

AI in Medicine I

Reinforcement learning

Julia Schnabel

I32 – Chair for Computational Imaging and AI in Medicine
Faculty of Informatics

Outline

Prelude

- Definitions and Gaming

Part I

- Introduction to Reinforcement Learning (RL)

Part II

- Deep Reinforcement Learning using Deep Q-networks (DQN)

Part III

- RL Applications in Healthcare

“Simple” definition

*“Reinforcement learning is learning what to do—how to map situations to **actions**—so as to maximize a numerical **reward** signal.*

*The learner is not told which actions to take, but instead must discover **which actions yield the most reward** by trying them.*

*In the most interesting and challenging cases, actions may affect **not only the immediate reward** but also the next situation and, through that, all **subsequent rewards**.*

*These two characteristics—**trial-and-error search** and **delayed reward**—are the two most important distinguishing features of reinforcement learning.”*

Reinforcement
Learning
An introduction
second edition
Richard S. Sutton and Andrew G. Barto



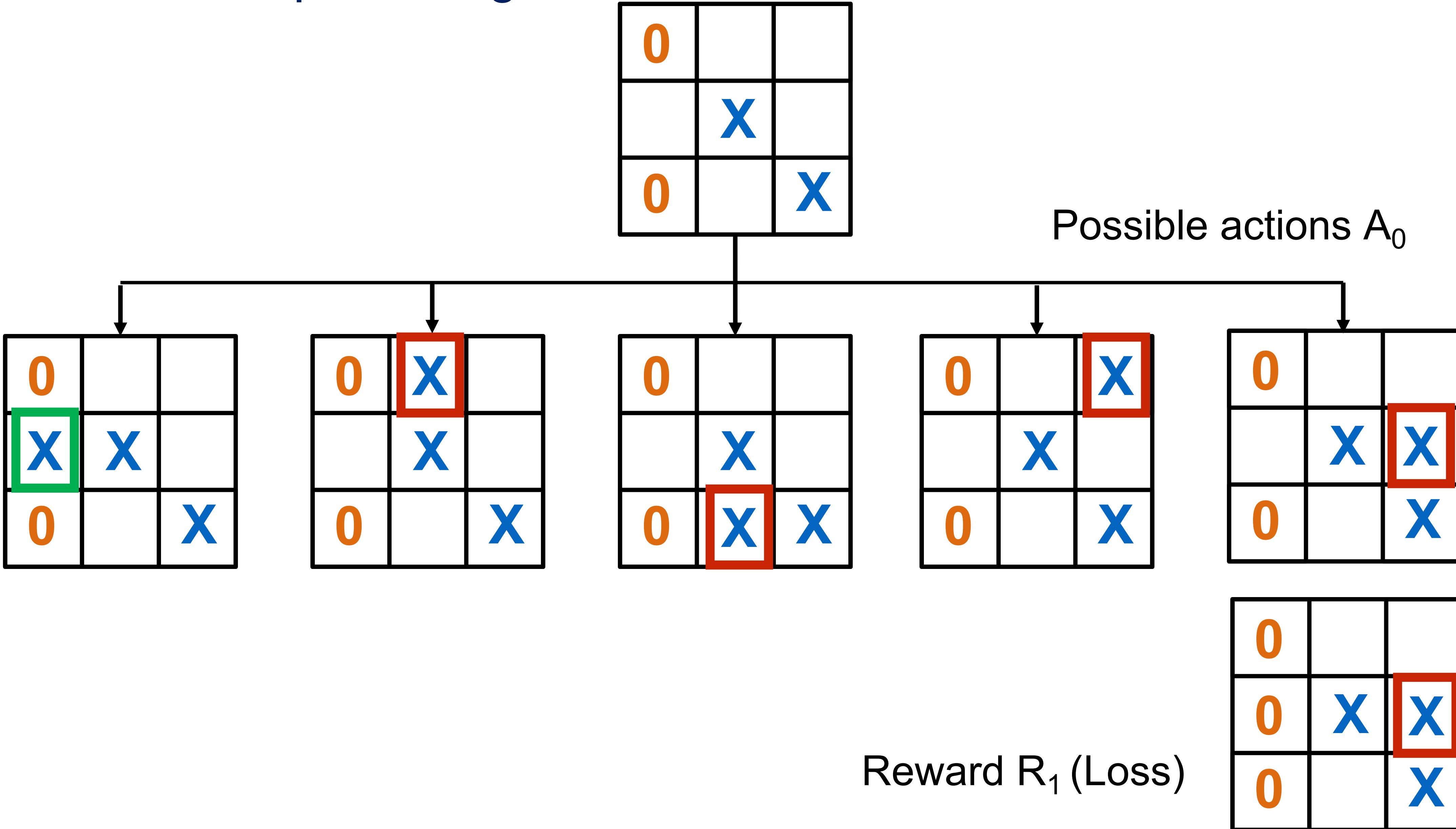
Reinforcement Learning: An Introduction. By R. S. Sutton and A. G. Barto

MIT Press, 1998 (2nd edition 2018)
<http://www.incompleteideas.net/book/RLbook2018trimmed.pdf>

RL featured in WarGames



Example: Noughts and Crosses – aka TicTacToe



Some more popular RL successes

- ▶ AlphaStar
- ▶ AlphaGo
- ▶ DQN Atari
- ▶ Open AI Five



AlphaStar © DeepMind, AlphaGo © source unknown, Atari © Nature/Google DeepMind/Atari Interactive, Dota 2 © Valve, and robots © Peter Pastor. All rights reserved. This content is excluded from our Creative Commons license.
For more information, see <https://ocw.mit.edu/help/faq-fair-use/>

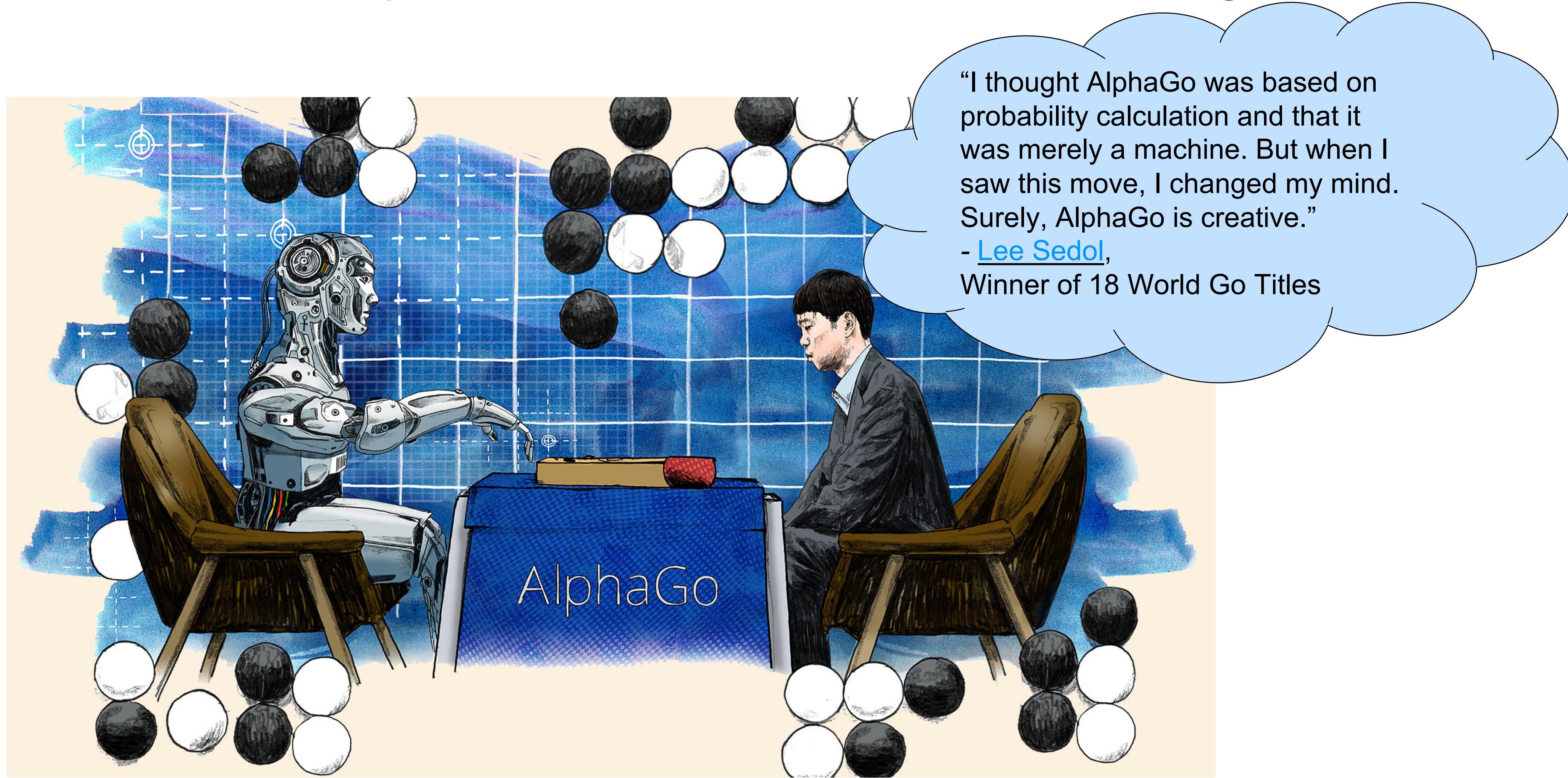
Reinforcement learning on Netflix



<https://www.alphagomovie.com/>

How DeepMind's AlphaGo Became the World's Top Go Player

The revolutionary power of reinforcement learning*



"I thought AlphaGo was based on probability calculation and that it was merely a machine. But when I saw this move, I changed my mind. Surely, AlphaGo is creative."
- [Lee Sedol](#),
Winner of 18 World Go Titles

* <https://ai.plainenglish.io/how-deepminds-alphago-became-the-world-s-top-go-player-5b275e553d6a>

Too smart to learn from others...

- “*AlphaGo variants became so powerful that the most skilled version yet, AlphaGo Zero, couldn’t train by playing against human Go champions like the first version of AlphaGo did because the skill inferiority of even the best human players would limit AlphaGo Zero’s capabilities.*”
- “*In the end, AlphaGo Zero’s only worthy match was itself... so it learned by playing against itself.*”

* <https://ai.plainenglish.io/how-deepminds-alphago-became-the-world-s-top-go-player-5b275e553d6a>

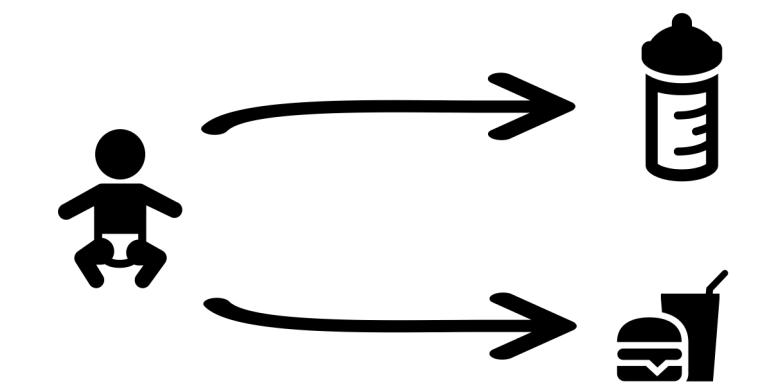
Part I

Introduction to Reinforcement Learning

A brief intro to the main elements of Reinforcement Learning

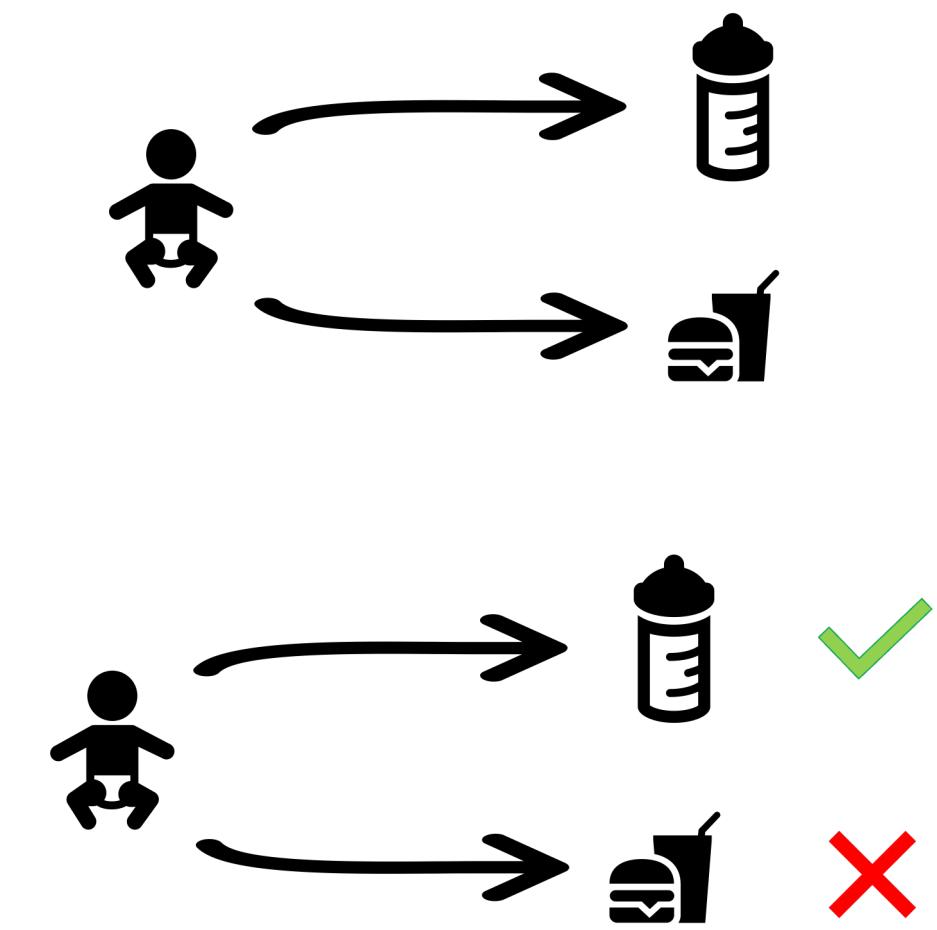
Re-visiting Machine Learning

- **Unsupervised learning**
 - Explores data and draws inferences from datasets to describe hidden structures from unlabeled data



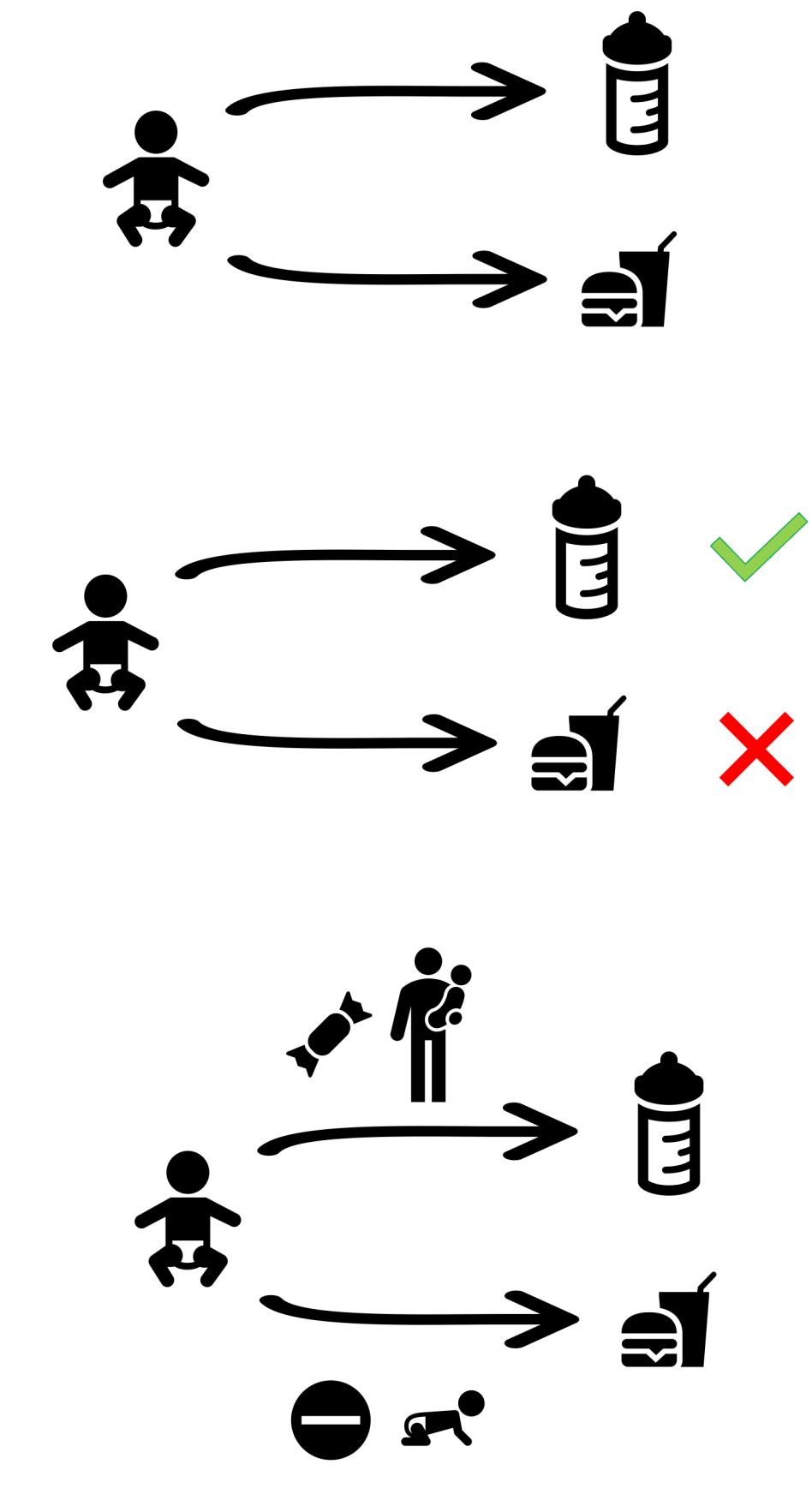
Re-visiting Machine Learning

- **Unsupervised learning**
 - Explores data and draws inferences from datasets to describe hidden structures from unlabeled data
- **Supervised Learning**
 - Learning from a training set of labeled examples provided by a knowledgeable external supervisor



Re-visiting Machine Learning

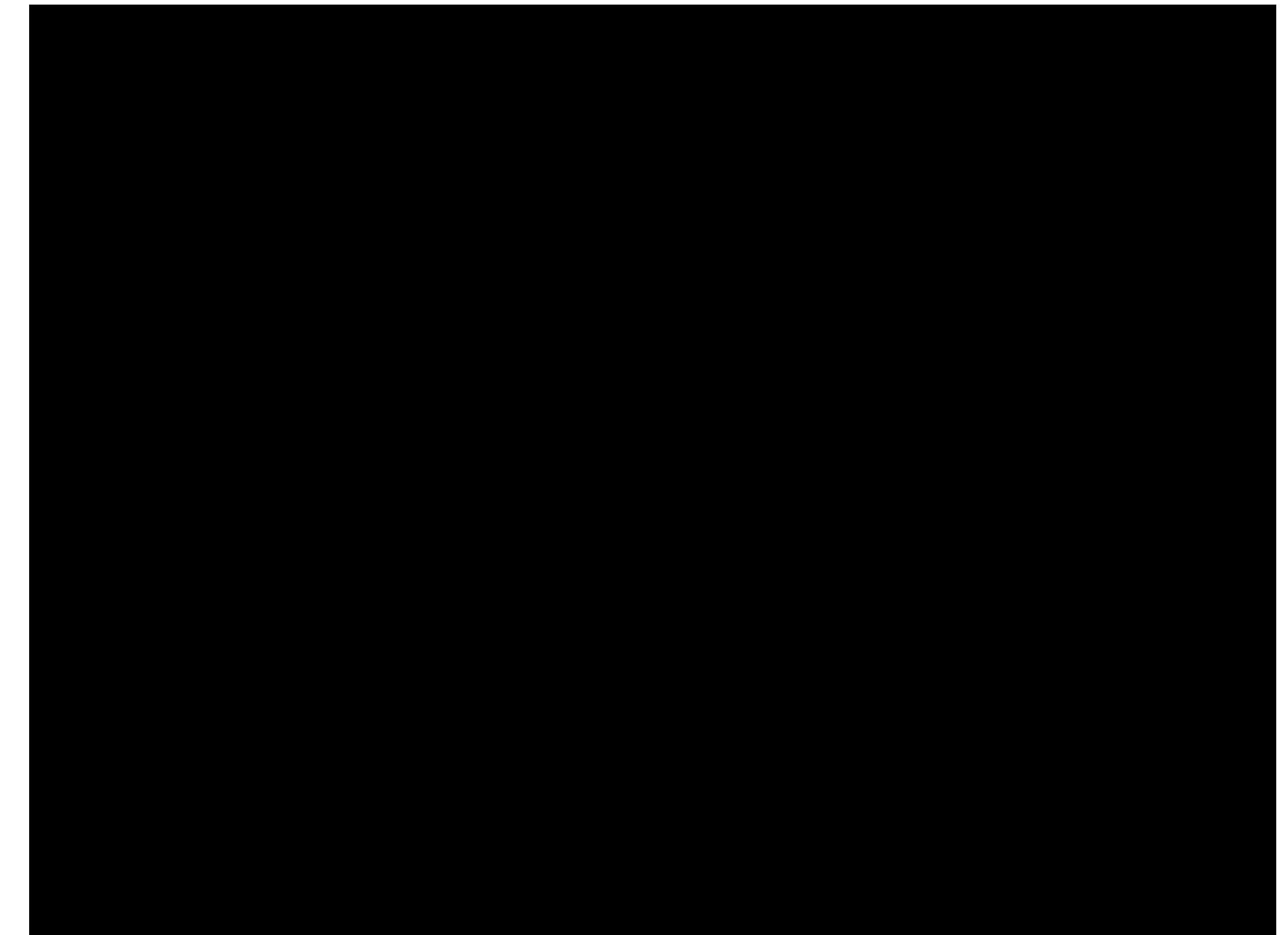
- **Unsupervised learning**
 - Explores data and draws inferences from datasets to describe hidden structures from unlabeled data
- **Supervised Learning**
 - Learning from a training set of labeled examples provided by a knowledgeable external supervisor
- **Reinforcement Learning**
 - Computational approach to learn by interacting with an environment
 - Single decision must be made
 - Multiple actions, where each action has a reward associated with it
 - Goal is to maximize reward: Pick an action with the highest reward
- **Deep Reinforcement learning**
 - Replaces “brain” of RL by deep neural network



Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning inspired by behaviourist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward

...Wikipedia



<https://www.youtube.com/watch?v=JA96Fba-WHk>

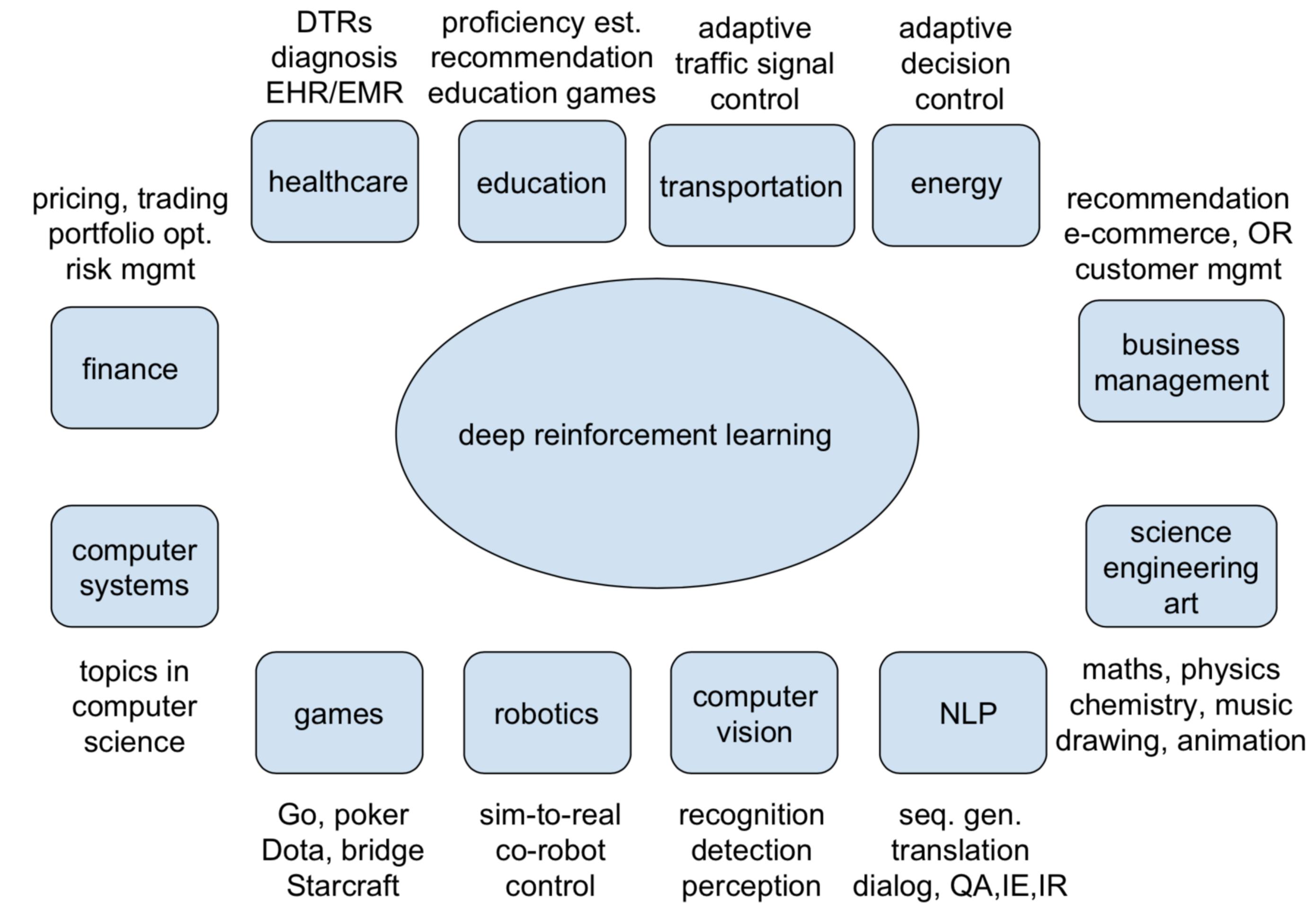
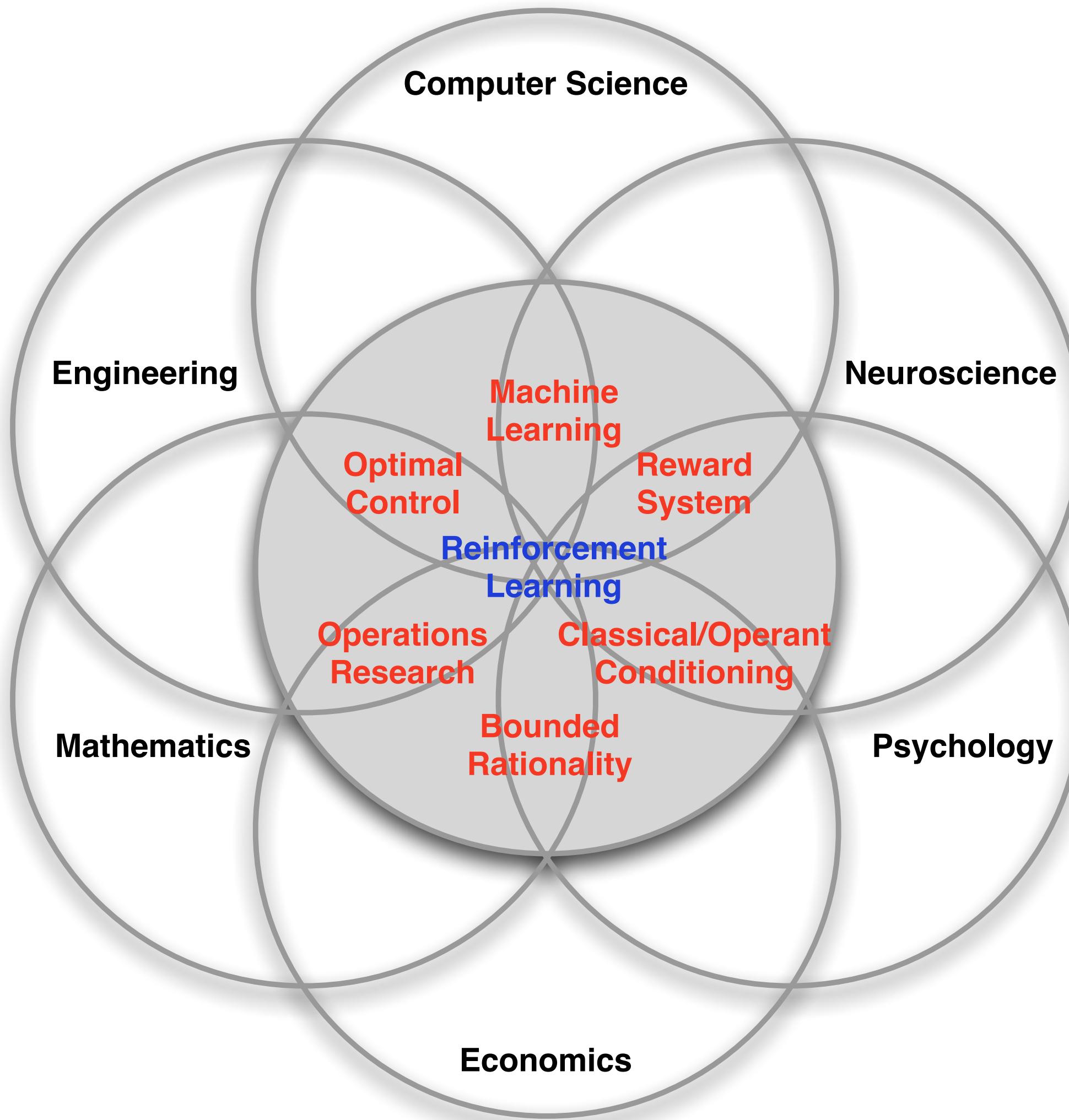
What is reinforcement learning?

- Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.
- Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

RL is not a new concept!

- Studied extensively in:
 - game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, and statistics.
- In control theory and operations research, goes under the name of: *approximate dynamic programming*, or *neuro-dynamic programming*.
 - Mostly concerned with the **existence and characterisation of optimal solutions**, and their **exact computation**
 - Less concerned with **learning or approximation**, particularly in the absence of a mathematical model of the environment.

(Deep) Reinforcement Learning



David Silver, Introduction to Reinforcement Learning

<https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver>

Yuxi Li, Deep Reinforcement Learning, arXiv, 2018

Sequential Decision Making

- Goal: *select actions to maximise total future reward*
- **Actions** may have long-term consequences
- **Reward** may be delayed
 - It may be better to sacrifice immediate reward to gain more long-term reward
- Examples:
 - A financial investment (may take months to mature)
 - Refuelling a helicopter (might prevent a crash in several hours)
 - Blocking opponent moves (might help winning chances many moves from now)

Sequential Decision Making

- The multi-armed bandit problem



Photo by [Carl Raw](#) on [Unsplash](#)

Sequential Decision Making

- The multi-armed bandit problem
- We have an **agent** which (or whom) we allow to choose **actions**
- Each **action** has a **reward** that is returned according to a given, underlying **probability distribution**.
- The game is played over **many episodes** (single actions in this case)
- The goal is to **maximize** your **reward**.

Sequential Decision Making

- Let's formalise this:
- **Action:** choose a lever as *action a_t* from a set A at *time t*
 - *i.e. which of the levers of a multi-armed bandit to pull, and when*
 - *or if you have several single-armed bandits, which bandit's lever to pull*
- **Reward:** *action a* has a different, unknown pdf $p(r | a)$ generating a reward r_t at *time t*
- **Policy:** choose an *action a* as pdf $\pi(a)$
 - *i.e. the likelihood of choosing an action*

Sequential Decision Making

- Define an action value function Q for each time step t so that:

$$\begin{aligned} Q_{t+1}(a) &= \frac{1}{t} \sum_{i=1}^t r_i = \frac{1}{t} \left(r_t + \sum_{i=1}^{t-1} r_i \right) = \frac{1}{t} \left(r_t + (t-1) \frac{\frac{1}{t-1} \sum_{i=1}^{t-1} r_i}{(t-1)} \right) \\ &= \frac{1}{t} (r_t + (t-1) Q_t(a)) \\ &= \frac{1}{t} (r_t + t Q_t(a) - Q_t(a)) \\ &= Q_t(a) + \frac{1}{t} (r_t - Q_t(a)) \end{aligned}$$

– Which gives us an **incremental update rule!**

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize} [\text{Target} - \text{OldEstimate}].$$

Sequential Decision Making

- Exploitation:
 - We can exploit a known good action
 - This is a deterministic policy called **greedy action selection**
- Exploration:
 - Explore also other moves (potentially more creative, but also riskier/potentially wrong)
- Exploitation is the right thing to do to maximize the expected reward on the one step.
- Exploration may produce the greater total reward in the long run.

Sequential Decision Making

- Sample discrete actions a from $\pi(a)$

- There is a choice of distributions!

- E.g. we can choose a uniform random distribution, $\pi(a) = \frac{1}{|A|}$

- Where $|A|$ is the cardinality of the set of different actions A

- Or we can choose Epsilon Greedy,

$$\pi(a) = \begin{cases} 1 - \varepsilon & \text{if } a = \max_a Q_t(a) \\ \frac{\varepsilon}{n-1} & \text{else} \end{cases}$$

- Where for a large (close to 1) value of ε this would mean more exploration

- And for a small (close to 0) value of ε this would mean more deterministic exploitation

- Or we can choose SoftMax,

$$\pi(a) = \frac{e^{\frac{Q_t(a)}{\tau_t}}}{\sum_{n=1}^{|A|} e^{\frac{Q_t(a_n)}{\tau_t}}}$$

- Where τ_t is called a time-varying temperature which decreases exploration over time

Sequential Decision Making

- A simple bandit algorithm:

A simple bandit algorithm

Initialize, for $a = 1$ to k :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Loop forever:

$$A \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with probability } 1 - \varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases} \quad (\text{breaking ties randomly})$$

$$R \leftarrow \text{bandit}(A)$$

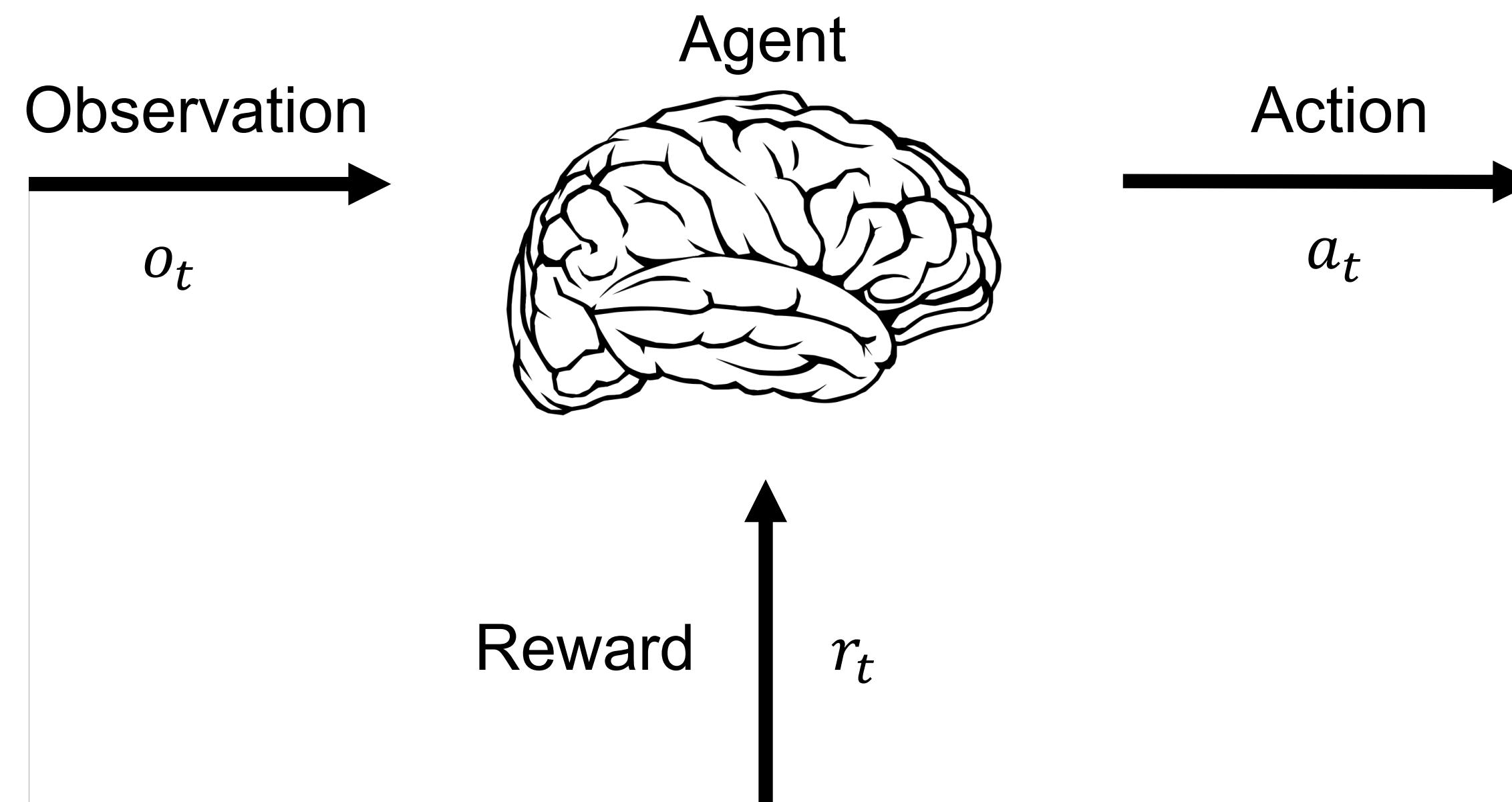
$$N(A) \leftarrow N(A) + 1$$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)} [R - Q(A)]$$

Summary on Sequential Decision Making

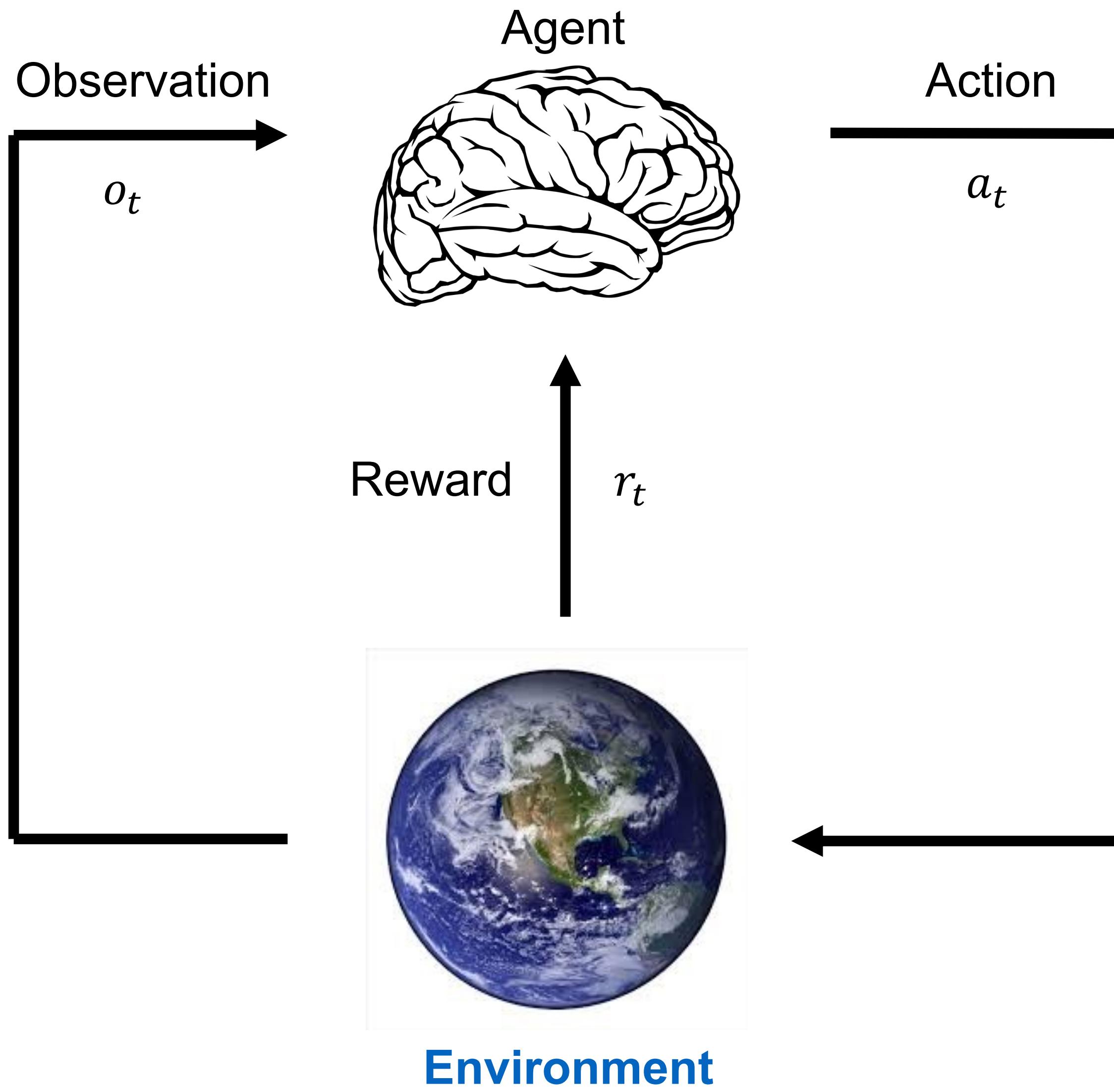
- We have considered sequential decision making in the setting of multi-armed bandits
- We found that by estimating an action value function $Q_t(a)$ and using an action selection policy $\pi(a)$ we can maximise a reward
- Exploration of different actions is necessary
 - We can increasingly move from exploration to exploitation
- For now we assumed actions do not depend on the state of the world, and that our action at time t **does not** influence future rewards
 - (but we will do so in the following)

Reinforcement Learning



- At each step t the agent:
 - Executes action a_t
 - Receives observation o_t
 - Receives scalar reward r_t

Reinforcement Learning



- At each step t the agent:
 - Executes action a_t
 - Receives observation o_t
 - Receives scalar reward r_t
- The **environment**:
 - Receives action a_t
 - Emits observation o_{t+1}
 - Emits scalar reward r_{t+1}
- t increments at env. step

History and State

- **History** is the sequence of observations, actions, rewards

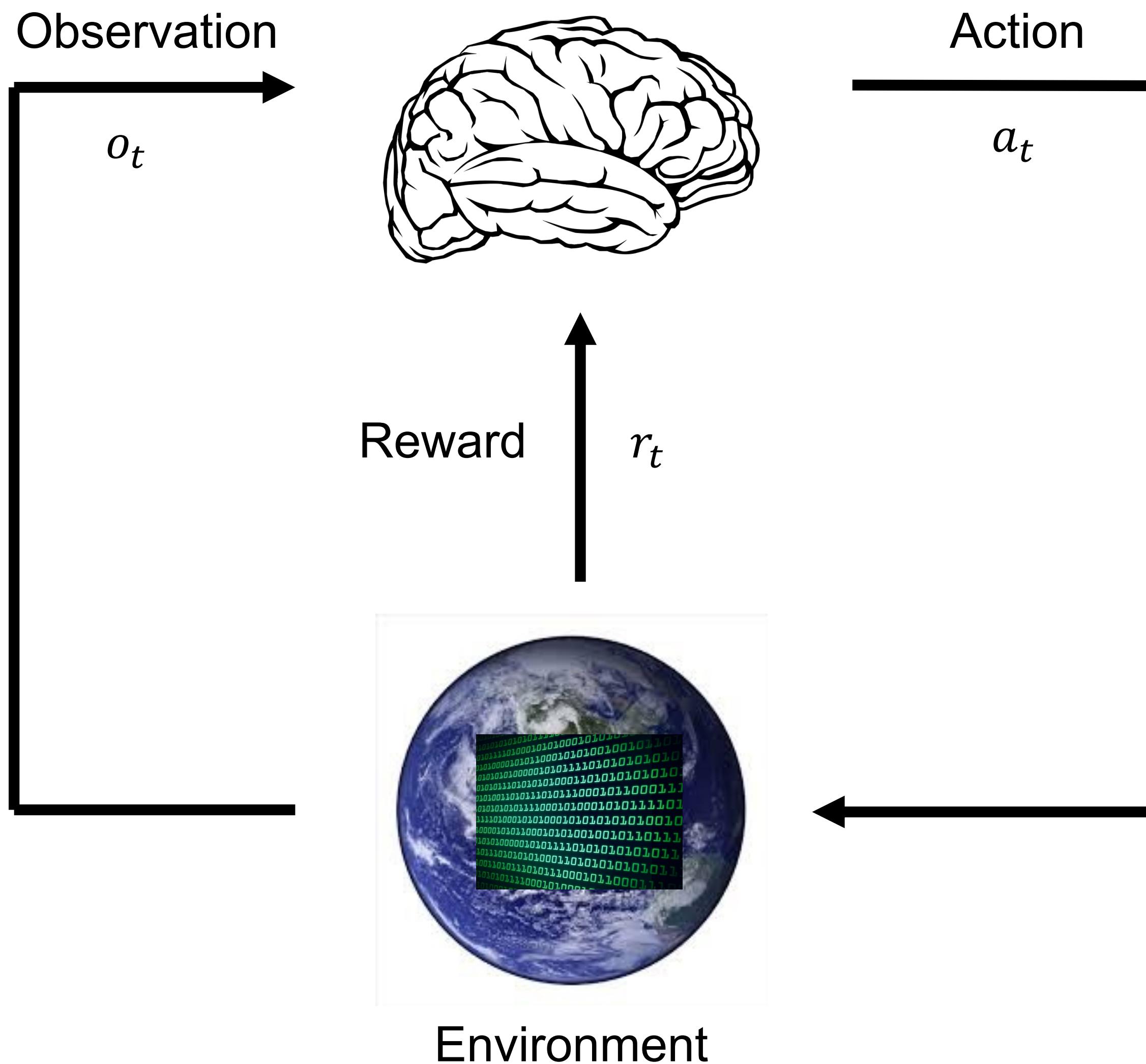
$$h_t = o_1, r_1, a_1, \dots, a_{t-1}, o_t, r_t$$

i.e. all observable variables up to time t

- What happens next depends on the history:
 - The agent selects actions
 - The environment provides observations/rewards
- **State** is the information used to determine what happens next
 - Formally, state is a function of the history:

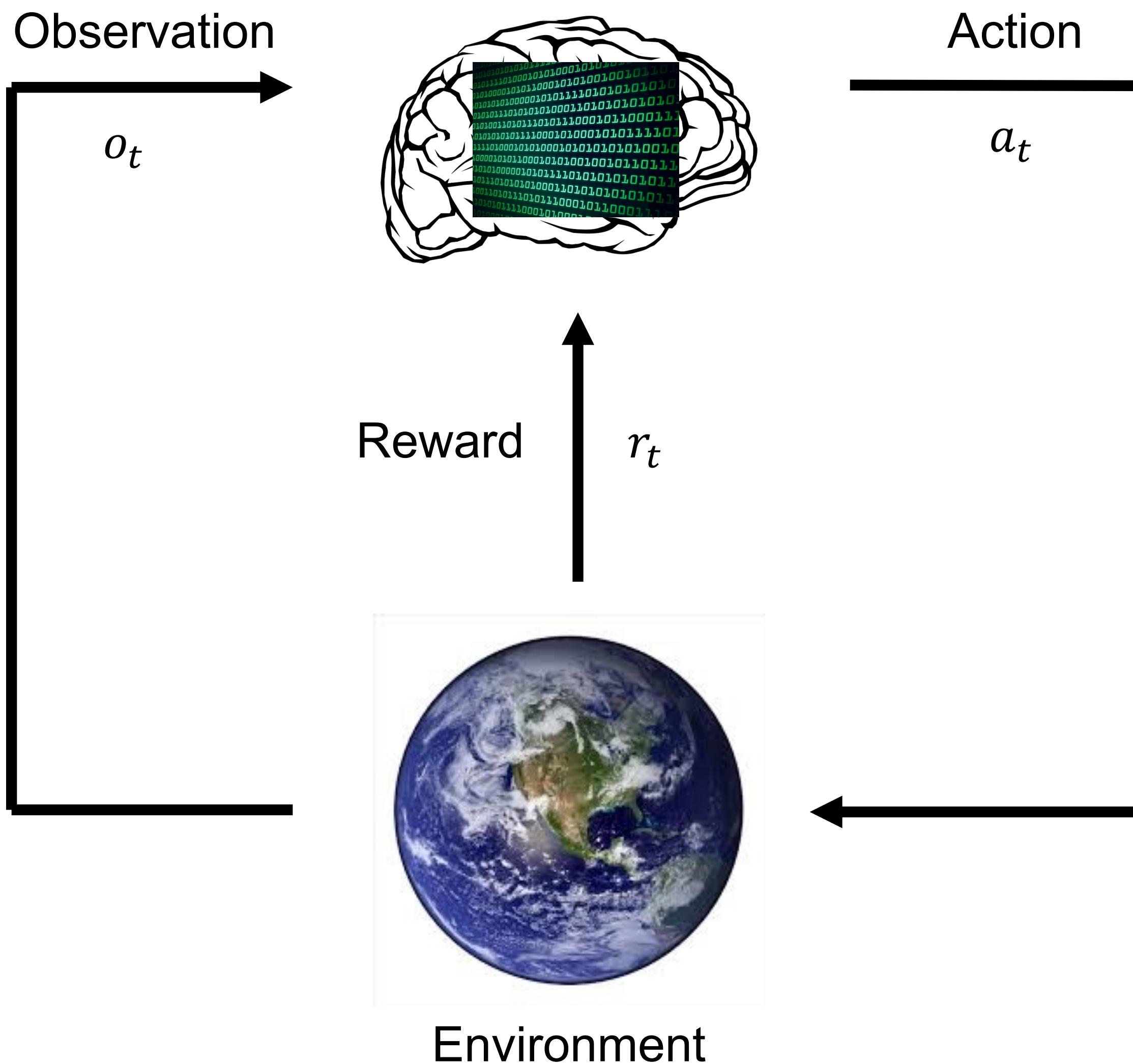
$$S_t = f(h_t)$$

Reinforcement Learning: Environment State



- The environment state s_t^e is the environment's private representation
 - i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
 - Even if s_t^e is visible, it may contain irrelevant information

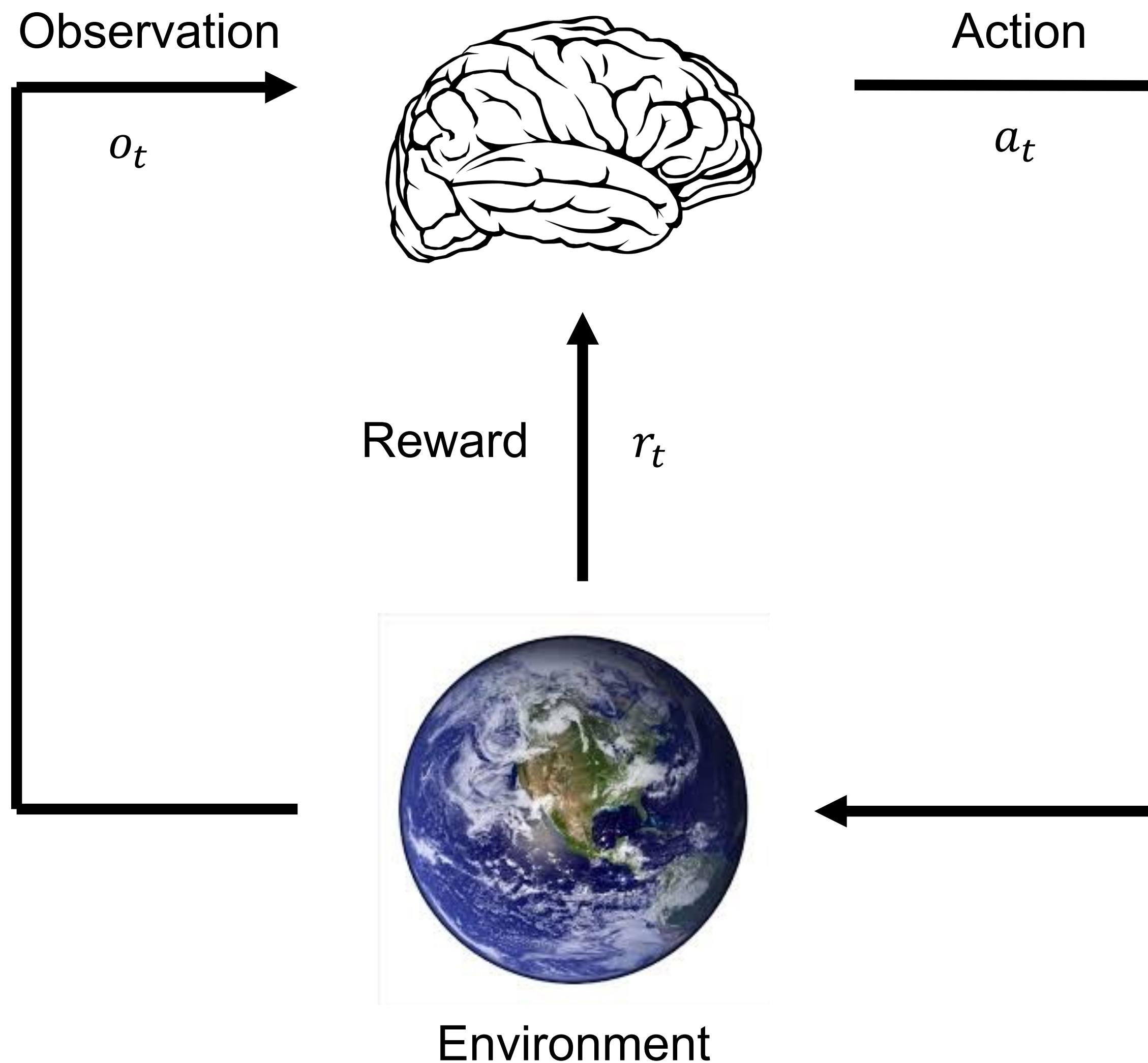
Reinforcement Learning: Agent State



- The agent state s_t^a is the agent's internal representation
 - i.e. whatever information the agent uses to pick the next action
 - i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history*:
$$s_t^a = f(h_t)$$

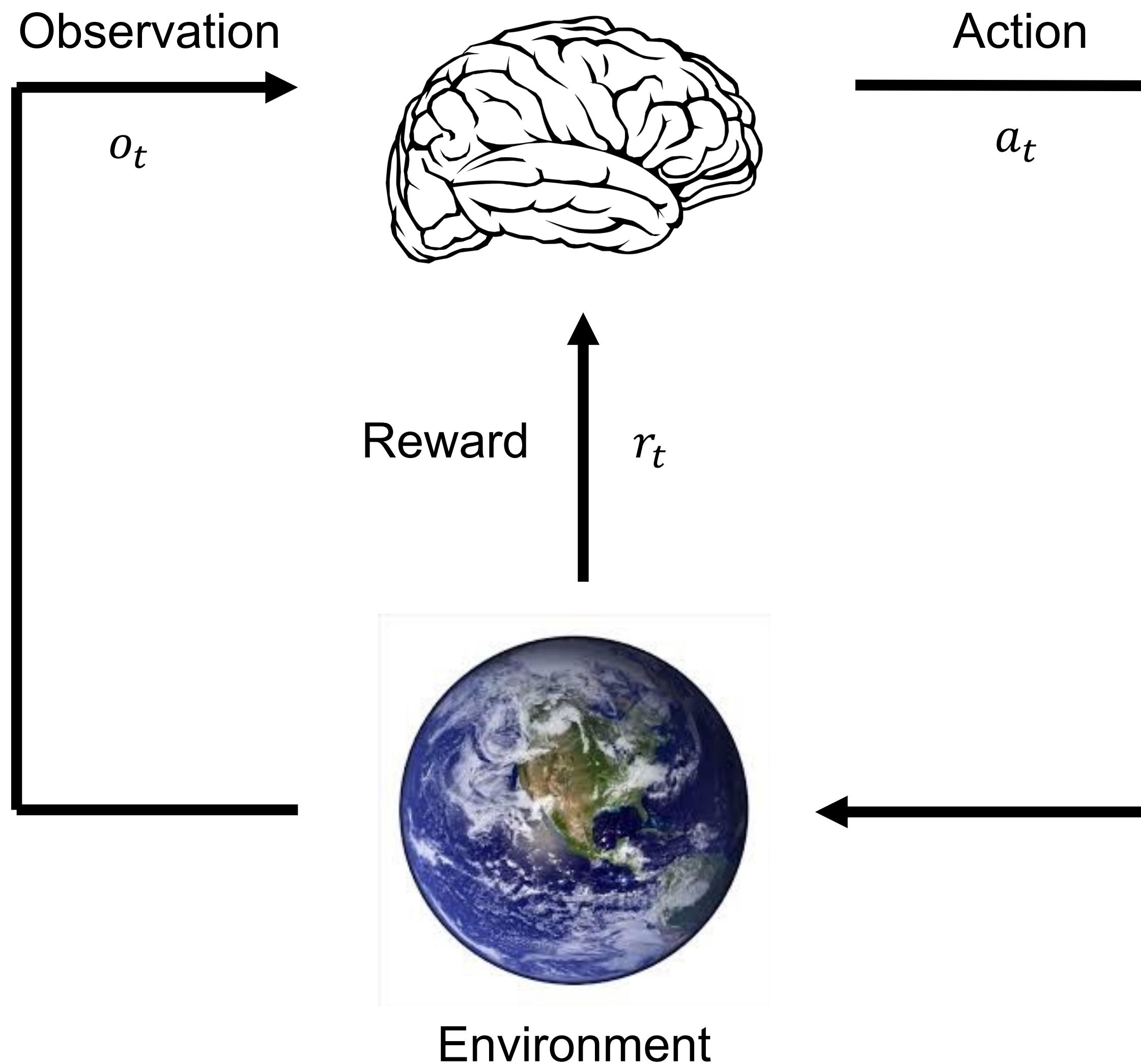
*Recall history: $h_t = o_1, r_1, a_1, \dots, a_{t-1}, o_t, r_t$

