Brain-Inspired Artificial Intelligence 5: Introduction to Deep Reinforcement Learning

Eiji Uchibe

Dept. of Brain Robot Interface

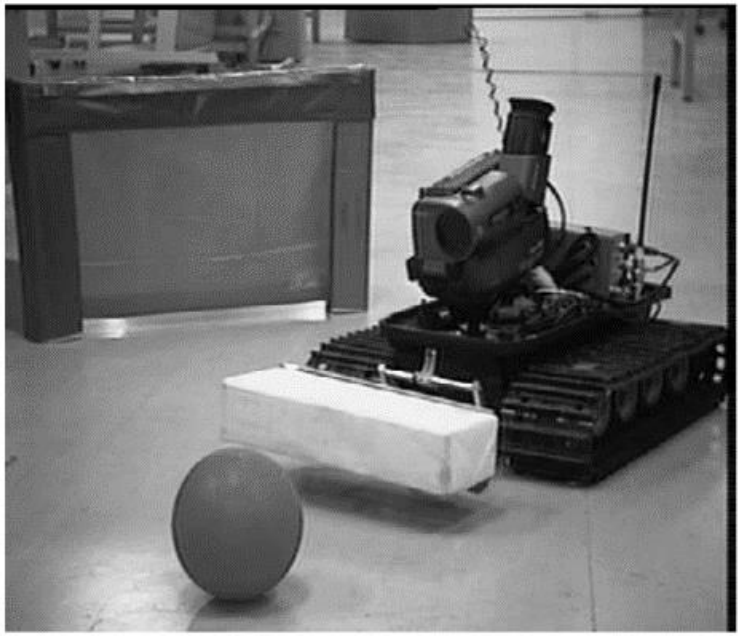
ATR Computational Neuroscience Labs.

Large-Scale Reinforcement Learning

- So far we have assumed that states and actions were discrete and it was possible to represent value functions and policy by a lookup table
- It is NOT true for realistic tasks
  - Go: 10^{170} states
  - Helicopter/Mountain Car: continuous state space
  - Robots: informal state space

Discretizing State Space

- When a state is given by a continuous vector, it is discretized manually
  - Soccer robot whose task is to shoot a ball into a goal



http://rraj.rsj-web.org/ja_history

M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine Learning, 23, 279-303.

Discretizing State Space

- 5 state variables: x-position and size of the ball and x-position, size and orientation of the goal

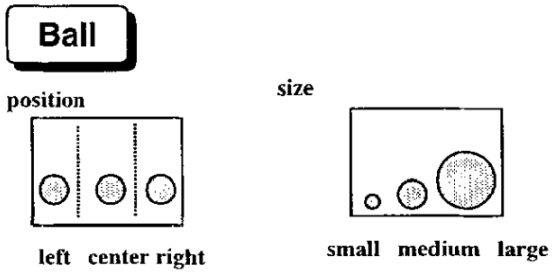
- The number of states grows exponentially as the number of features increases

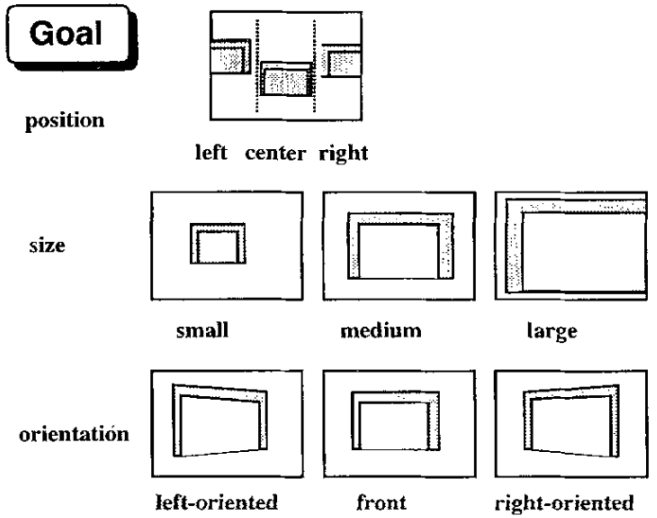
$$3^2 \times 3^3 = 243$$

the number of ball states

the number of goal states

the total number of states





- So we need to approximate value functions and policy

M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. (1996). Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine Learning, 23, 279-303.

Function approximation

- Reminder: Update rule of Q-learning for discrete systems

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t$$
$$\delta = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)$$

- If $Q(s, a)$ is approximated by some function $\hat{Q}(s, a; \boldsymbol{w})$ parameterized by $\boldsymbol{w}$, the update rule is given by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\delta\nabla_{\boldsymbol{w}}\hat{Q}(s, a; \boldsymbol{w})$$
$$\delta = r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{w}) - \hat{Q}(s_t, a_t; \boldsymbol{w})$$

- What kind of approximators should we use?

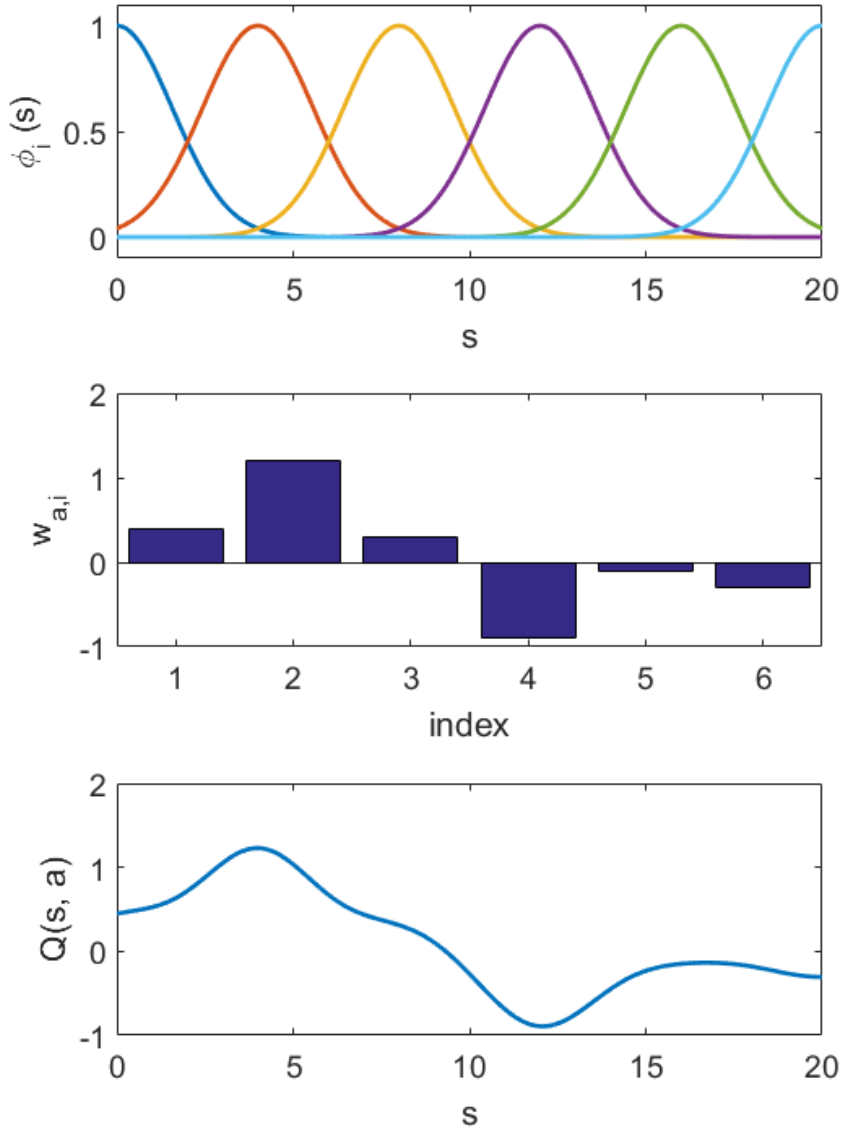
Linear Function Approximation

- A linear function approximator is introduced to deal with continuous states and discrete actions

$$\hat{Q}(s, a; \boldsymbol{w}) = \boldsymbol{w}_a^\top \boldsymbol{\phi}(s)$$

  - $\boldsymbol{w}_a$: weight parameter vector for action a
  - $\boldsymbol{w} = \{\boldsymbol{w}_a\}$
  - $\boldsymbol{\phi}(s)$: basis function vector

- Under some assumptions, convergence proof is given



Failure of Nonlinear Function Approximation

- Task: Mountain-Car
  - Driving an underpowered car up a steep mountain road
- V^{π} is approximated by a neural network with 80 hidden units
- Value iteration results in divergence

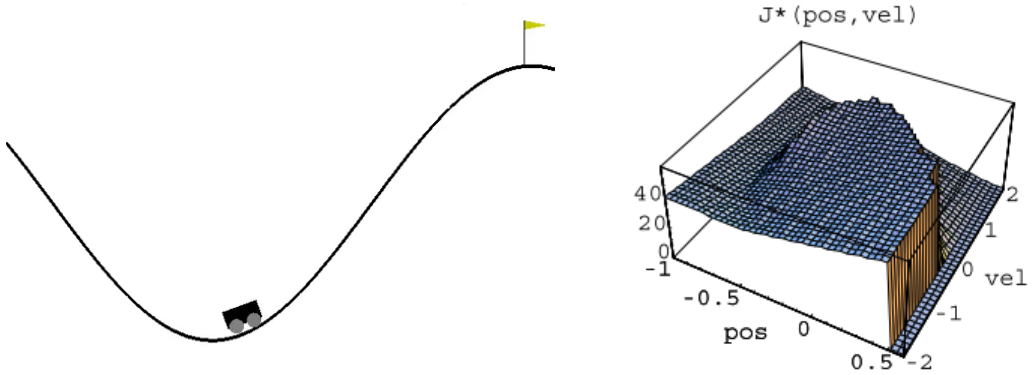


Figure 5: The car-on-the-hill domain. When the velocity is below a threshold, the car must reverse up the left hill to gain enough speed to reach the goal, so J^* is discontinuous.

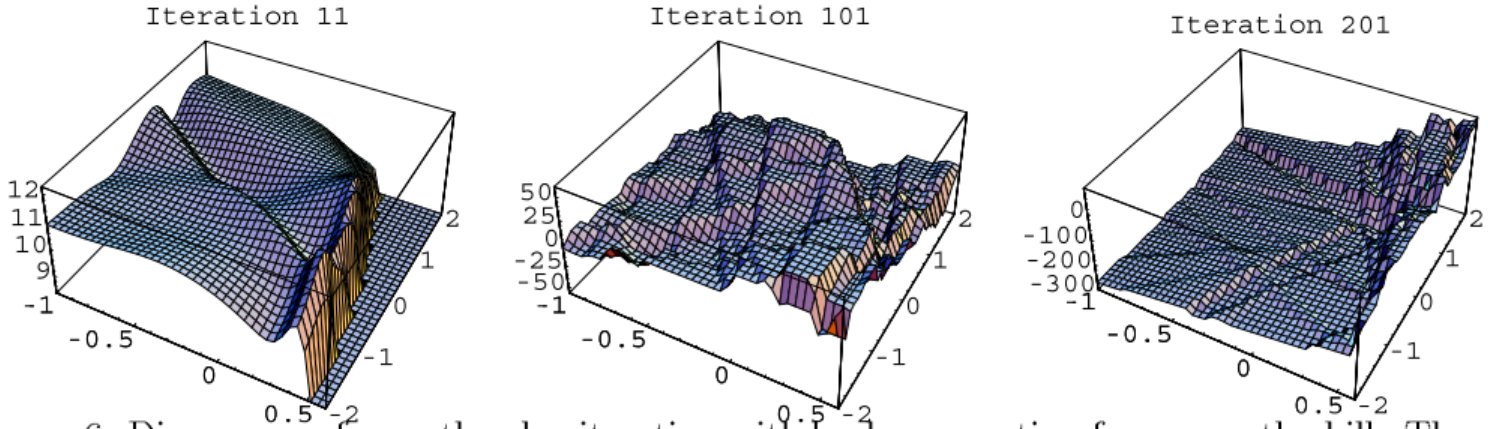
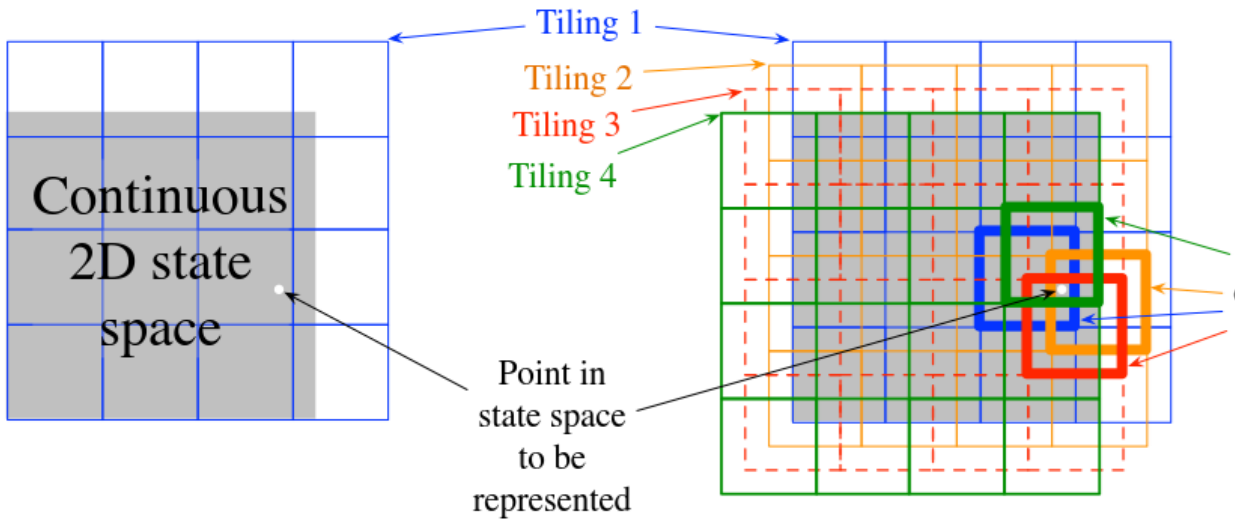


