\pi_\theta(a_t|s_t)$$

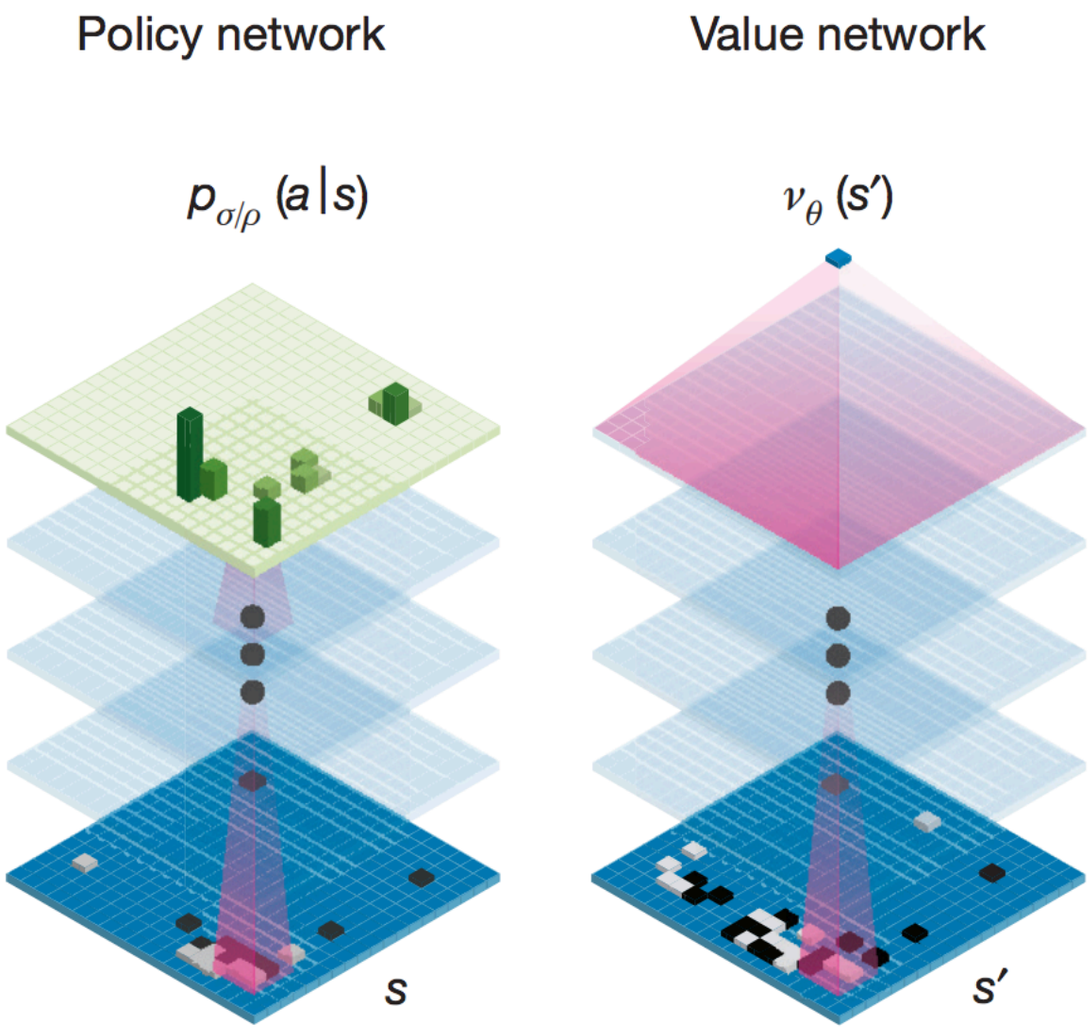
- Objective

$$\max_{\pi_\theta} \mathbb{E}[\sum_t \gamma^t r(s_t, a_t)]$$

https://medium.com/@jonathan_hui/rl-model-based-reinforcement-learning-3c2b6f0aa323

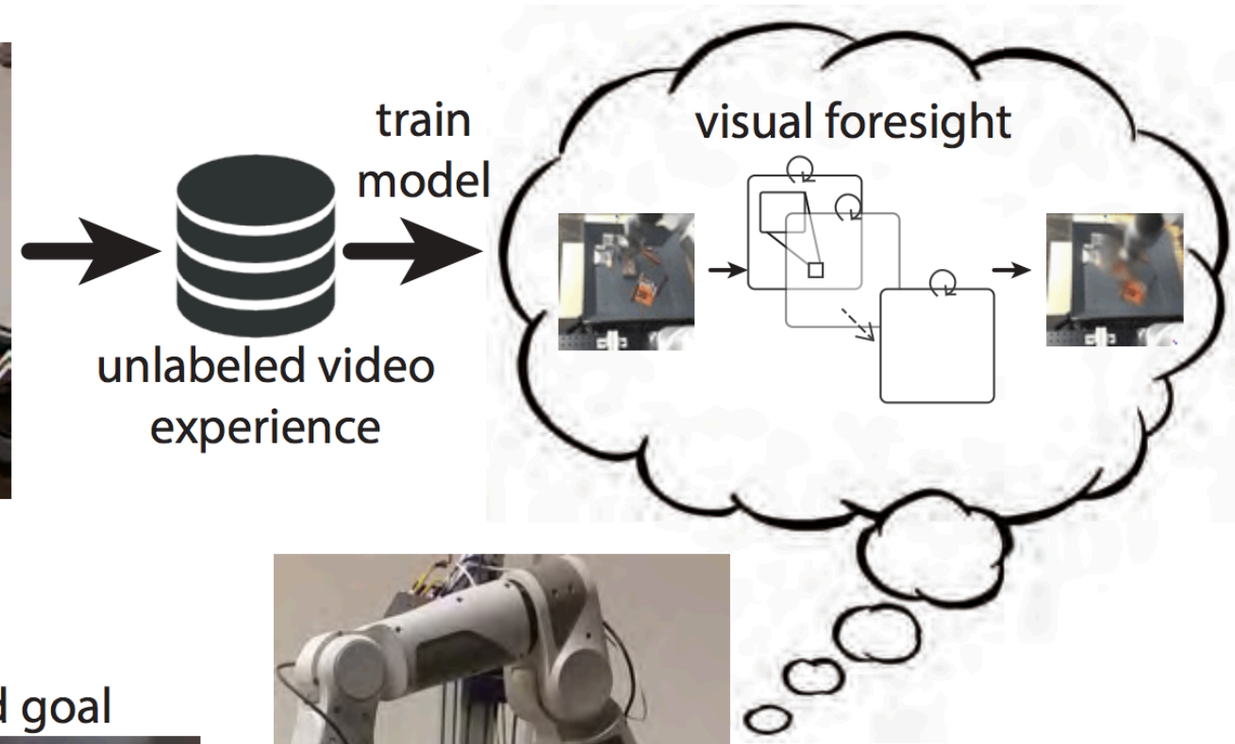
Model-free reinforcement learning

- The headlines started with Deepmind
  - Deep-Q learning - Atari
  - AlphaGo - Go
  - Alpha Zero - Chess
- Replaced years of hand-crafted algorithms and features to play with a computer
- Achieved in simulation domain



