地平线 Horizon Robotics

DEECAMP

A Brief Introduction to Reinforcement Learning

lisen.mu@hobot.cc

- **Recap and Concepts**
- Reinforcement Learning Basics
- Advanced Reinforcement Learning
- Challenges and Approaches

Recap and Concepts

- Machine Learning
- Supervised Learning
- Reinforcement Learning

Machine Learning?

- A computer program is said to learn:
  - from experience **E**
  - with respect to some class of tasks **T**
  - and performance measure **P**
- if its performance at tasks in T, as measured by P, improves with experience E

- Tom M. Mitchell

Supervised Learning Example:
Task A: Image Classification

Correctly associate labels with images: dog, plane, flower, cellphone, etc.

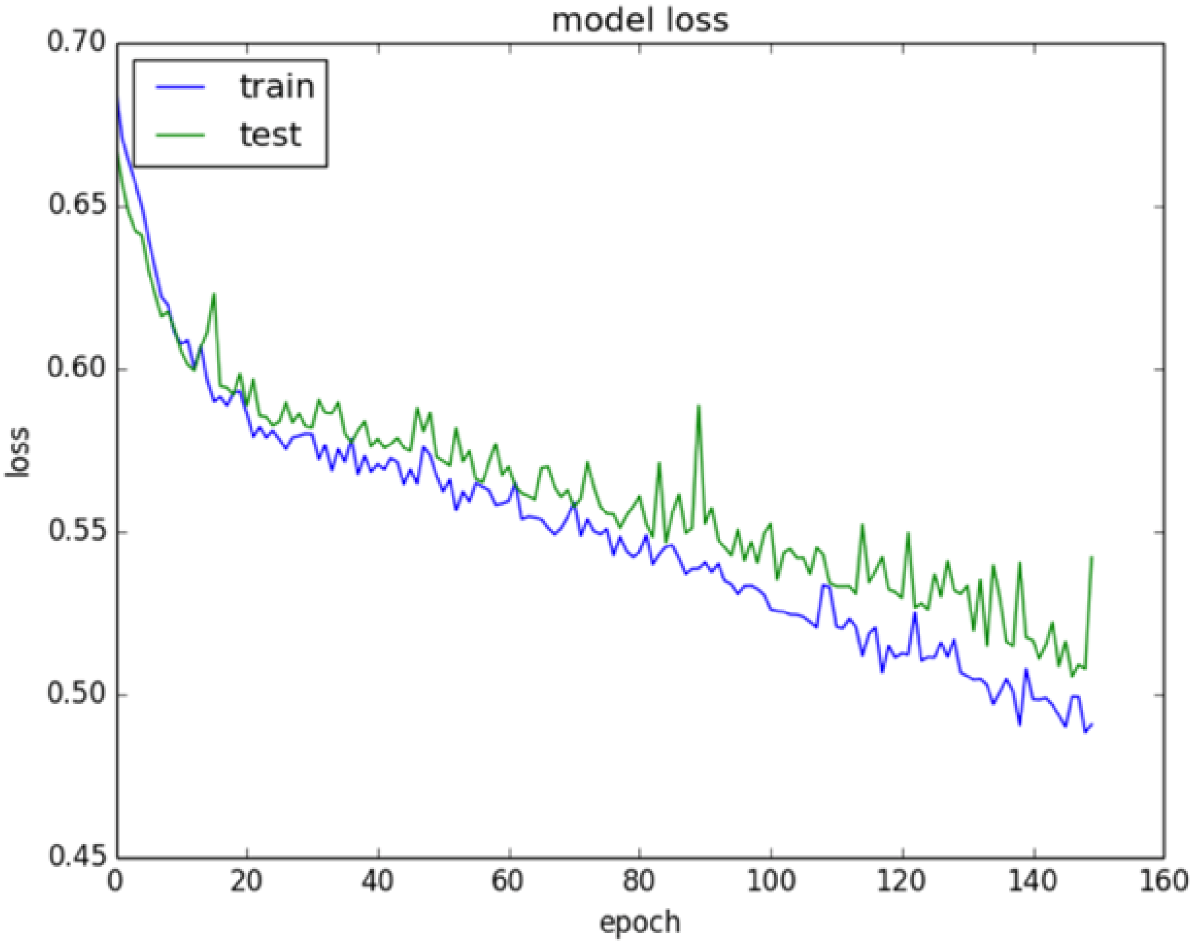
Supervised Learning Example: Task A: Image Classification

Task	Image Classification
Experience	Labelled Image
Performance Measure	Precision/Recall