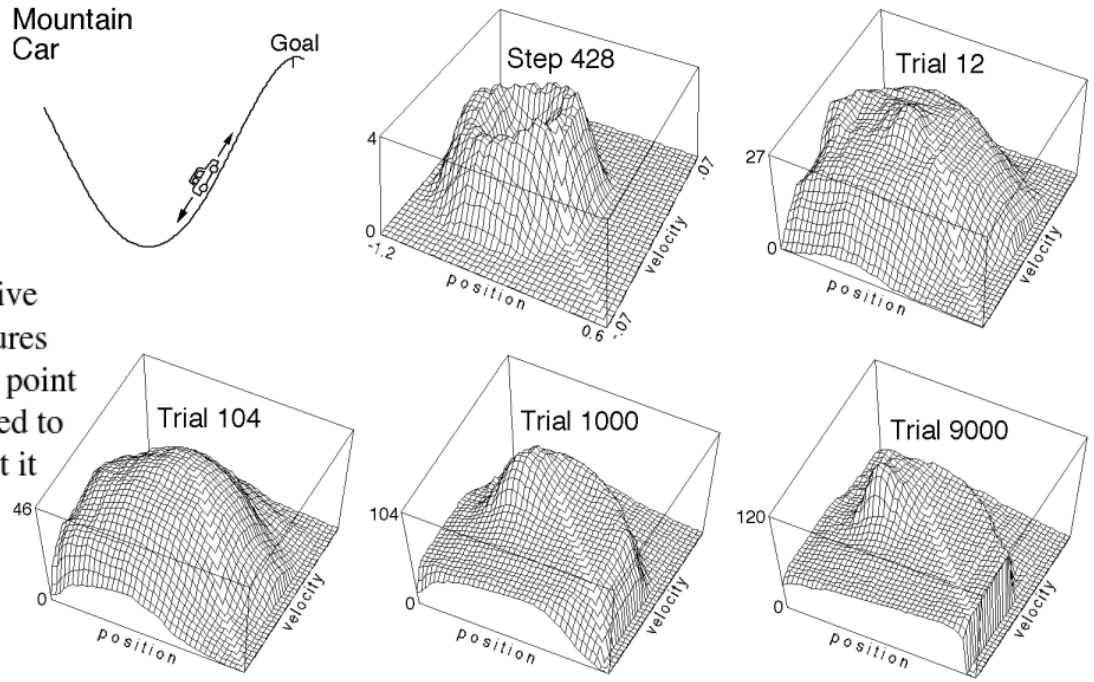
Figure 6: Divergence of smooth value iteration with backpropagation for car-on-the-hill. The neural net, a 2-layer MLP with 80 hidden units, was trained for 2000 epochs per iteration.

J.A. Boyan, and A.W. Moore. (1995). Generalization in Reinforcement Learning: Safely Approximating the Value Function. NIPS 7, pp.369-376.

Success of Linear Function Approximation

- Tile coding successfully approximated the optimal value function because basis functions are localized



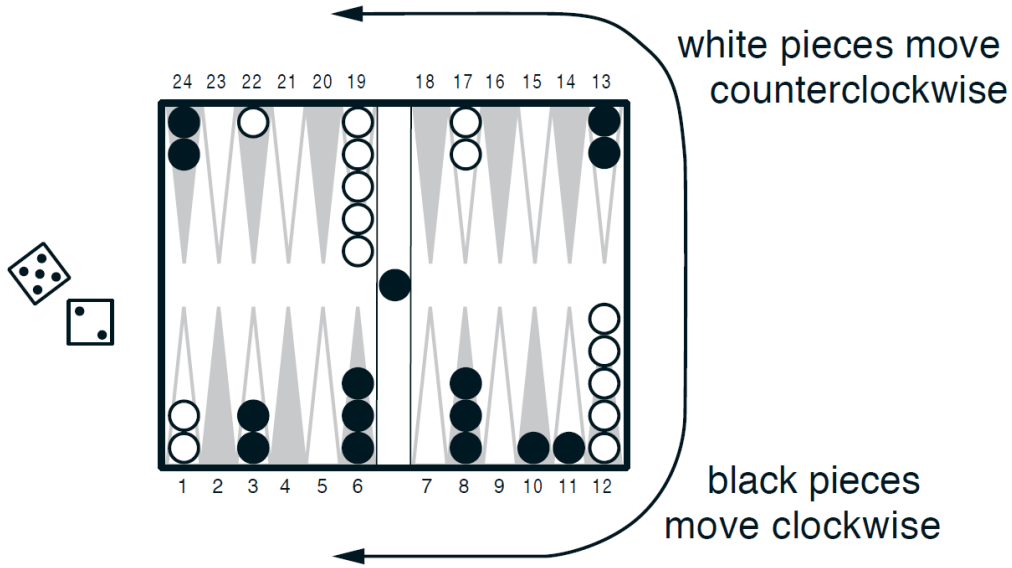


- Difficult to apply tile coding to a high-dimensional state space

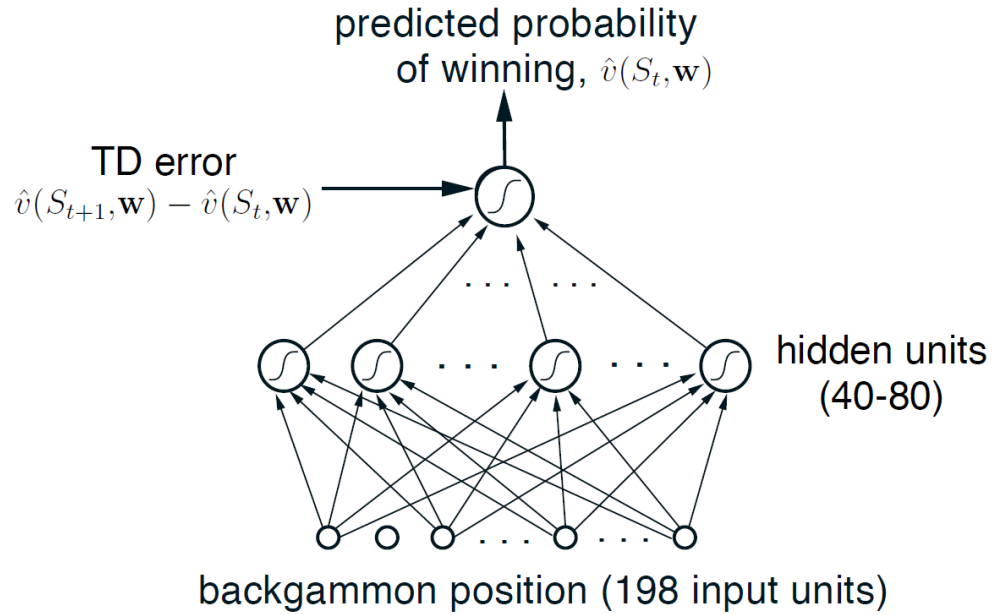
R.S. Sutton. (1996). Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. NIPS 8, pp. 1038-1044.

1991-95: TD-gammon

- RL backgammon program
- The big early success of RL + NN
- 40-80 sigmoid hidden units
- Up to 1,500,000 games of self-play
- Delayed rewards at end of the game
- At or near best human player level



A backgammon position



G. Tesauro. (1992). Practical issues in temporal difference learning. Machine Learning. Vol. 8, no. 3, pages 257-277.
R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

TD-Gammon results

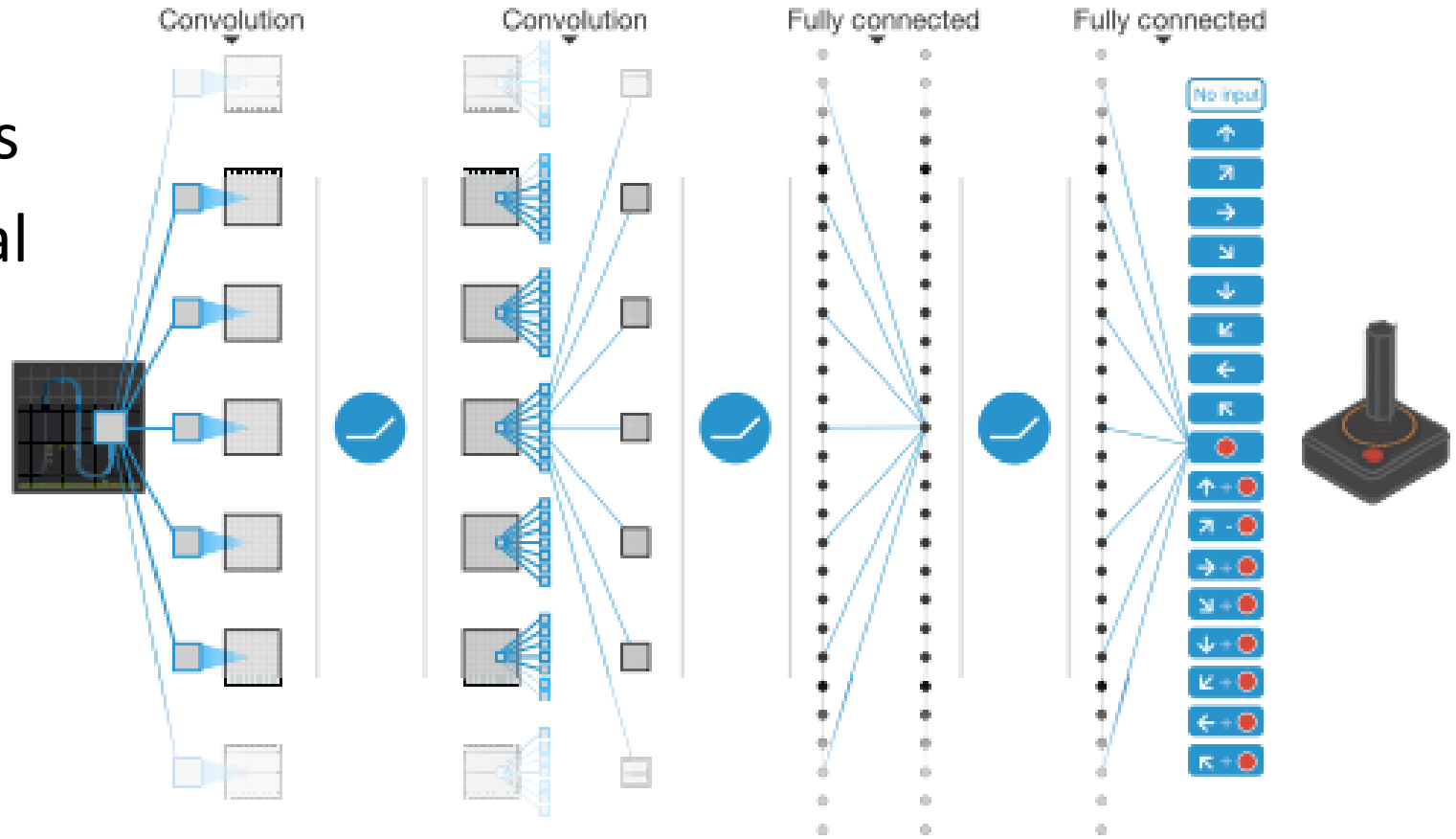
Program	Hidden units	Training games	Opponents	Results
TD-Gammon 0.0	40	300,000	other programs	tied for best
TD-Gammon 1.0	80	300,000	Robertie, Magriel, ...	-13 pts / 51 games
TD-Gammon 2.0	40	800,000	various grandmasters	-7 pts / 38 games
TD-Gammon 2.1	80	1,500,000	Robertie	-1 pt / 40 games
TD-Gammon 3.0	80	1,500,000	Kazaros	+6 pts / 20 games

G. Tesauro. (1992). Practical issues in temporal difference learning. Machine Learning. Vol. 8, no. 3, pages 257-277.
R. Sutton and A.G. Barto. (2018). Reinforcement Learning: An Introduction. (2nd edition). MIT Press.