Model-free reinforcement learning

- Starting to be [applied to robots]



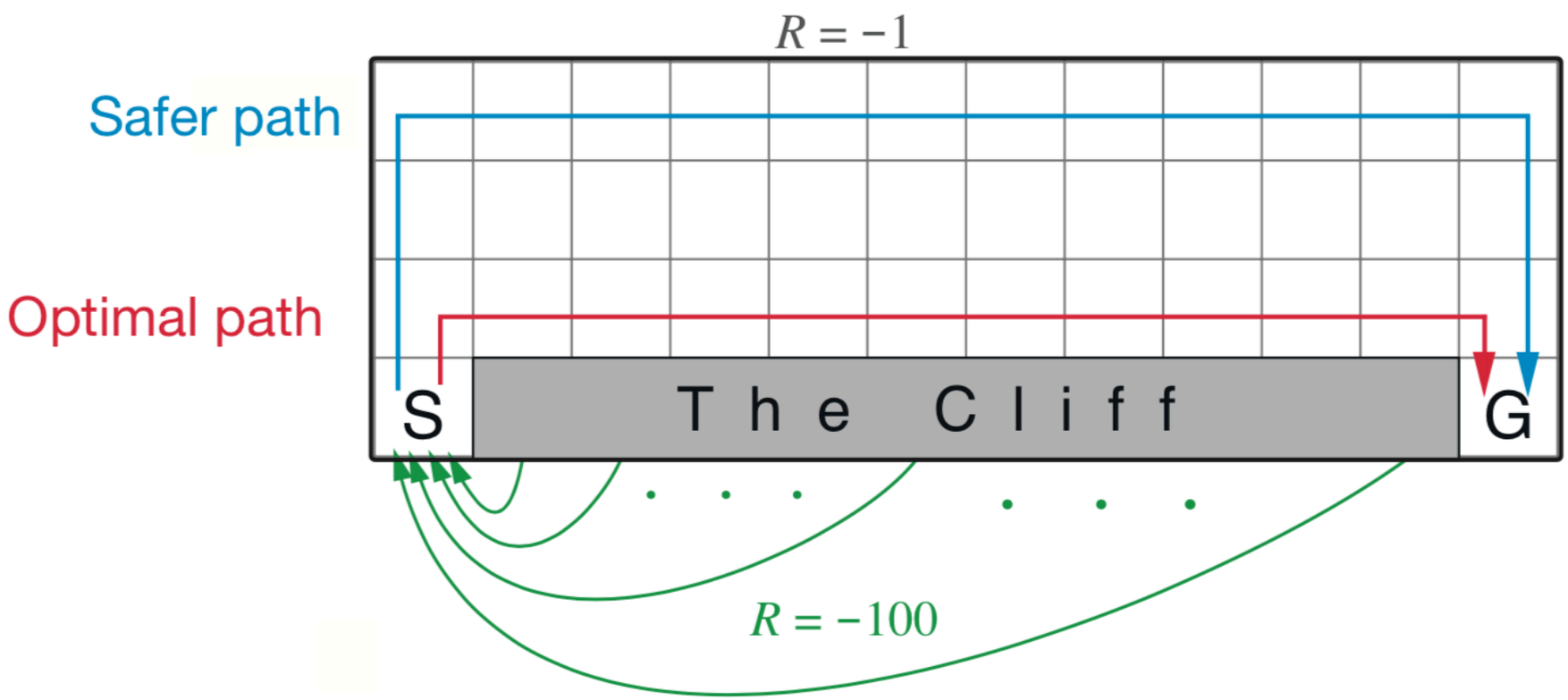
Finn, Chelsea, and Sergey Levine. "Deep visual foresight for planning robot motion." *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017.

Q-learning and value functions

- State-action pairs
- Markov assumption

$$V^*(s_t) = \max_{a_t} Q^*(s_t, a_t)$$

$$Q^*(s_t, a_t) = \max_a \mathbb{E}[r_t + \gamma r_{t+1} + \ldots | s_t, a_t]$$



Sutton and Barto: http://incompleteideas.net/book/the-book-2nd.html, p.132
https://en.wikipedia.org/wiki/Q-learning

Q-learning and value functions

- Q-learning state-action value estimates are updated with new experience
- Exploration required to get new relevant experiences