Reinforcement Learning: Partially observable environments



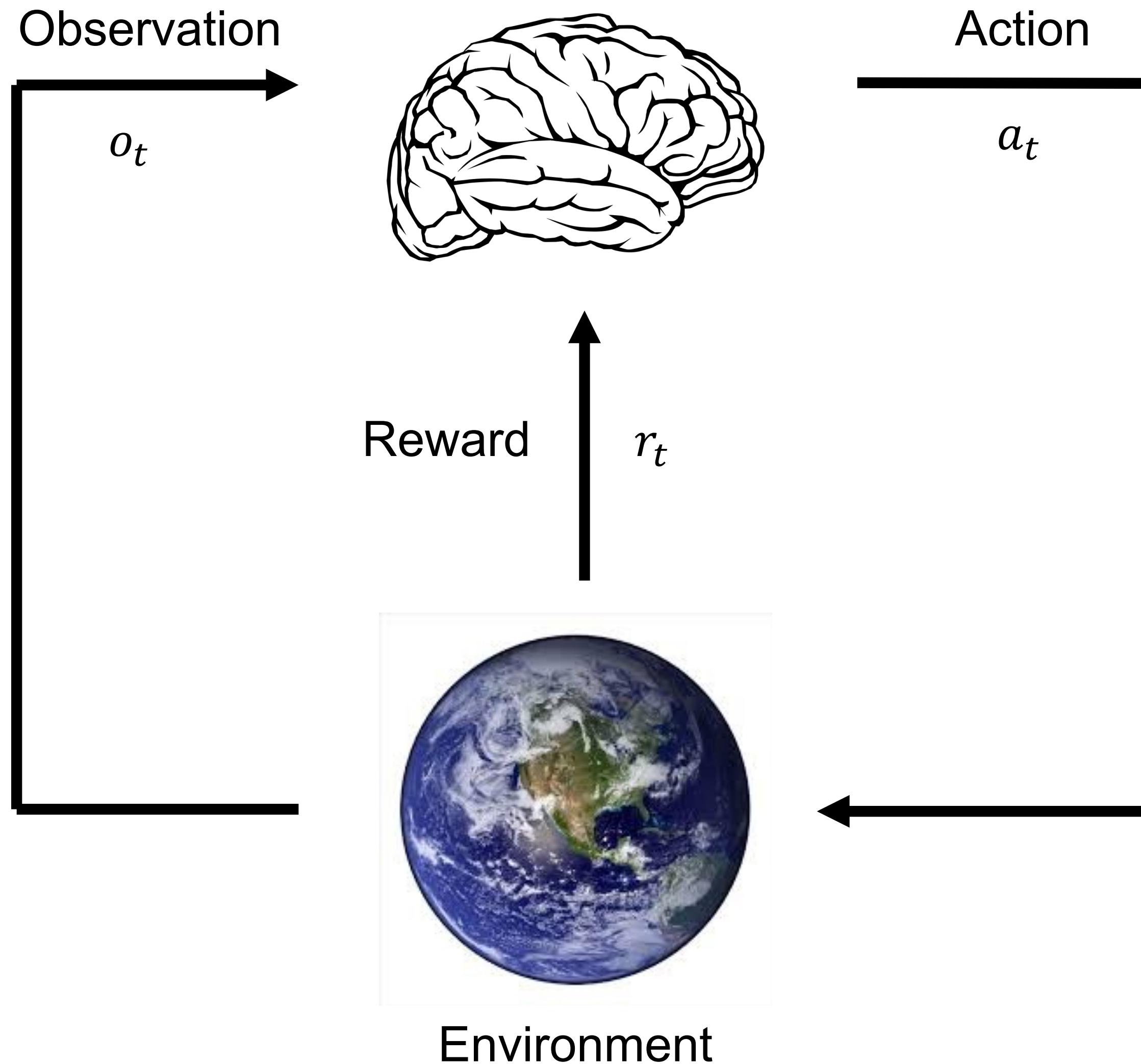
- **Partial observability**, i.e. agent indirectly observes environment:
 - A robot with camera vision isn't told its absolute location
 - A trading agent only observes current prices
 - A poker playing agent only observes public cards
- Agent state \neq environment state
- This is a partially observable Markov decision process (POMDP)

Reinforcement Learning: Fully observable environments



- **Special case** of full observability:
agent directly observes
environment state
- Agent state = environment state =
information state
- Decisions are based on the
information state
- Formally, this is a **Markov
decision process (MDP)**

Reinforcement Learning: Mathematical Formulation



- **Markov Decision Process (MDP)***
 - Set of states S
 - Set of actions A
 - Reward signal $R: s_t \times a_t \times s_{t+1} \rightarrow R$
 - Transition function $T: s_t \times a_t \rightarrow s_{t+1} \equiv P(s_{t+1}|s_t, a_t)$
 - *Discount factor γ : we'll come to that later*
- **Markov assumption**
 - s_t and a_t are conditionally independent of all previous states and actions

*Also called Markov Reward Process (MRP)

Reinforcement Learning

Reinforcement learning (RL)

- Learning what to do (how to map situations to action) so as to maximize sum of rewards seen over the learner's lifetime, i.e. learn a policy π

$$\pi: S \rightarrow A$$

Two distinguishing characteristics of RL

- Trial-and-error search
 - The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them (there is no supervisor, only a reward signal)
- Delayed rewards
 - Actions may affect not only the immediate reward but also all subsequent rewards

Reinforcement learning: Terminology

- **State**
 - Whatever information is available to the agent about its environment
- **Terminal state**
 - The final state where there are no more available actions, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states
- **Episode**
 - Complete play from the initial to final state $(s_0, a_0, r_0), (s_1, a_1, r_1), \dots, (s_n, a_n, r_n)$
- **Cumulative Reward**
 - The discounted sum of rewards accumulated throughout an episode

$$R = \sum_{t=0}^n \gamma^t r_{t+1}$$

γ : discount factor or discount-rate parameter

Future rewards are discounted by $0 < \gamma \leq 1$

RL components

- **Policy** π
 - The agent's strategy to choose an action at each state (maps state to action). Can be
 - **deterministic** $\pi(s) = a$, or
 - **stochastic** $\pi(a|s) = P(A_t = a|S_t = s)$
 - Optimal policy π^* is the theoretical policy that maximizes the expectation of cumulative rewards
- **Reward** r
 - A reward r_t is a scalar feedback signal
 - Indicates how well agent is doing at step t

RL components (cnt'd)

- **State value function V**
 - **Prediction of future total reward** an agent can expect to accumulate over the future
 - Used to evaluate the goodness/badness of states
- **State action value function Q**
 - **Expected return** for taking action in a state and thereafter following an optimal policy.
- **Model of the environment (optional)**
 - **Planning** - given a state and action, the model might predict the resultant next state and next reward (model-based methods)
 - **Model-free methods** that are explicitly trial-and-error learners (almost the opposite of planning)

Exploitation-Exploration Dilemma

- **The goal of the agent is to maximize its future expected reward**
 - **Exploitation** - prefer actions that it has tried in the past and found to be effective in producing reward
 - **Exploration** - discover such actions, it has to try actions that it has not selected before
- **Simple exploration methods are the most practical:**
 - ϵ -greedy - the agent chooses an action uniformly at random with probability $(1- \epsilon)$
 - ϵ -soft - similar to ϵ -greedy but the probability is divided by number of actions
 - Softmax
 - ...
- Note exploration and exploitation ***do not arise*** in supervised and unsupervised learning

Value Function and Q-Function

Remember:

- A value function is defined as prediction of the expected, accumulated, discounted, future reward in order to measure how good each state or state-action

How good is a state?

- **State value function** estimates the expected cumulative reward from following the policy π from state s :

$$V^\pi(s) = \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi \right]$$

How good is a state-action pair?

- **State-action value function** estimates a value of each action a in each state s under policy π

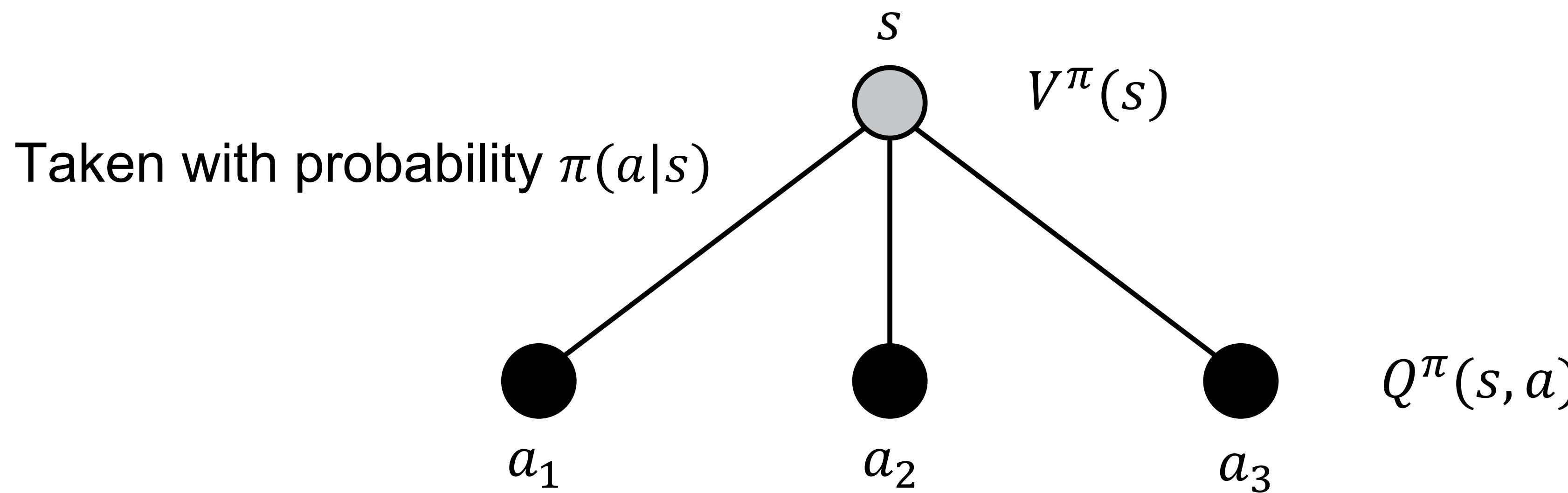
$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

Value Function and Q-Function

Optimal policy achieves the best expected return from any initial state

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$



Bellman Equation

- The optimal Q-value function Q^* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

- Q^* satisfies the following **Bellman equation***:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q^*(s', a') | s, a]$$

- **Intuition:** if the optimal state-action values for the next time-step $Q^*(s^*, a^*)$ are known, optimal strategy is to take the action that maximizes the expected value of $r + \gamma Q^*(s', a')$
- The optimal policy π^* corresponds to taking the best action in any state as specified by Q^*

*Bellman equation rewrites problem as a recursive definition of the value function

Solving for the optimal policy

- **Value iteration** algorithm: Use Bellman equation as an iterative update

$$Q_{i+1}(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q_i(s', a') | s, a]$$

- Q_i will converge to Q^* as $i \rightarrow \infty$

Q-learning (Watkins 1989 & 1992)

- The action that is optimal for each state is the action that has the highest long-term reward $Q^*(s, a)$

$$Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)$$

learned value

- Updates the state-action value function iteratively
- Learn the value function and policy simultaneously
- Keep an estimate of $Q(s, a)$ in a table
- What's the problem with this?**

		Action			
		0	1	2	3
State	0	-1	-1	0	-1
	1	-1	0	-1	100
	2	-1	0	-1	-1
	3	0	-1	0	100

Q-learning (Watkins 1989 & 1992)

- The action that is optimal for each state is the action that has the highest long-term reward $Q^*(s, a)$

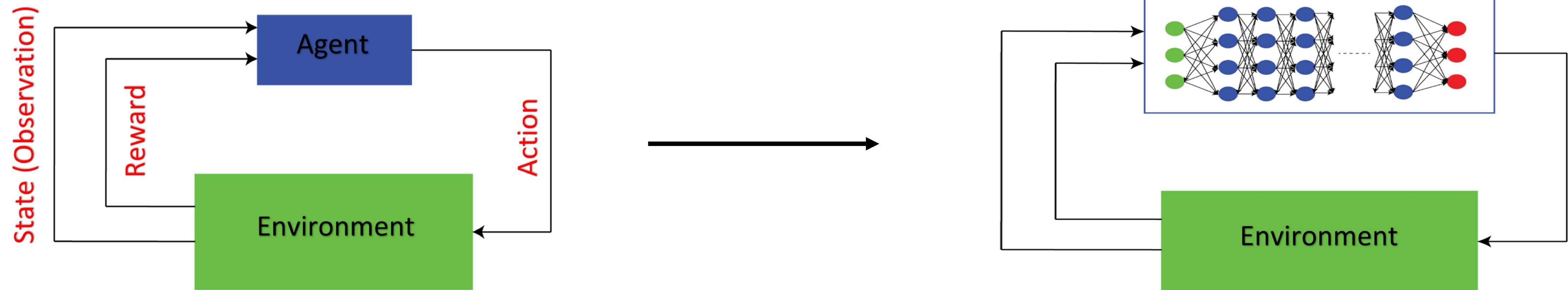
$$Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)$$

learned value

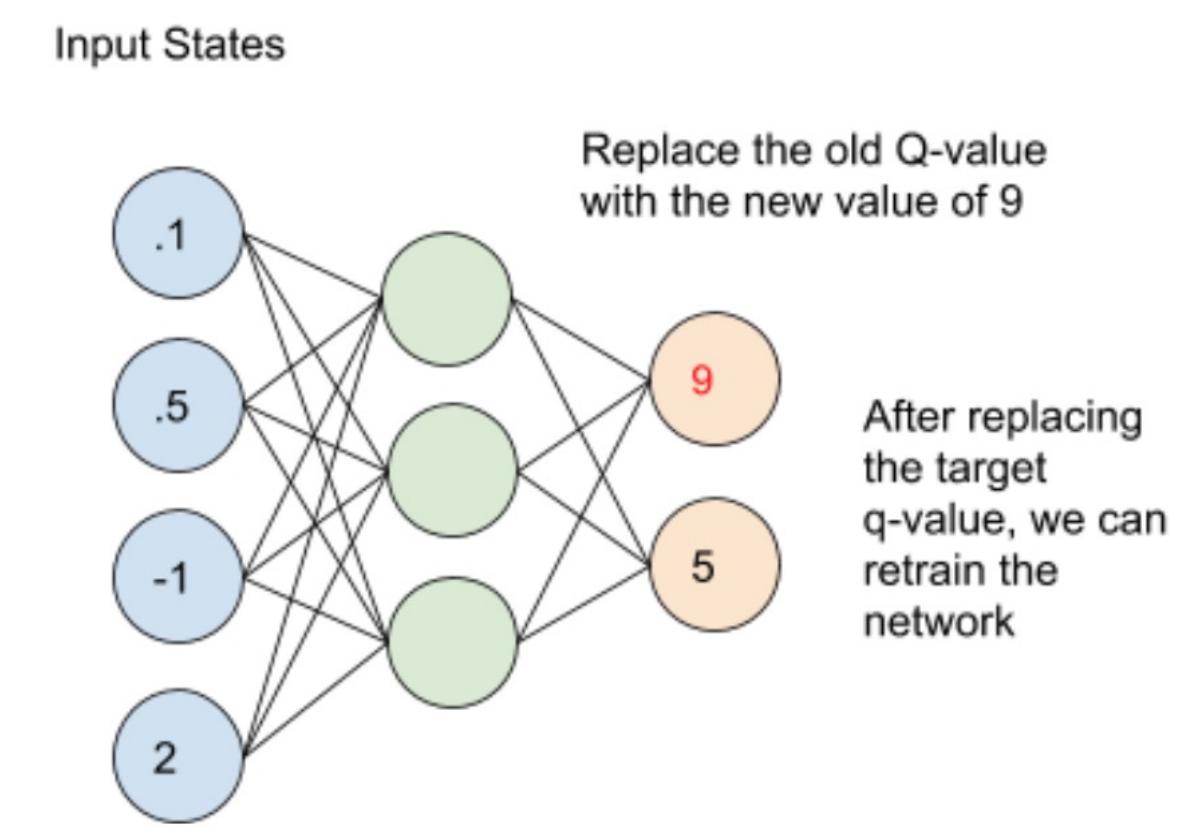
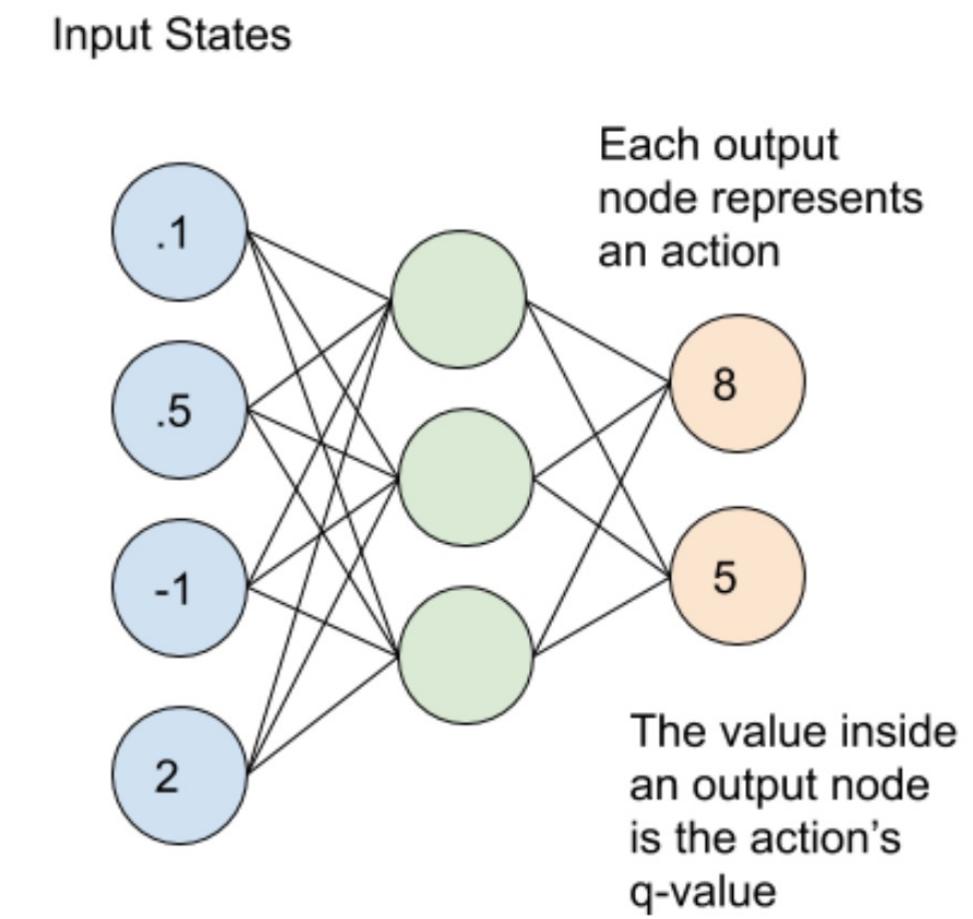
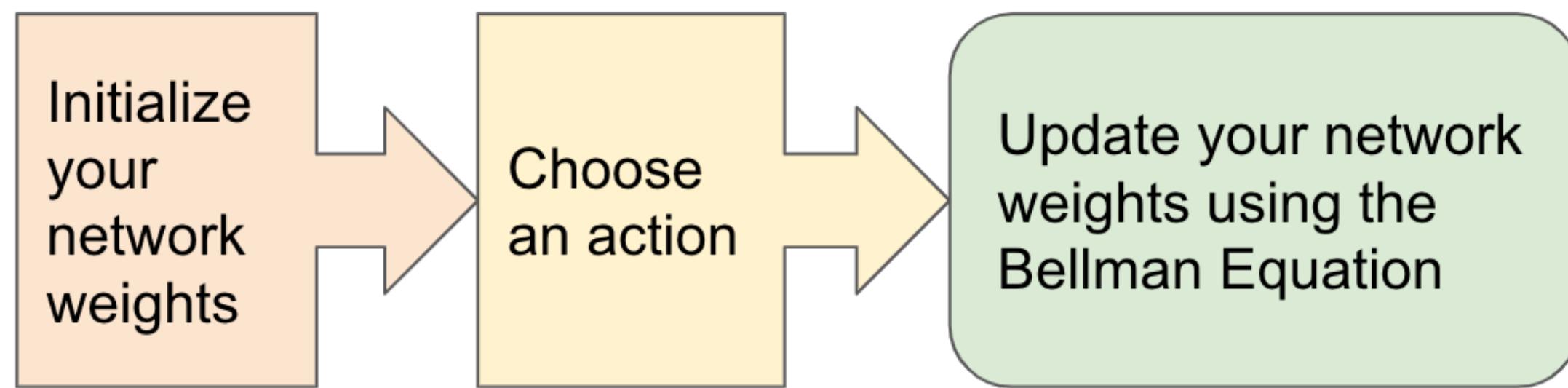
- Not scalable.** Must compute $Q(s, a)$ for every state-action pair. If state is e.g. current game state pixels, computationally infeasible to compute for entire state space!
- Solution:** use a function approximator to estimate $Q(s, a)$, e.g. a neural network!

Part II

Deep Reinforcement learning using Deep Q-networks (DQN)



Deep Q-Networks



<https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abff>

- Deep Q-Networks
- Double DQN
- Duel DQN

Deep Q-Networks (DQN, Mnih 2013)

- DQN is an implementation of a standard Q-learning algorithm with function approximation using a ConvNet

$$Q^\pi(s, a) \approx Q^\pi(s, a; \theta)$$

- Objective function: MSE in Q-values

$$L(\theta) = E \left[\underbrace{\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta) \right)^2}_{\text{Target}} \right]$$

- Optimize end-to-end by SGD, using $\frac{\delta L(\theta)}{\delta \theta}$

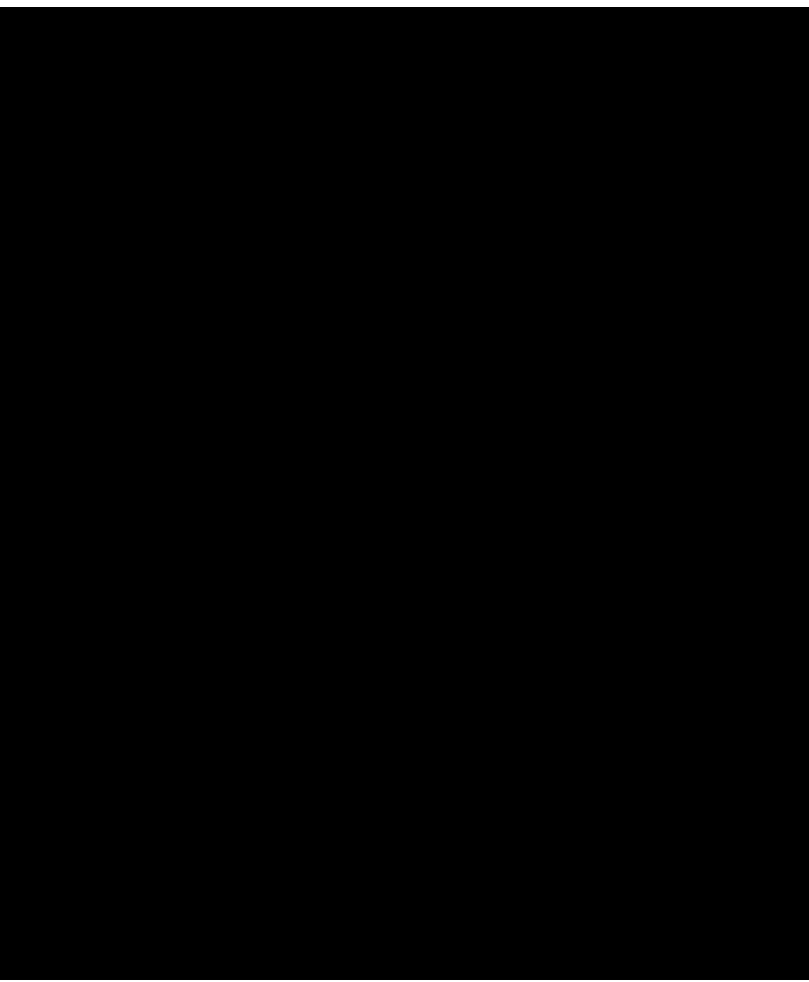
Instability and Divergence

- Sequential Data
 - Successive samples are correlated (non-i.i.d.)
- Rapid policy changes
 - Policy may oscillate because of the changes to Q-values
 - Distribution of the data can swing from one extreme to another
- Unknown reward/Q-value range
 - Gradients can be large and unstable when backpropagated

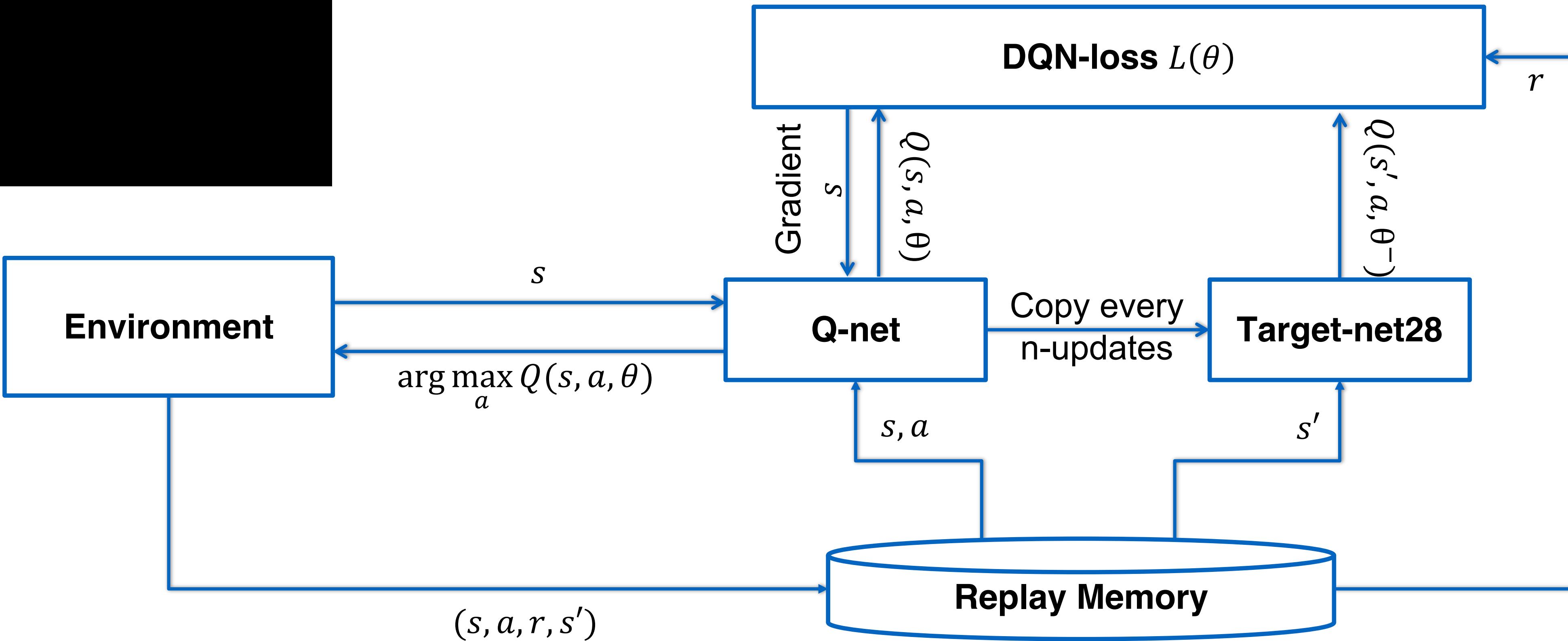
Tricks for Stability

- Experience replay memory
 - Take action according to ϵ -greedy policy
 - Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory D
 - Sample random mini-batch of transitions (s, a, r, s') from D
- Freeze target Q-network
 - Compute Q-value targets w.r.t. (old) fixed parameters θ^-
 - Periodically update fixed parameters $\theta^- \leftarrow \theta$
$$L(\theta) = \mathbb{E}_{s,r,a,s' \sim D} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right]$$
- Clip rewards
 - Clip the rewards to $[-1, +1]$
 - Prevent Q-values from becoming too large and ensures gradients are well-conditioned
 - Drawback: can't tell difference between small and large rewards

DQN Algorithm



$$L(\theta) = E_{s,r,a,s' \sim D} \left[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2 \right]$$



Improvements on DQN

Prioritized Experience Replay

- Proposed by T. Schaul in 2015
- Key idea
 - **Some experiences may be more important than others** for our training, but might occur less frequently.
 - Because we sample the batch uniformly (selecting the experiences randomly) these rich experiences that occur rarely have practically no chance to be selected.
- Store experience in priority queue according to DQN error:

$$\left| r + \gamma \max_{a'} Q(s', a', \theta^-) - Q(s, a; \theta) \right|$$

Improvements on DQN

Double Q-learning (Van Hasselt 2015)

- DQN algorithm might overestimate the true rewards
 - Q-values think the agent is going to obtain a higher return than what it will obtain in reality.
- Double Q-learning removes upward bias caused by maximum approximated action value
 - Current Q-net θ is used to select actions
 - Older target Q-net θ^- is used to evaluate actions

$$L(\theta) = E_{s,r,a,s' \sim D} \left[\left(r + \gamma \max_{a'} Q(s', Q(s', a'; \theta), \theta^-) - Q(s, a; \theta) \right)^2 \right]$$

Double DQN

- Combines DQN with double Q-learning
 - Decouple the **action selection** from the **action evaluation**
 - Instead of using the same Bellman equation as in the DQN algorithm, we use:
$$Q(s, a; \theta) = r + \gamma \text{argmax}_{a'} Q(s', a'; \theta); \theta')$$
 - First, main neural network θ decides which is best next action a' among all available next actions
 - Then, the target neural network evaluates this action to find its Q -value.
- This was shown to reduce overestimations, resulting in better final policies:
 - Double Q-learning is an **off-policy RL algorithm**, where a different policy is used for value evaluation than what is used to select the next action.