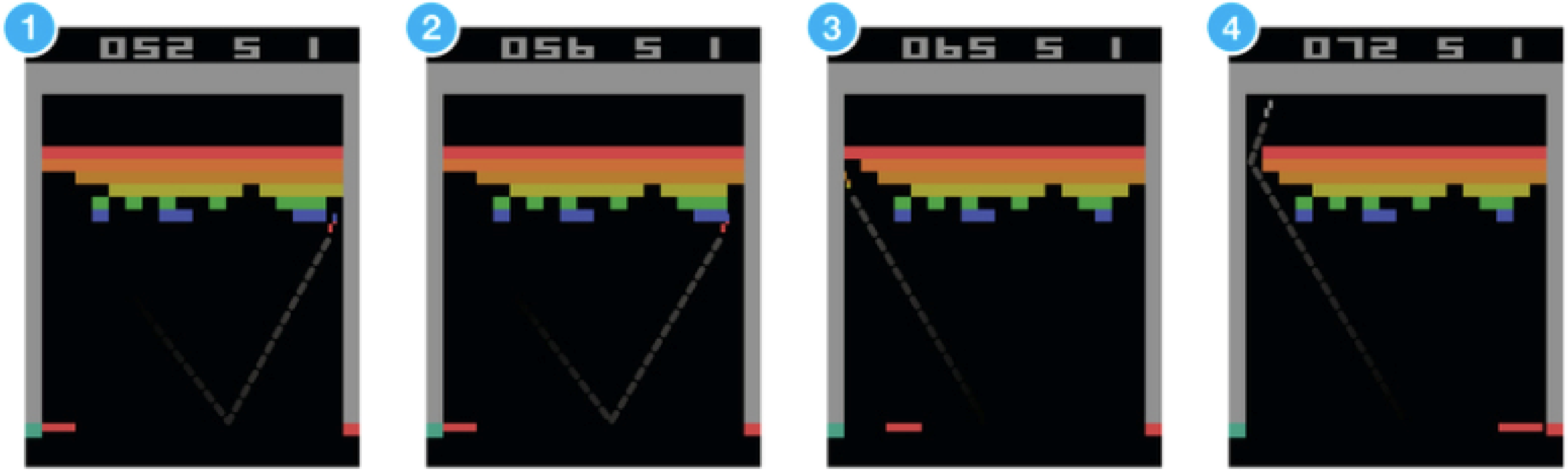
Learning with Model

- True(unknown) function
  - *"how human classify objects in image?"*
- Hypothesis class(Model)
  - *function: $y = f(x)$ with parameters*
- Search within hypothesis space
  - according to dataset
  - with optimization methods

Learning with Model



Reinforcement Learning Example: Task B: Play Game

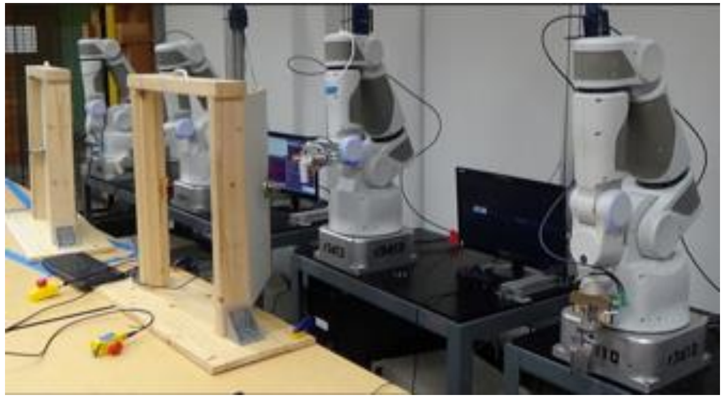


Survive, and get highest score as possible

Reinforcement Learning Example: Task B: Play Game

Task	Play Game
Experience	Play game over and over again
Performance Measure	Overall score

Reinforcement Learning Example: Other Tasks



Reinforcement Learning

- How software agents ought to take actions
- in an environment, so as to maximize some notion of cumulative reward.

Reinforcement Learning

- 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.