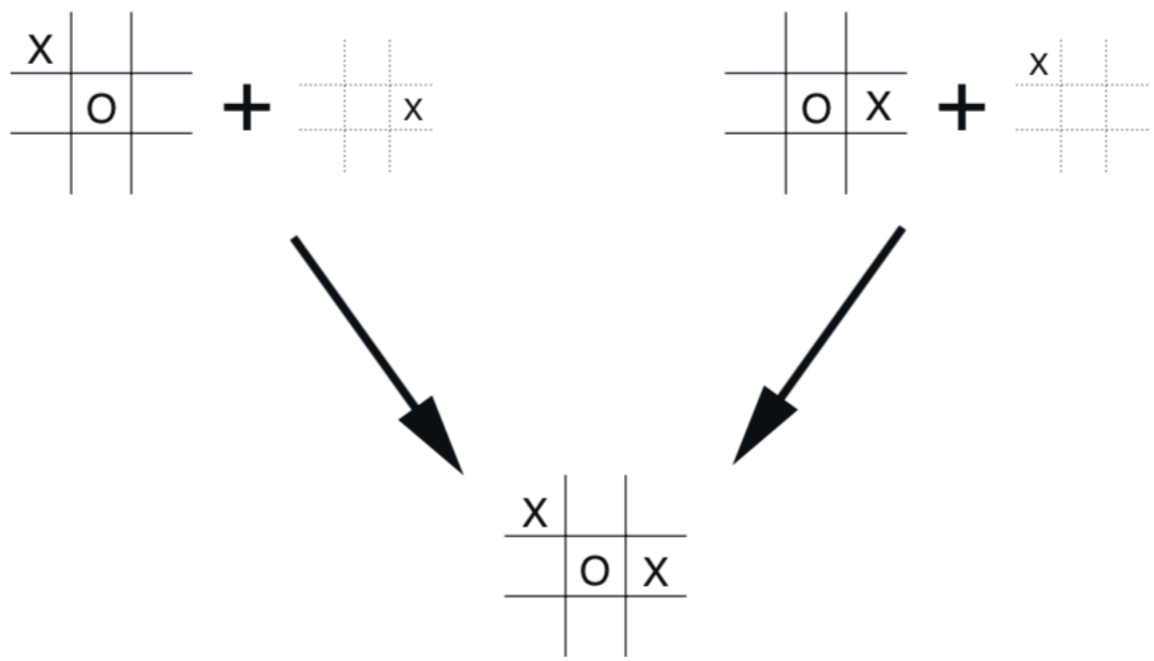
$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)\underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount}} \overbrace{\underbrace{\max_a Q(s_{t+1}, a)}_{\text{est. of optimal future value}}}^{\text{learned value}} \Big)$$

Sutton and Barto: http://incompleteideas.net/book/the-book-2nd.html, p.131

https://en.wikipedia.org/wiki/Q-learning

Q-learning and value functions

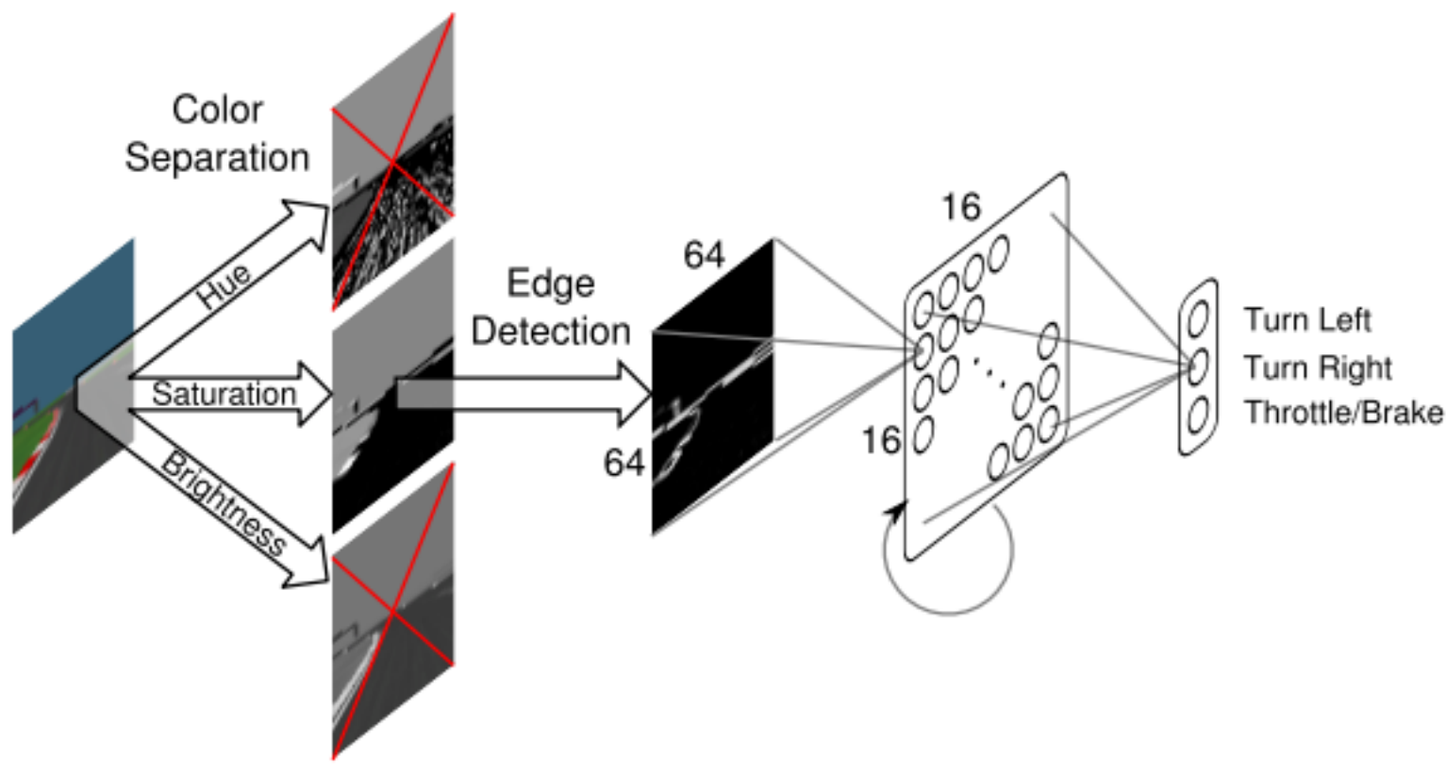
- Example applications - Tic, tac, toe and grid-worlds
- Large state-action spaces would be prohibitive



Sutton and Barto: http://incompleteideas.net/book/the-book-2nd.html, p.137
For complete tic, tac, toe solution: https://xkcd.com/832/