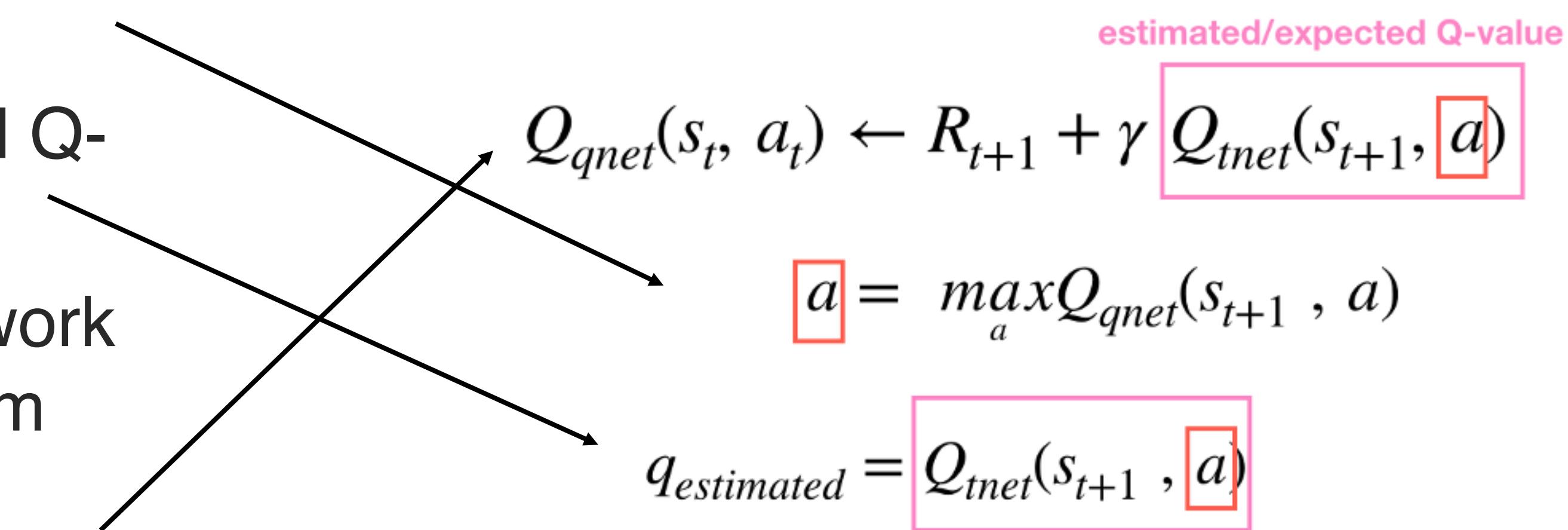
Double DQN

- Deep Q Network: select best action a with maximum Q-value of next state.
- Target Network: calculate estimated Q-value with action a selected above.
- Update the Q-value of Deep Q Network based on the estimated Q-value from Target Network
- Update Target Network based on Deep Q Network per several iterations.

$$Q_{tnet}(s, a) = Q_{qnet}(s, a)$$

- Update the parameters of Deep Q Network

Double Q Network



Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

```

Initialize primary network  $Q_\theta$ , target network  $Q_{\theta'}$ , replay buffer  $\mathcal{D}$ ,  $\tau \ll 1$ 
for each iteration do
    for each environment step do
        Observe state  $s_t$  and select  $a_t \sim \pi(a_t, s_t)$ 
        Execute  $a_t$  and observe next state  $s_{t+1}$  and reward  $r_t = R(s_t, a_t)$ 
        Store  $(s_t, a_t, r_t, s_{t+1})$  in replay buffer  $\mathcal{D}$ 
    for each update step do
        sample  $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$ 
        Compute target Q value:
            
$$Q^*(s_t, a_t) \approx r_t + \gamma \max_{a'} Q_{\theta'}(s_{t+1}, a')$$

        Perform gradient descent step on  $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$ 
        Update target network parameters:
            
$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$


```

Dueling DQN

- This algorithm splits the Q-values in two different parts, the value function $V(s)$ and the advantage function $A(s, a)$.
 - The (action-independent) value function $V(s)$ tells us how much reward we will collect from state s .
 - The (action-dependent) advantage function $A(s, a)$ tells us how much better one action is compared to the other actions.

- Combining the value V and the advantage A for each action, we can get the Q-values:

$$Q(s, a) = V(s) + A(s, a)$$

- Through this decoupling, the Dueling DQN can learn which states are (not) valuable without having to learn the effect of each action at each state.

Dueling DQN

Dueling DQN split Q-net into two channels

- Proposed by Z. Wang in 2015*

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'; \theta, \alpha) \right)$$

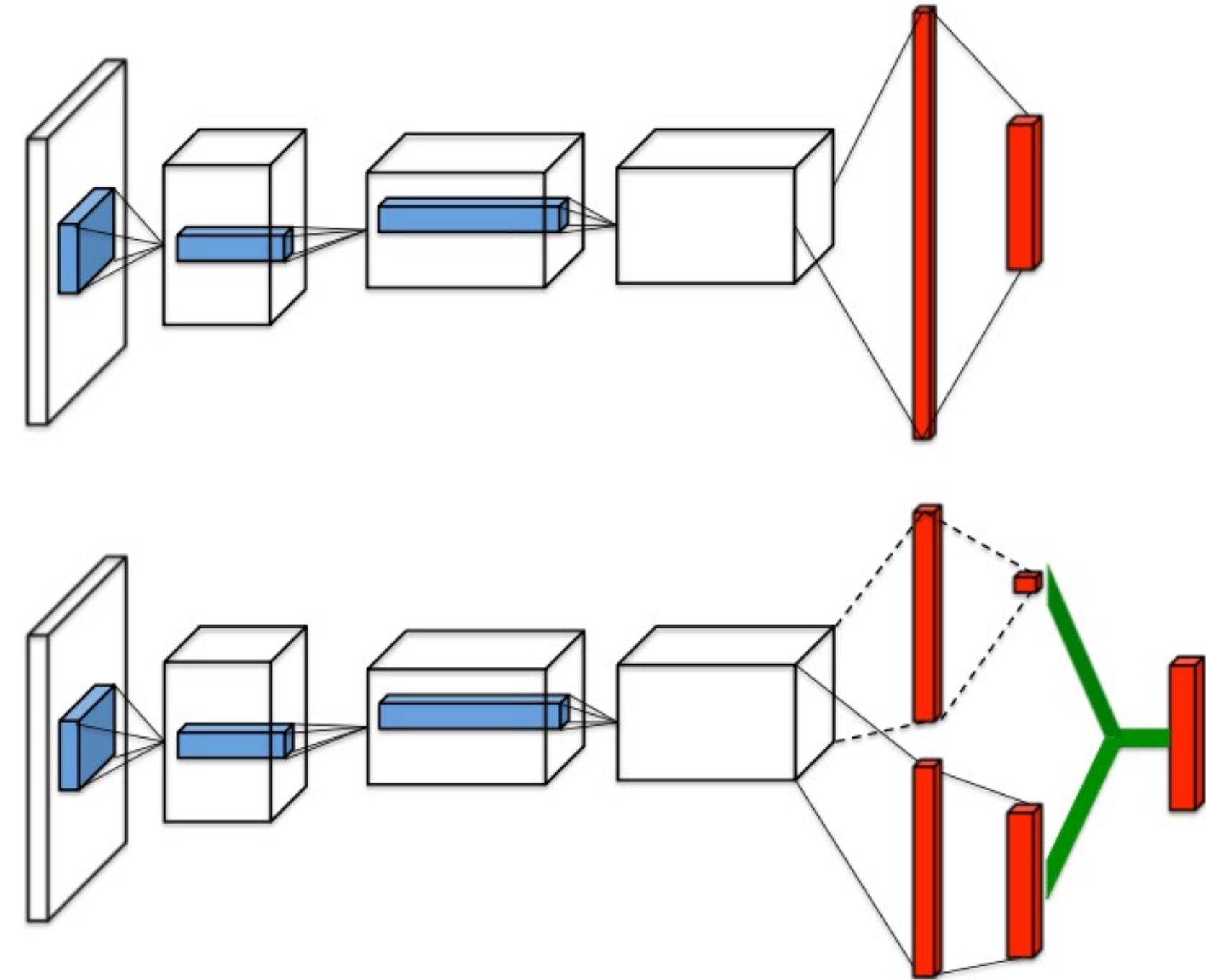


Figure 1. A popular single stream Q -network (**top**) and the dueling Q -network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q -values for each action.

* <https://arxiv.org/abs/1511.06581>

DQN vs Dueling DQN

S.K. Zhou, H.N. Le, K. Luu et al.

Medical Image Analysis 73 (2021) 102193

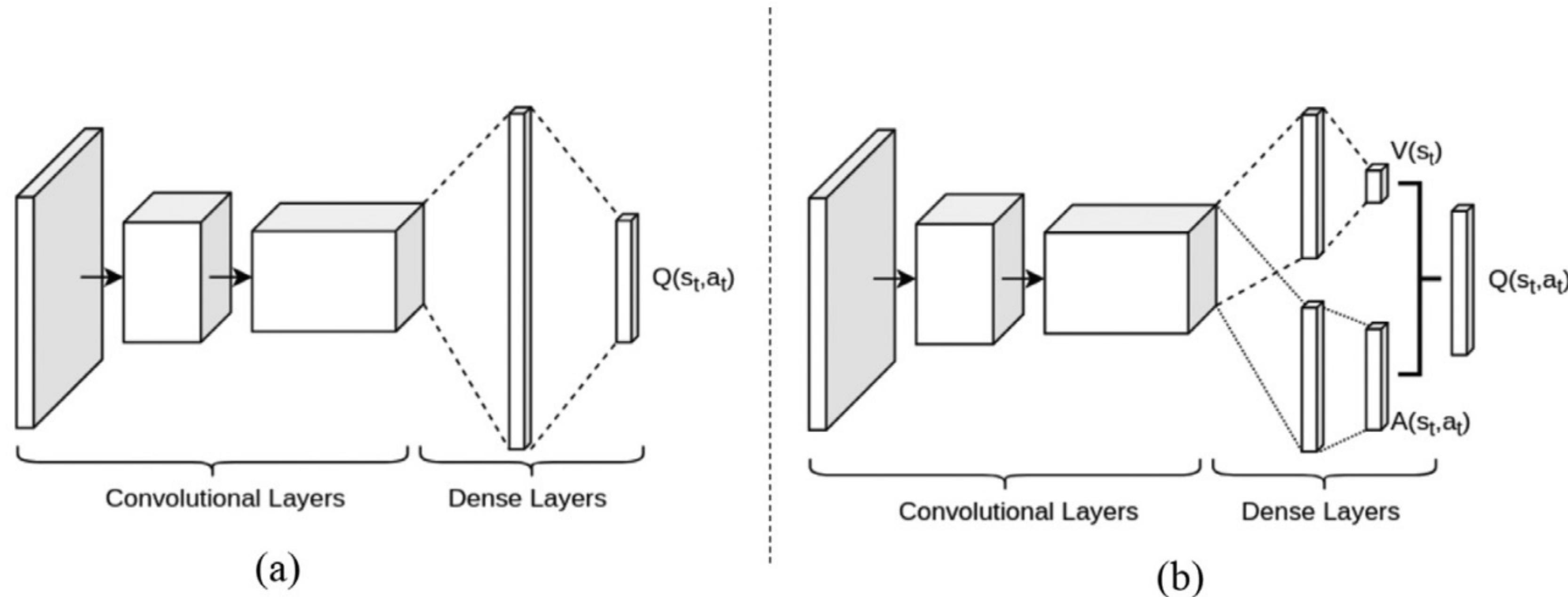


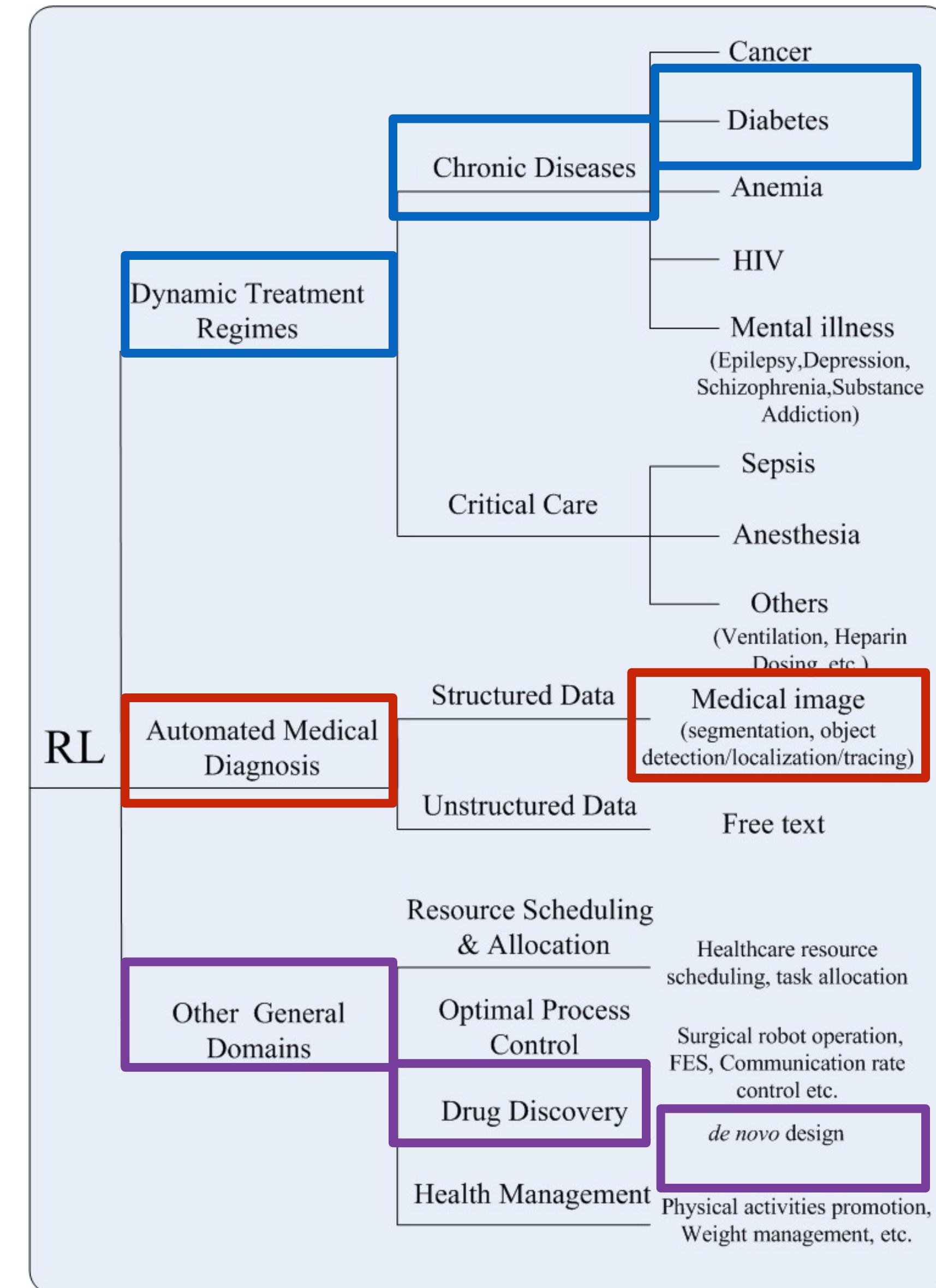
Fig. 4. (a): Network structure of Deep Q-Network (DQN), where Q-values $Q(s,a)$ are generated for all actions for a given state. (b): Network structure of Dueling DQN, where value function $V(s)$ and advantage function $A(s, a)$ are combined to predict Q-values $Q(s, a)$ for all actions for a given state.

Part III

RL applications in healthcare

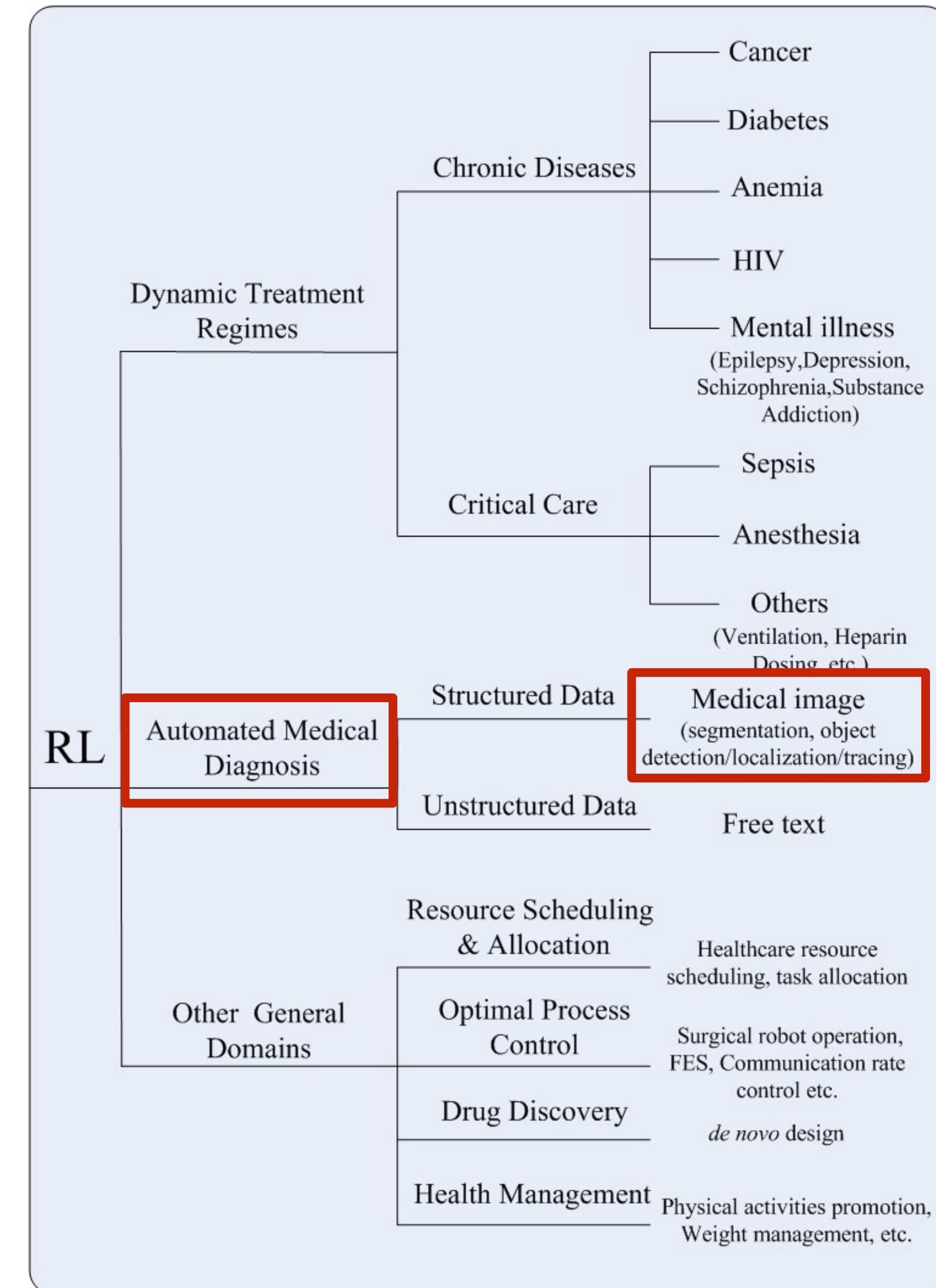


RL in healthcare applications



Reinforcement Learning in Healthcare: A Survey
You et al., ACM Computing Surveys (2023)
<https://doi.org/10.1145/3477600>
<https://arxiv.org/pdf/1908.08796.pdf>

RL in healthcare applications



Reinforcement Learning in Healthcare: A Survey
You et al., ACM Computing Surveys (2023)
<https://doi.org/10.1145/3477600>
<https://arxiv.org/pdf/1908.08796.pdf>

RL in medical imaging

Image Segmentation

- RL for image thresholding and segmentation

**Shokri, M. et al.
(2003)**
**Sahba, F. et al.
(2006)**

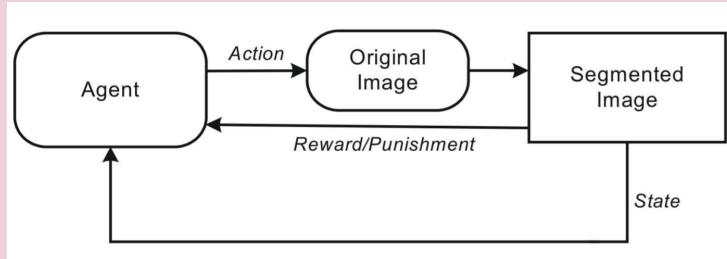
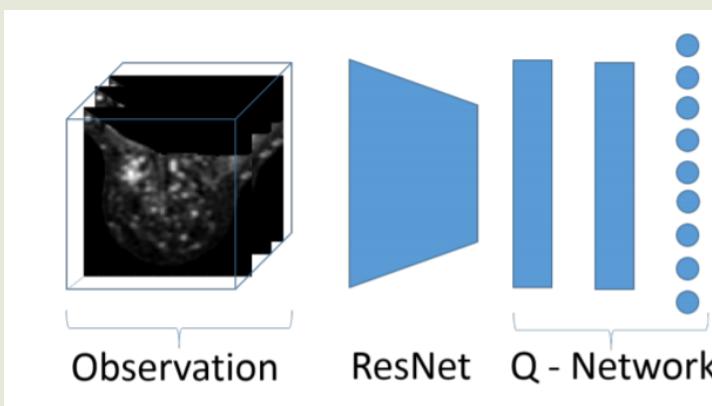


Image Localization

- Deep RL for Active Breast Lesion Detection from DCE-MRI

**Maicas, G. et al.
(2017)**



Landmark Detection

- Artificial agent for anatomical landmark detection in medical images

**Ghesu, FC. et al.
(2016, 2017)**
Alansary, A. (2019)

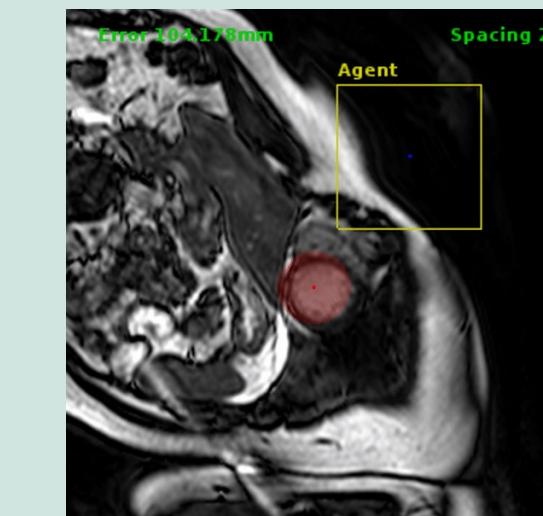
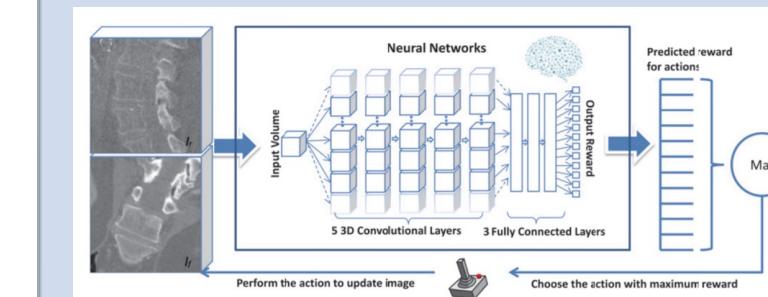


Image Registration

- Artificial Agent for Robust Image Registration (rigid, non-rigid, 2D/3D)

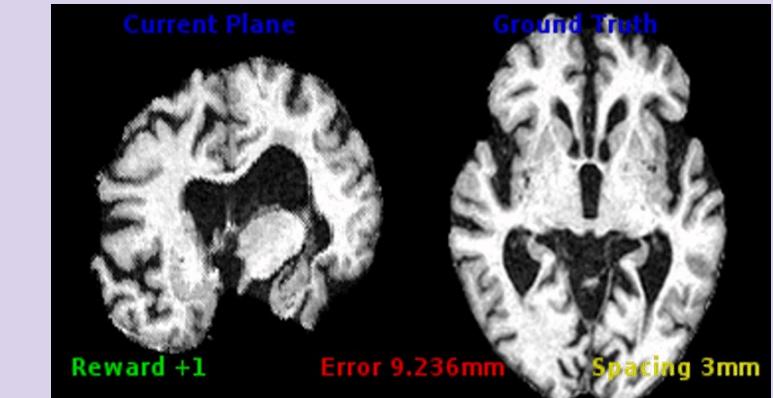
Liao, R. et al. (2017)
Krebs J. et al. (2017)
Miao, S. et al. (2017)