Reinforcement Learning

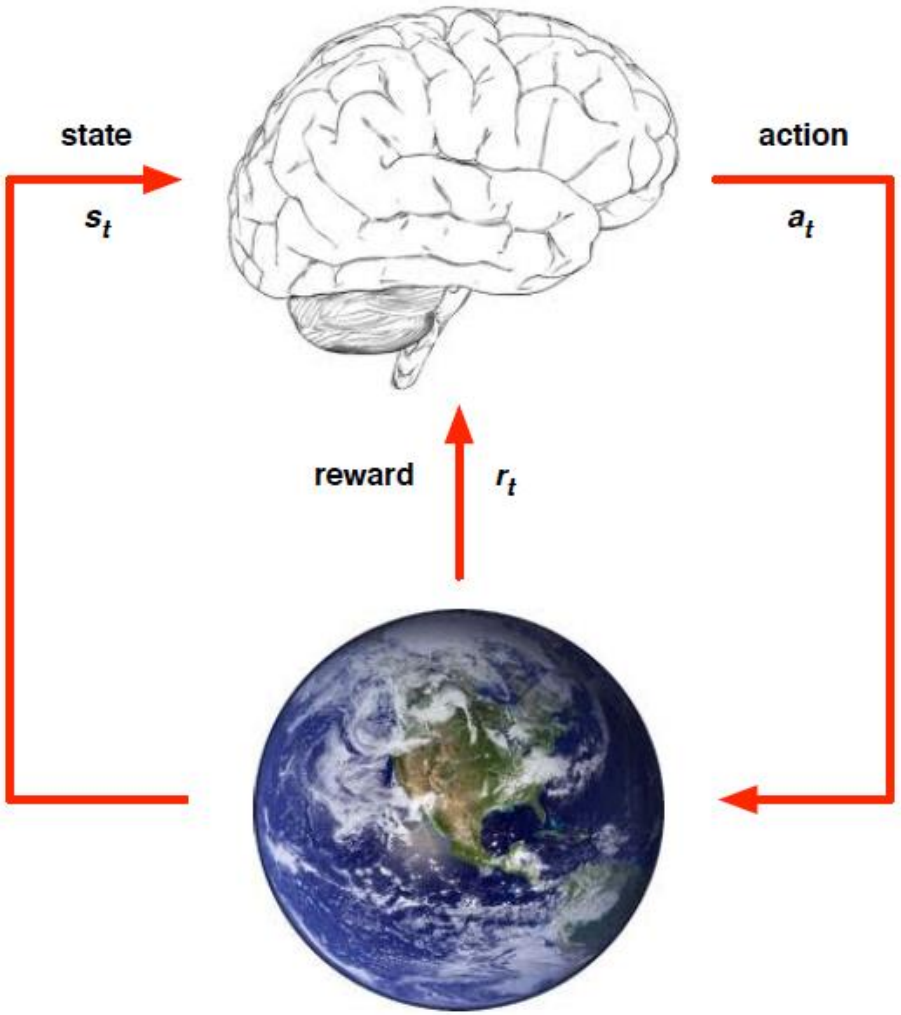
Task	Optimal policy for specific task, under specific environment
Experience	Interaction Experience with environment
Performance Measure	Discounted Future Reward

T: Markov Decision Process

- $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$
- $\mathcal{S}$: State Space
- $\mathcal{A}$: Action Space
- $\mathcal{R}: \mathcal{S} \times \mathcal{A} \rightarrow R$: Reward Function
- $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$: Transition Function

E: Interaction Experience

- At each step t, the agent
  - receive state s_t
  - execute action a_t
- The environment
  - receive action a_t
  - emit scalar reward r_t
  - enter state s_{t+1}
- Transition: $\langle s_t, a_t, r_t, s_{t+1} \rangle$



P: Discounted Future Reward

- Total reward:

$$R = r_1 + r_2 + r_3 + \cdots + r_n$$

- Total future reward:

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n$$

- Discounted future reward:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n, \quad \gamma \in [0,1]$$

Policy, Value and Transition Model

- Policy is a behavior function choosing actions given states

$$a = \pi(s) \text{ or } p^{\pi}(a|s)$$

- Value function is expected discounted future award, starting from a given state and performing a given action

$$Q(s_t, a_t) = \mathbb{E}(R_t | s_t, a_t)$$

- Transition model estimates the future state(and reward)

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

Approaches to RL

- Value-based RL
  - Estimate the optimal value function $Q^*(s, a)$
  - This is the maximum value achievable under any policy
- Policy-based RL
  - Search directly for the optimal policy π^*
  - This is the policy achieving maximum future reward
- Model-based RL
  - Build a transition model of the environment
  - Plan (e.g. by lookahead) using model $p(s_{t+1}, r_t | s_t, a_t)$

- Recap and Concepts
- **Reinforcement Learning Basics**
- Advanced Reinforcement Learning
- Challenges and Approaches