RECENT STUDIES ON REINFORCEMENT LEARNING

Deep Q Networks (DQN), playing ATARI 2600 games

- Learned a mapping from a sequence of game screens to the value of action
- Performed at a level that is comparable to professional human game testers



V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

What is the Atari 2600 games?

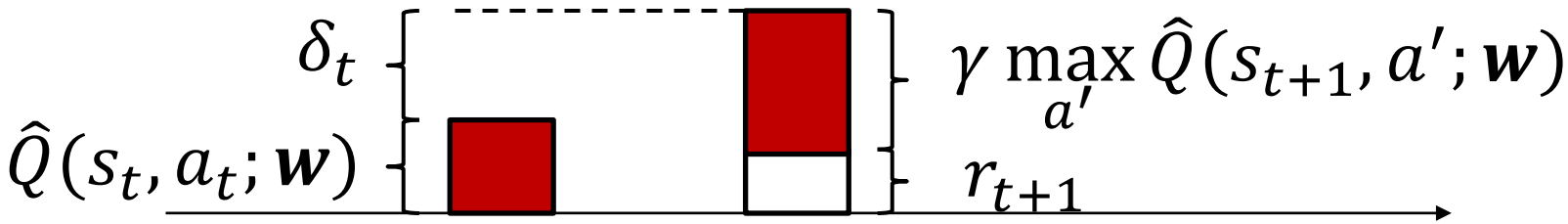
- Home video console released in 1977
- 210 x 160 color video at 60 Hz
- Pac-Man (‘82) sold 7.7 million copies

from Wikipedia

Why are nonlinear approximators unstable

- The target value changes when the parameters are updated!

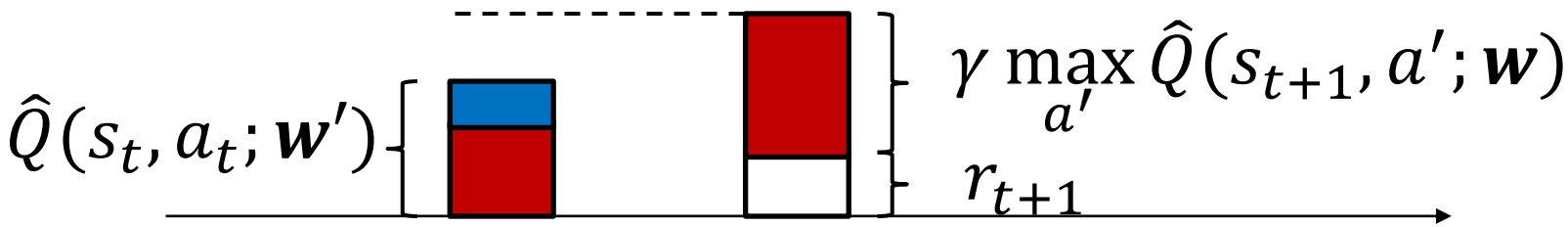
$$\delta_t = r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{w}) - \hat{Q}(s_t, a_t; \boldsymbol{w})$$



Why are nonlinear approximators unstable

- The target value changes when the parameters are updated!

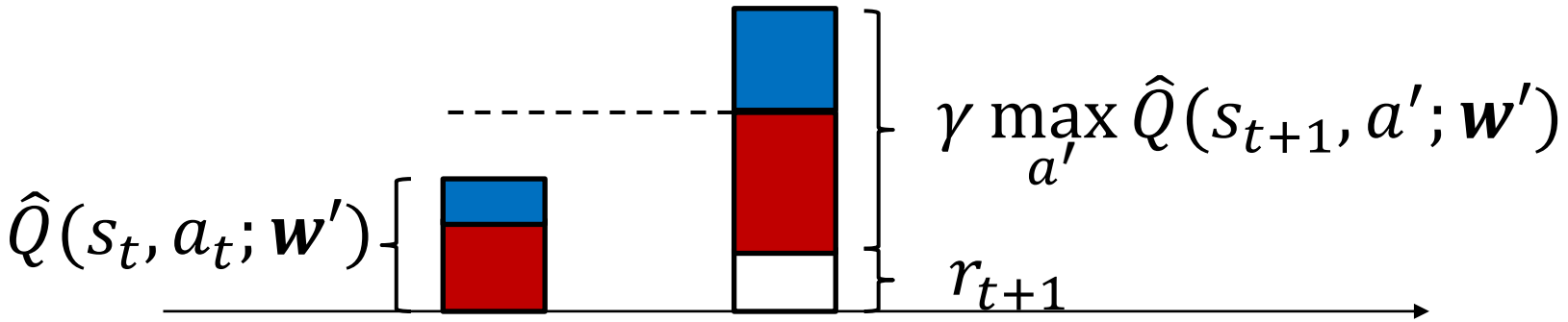
$$\delta_t = r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{w}) - \hat{Q}(s_t, a_t; \boldsymbol{w})$$



Why are nonlinear approximators unstable

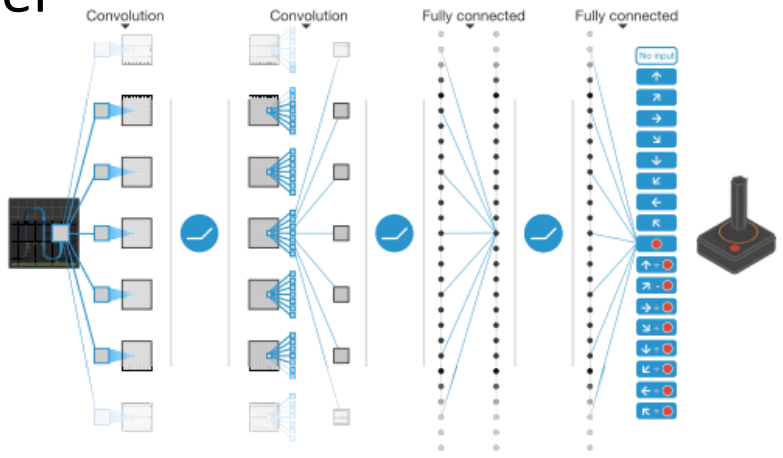
- The target value changes when the parameters are updated!

$$\delta_t = r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{w}) - \hat{Q}(s_t, a_t; \boldsymbol{w})$$



2013-15: Deep Q network (DQN)

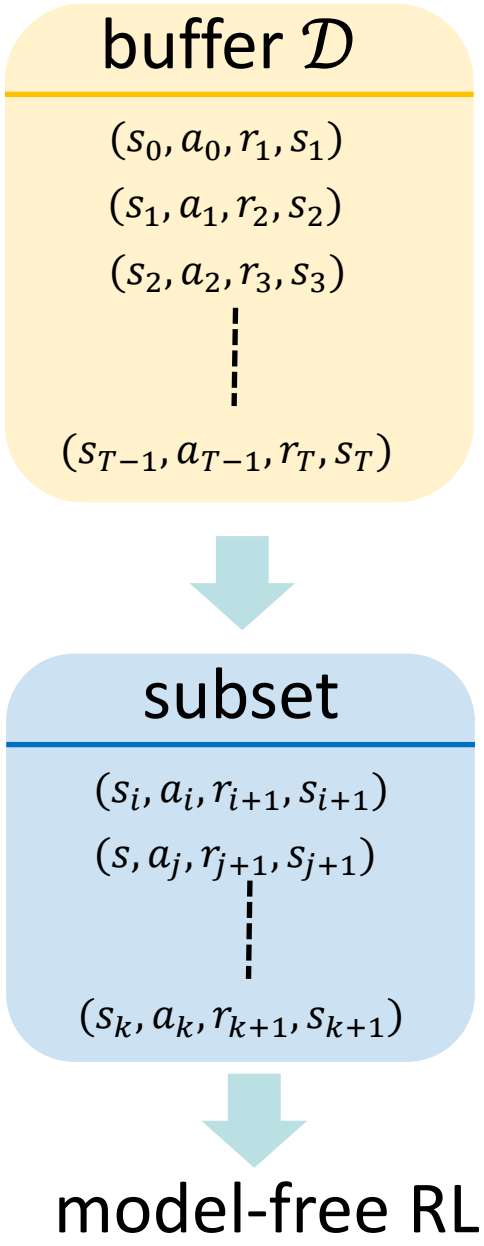
- DQN achieves remarkable success in computer games by learning deeply encoded representation from convolution networks
- What are the key issues in DQN?
  - Neural fitted Q iteration: Convert reinforcement learning to supervised learning
  - Experience replay: Store experiences and reuse them later
  - (Clipping the values of rewards)



V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

Experience Replay

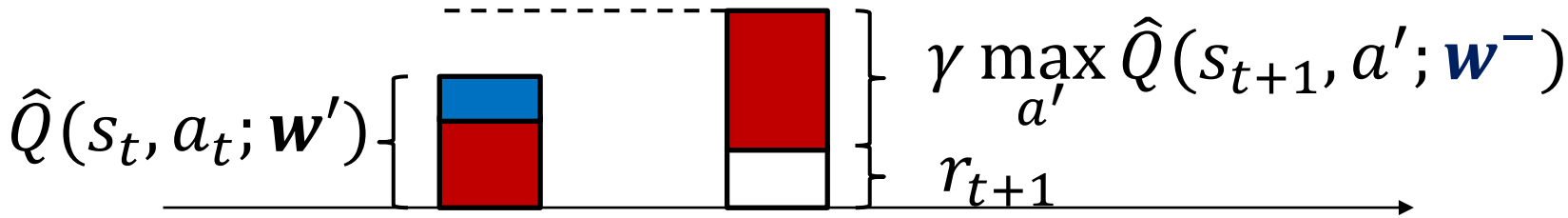
- Experienced transitions are stored in the buffer $\mathcal{D}$
- Same transitions are repeatedly sampled from $\mathcal{D}$ and use off-policy RL algorithms
- Uniform sampling is used
  - If there are N experienced transitions, the probability to select i-th transition is $1/N$
- One of widely used techniques in modern RL studies



L.-J. Lin. (1991). Programming Robots Using Reinforcement Learning and Teaching. In Proc. of AAAI, pages 781-786.
V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

Neural fitted Q-iteration

- Two action-value functions are maintained
  - $\hat{Q}(s, a; \boldsymbol{w}^{-})$: Target Q function to compute the target value
  - $\hat{Q}(s, a; \boldsymbol{w})$: Learning Q function to be trained
- Fix the target Q-function for a while when computing TD error



$$\delta_t = r_{t+1} + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \boldsymbol{w}^{-}) - \hat{Q}(s_t, a_t; \boldsymbol{w})$$