Deep reinforcement learning

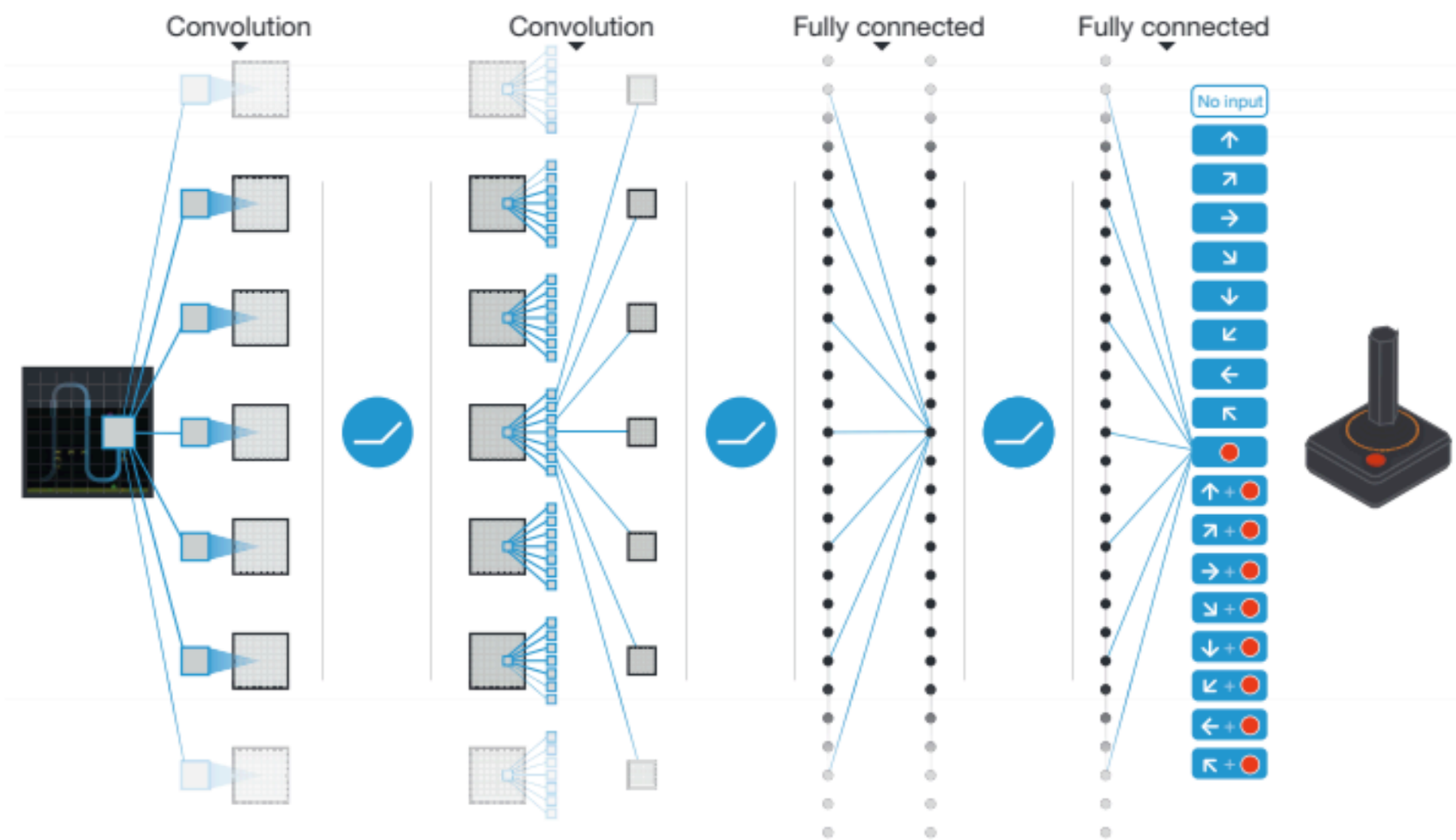
- Learning to drive in a simulator using evolutionary algorithms



Koutník, Jan, et al. "Evolving large-scale neural networks for vision-based reinforcement learning." *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013.

Deep Q-learning for human-level Atari

- Playing Atari games with Deep Q-networks



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

Training Deep Q-networks

- For episode = 1, M do
  - For t=1,T do
    - Select **epsilon-greedy action**
    - Execute **action, observe reward and image, store transition**
    - Sample **batch of saved transitions**
    - Perform **gradient descent step** w.r.t. network parameters
    - Every C steps **reset fixed target network**
  - End For
- End For

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

Training DeepQ and similar networks

- **Memory replay buffer - OFF POLICY LEARNING:**
  - Each transition (s, a, r, s') gets stored
  - Batches of transitions get sampled uniformly
  - Avoids correlating updates along a trajectory

- Inspired by what may happen during "dreaming" in humans

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

Training DeepQ and similar networks

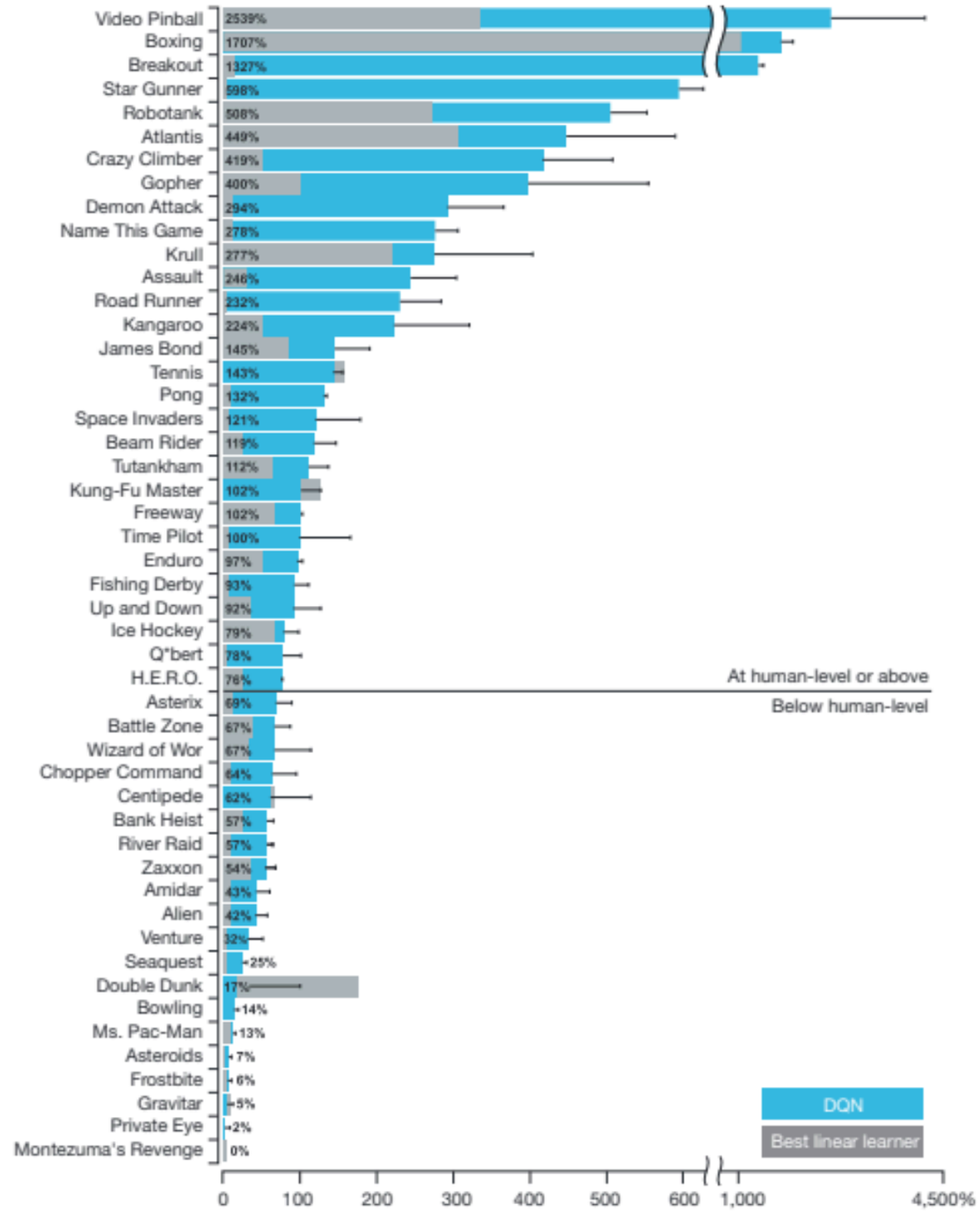
- **Loss function:**
  - One fixed target network
  - Tries to approximate reward + future value function