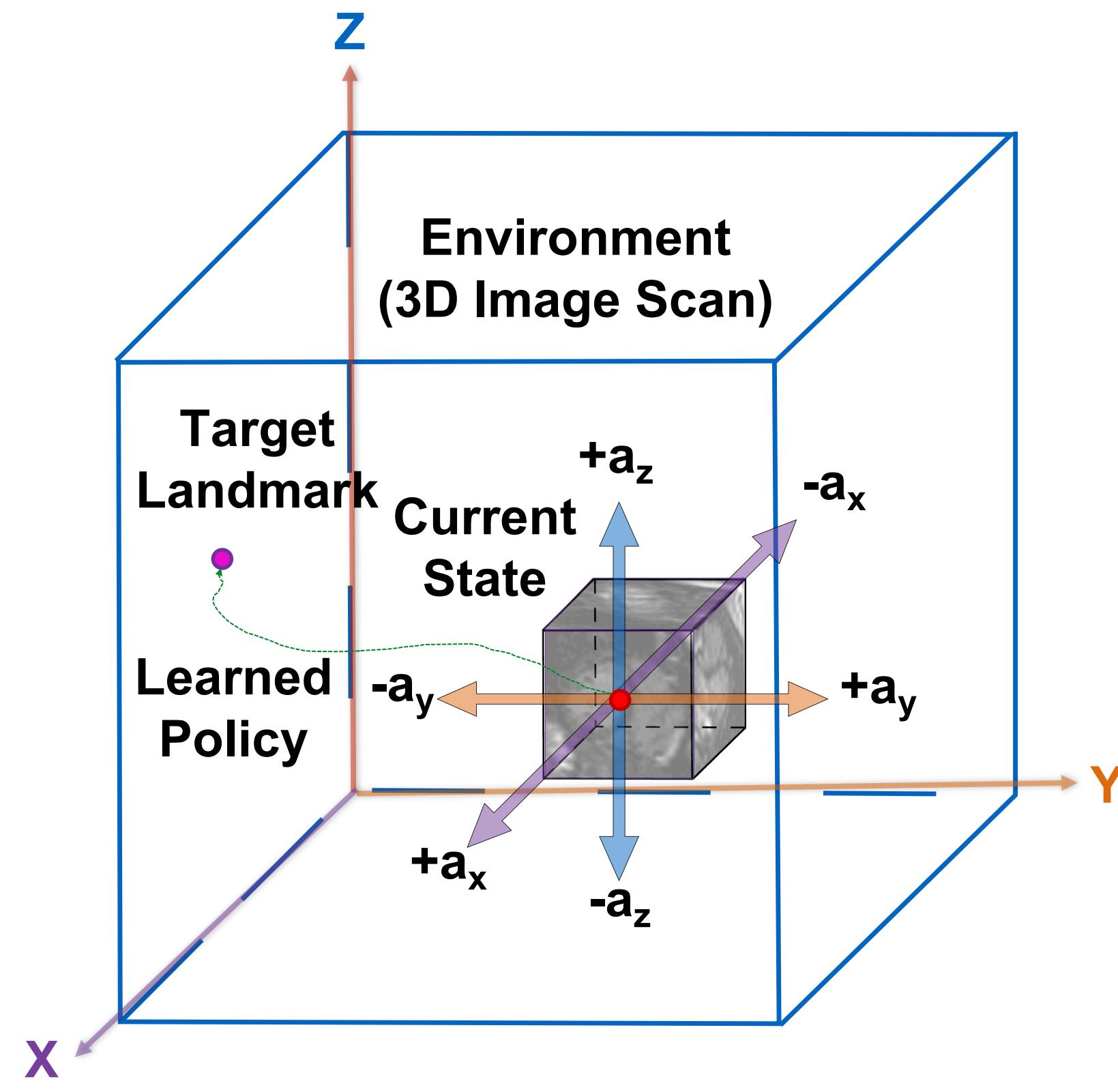
View Planning

- Automatic view planning using deep RL agents

Alansary, A. (2018)

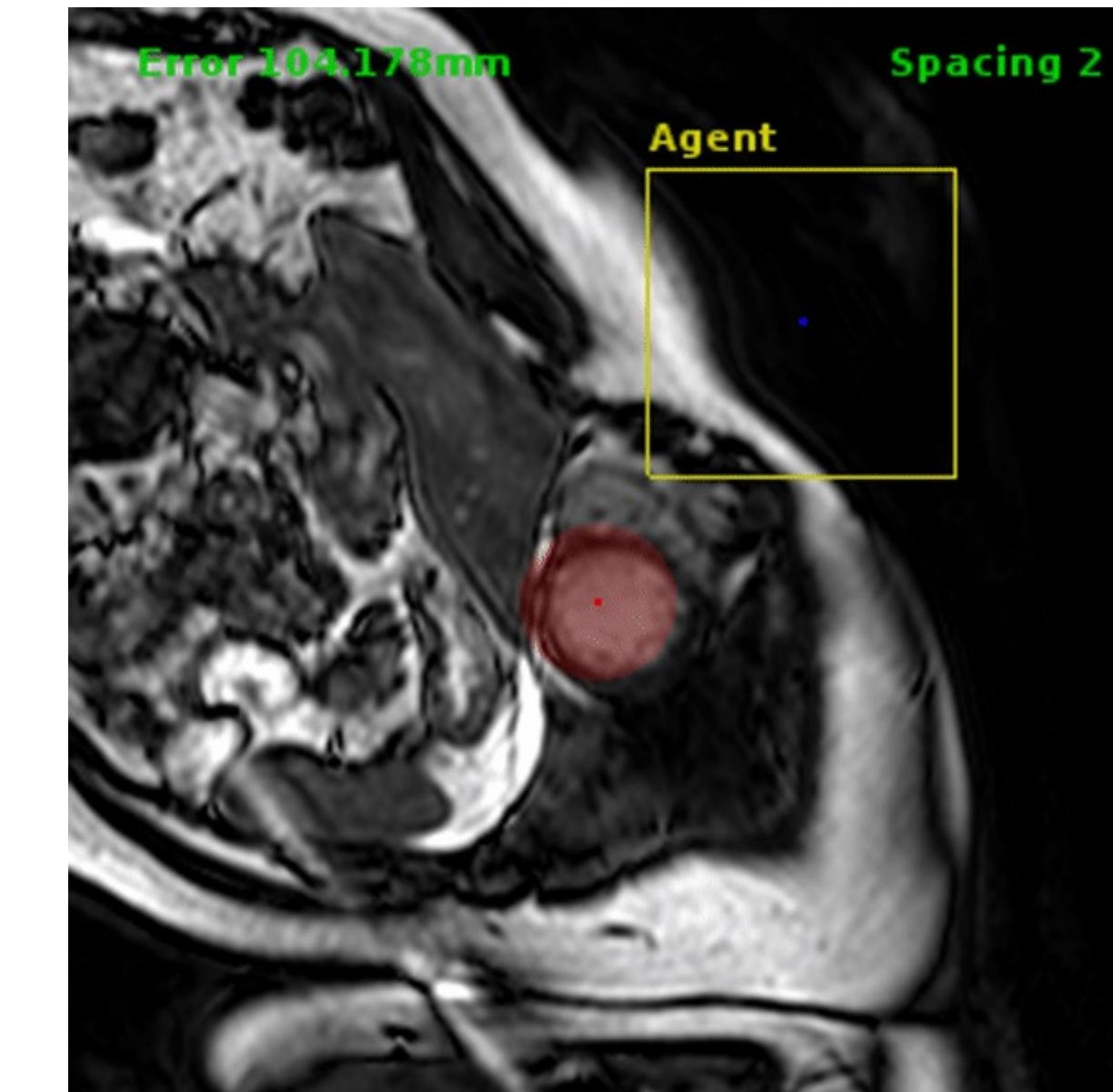


RL for anatomical landmark detection*



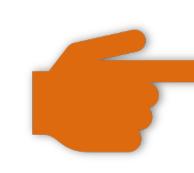
Agent

The agent learns to navigate in a 3D image towards the target landmark (sequential decision process)



States

3D region of interest centered around the landmark (current location)



Navigation actions

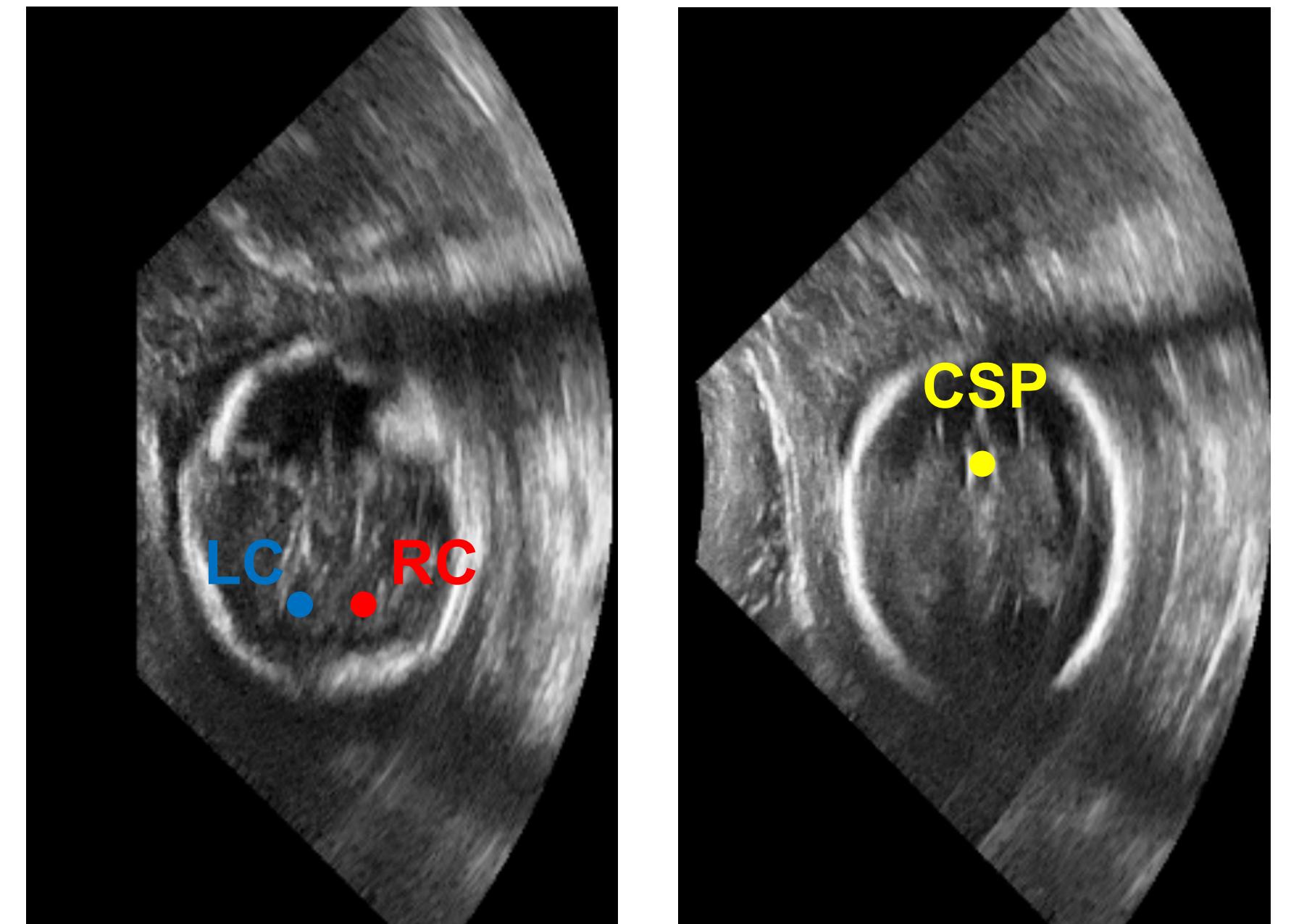
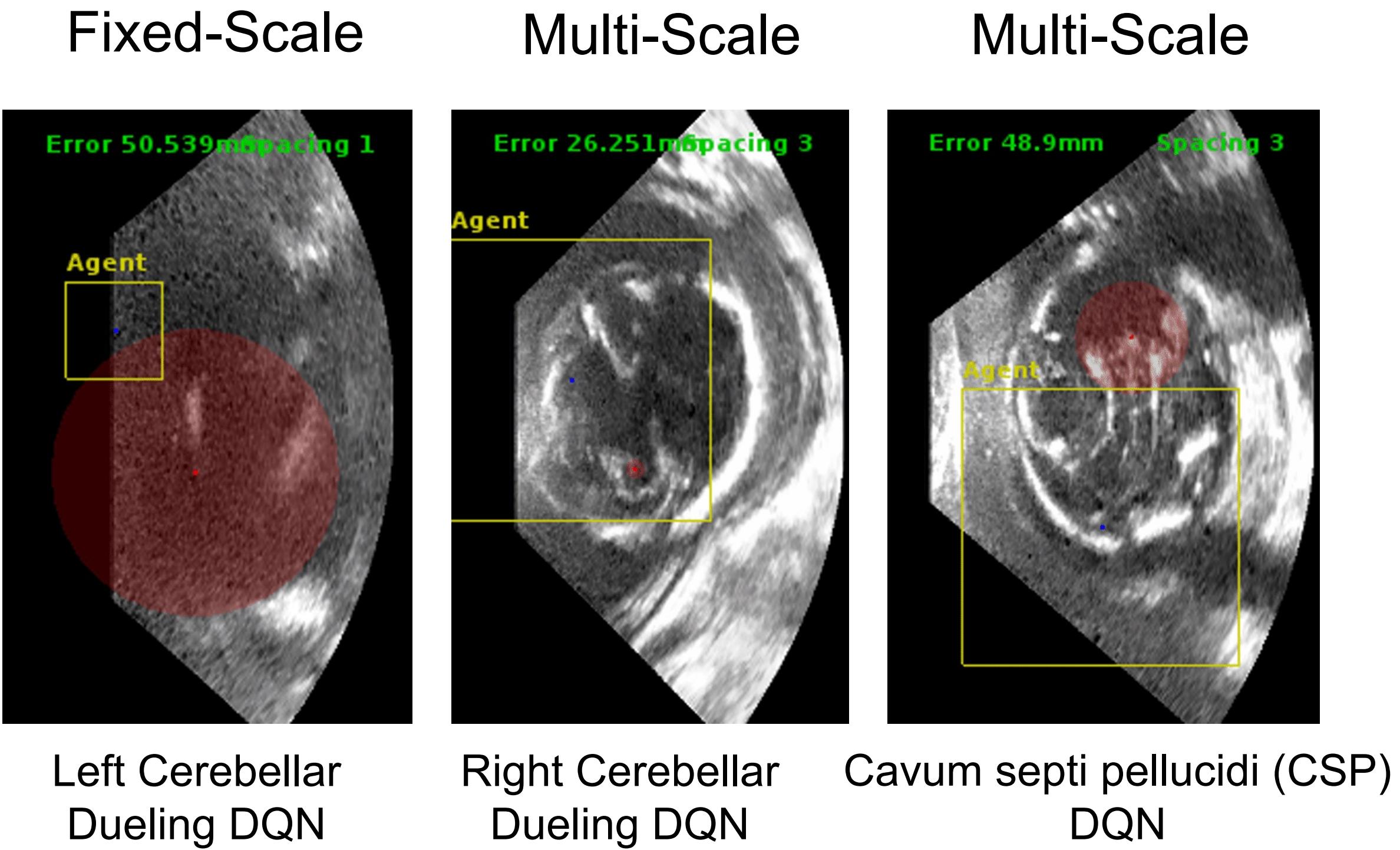
[left, right, up, down, forward, backward]

RL for anatomical landmark detection*

- **Reward function** R , design can be difficult due to danger of overfitting.
 - Here: $R = D(P_{i-1}, P_t) - D(P_i, P_t)$, where D : Euclidean distance between two points;
 P_i : current position at step i ; P_t : target ground truth landmark's location
- **Action space** is simply:
 - Right, left, forward, backward up, down, terminal
- **Terminal state**:
 - Target landmark has been found, ie distances between current estimated and ground truth parameters are less than a threshold T_θ
- **Multi-scale agent** (*Ghesu et al. 2017*)
 - Captures spatial relations within a global neighborhood, in a coarse-to-fine fashion
 - Coarser levels enables the agent to see more structural information
 - Finer scales provide more precise adjustments for the final estimation
 - Requires hierarchical action steps
 - Larger steps speed convergence, smaller steps fine-tune the final estimation

RL for anatomical landmark detection*

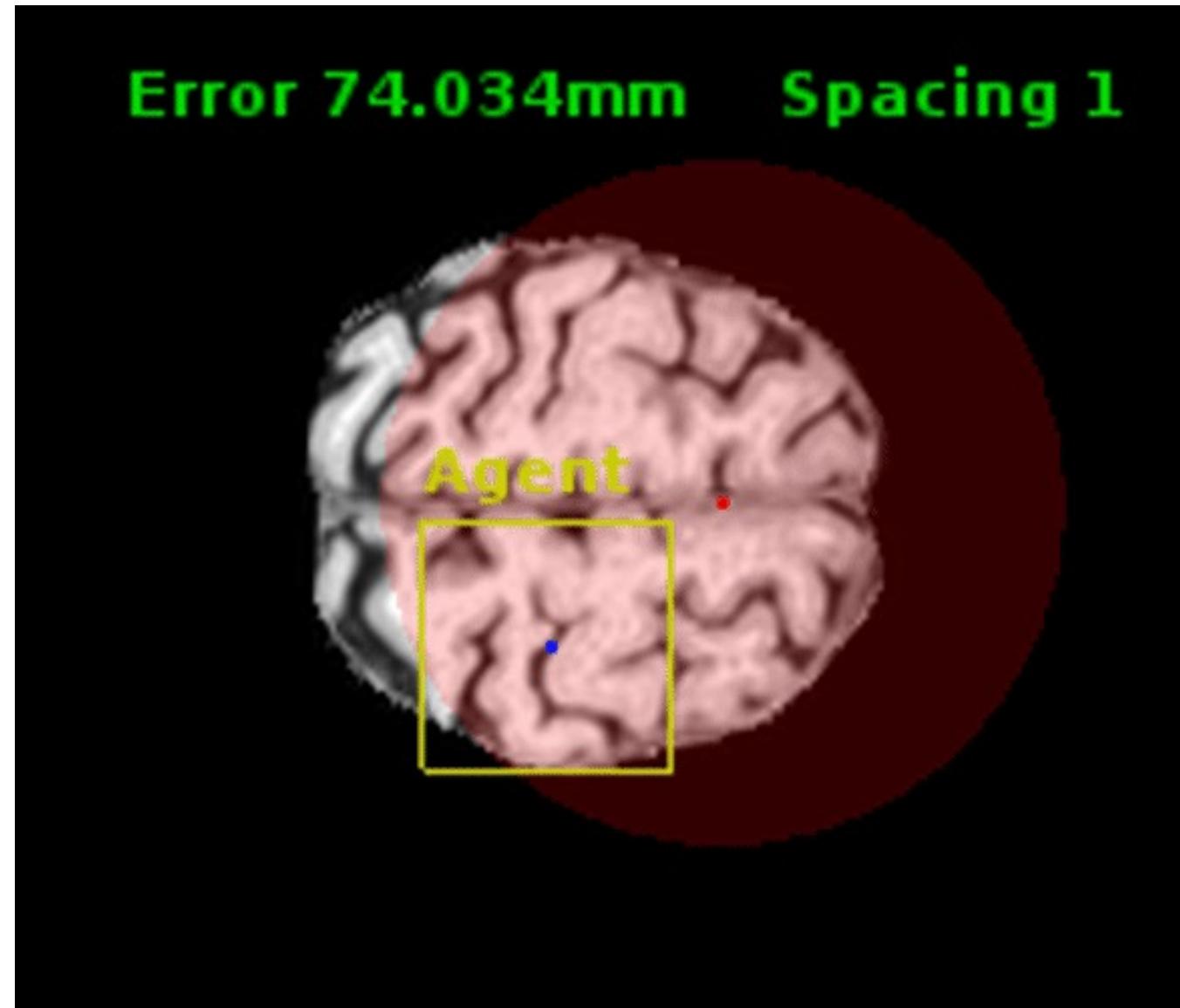
- **Fetal head ultrasound:** challenging task because of the shadowing, mirror images, refraction, and fetal motion



RL for anatomical landmark detection*

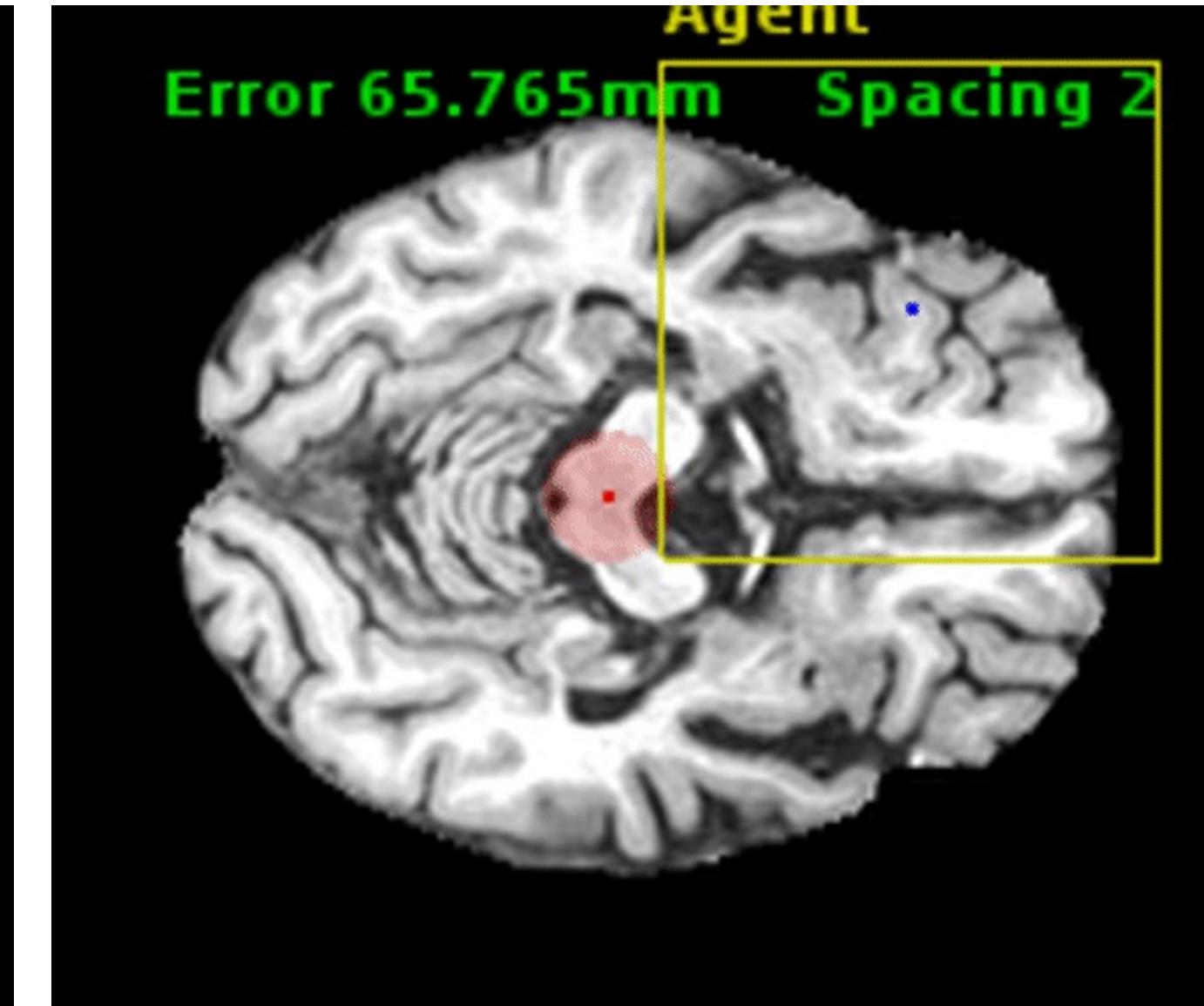
- **Brain MRI:** Anterior and posterior commissure (AC - PC) commonly used by the neuroimaging community to define the axial plane during image acquisition