Reinforcement Learning Basics

- **Value Based RL**
- Policy Based RL
- Model Based RL

Q Function

- Value function is expected discounted future award, starting from a given state and performing a given action

$$Q(s_t, a_t) = \mathbb{E}(R_t | s_t, a_t)$$

Bellman Equation

- Optimal value function can be unrolled recursively

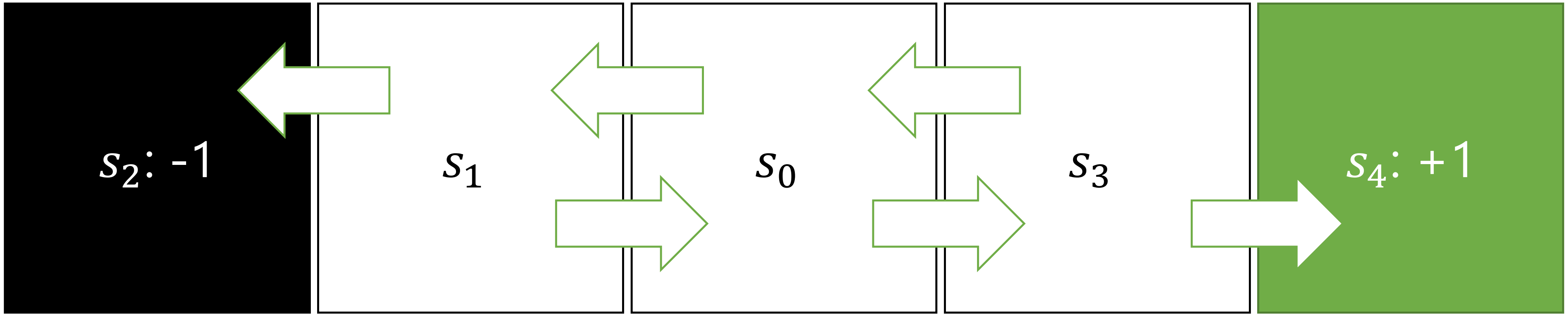
$$Q^*(s,a) = \mathbb{E}_{s'}\left(r + \gamma \max_{a'} Q^*(s',a') \,|s,a\right)$$

- Q-function can be iteratively updated by using the Bellman equation

$$Q_{i+1}(s,a) \leftarrow \mathbb{E}_{s'}\left(r + \gamma \max_{a'} Q_i(s',a') \,|s,a\right)$$

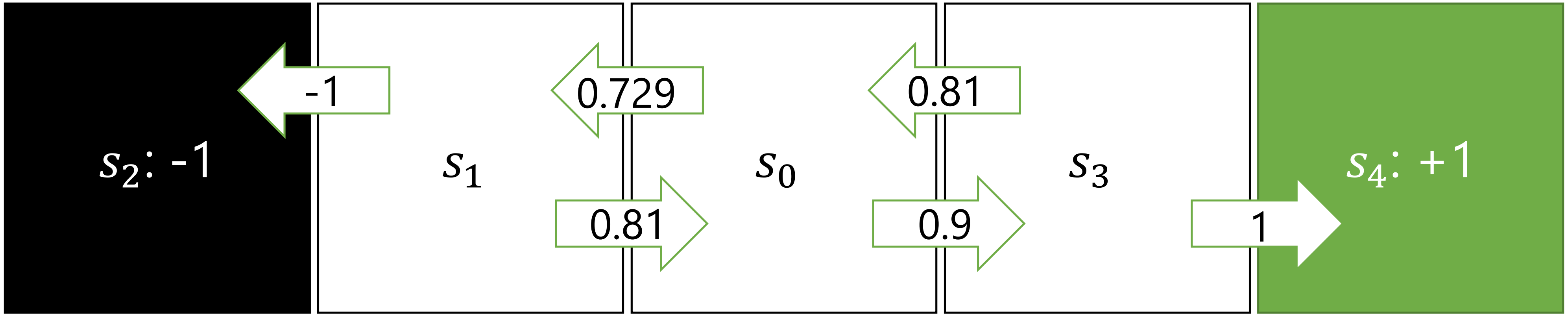
Example Problem: Grid World

- Q Function:
  - $Q(s_0, a_l), Q(s_0, a_r), Q(s_1, a_l), Q(s_1, a_r), Q(s_3, a_l), Q(s_3, a_r)$
  - $\gamma = 0.9$



Example Problem: Grid World

- $Q^*(s,a) = \mathbb{E}_{s'}\left(r + \gamma \max_{a'} Q^*(s',a') \,|s,a\right)$
- $Q(s_3,a_r), Q(s_1,a_l), Q(s_0,a_r), Q(s_1,a_r), Q(s_0,a_l), Q(s_3,a_l)$
- $\gamma = 0.9$