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')}[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)^2]$$

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

Training DeepQ and similar networks

- **Results**:
- Better performance than reference game testers in some games
- Not good at long-term planning
- Especially good at “reflex” games



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

Robotic priors

- How to make the best of the model-based and model-free world?

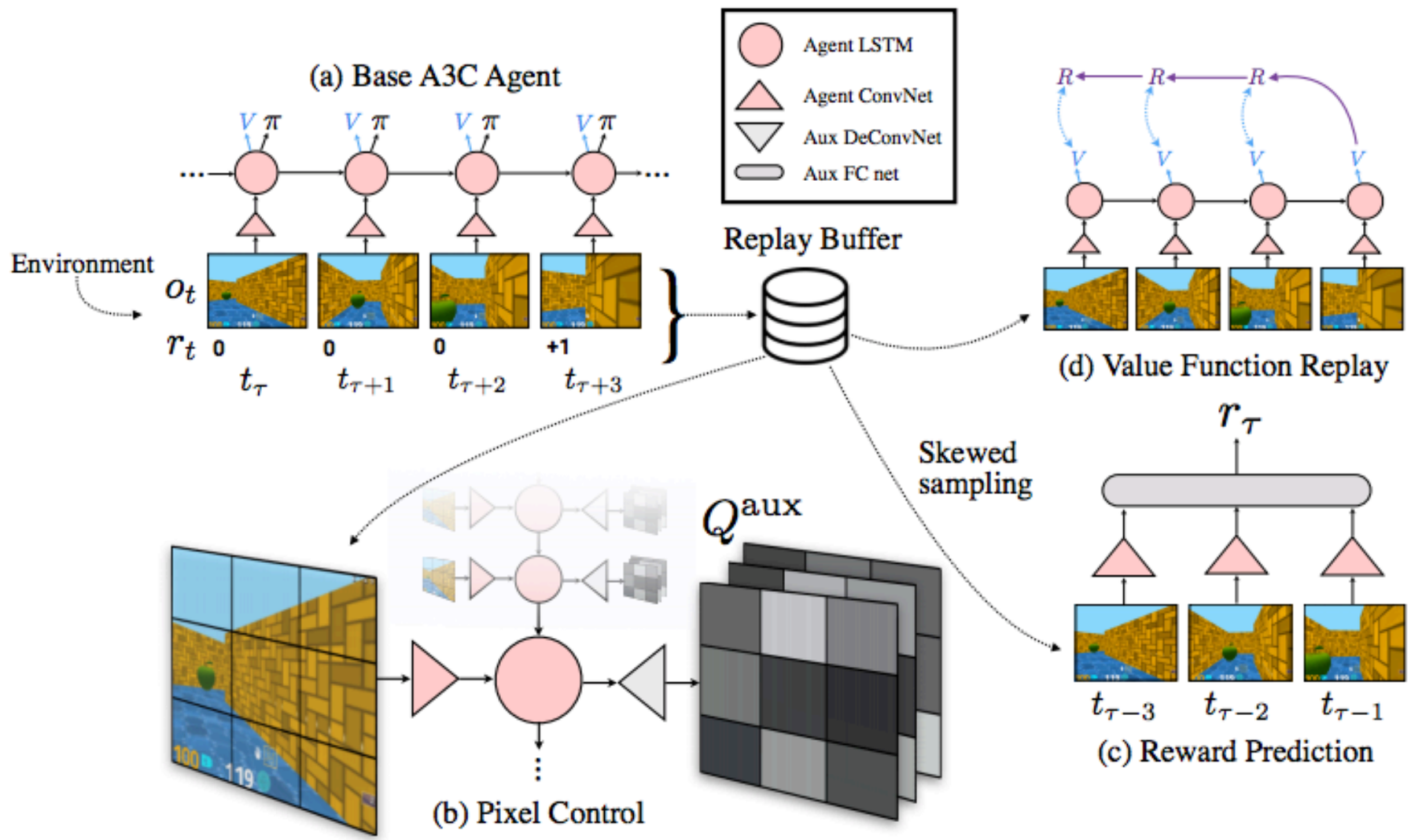
  - Model-free where structure is unknown
  - Model-based where structure is known

- Depends on data availability - more constraints for less data

Jonschkowski, Rico, and Oliver Brock. "Learning state representations with robotic priors." *Autonomous Robots* 39.3 (2015): 407-428.