M. Riedmiller. (2005). Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In Proc. of ECML, pages 317-328.

V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

Neural fitted Q-iteration

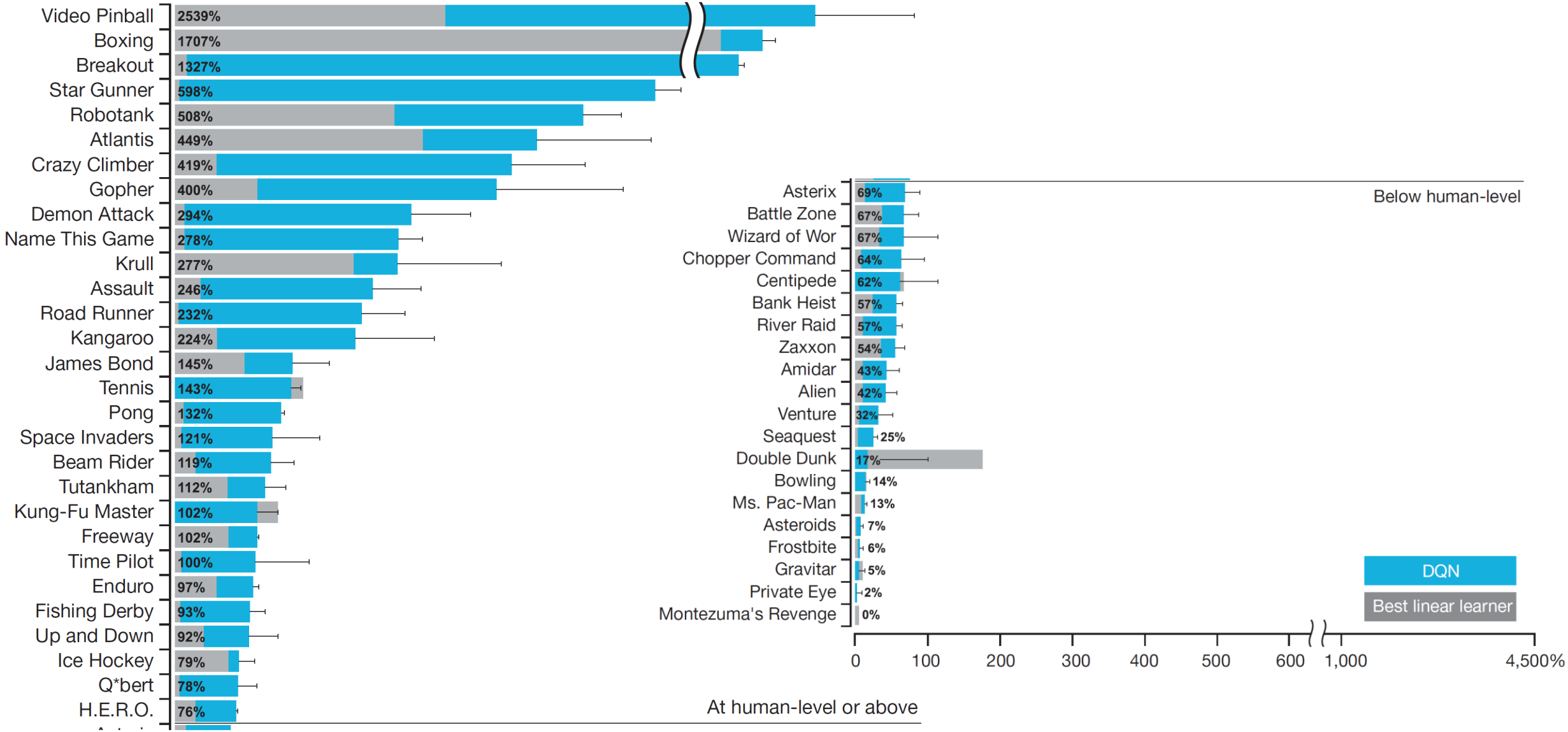
- Frequency to update the target network is critical

  ⇒ Fast, but unstable learning for frequent update

  ⇒ Stable, but slow learning when we rarely update $\boldsymbol{w}^{-} \leftarrow \boldsymbol{w}$

M. Riedmiller. (2005). Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In Proc. of ECML, pages 317-328.
V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

Performance of DQN



V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

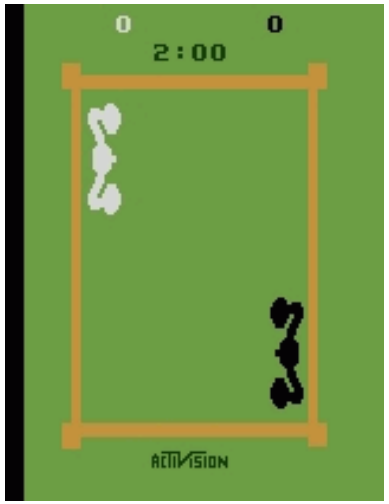
Examples of the Atari 2600 games

Better than human

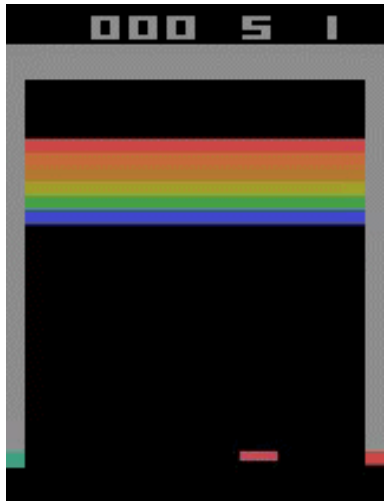
- Reactive games

Worse than human

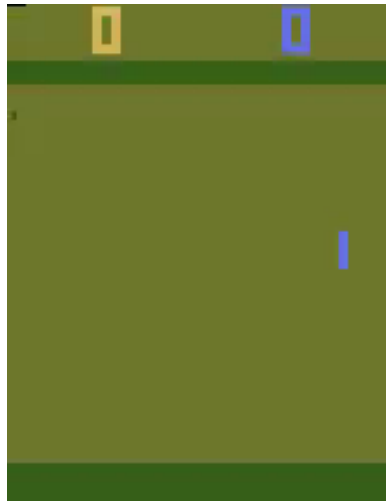
- Games demanding more temporally extended planning strategies



Boxing
1,707 %

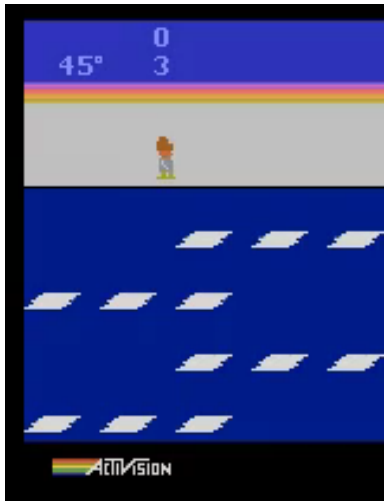


Breakout
1,327 %

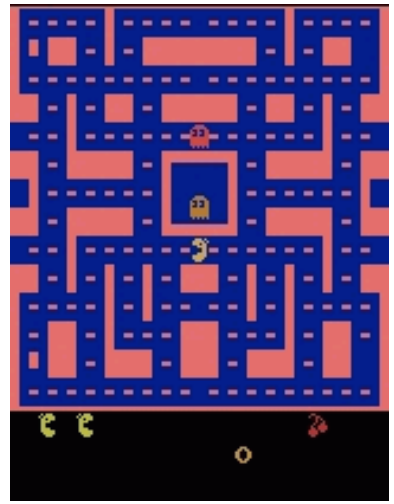


Pong
132 %

Seaquest
25 %



Frostbite
6 %



Ms. Pac-Man
13 %

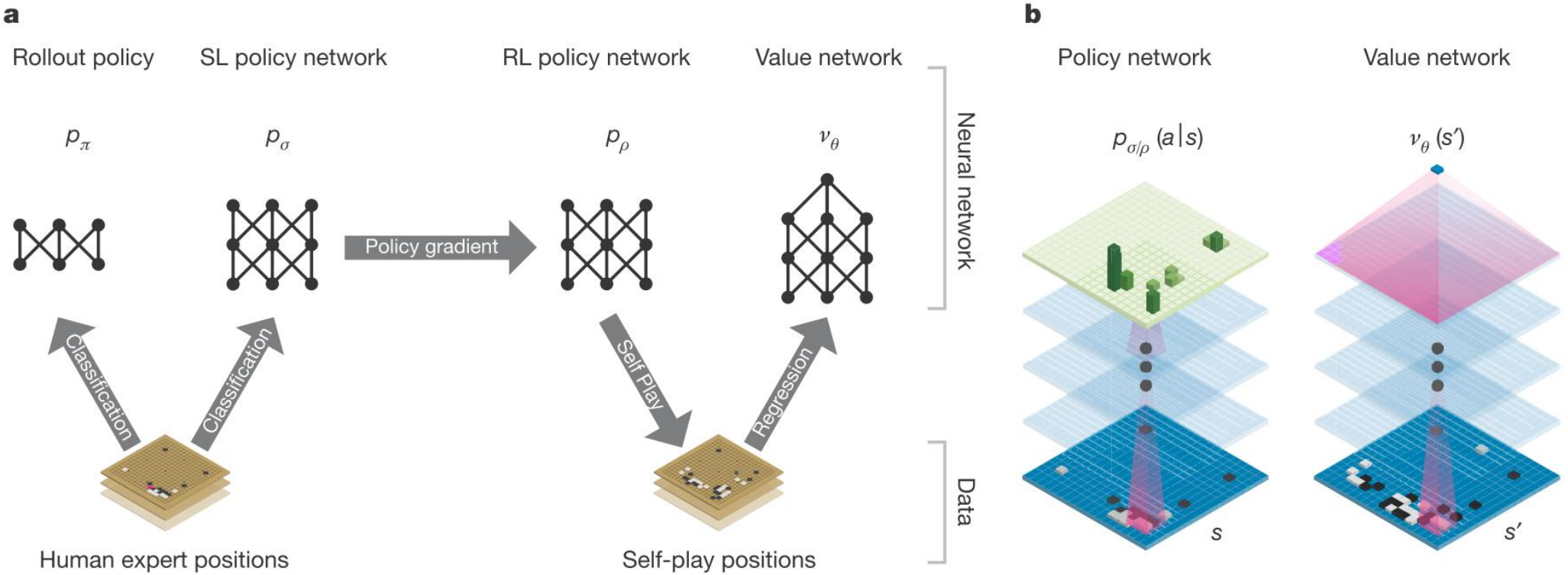
V. Mnih et al. (2015). Human-level control through deep reinforcement learning. Nature, vol. 518, no. 7540, pp. 529–533.