Fixed-Scale

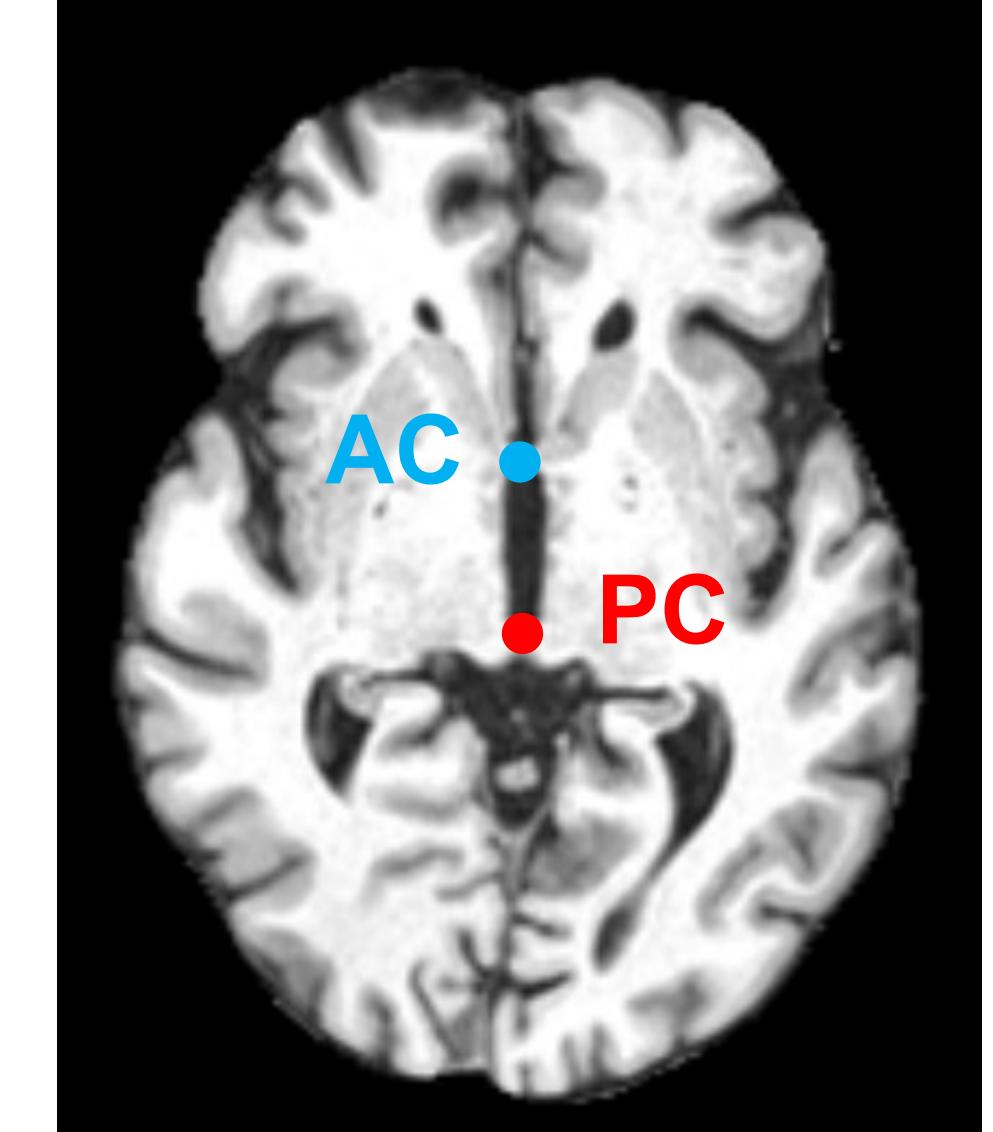


AC - DuelDoubleDQN

Multi-Scale

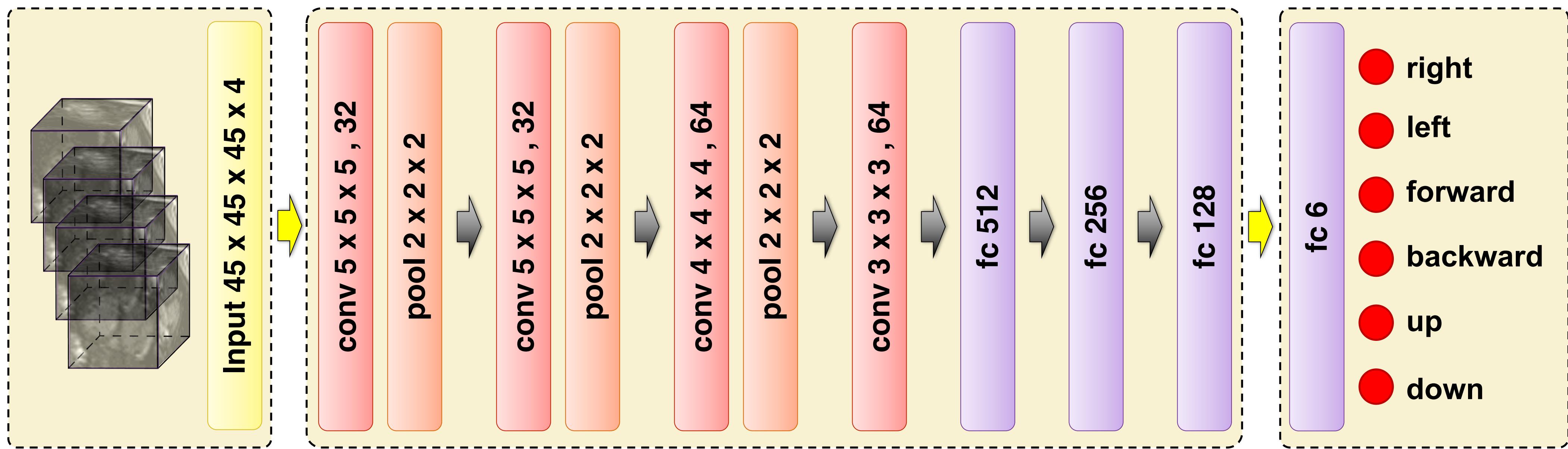


PC – Double DQN



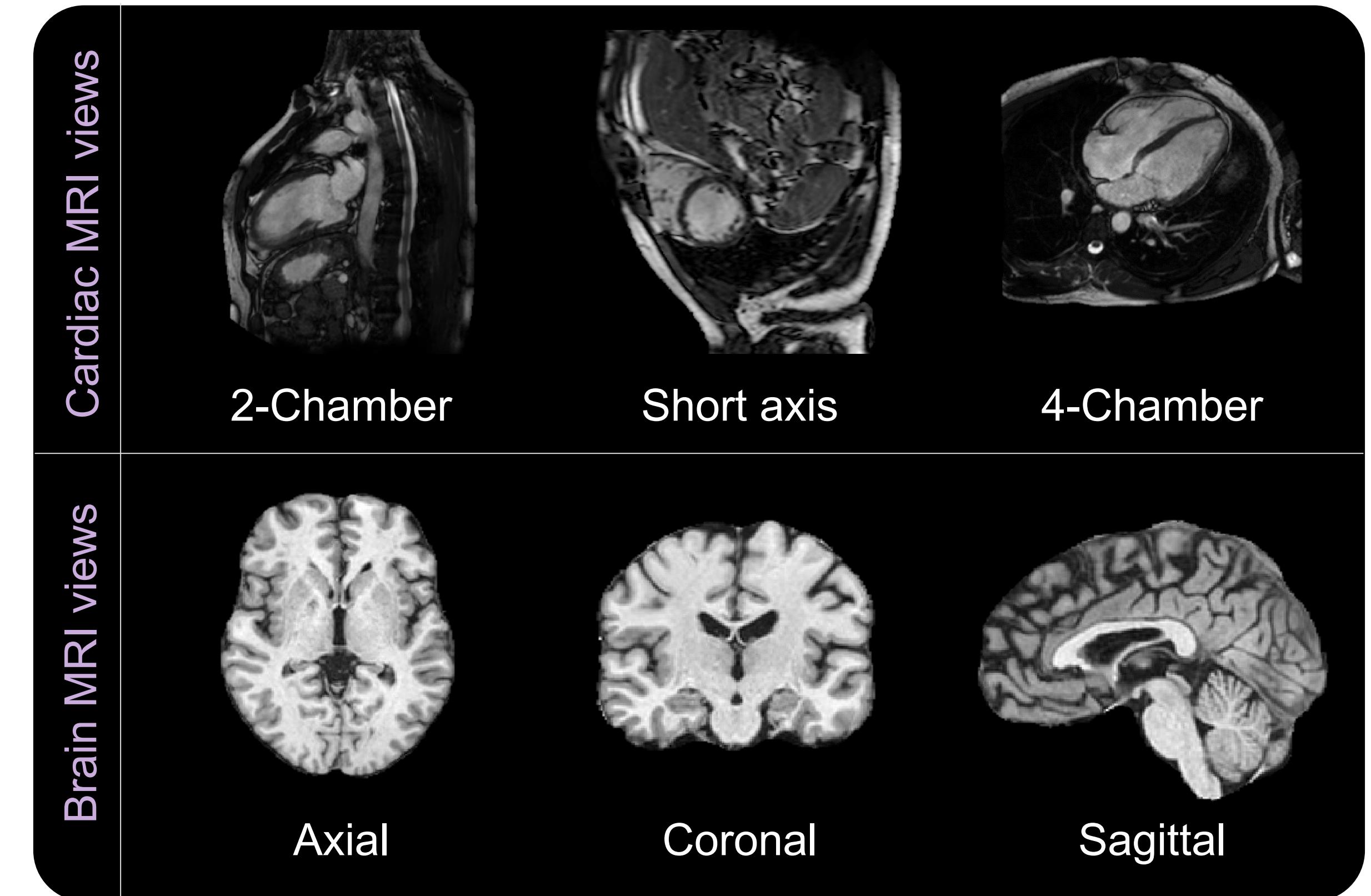
RL for anatomical landmark detection*

- Navigation actions are based on the estimated Q-values from the output of DQN



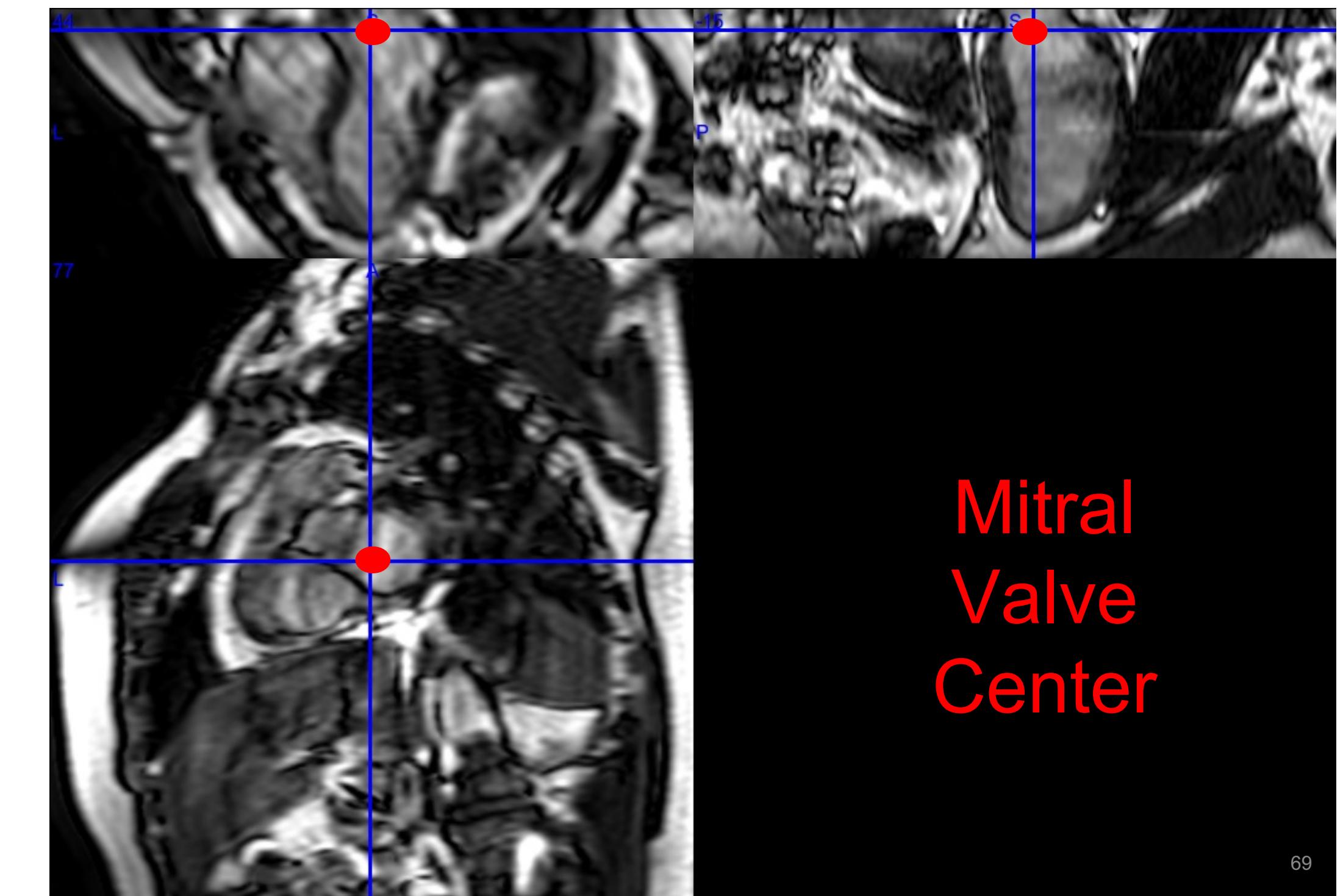
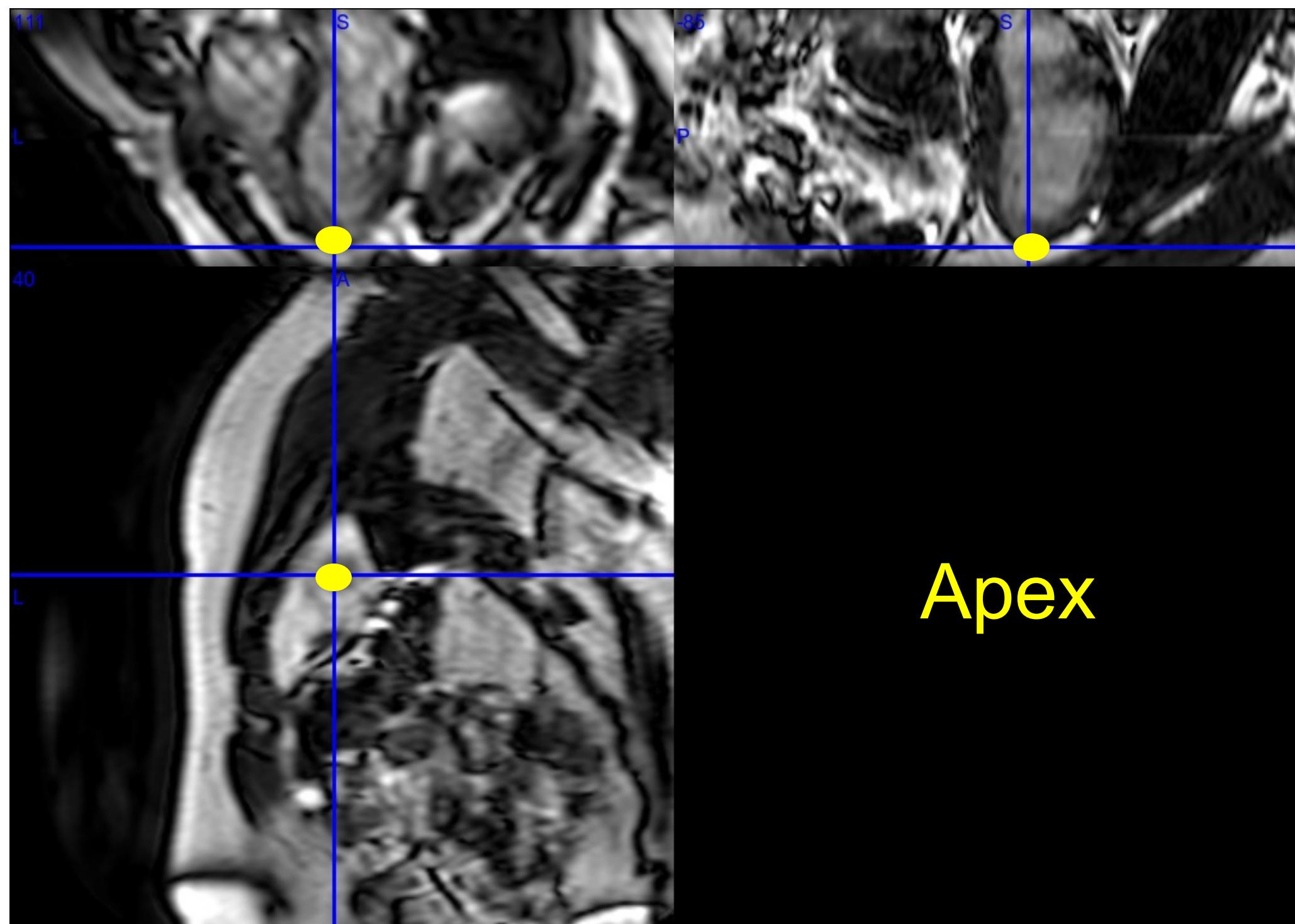
RL for automated view planning*

- Commonly used in clinical practice to establish comparable metrics
- Essential for diagnosis and supporting critical decision making
- Used to initialize image registration methods, or to evaluate and assess anomalies



RL for automated view planning*

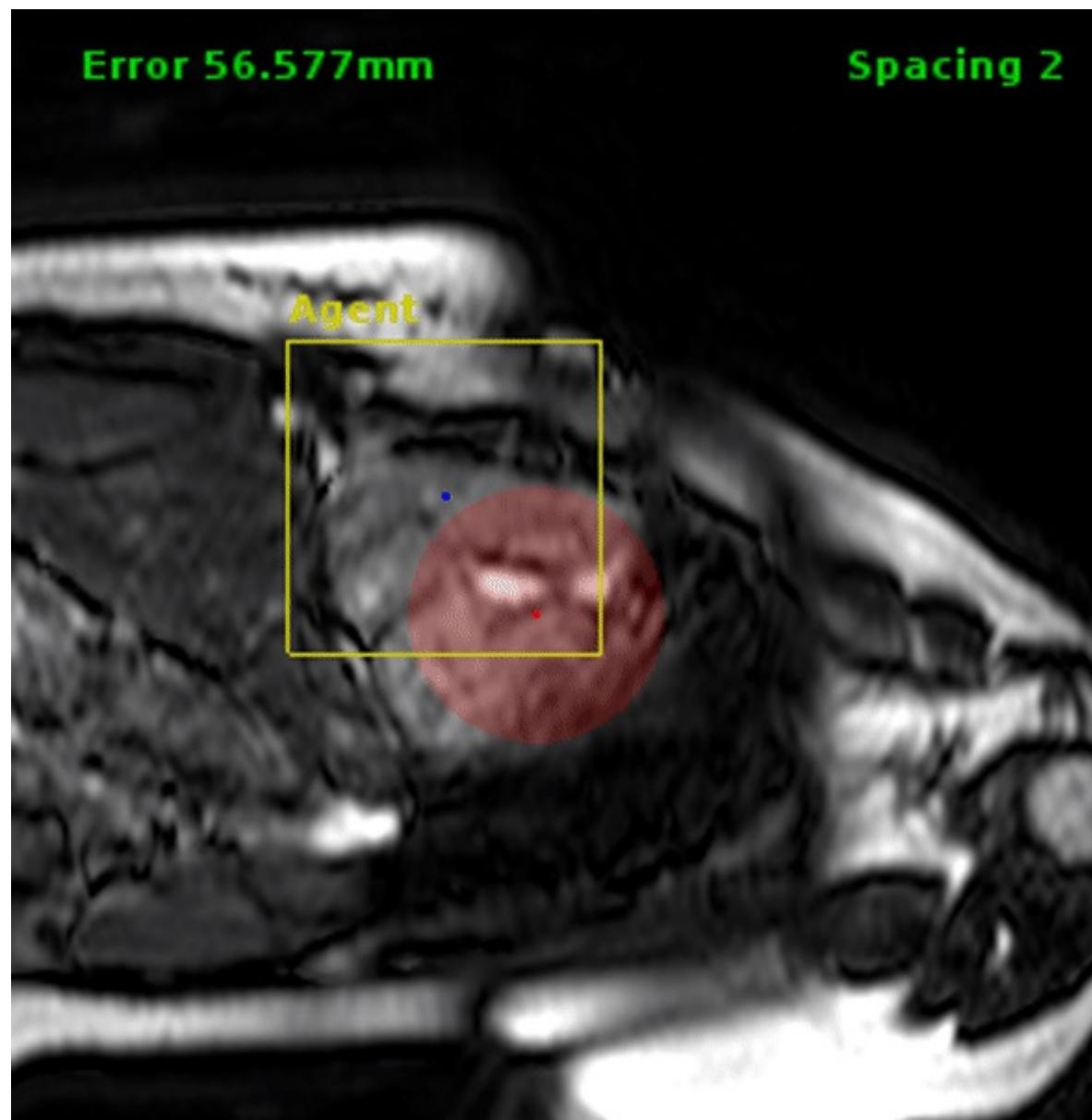
- **Cardiac MRI:** Apex and center of mitral valve, commonly used for defining the short axis view (SAX) during image acquisitions
 - Also used to assist automatic segmentation methods by defining starting and ending slices in the acquired cardiac stack of 2D image sequence



RL for automated view planning*

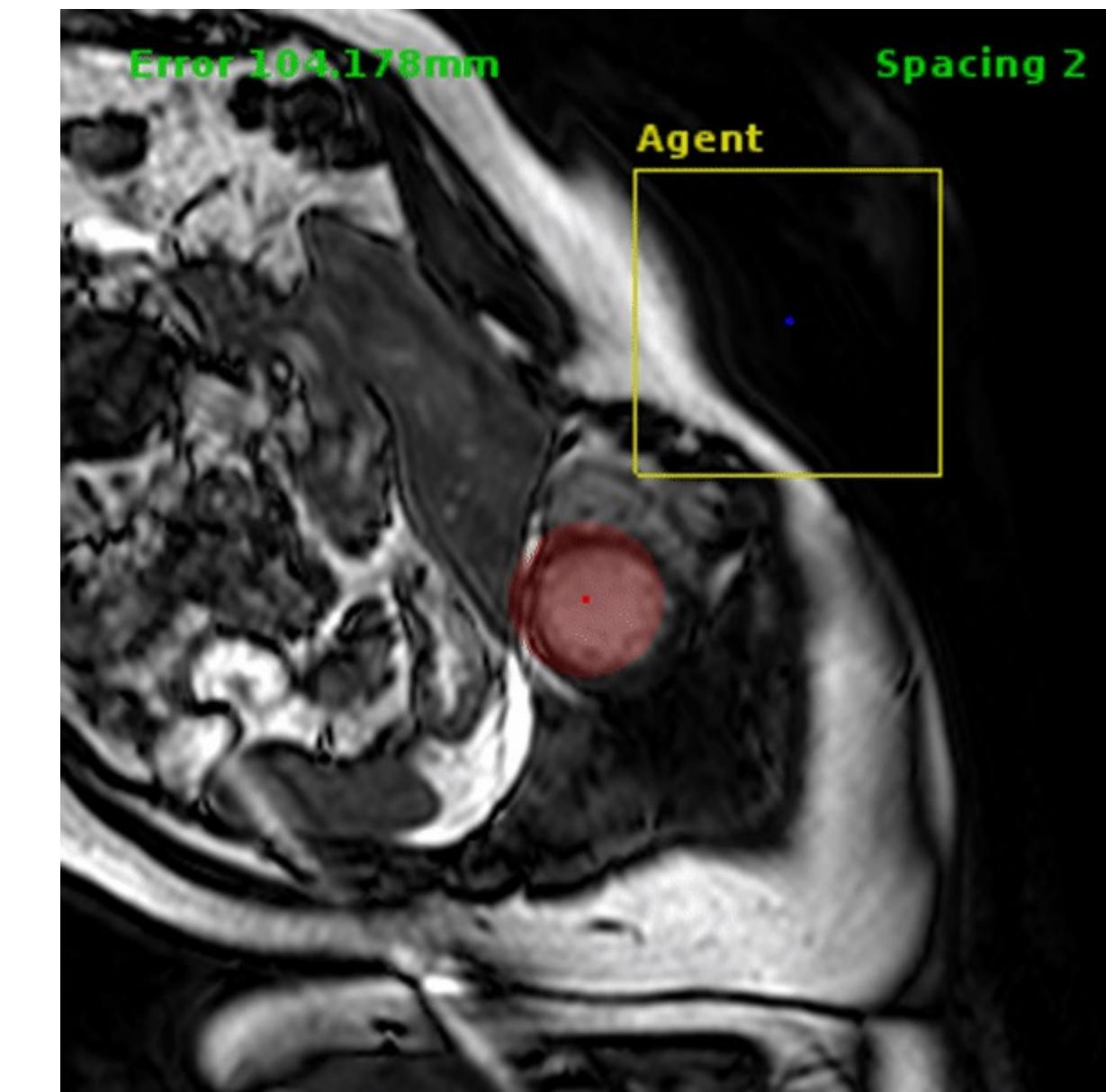
- **Cardiac MRI:** Apex and center of mitral valve, commonly used for defining the short axis view (SAX) during image acquisitions

Multi-Scale



Mitral Double DQN

Multi-Scale

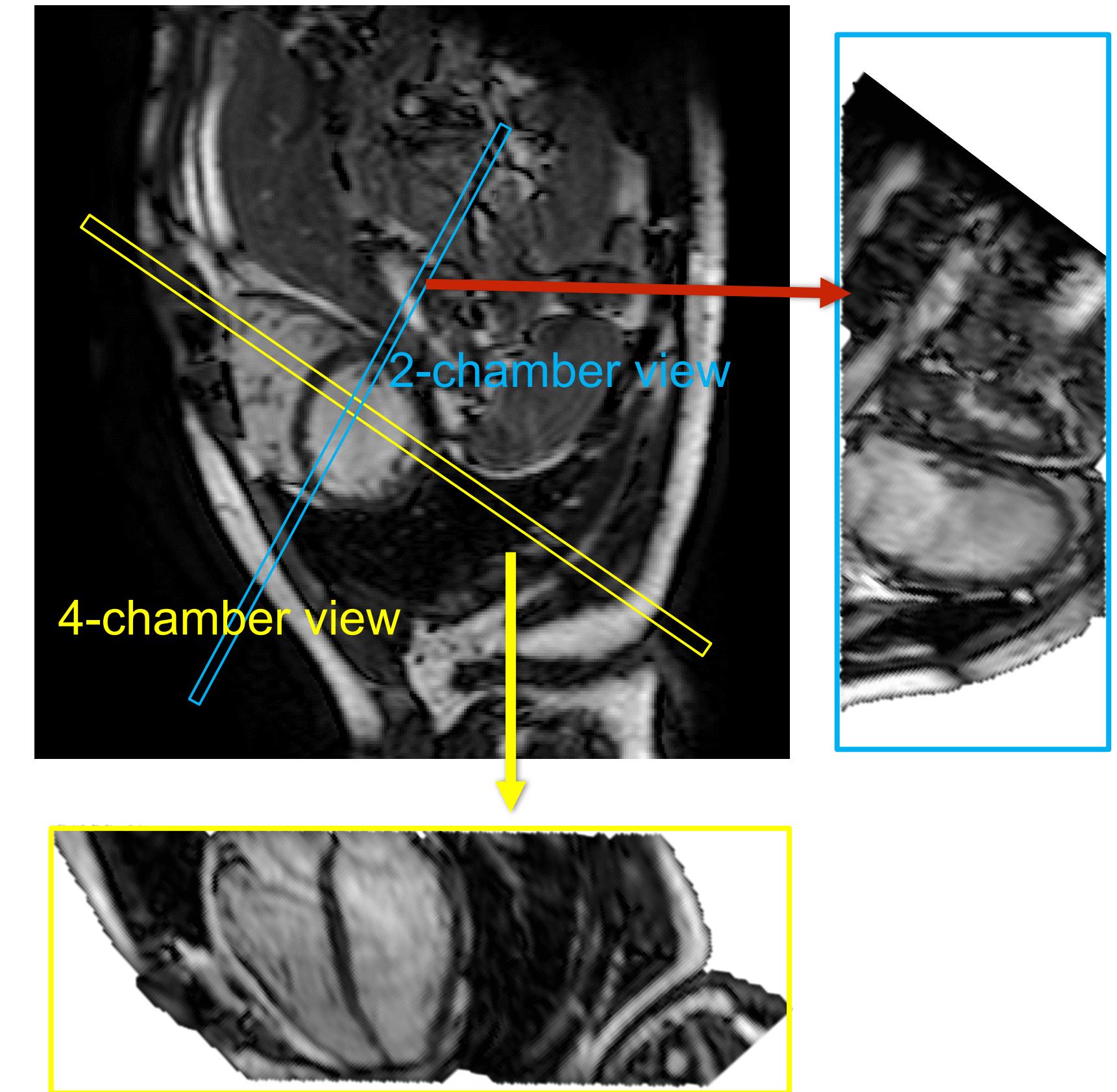


Apex Dueling DQN

RL for automated view planning*

Cardiac MRI: 4-Chamber view in 5 steps

1. Localize 3 planes: axial, coronal, and sagittal
2. Acquire axial stack - above the aortic arch to below the level of the heart
3. Define 2-chamber (2CH) view plane:
 - Perpendicular to axial plane
 - Parallel to interventricular septum (IVS)
 - In the middle of left ventricle (LV)