Combining model-based and model-free aspects

- Auxiliary loss functions to improve learning



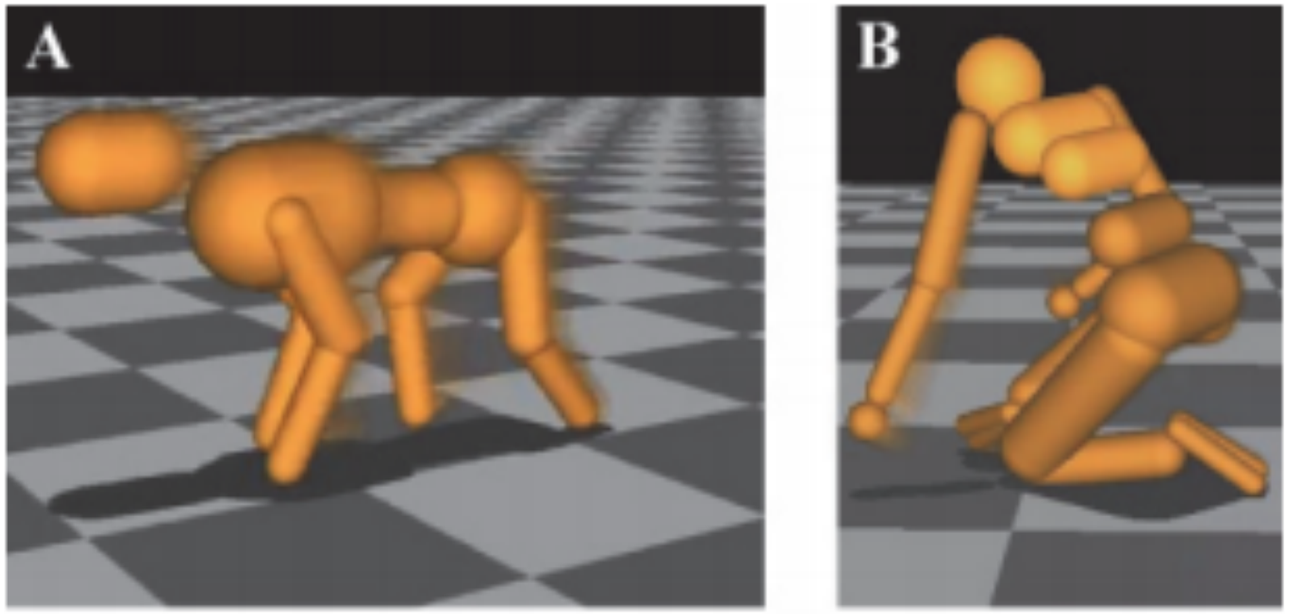
Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *arXiv preprint arXiv:1611.05397*(2016).

Words of caution: Deep RL

- https://xkcd.com/1838/
- https://imgs.xkcd.com/comics/machine_learning.png

Words of caution: Sample complexity

- Random search can compete with Deep RL methods



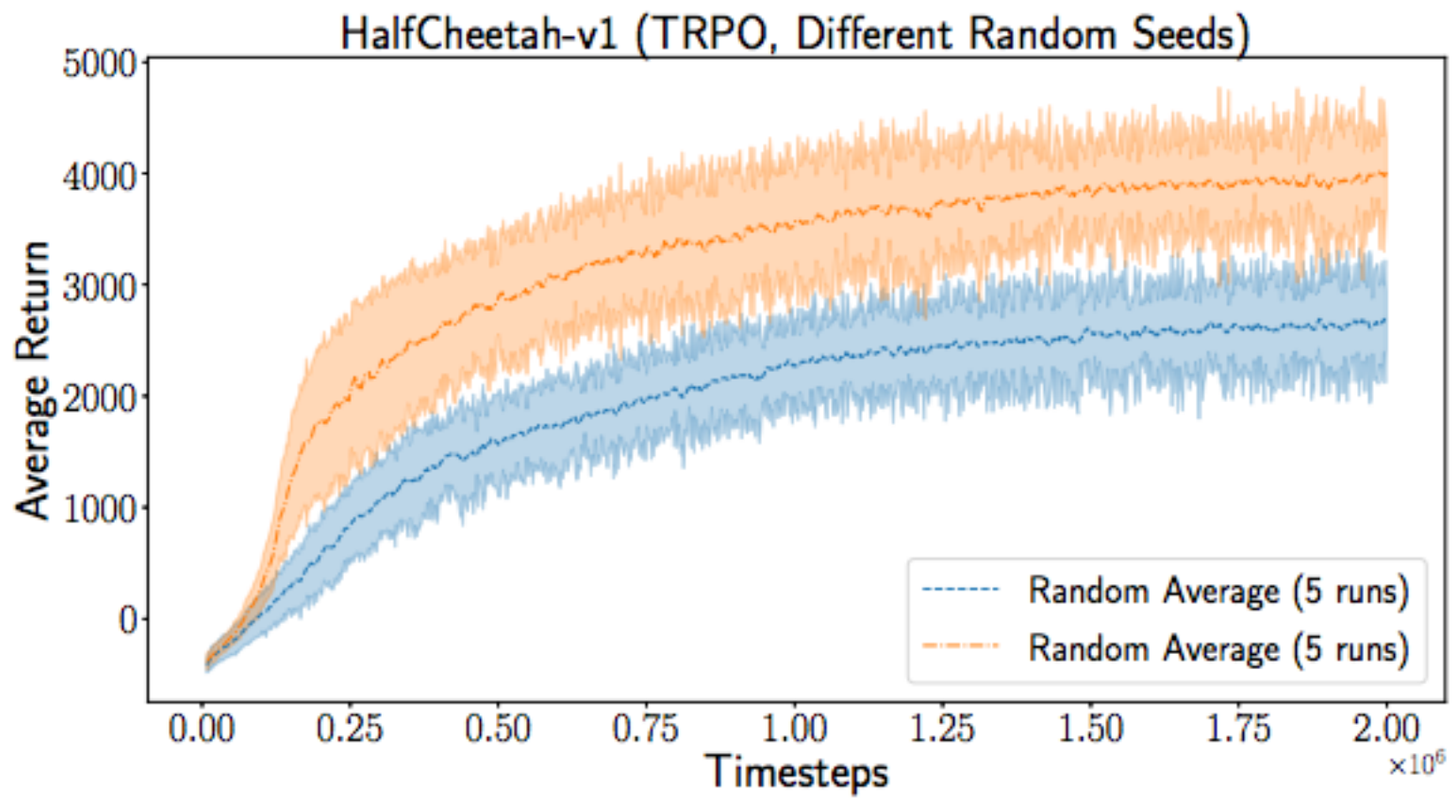
		Average # episodes to reach reward threshold						
Task	**Threshold**	**ARS**				**NG-lin**	**NG-rbf**	**TRPO-nn**
		V1	**V1-t**	**V2**	**V2-t**			
Swimmer-v1	325	100	100	427	427	1450	1550	N/A [4]
Hopper-v1	3120	89493	51840	3013	1973	13920	8640	10000
HalfCheetah-v1	3430	10240	8106	2720	1707	11250	6000	4250
Walker2d-v1	4390	392000	166133	89600	24000	36840	25680	14250
Ant-v1	3580	101066	58133	60533	20800	39240	30000	73500
Humanoid-v1	6000	N/A	N/A	142600	142600	≈130000	≈130000	UNK[5]