AlphaGo

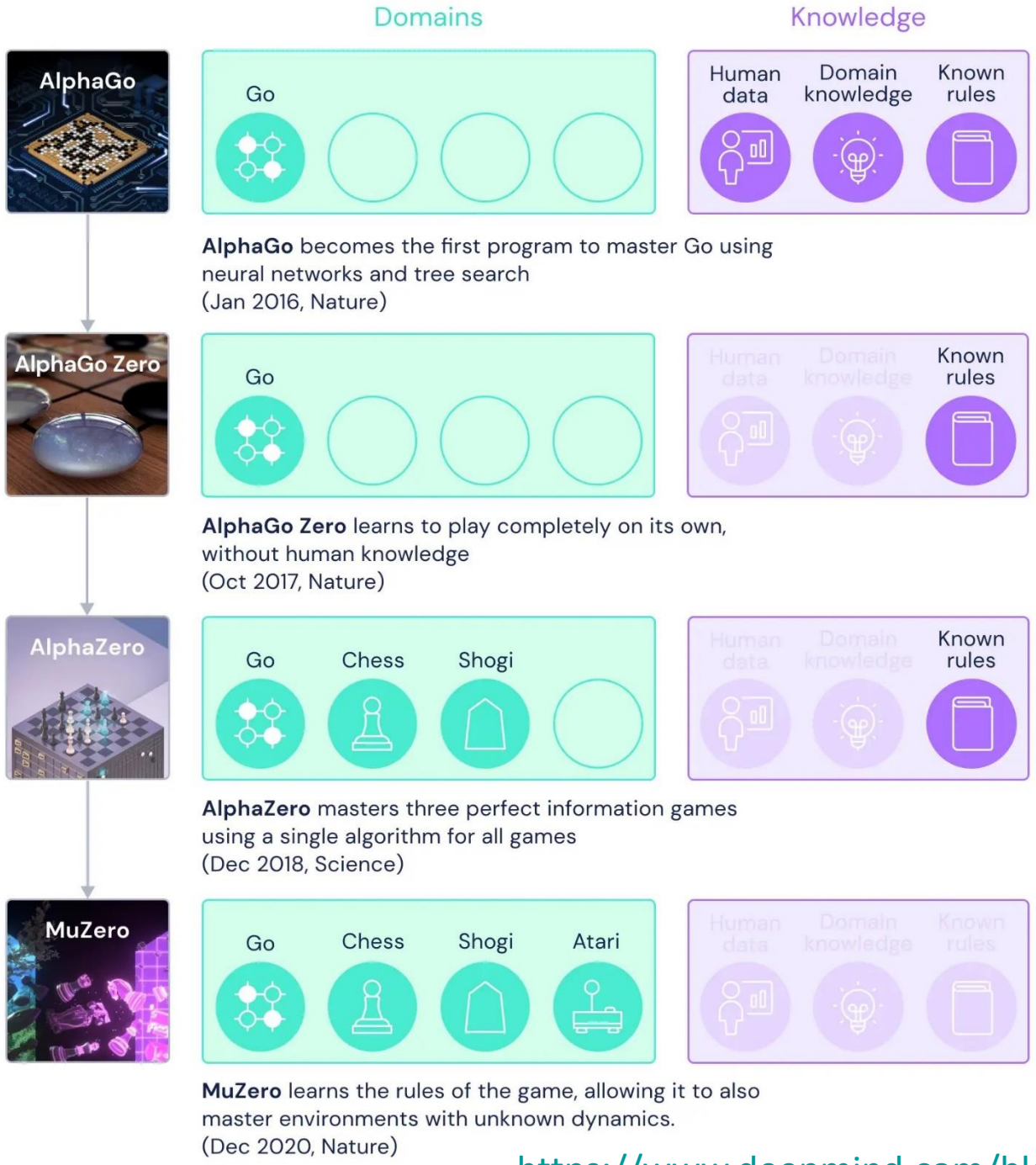
- Outperform the human-champion

Go ratings

Rank	Name	♂♀	Flag	Elo
1	Ke Jie	♂		3615
2	Google AlphaGo			3585
3	Park Jungwhan	♂		3569
4	Iyama Yuta	♂		3532
5	Lee Sedol	♂		3519



D. Silver et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, vol. 529, no. 7587, pp. 484–489.



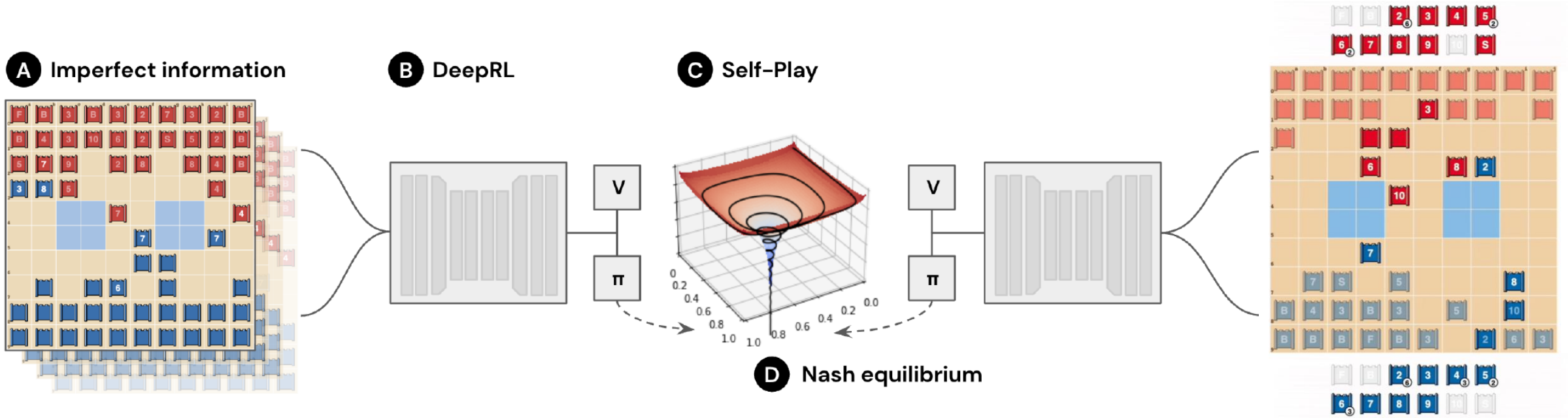
History of Alpha-series

- AlphaGo Zero does not use human-data
- AlphaZero can be applied to Chess and Shogi as well as Go
- MuZero does not know the game rules (model-free RL!)

https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules

DeepNash: Imperfect Strategy Game Stratego (Similar to Japanese Militar Shogi 軍人将棋)

- Model-free Deep RL + Self-play + Nash equilibrium



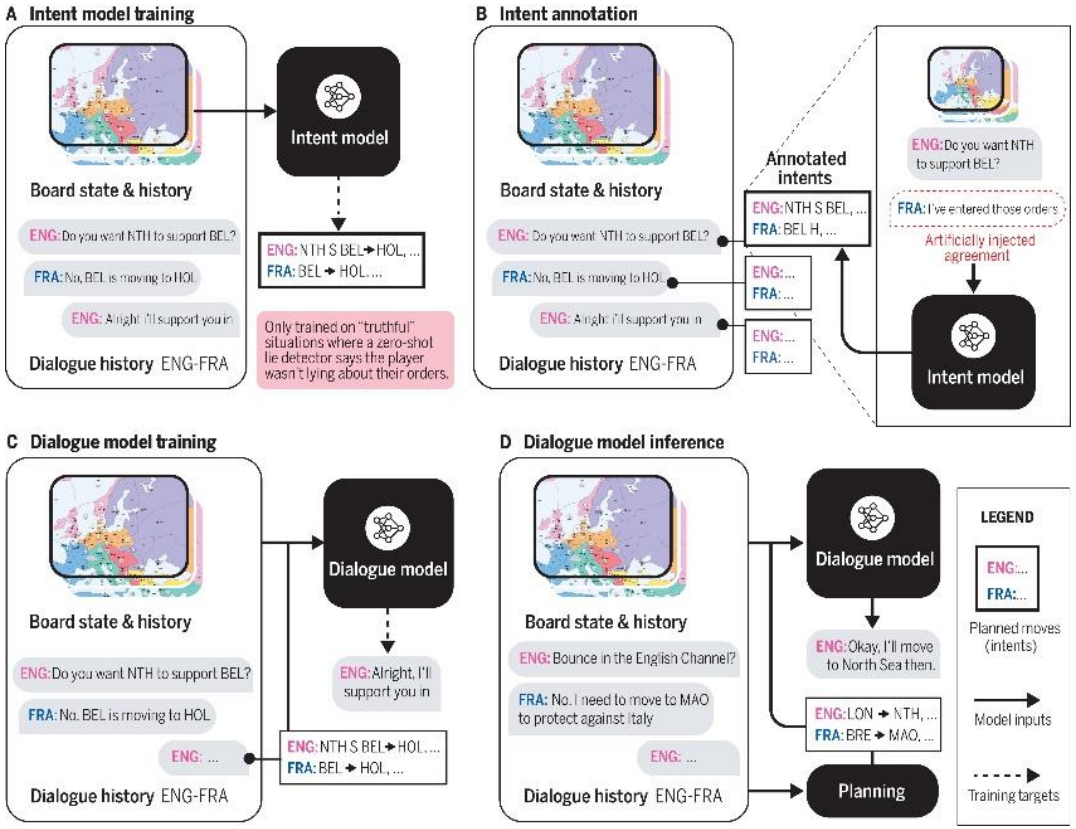
Replicator dynamics: $\frac{d}{d\tau}\pi_\tau^i(a^i) = \pi_\tau^i(a^i)\left[Q_{\pi_\tau}^i(a^i) - \sum_{b^i}\pi_\tau^i(b^i)Q_{\pi_\tau}^i(b^i)\right]$

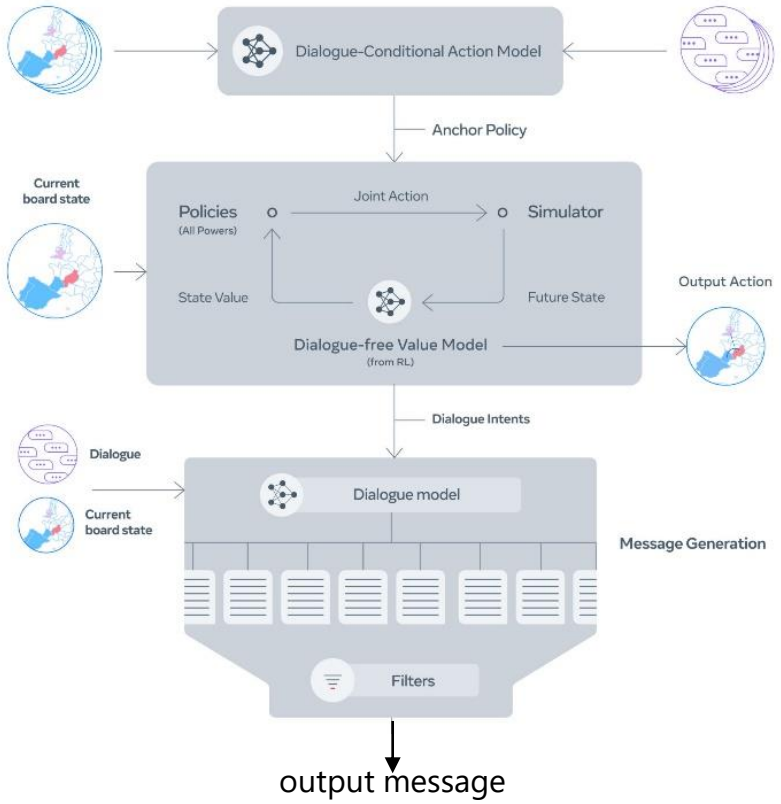
Reward transformation: $r^i(\pi^i, \pi^{-i}, a^i, a^{-i}) = r^i(a^i, a^{-i}) - \eta\log\left(\frac{\pi^i(a^i)}{\pi_{\text{reg}}^i(a^i)}\right) + \eta\log\left(\frac{\pi^{-i}(a^{-i})}{\pi_{\text{reg}}^{-i}(a^{-i})}\right)$

Perolat, J., De Vylder, B., Hennes, B., et al. (2022). Mastering the game of Stratego with model-free multiagent reinforcement learning. Science, 378: 990-996.

Cicero: Imperfect Strategy Game Diplomacy

- Set in Europe in the years leading to the First World War, Diplomacy is played by two to seven players





META Fundamental AI Research Diplomacy Team (FAIR). (2022). Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378, 1067-1074.