RL for automated view planning*

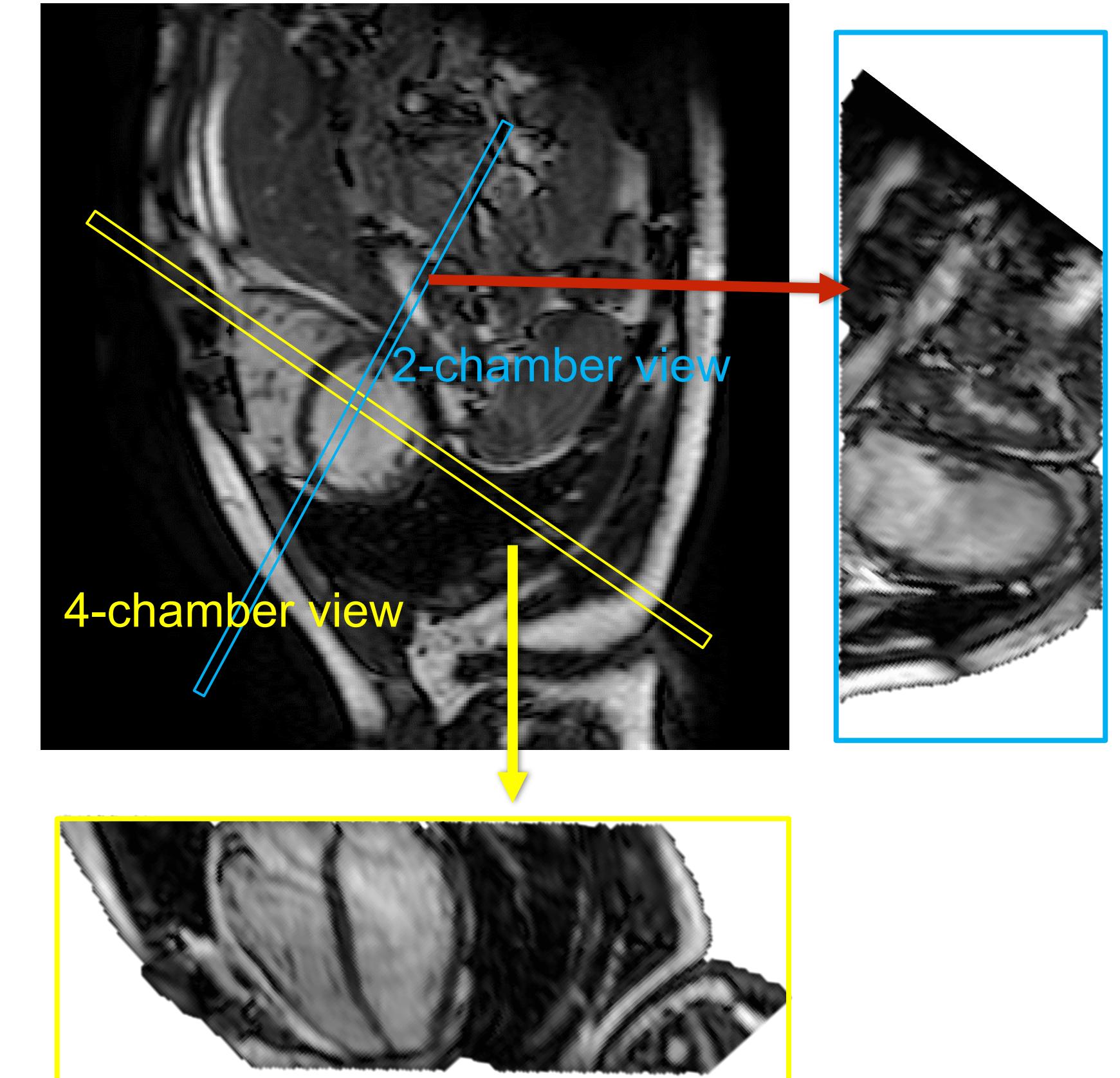
Cardiac MRI: 4-Chamber view in 5 steps

4. Define pseudo short axis (SAX) view plane:

- Perpendicular to the 2CH view
- Align with mitral valve

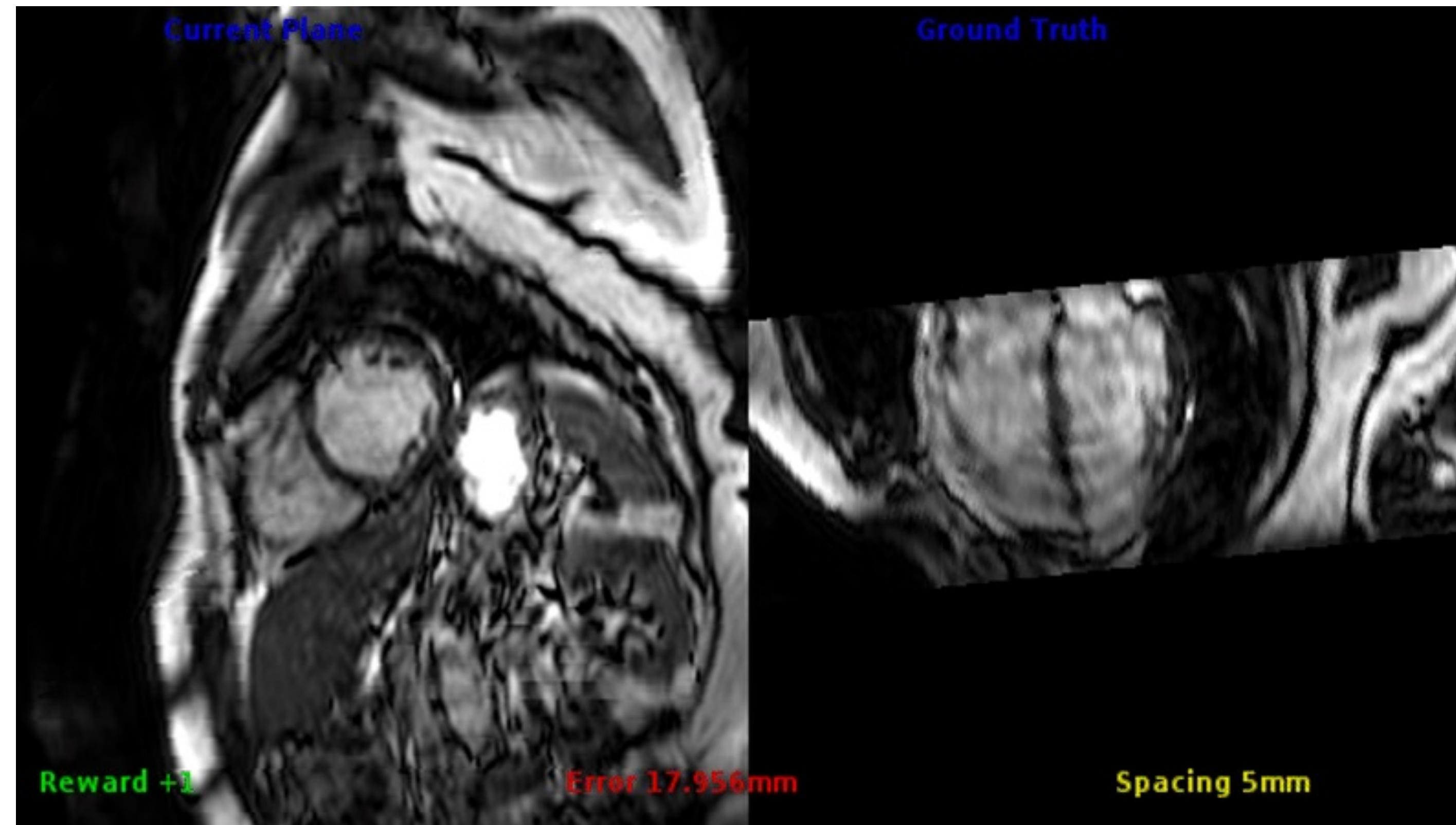
5. Define 4-chamber (4CH) view plane:

- In SAX view:
 - Perpendicular to SA
 - Goes through the center LV and the intersection of interior and inferior of the free wall
- In 2CH view:
 - Divides the heart along the long axis



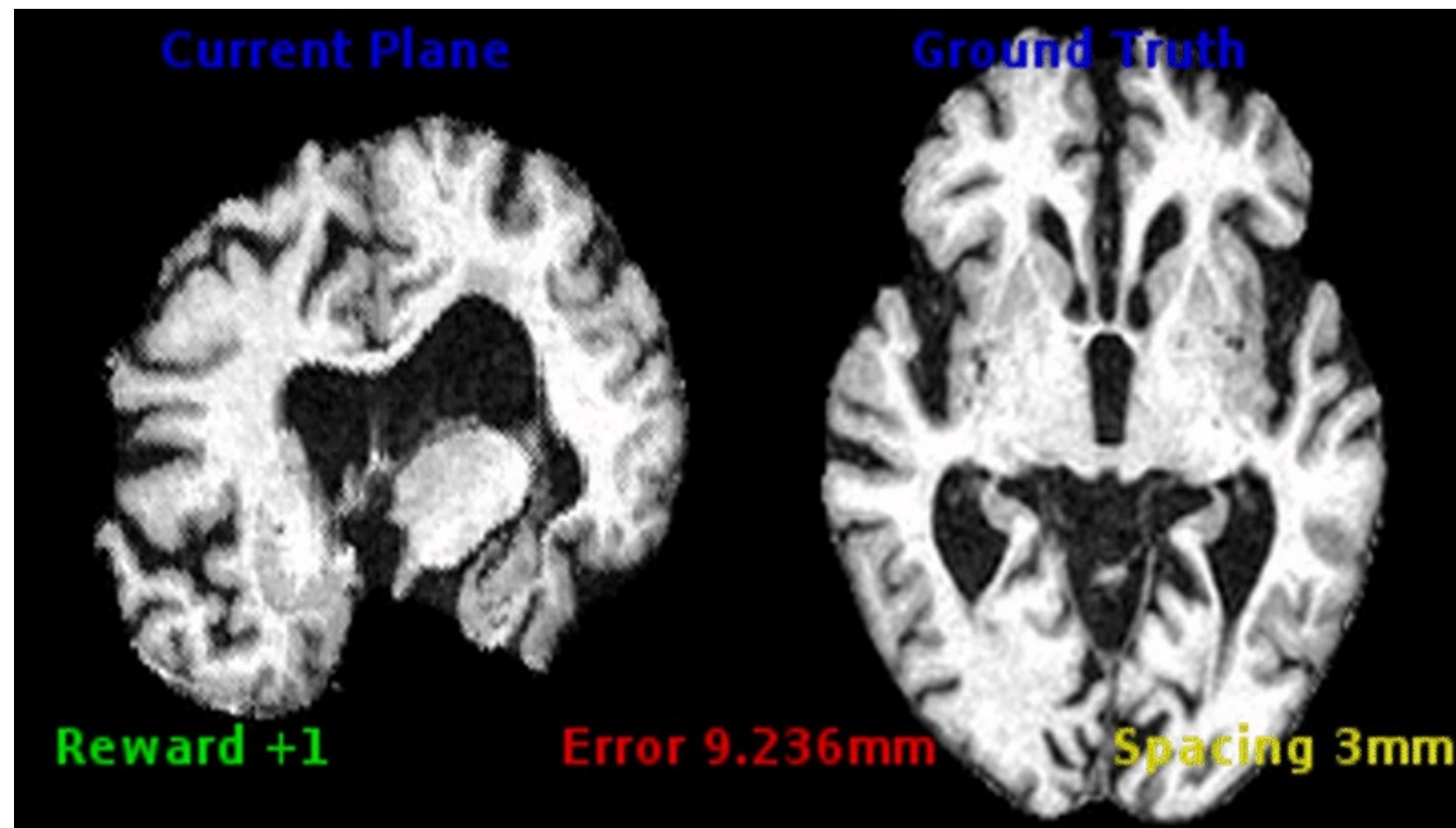
RL for automated view planning*

- Cardiac MRI: 4-Chamber view in 5 steps

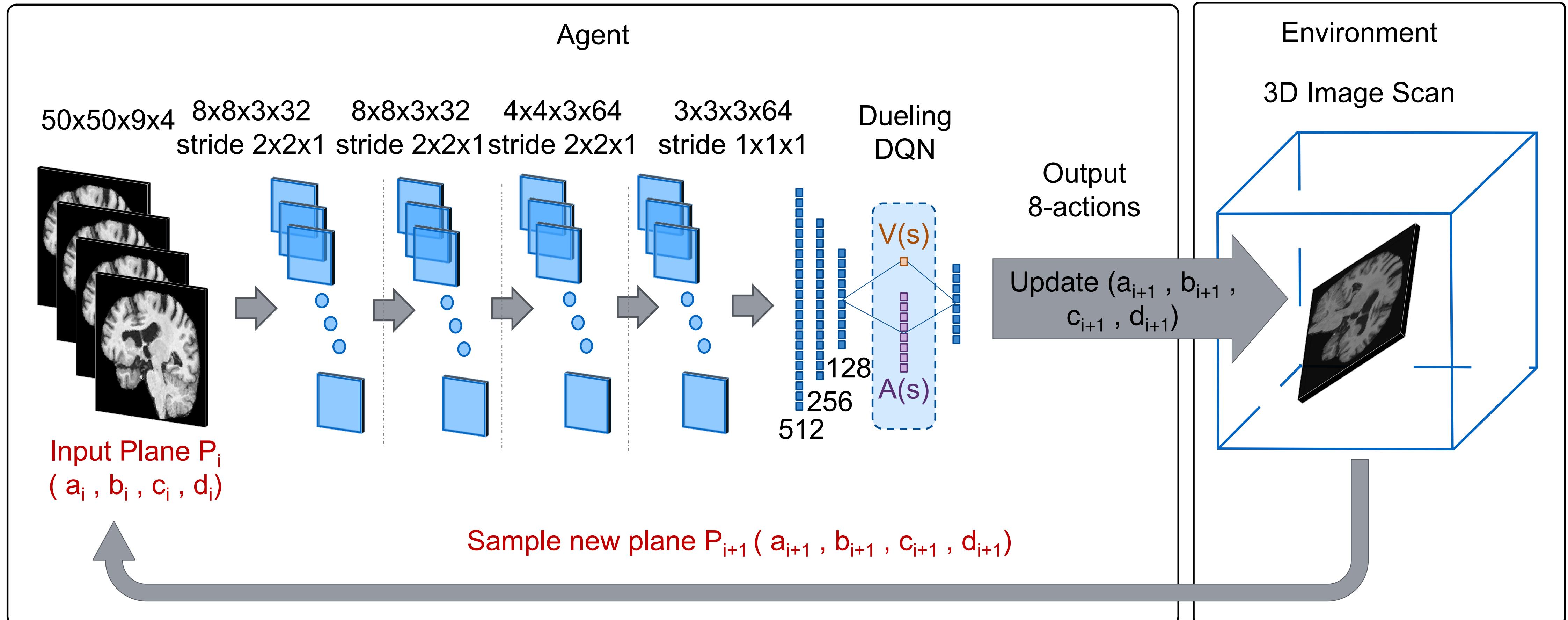


RL for automated view planning*

- Brain MRI: AC-PC Axial Plane



RL for automated view planning*



RL for automated view planning*



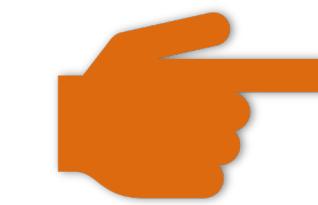
Agent

RL-agent learns to navigate in an environment by sampling new planes towards the target plane using discrete action-steps



States

Sampled 3D planes $[a^*x + b^*y + c^*z + d]$ from the input 3D image scan



Navigation actions

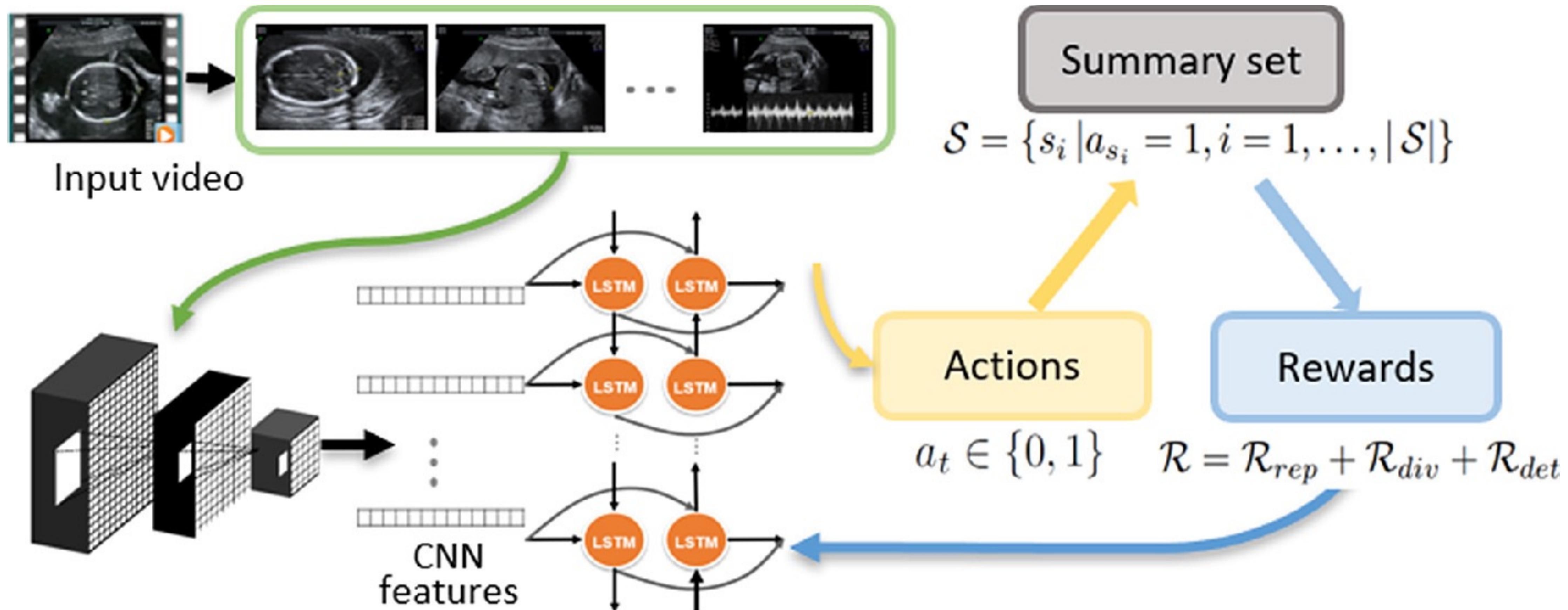
At every step, the agent selects an action to update plane parameters for the next plane
 $\{\pm a_{\theta x}, \pm a_{\theta y}, \pm a_{\theta z}, \pm a_d\}$

Reward Function

$$R = \text{sgn}(D(P_{i-1}, P_t) - D(P_i, P_t))$$

D : Euclidean distance between plane parameters
 P_i : Current plane at step i
 P_t : Target ground truth plane

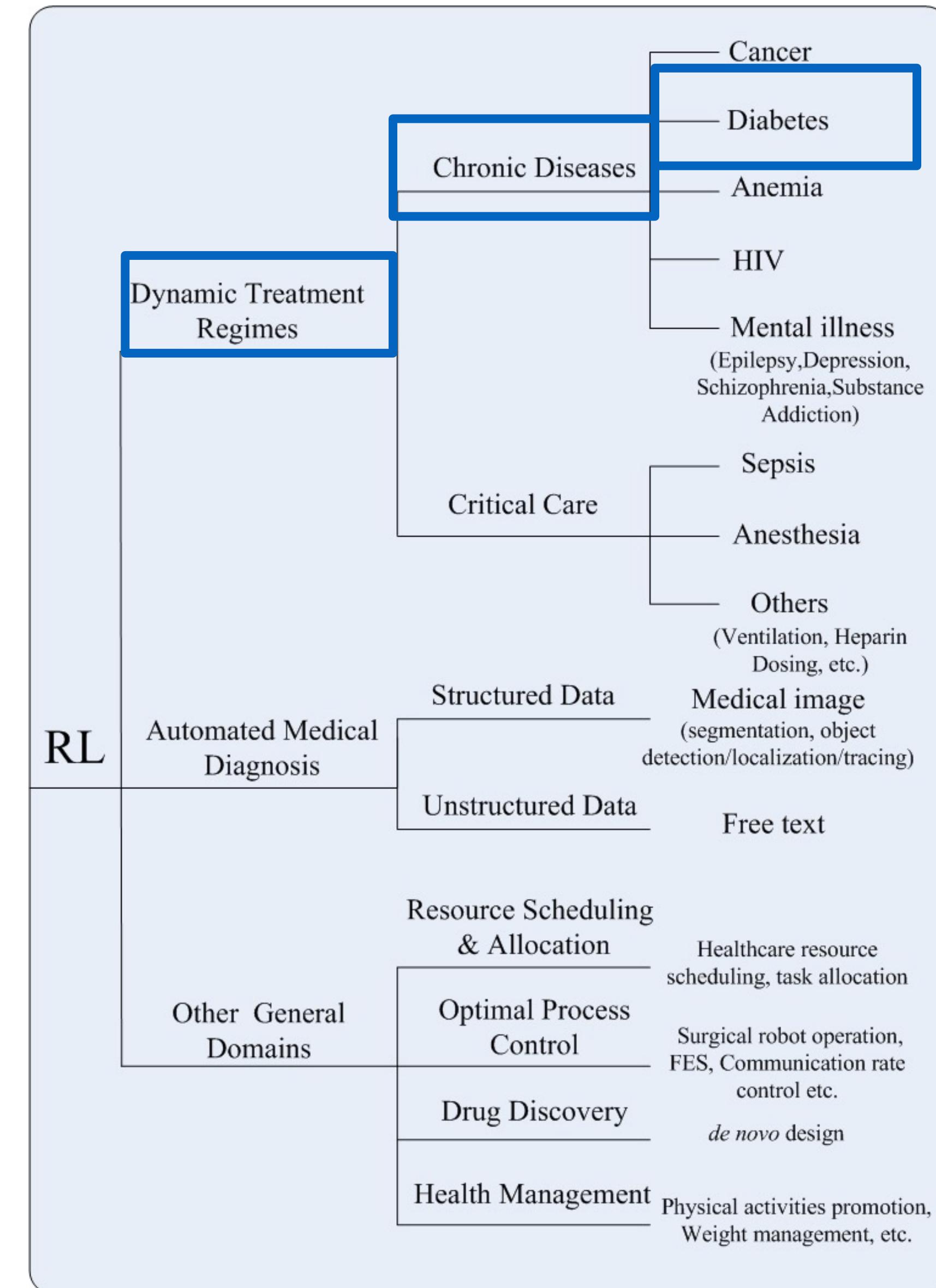
RL for fetal ultrasound summarization*



*Liu, T., Meng, Q., Vlontzos, A., Tan, J., Rueckert, D., Kainz, B. (2020). MICCAI 2020.

https://doi.org/10.1007/978-3-030-59716-0_46

RL in healthcare applications

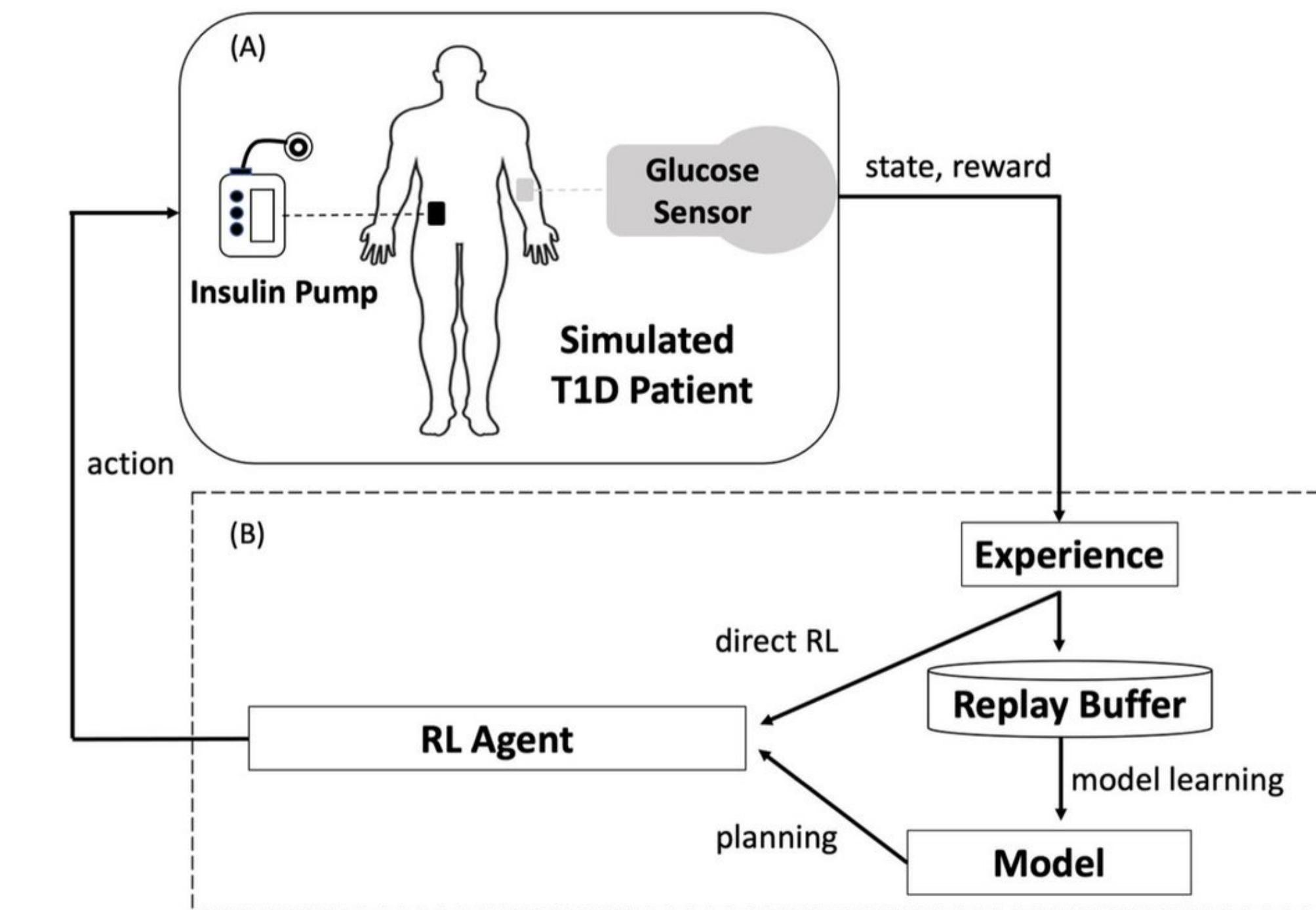


Reinforcement Learning in Healthcare: A Survey
You et al., ACM Computing Surveys (2023)
<https://doi.org/10.1145/3477600>
<https://arxiv.org/pdf/1908.08796.pdf>

RL for diabetes management

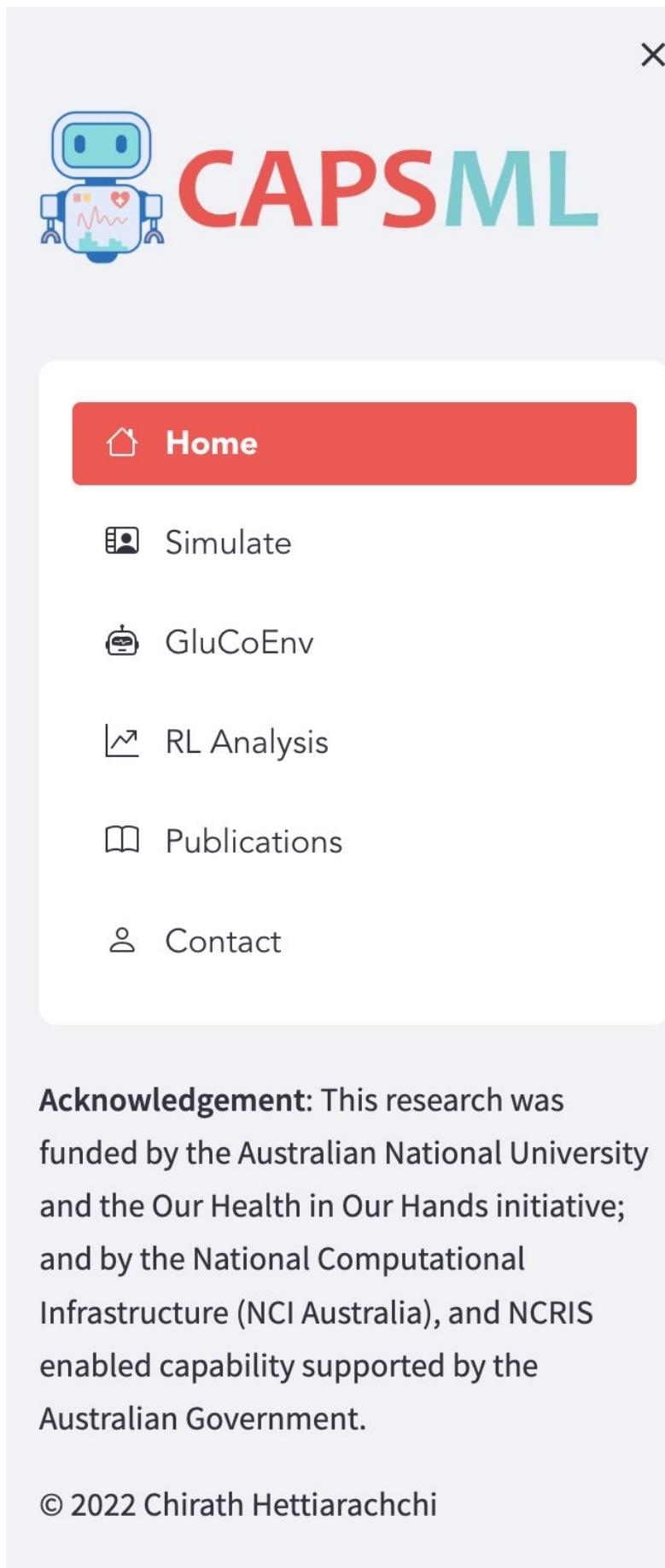
- Blood glucose needs to be ideally maintained, while any meals consumed fluctuate the glucose levels
- RL to control blood glucose levels by making insulin infusion decisions to the body:
 - During training, RL agent is only given a reward based on the glucose value and no specific instructions related to insulin treatment.
 - RL agent fails at the very beginning, resulting in severe hypoglycemia, and the simulation ends quickly (death).
 - With experience (more training), it learns to make better insulin infusion decisions.
 - Finally, agent learns a strategy close to a clinical strategy we see in real life (utilising a basal insulin rate and bolus insulin).

G2P2C: Glucose Control by Glucose Prediction and Planning



https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4226648

RL for diabetes management

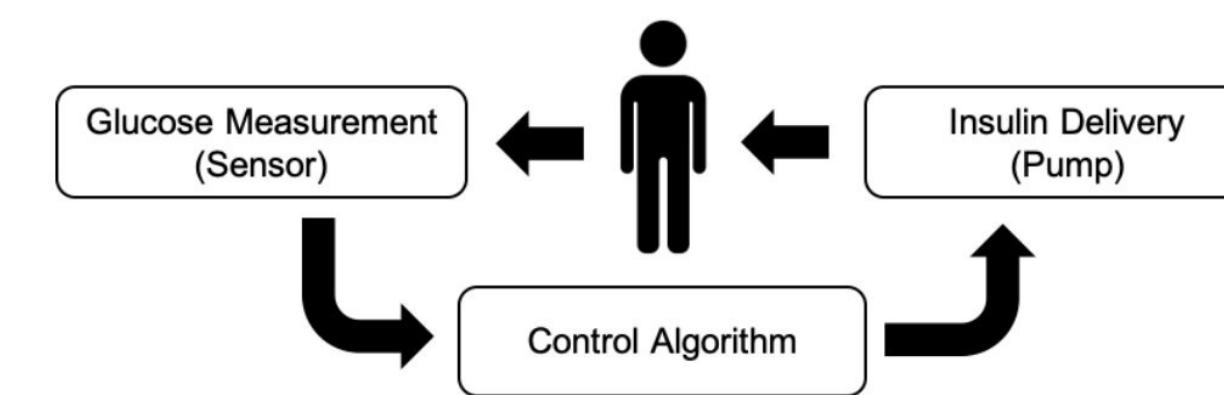


The screenshot shows the CAPSML website. At the top is a logo of a blue robot with a heart and ECG lines, next to the text "CAPSML". Below the logo is a navigation bar with a red "Home" button and other links: "Simulate", "GluCoEnv", "RL Analysis", "Publications", and "Contact". A sidebar on the left contains an "Acknowledgement" section and a copyright notice. The acknowledgement text reads: "Acknowledgement: This research was funded by the Australian National University and the Our Health in Our Hands initiative; and by the National Computational Infrastructure (NCI Australia), and NCRIS enabled capability supported by the Australian Government." The copyright notice at the bottom says "© 2022 Chirath Hettiarachchi".