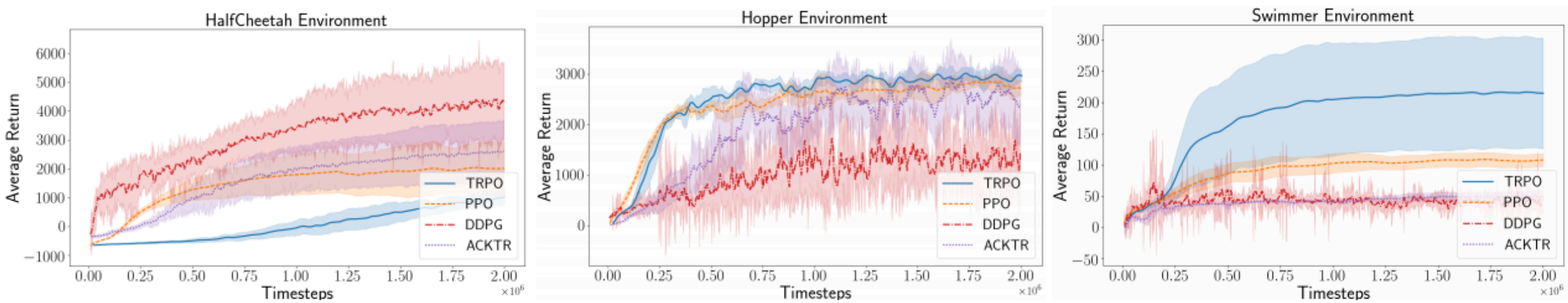
Todorov, Emanuel, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control." *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

Mania, Horia, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning." *arXiv preprint arXiv: 1803.07055*(2018).