InstructGPT (Ancestor of ChatGPT)

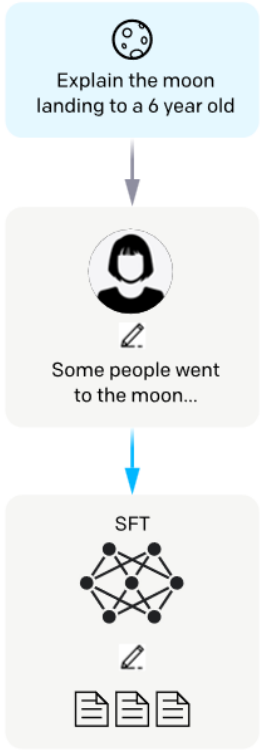
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



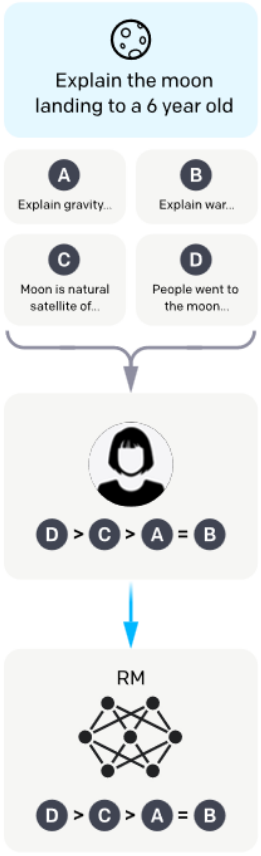
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

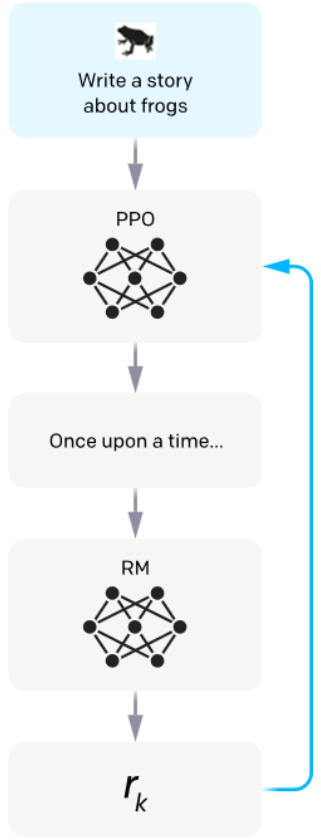
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

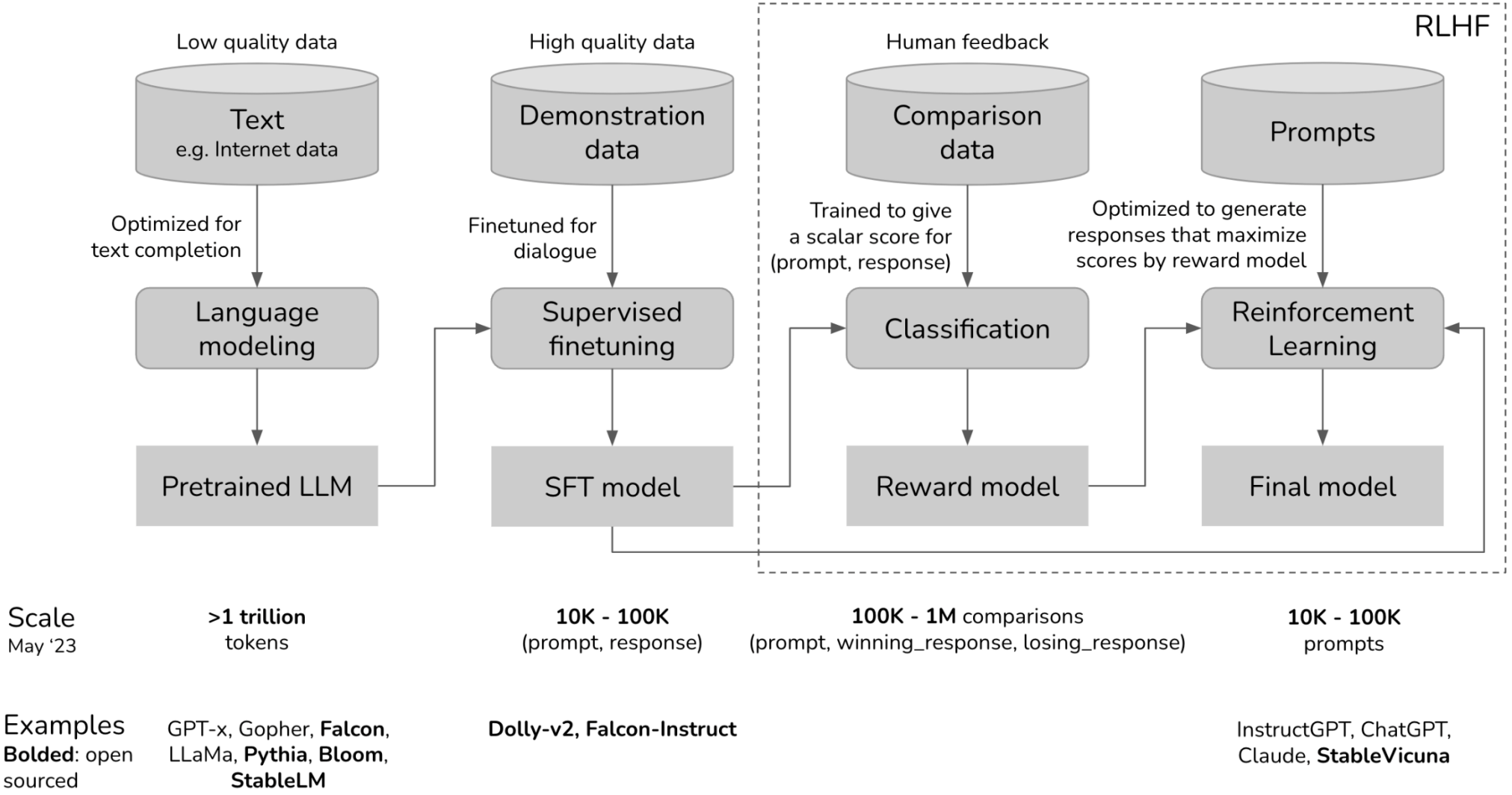
The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



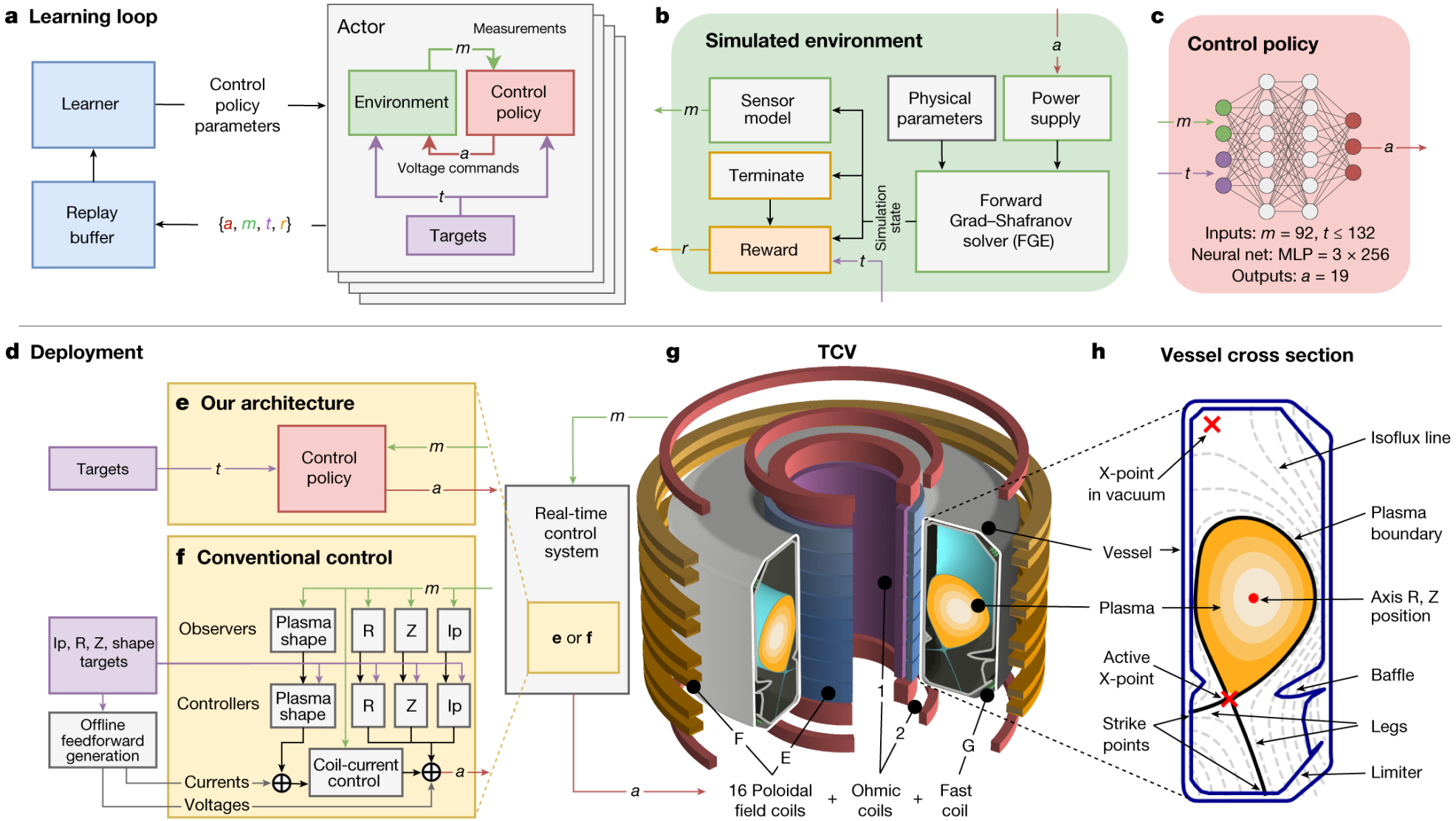
L. Ouyang et al. (2022). Training language models to follow instructions with human feedback. NeurIPS 35.

RLHF: Reinforcement Learning from Human Feedback



C. Huyen. (2023). RLHF: Reinforcement Learning from Human Feedback.

Magnetic control of tokamak plasmas



J. Degrave et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. Nature, vol. 602, no. 7897, pp. 414-419.

Tax Collections Optimizers

- Optimally managing the tax collection processes at financial institutions

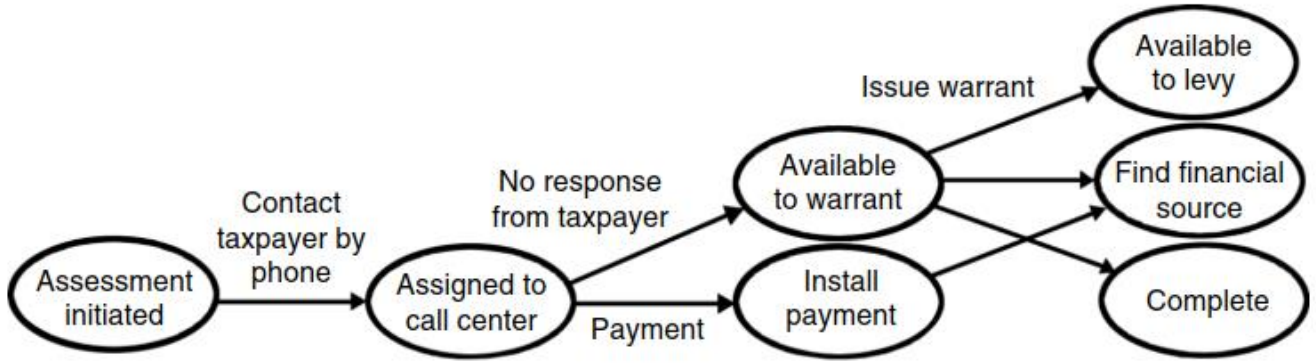
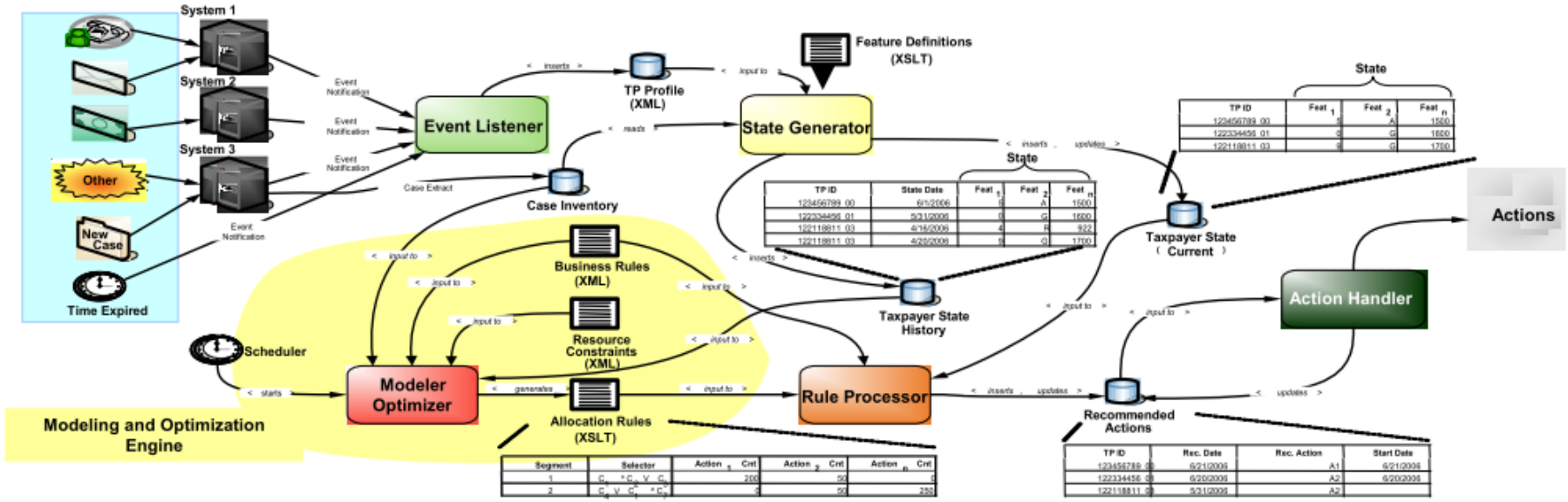


Figure 1: The graphic illustrates an MDP formulation of the collections process.