Controlling Artificial Pancreas Systems through Machine Learning

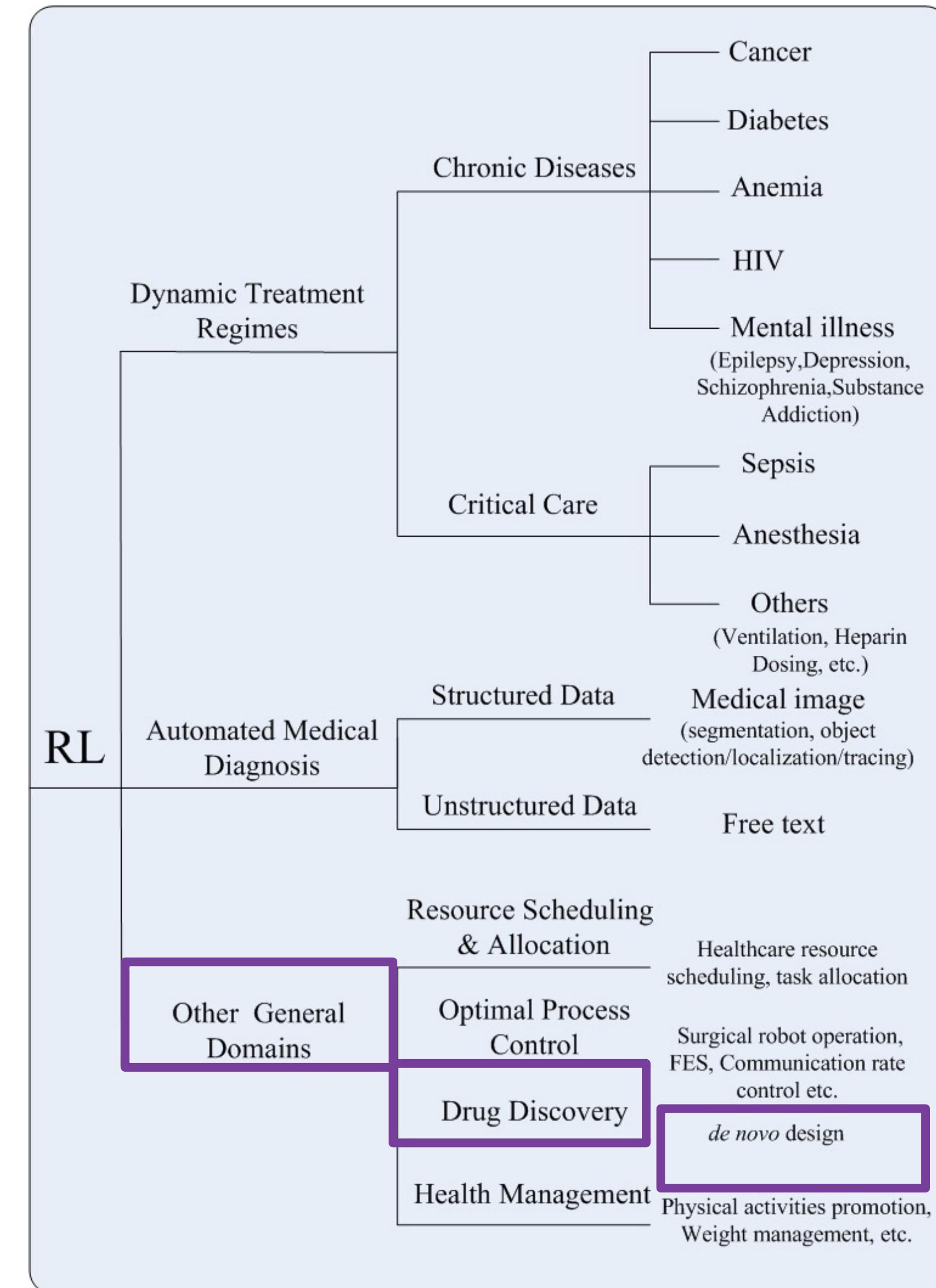
Abstract

Type 1 Diabetes (T1D) requires the administration of insulin externally to maintain glucose levels, which is crucial as both low and high glucose levels are detrimental. This is usually done through an insulin pump attached to the body. A continuous glucose sensor is also attached to measure the glucose levels so that a control algorithm can estimate the appropriate insulin dose. I design Reinforcement Learning (RL) algorithms for this control problem. The figure below summarises the main components of an Artificial Pancreas System (APS) to treat T1D.



Maintaining glucose levels is a life-long optimisation problem, complicated due to the disturbances associated with daily events (meals, exercise, stress.. etc), delays present in glucose sensing and insulin action, partial observability, and safety constraints among others. On the right you can see a simulated glucose control strategy of a RL algorithm.

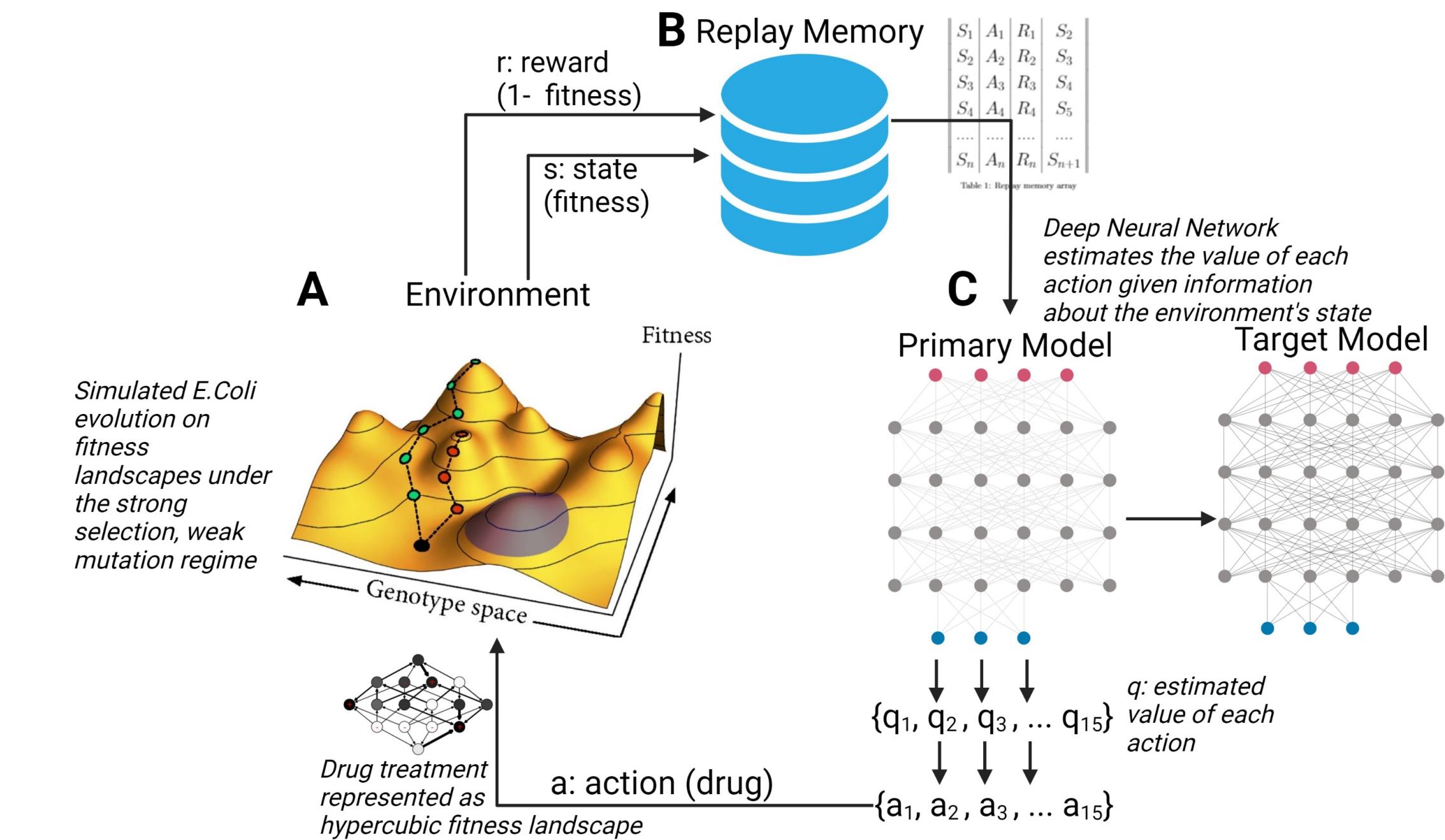
RL in healthcare applications



Reinforcement Learning in Healthcare: A Survey
You et al., ACM Computing Surveys (2023)
<https://doi.org/10.1145/3477600>
<https://arxiv.org/pdf/1908.08796.pdf>

RL for developing effective drug cycling policies

- RL framework capable of controlling an evolving population of E. Coli in silico
- RL agents stably learn multi-drug combinations that were state-specific and reliably outperformed a random drug cycling policy as well as all possible two-drug cycling policies
- Proof-of-concept for AI-based **evolutionary control**, an emerging field with the potential to revolutionize clinical medicine.



Schematic of artificial intelligence system for controlling evolving cell populations.
<https://www.biorxiv.org/content/10.1101/2023.01.12.523765v1>

Summary

- Reinforcement learning is a computational approach for goal-directed learning and decision making.
- Specifically, it puts emphasis on learning by an agent from direct interaction with its environment.
- It does **not** require supervision nor complete models of the environment.

← [Tweet](#) ...



Jeffrey Li 💙💛
@askerlee

Why machine learning people are more pessimistic than reinforcement learning people?
Because machine learning people are dealing with losses, and reinforcement learning people are dealing with rewards.

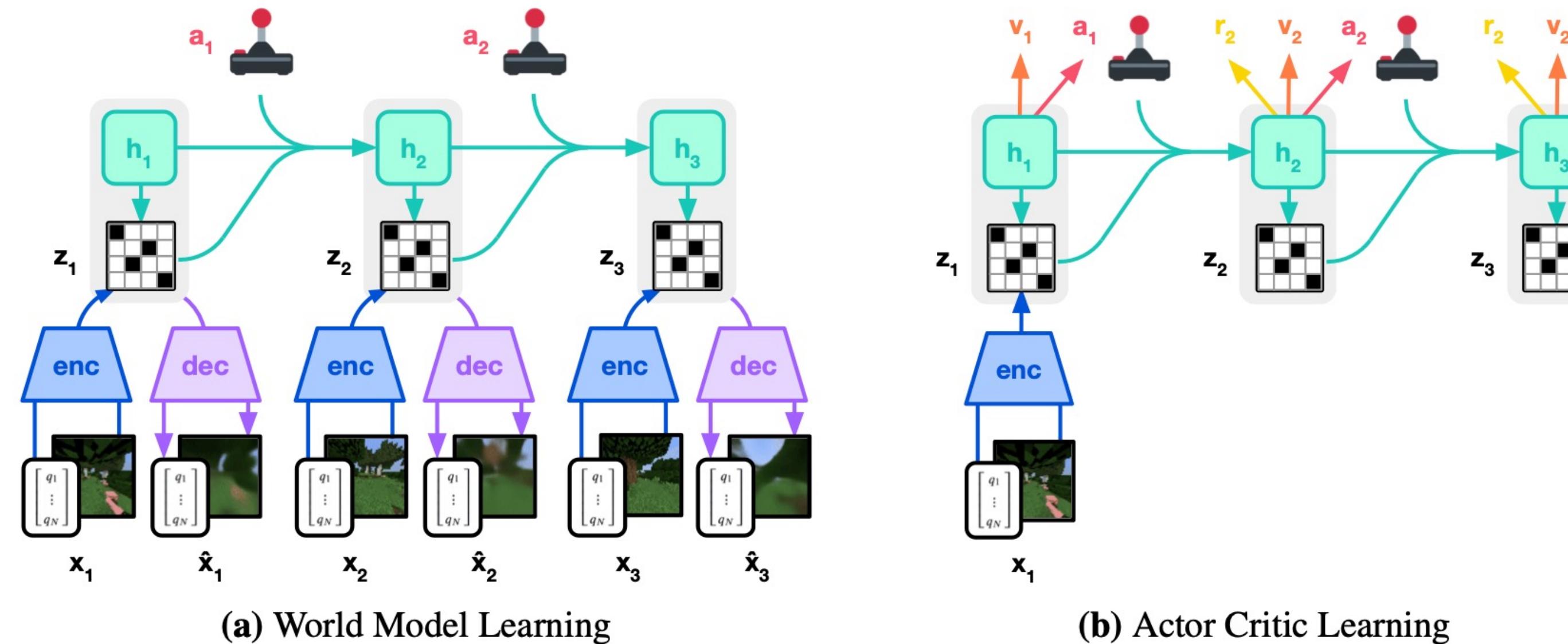
14:27 · 13/01/2023 · 29.7K Views

Beyond this lecture but worth noting

- Multi-agent RL
 - multiple learning agents in a shared environment, each motivated by own rewards, performing its own actions. May lead to opposing interests and complex group dynamic.
- Model-free RL
 - agent learns to take actions directly from experiences, without ever modelling transitions in the environment or reward functions
- Model-based RL
 - agent learns to predict the environment ‘in its head’, so it no longer needs to interact directly with the environment in order to learn an optimal policy.
- RL for Transformers, and Transformers for RL
 - Transformers coming up in AIM-2

Hot off the press: DeepMind DreamerV3*

- General and scalable RL algorithm that masters a wide range of domains with fixed hyperparameters.



- First algorithm to obtain diamonds in Minecraft end-to-end from sparse rewards

* <https://arxiv.org/abs/2301.04104v1>

Last not least: ChatGPT and friends

- **RL from Human Feedback (RLHF):**

- use RL to directly optimize a (large) language model with human feedback.

- **Basic method:**

1. Pretrain a language model (LM)
2. Gather data and train a reward model
3. Fine-tune the LM with RL

LE

Can you explain to me how RL with human feedback works for language models like I am five?



Sure! So, imagine you have a robot named Rufus who wants to learn how to talk like a human. Rufus has a language model that helps him understand words and sentences.

First, Rufus will say something using his language model. For example, he might say "I am a robot."

Then, a human will listen to what Rufus said and give him feedback on whether it sounded like a natural sentence a human would say. The human might say, "That's not quite right, Rufus. Humans don't usually say 'I am a robot.' They might say 'I'm a robot' or 'I am a machine.'"

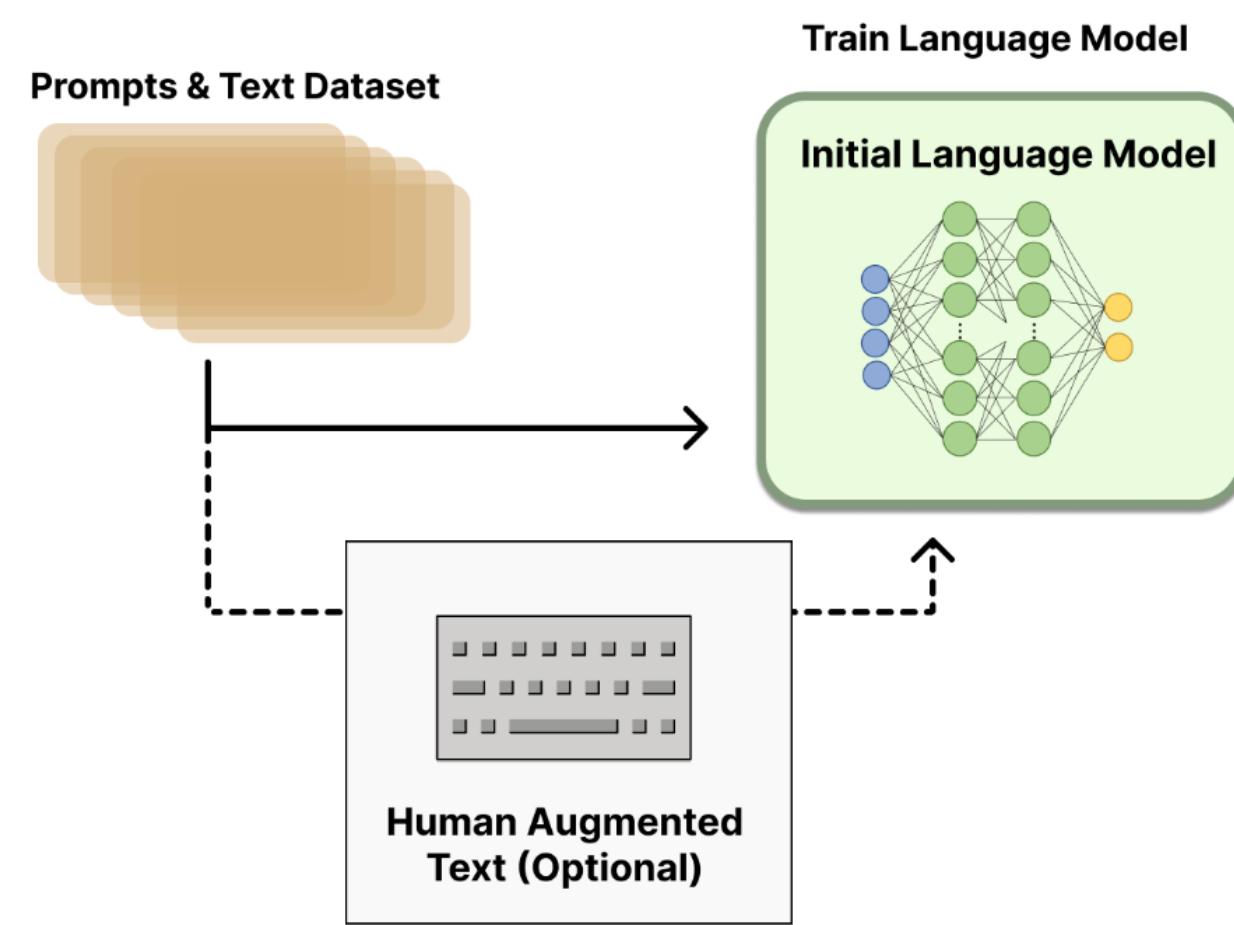
Rufus will take this feedback and use it to update his language model. He will try to say the sentence again, using the new information he received from the human. This time, he might say "I'm a robot."

The human will listen again and give Rufus more feedback. This process will continue until Rufus can say sentences that sound natural to a human.

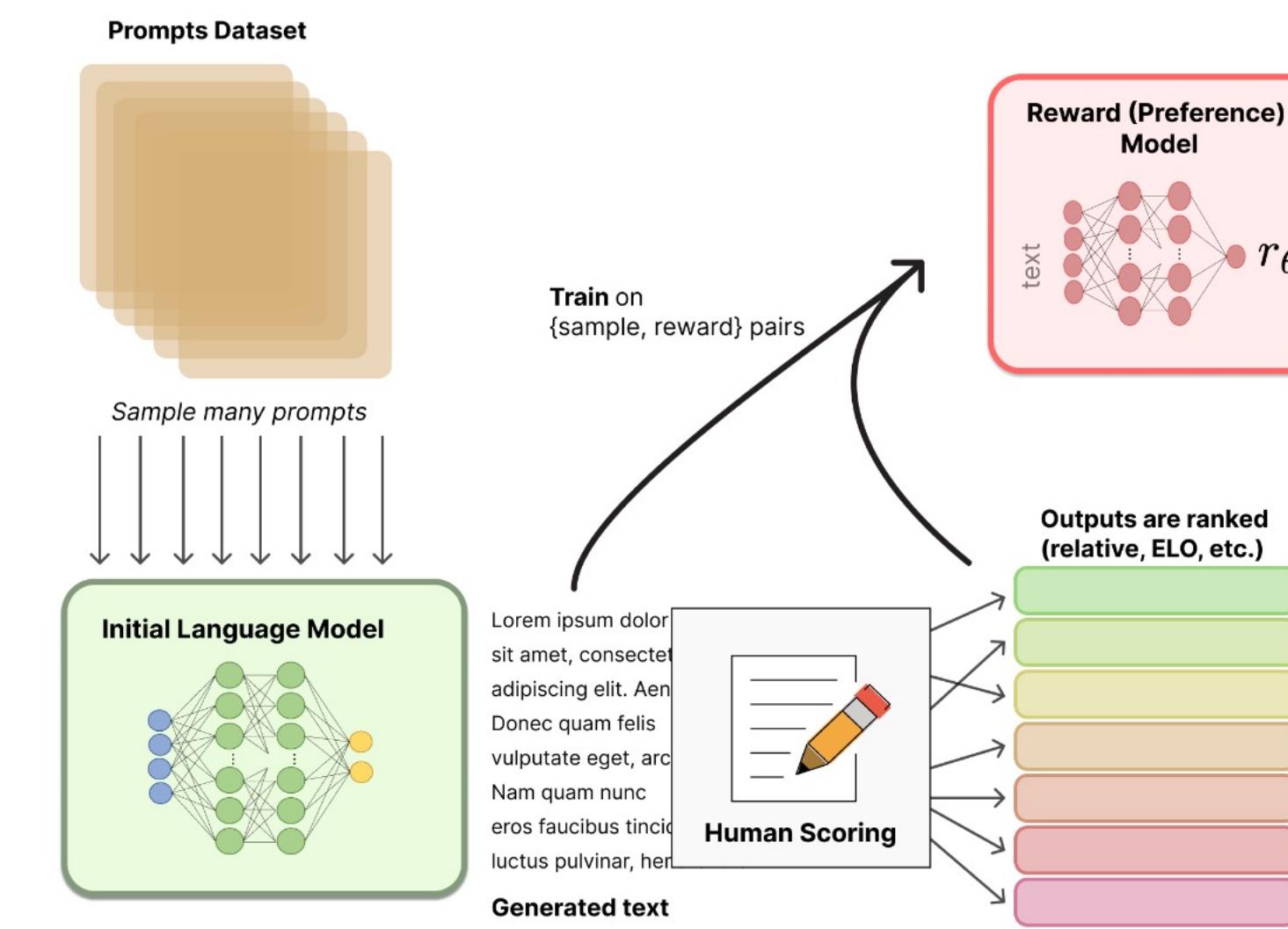
Over time, Rufus will learn how to talk like a human thanks to the feedback he receives from humans. This is how language models can be improved using RL with human feedback.

Last not least: ChatGPT and friends

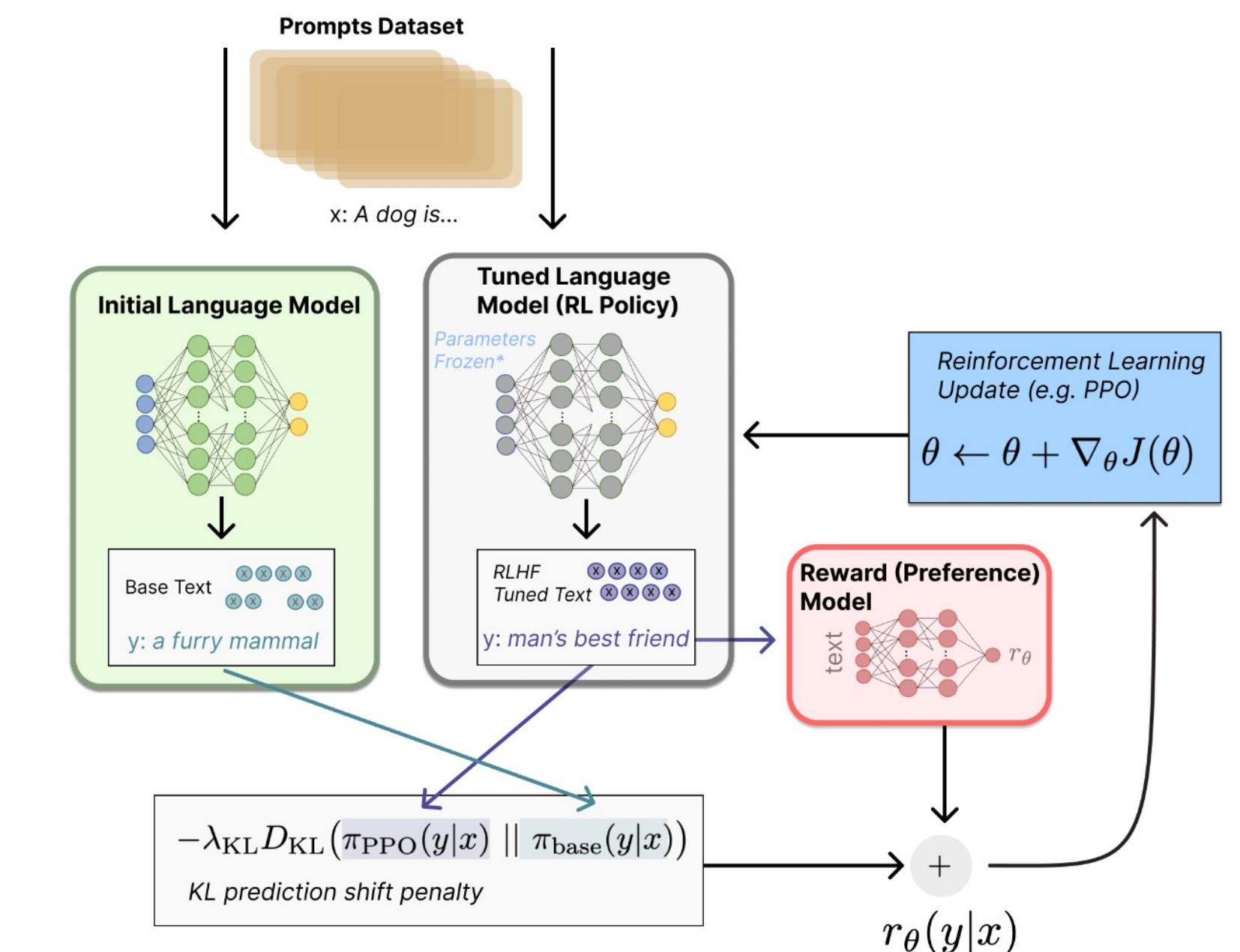
- RLHF:



1. Pretrain a language model



2. Reward model training



3. Fine-tuning with RL

General References

- Reinforcement Learning: An Introduction.
 - R. S. Sutton and A. G. Barto. MIT Press, 1998 (2nd edition 2018)
 - <http://www.incompleteideas.net/book/RLbook2018trimmed.pdf>
- Algorithms for Reinforcement Learning.
 - By C. Szepesvari. Morgan and Claypool, 2010
 - <https://sites.ualberta.ca/~szepesva/RLBook.html>
- Deep Reinforcement Learning: An Overview
 - By Y. Li <https://doi.org/10.48550/arXiv.1701.07274>
- A Succinct Summary of Reinforcement Learning. S. Ahilan.
 - <https://arxiv.org/abs/2301.01379>
 - **Acknowledgements:** I would like to thank Peter Dayan, David Silver, Chris Watkins and ChatGPT for helpful feedback.

References to RL in healthcare

- Deep reinforcement learning in medical imaging: A literature review
 - Zhou et al. MedIA (2021) <https://doi.org/10.1016/j.media.2021.102193>
- Reinforcement Learning in Healthcare: A Survey
 - You et al., ACM Computing Surveys (2023) Preprint: <https://arxiv.org/pdf/1908.08796.pdf>
- Evaluating reinforcement learning agents for anatomical landmark detection
 - Alansary et al. MedIA (2019) <https://doi.org/10.1016/j.media.2019.02.007>

Software references

- <https://www.deepmind.com/open-source/dqn>
- <https://openai.com/blog/openai-baselines-dqn/>
- <https://paperswithcode.com/method/dqn>