Words of caution: Non-reproducibility

- Random seeds change solution
- Performance changes across environments

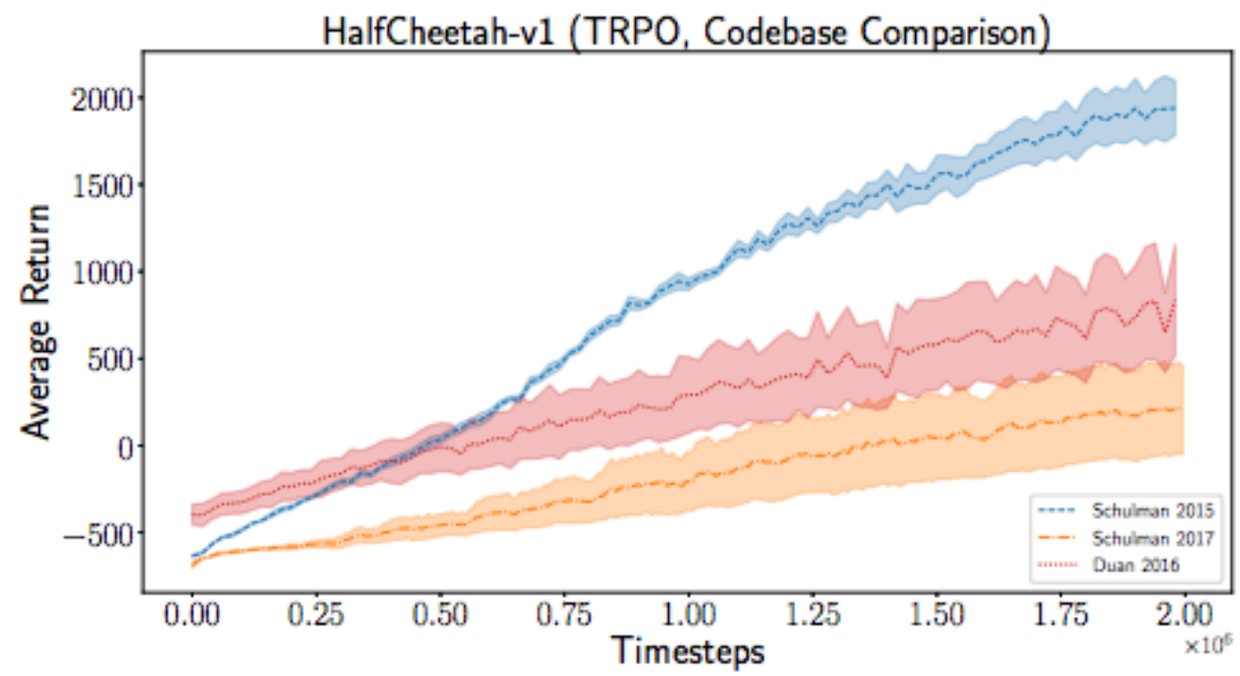


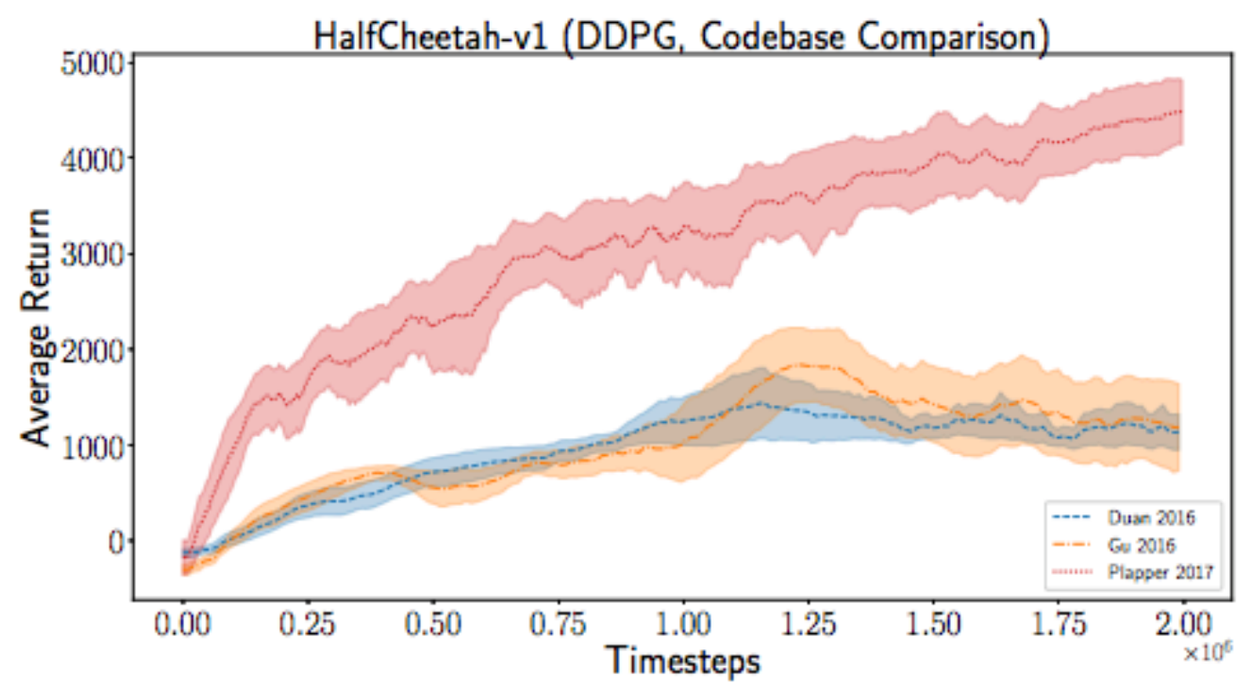


Henderson, Peter, et al. "Deep reinforcement learning that matters." *arXiv preprint arXiv:1709.06560* (2017).

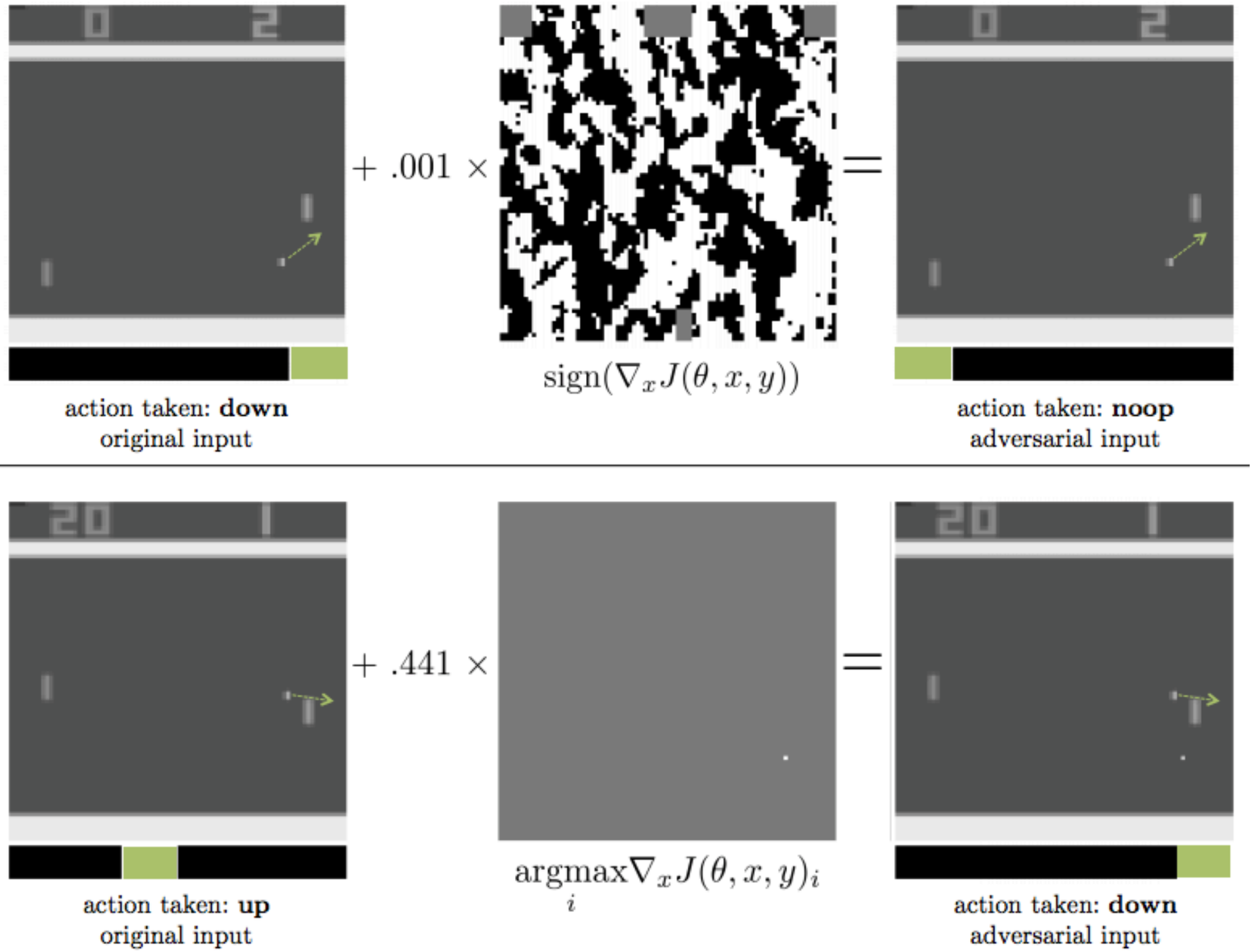
Words of caution: Non-reproducibility

- Different code bases - same algorithm, same hyperparameter settings





Words of caution: Adversarial attacks



Huang, Sandy, et al. "Adversarial attacks on neural network policies." *arXiv preprint arXiv:1702.02284* (2017).

Takeaway

- **What is reinforcement learning? Why is it important?**

- **Model-based vs. model-free reinforcement learning**
  - Different sample complexity regimes and application areas

- **Words of caution**
  - Key difficulties still to overcome
  - Further study: Deep Mind youtube series