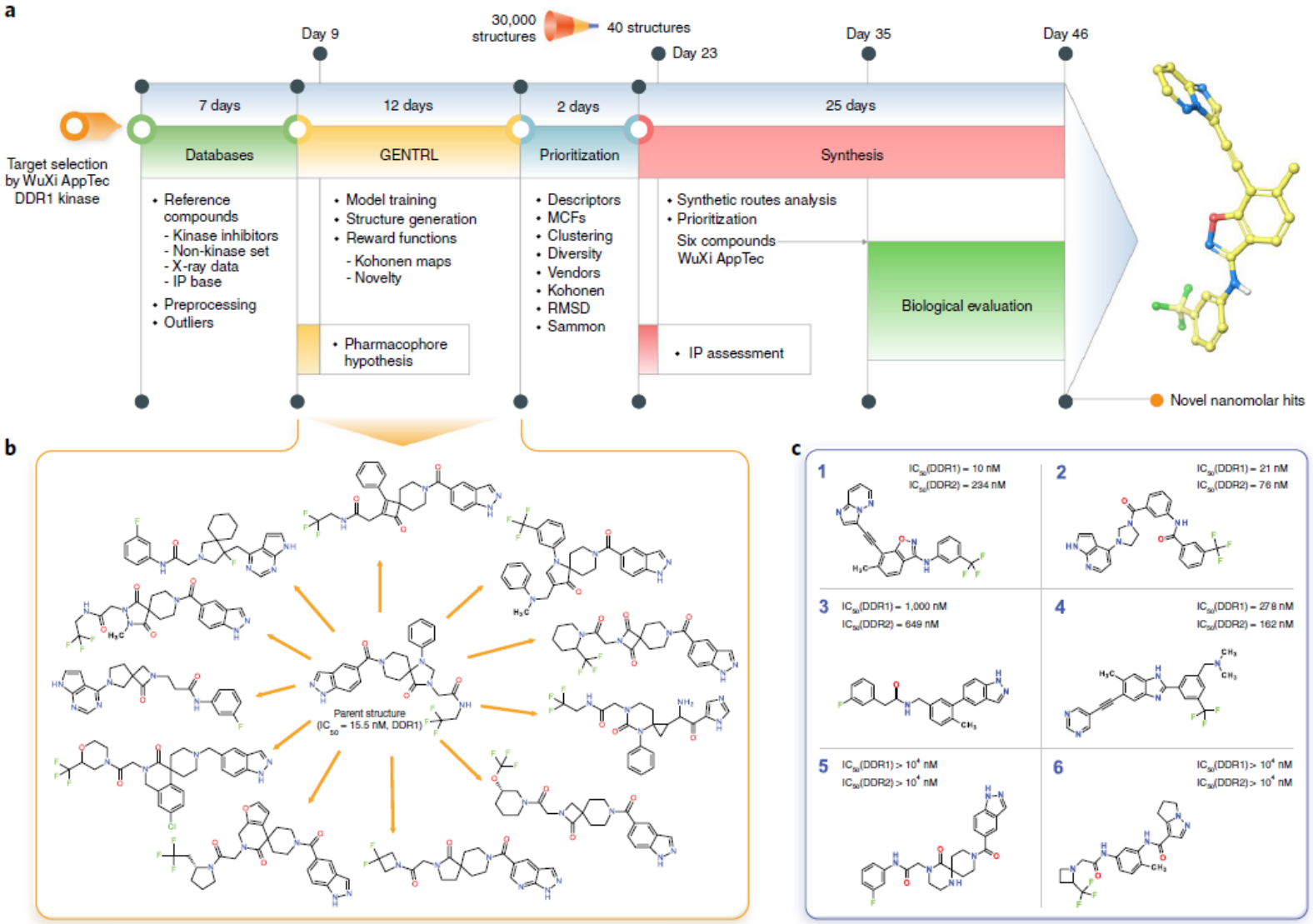
- Use actually in New York State Department of Finance



N. Abe, et al. (2010). Optimizing debt collections using constrained reinforcement learning. In Proc. of the 16th ACM SIGKDD, pp. 75-84.

G. Miller, et al. (2012). Tax collection optimization for New York State. Interfaces, 42(1), 74-84, 2012.

Rapid identification of potent DDR1 kinase inhibitors



A. Zhavoronkov, et al. (2019). Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nature Biotechnology. Vol 37, no. 9, pages 1038-1040.

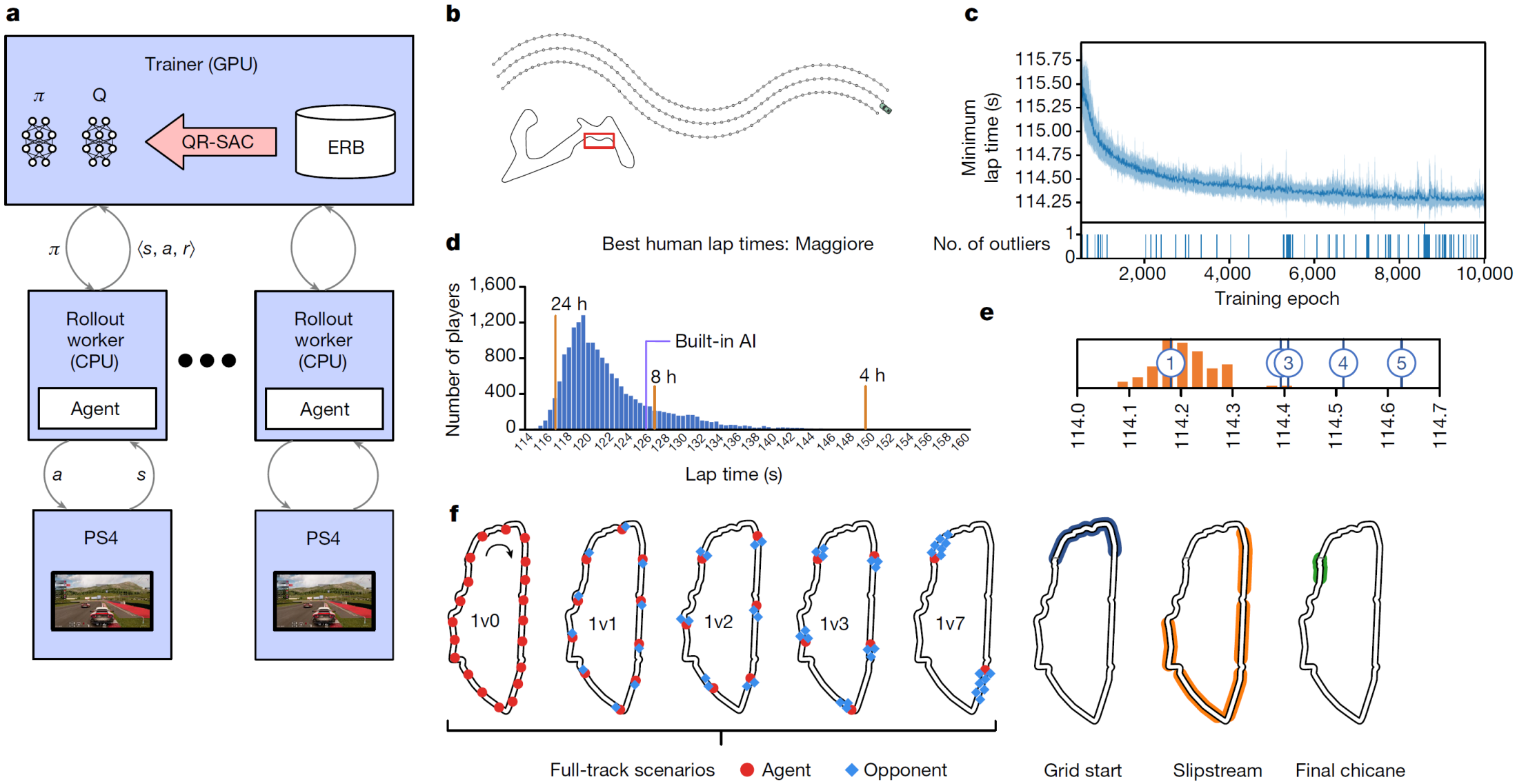
Gran Turismo Sophy

- Outperform the human-champion game players

https://www.gran-turismo.com/jp/gran-turismo-sophy

P.R. Wurman et al. (2022). Outracing champion Gran Turismo drivers with deep reinforcement learning, Nature, vol. 602, no. 7896, pp. 223–228.

Gran Turismo Sophy



P.R. Wurman et al. (2022). Outracing champion Gran Turismo drivers with deep reinforcement learning, Nature, vol. 602, no. 7896, pp. 223–228.

AlphaStar (Playing Star Craft II)

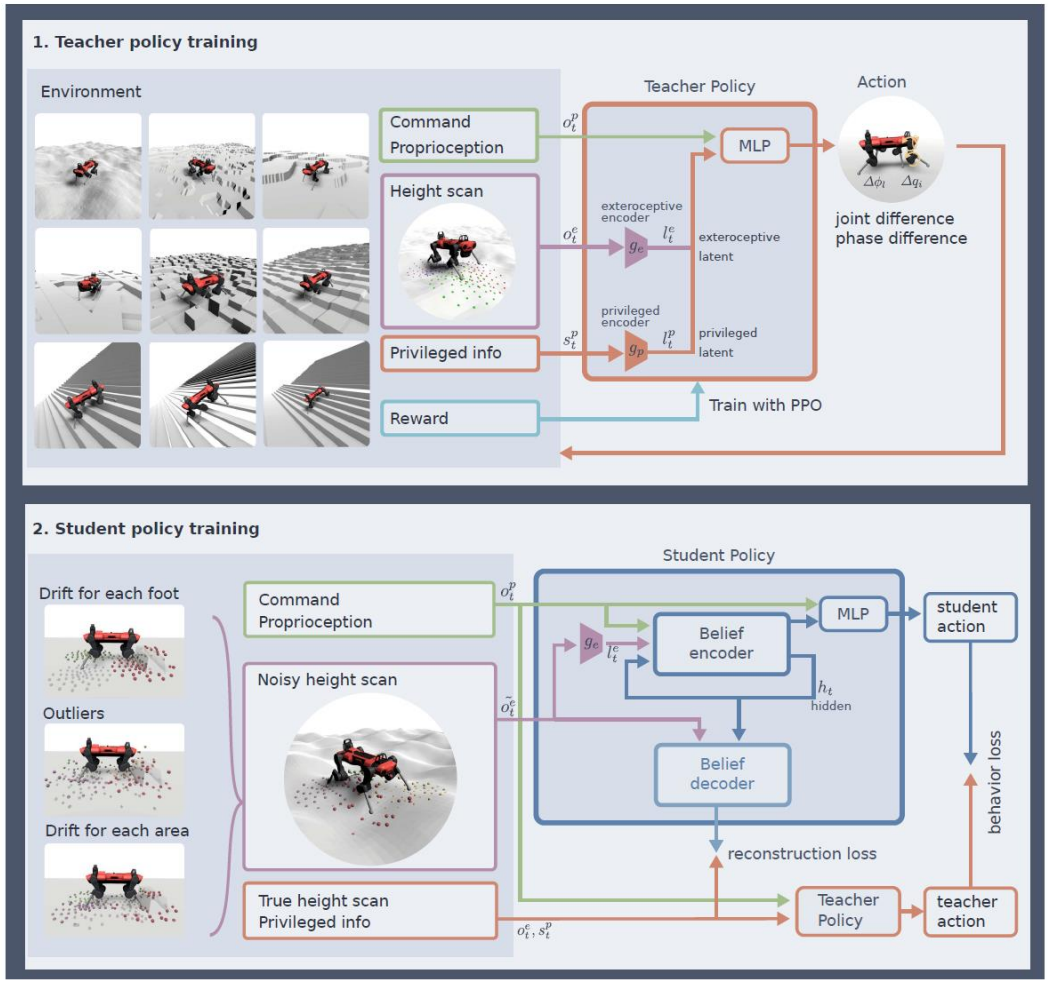
- AlphaStar (Vinyals et al., 2019)
- multiagent real-time strategy game RL + supervised learning
- 200 years

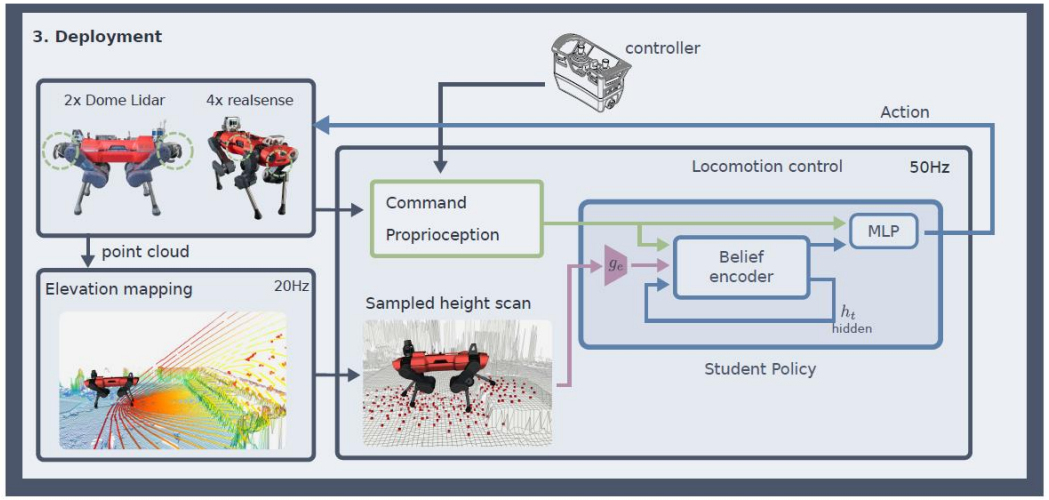
O. Vinyals, et al. (2019). Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. Nature 575, no. 7782: 350–54.

Robust locomotion in the wild

T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. (2022). Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.*, vol. 7, no. 62, p. eabk2822.

Robust locomotion in the wild





T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. (2022). Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.*, vol. 7, no. 62, p. eabk2822.

Computational resources

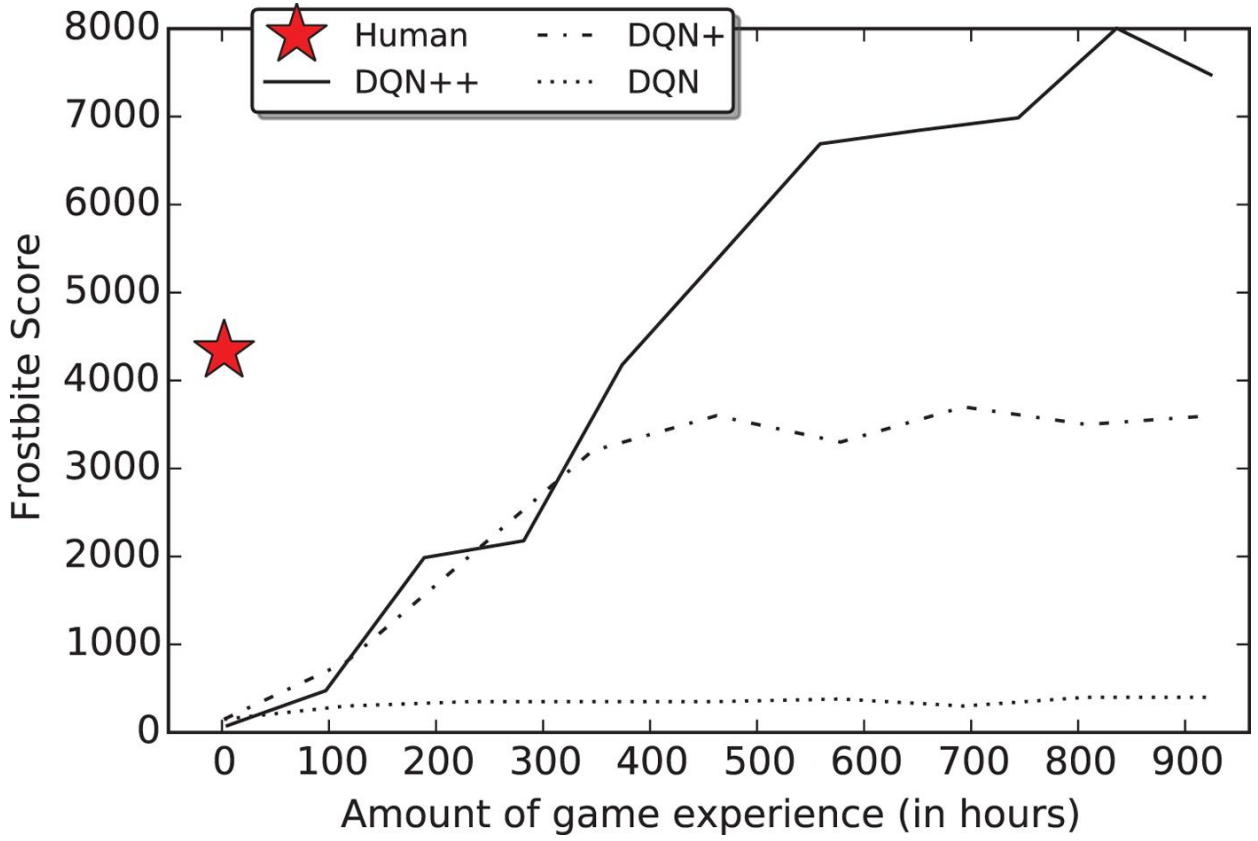
- AlphaZero independently learns to play chess, shogi, and go
  - 5,000 first-generation TPUs to generate the games
  - 64 second-generation TPUs to train the neural networks

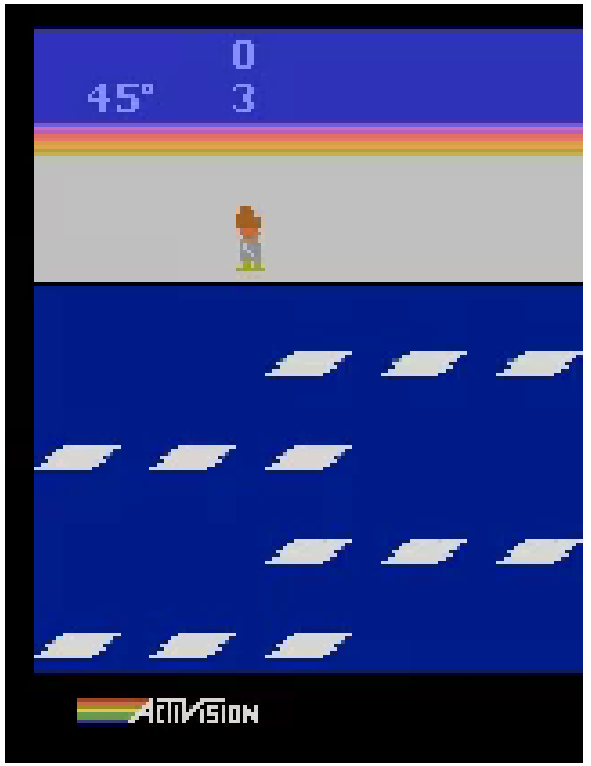
Reported in paper (1)	3-Day	40-Day
Hours of training	72	960
Matches played	4,900,000	29,000,000
Seconds per move	0.4	0.8
Training: # GPUs	64	64
Training: # CPUs	19	19
Self-play: # TPUs/player	4	4
GCP Data (2)		
TPU $/h	$6.50	$6.50
GPU $/h (best)	$0.31	$0.31
CPU $/h (best)	$0.01	$0.01
Estimates		
# moves in a Go match (3)	211	211
How many TPUs were required?		
Seconds/match/player	84.40	168.80
Matches/hour/player	42.65	21.33
Matches/total time/player	3071.09	20473.93
# Players required	1595.52	1416.44
# TPUs required	6382.10	5665.74
Costs		
Training cost (GPU)	$1,428.48	$19,046.40
Training cost (CPU)	$9.55	$127.32
Self-play: TPU cost	$2,986,822.22	$35,354,222.22
Final Estimate	$2,988,260.25	$35,373,395.94

https://www.yuzeh.com/data/agz-cost.html

Comparing learning speed for people versus DQNs

- 463 hours to reach 80% of human level

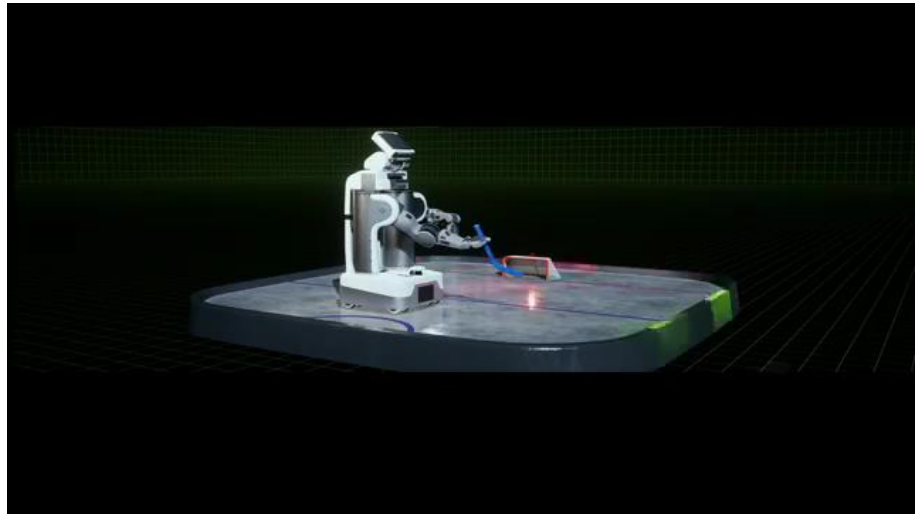




B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. (2017). Building machines that learn and think like people. Behavioral and Brain Sciences, vol. 4.

Why is deep reinforcement learning difficult?

- In Reinforcement Learning, the learning agent should collect data by itself
  - An initial policy, which is randomly initialized, cannot explore the environment efficiently, and therefore, useful data is not gathered
- Using simulators
  - Pay attention to simulation-to-reality gap



https://www.youtube.com/watch?v=oa__wkSmWUw

Why is deep reinforcement learning difficult?

- In Reinforcement Learning, the learning agent should collect data by itself
  - An initial policy, which is randomly initialized, cannot explore the environment efficiently, and therefore, useful data is not gathered
- Using multiple real robots in parallel
  - Expensive
  - The amount of data is still limited

https://www.youtube.com/watch?v=cXaic_k80uM

Summary

- Introduction to deep reinforcement learning
- Recent studies achieve remarkable success in games, but it is still challenging to apply to real problems
- Promising approaches
  - Imitation learning
  - Offline reinforcement learning

Report

- Please select one topic and write your report

1. Consider the application of reinforcement learning

2. How can we reduce the training data and time?