Value Based Method example: Q-Learning

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

Reinforcement Learning Basics

- Value Based RL
- **Policy Based RL**
- Model Based RL

Policy Function

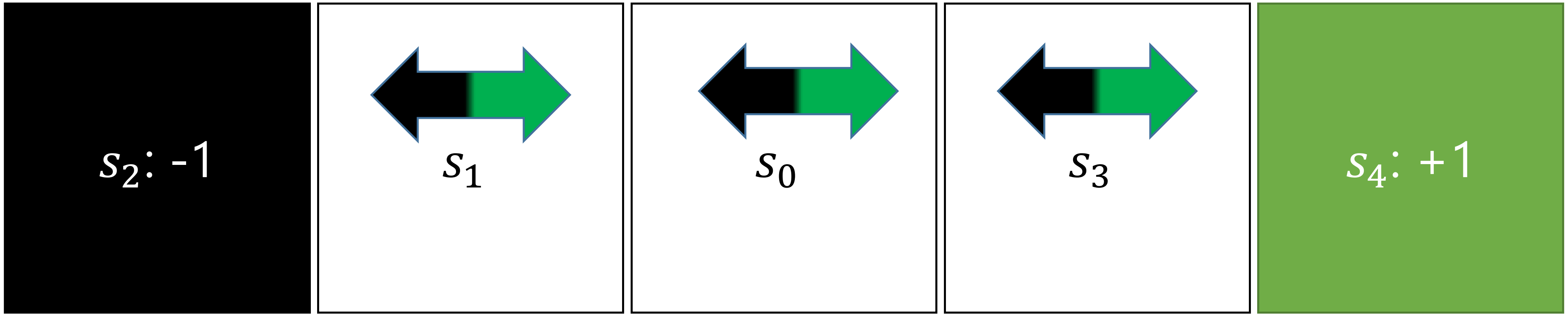
- Policy is a behavior function choosing actions given states

$$a = \pi(s) \quad \text{or} \quad p^{\pi}(a|s)$$

- Policy Based RL directly optimizes $\mathbb{E}[R]$ by searching in policy space

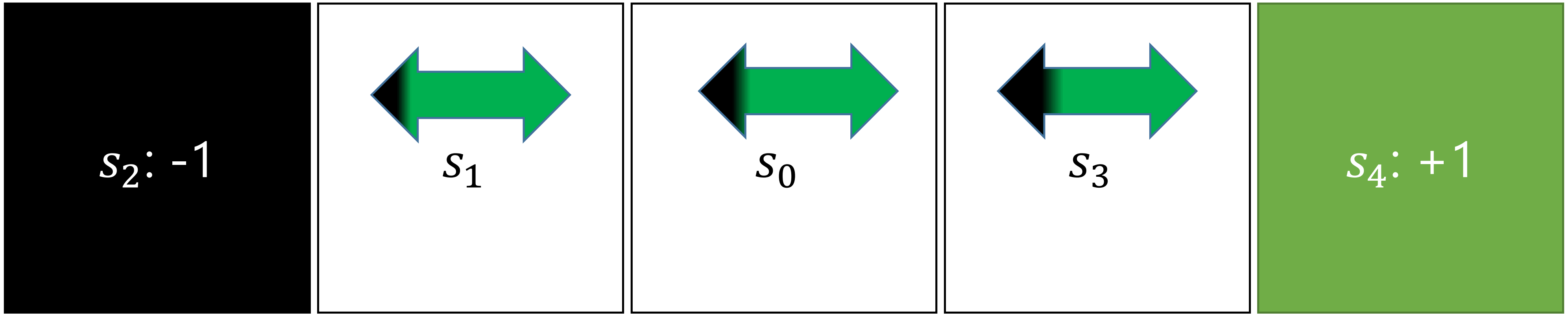
Example Problem: Grid World

- Policy Function:
  - $P(a_l|s_0), P(a_r|s_0), P(a_l|s_1), P(a_r|s_1), P(a_l|s_3), P(a_r|s_3)$
  - $\gamma = 0.9$



Example Problem: Grid World

- Policy Function:
  - $P(a_l|s_0), P(a_r|s_0), P(a_l|s_1), P(a_r|s_1), P(a_l|s_3), P(a_r|s_3)$
  - $\gamma = 0.9$



Policy Based Method example: Policy Gradient

- Define the loss function of policy $\pi(*:\theta)$ as

$$\mathcal{H}(\theta) = \mathbb{E}[R_t|\pi(*:\theta)] = \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots |\pi(*:\theta)\right]$$

- For stochastic policy $\pi(a|s:\mu)$

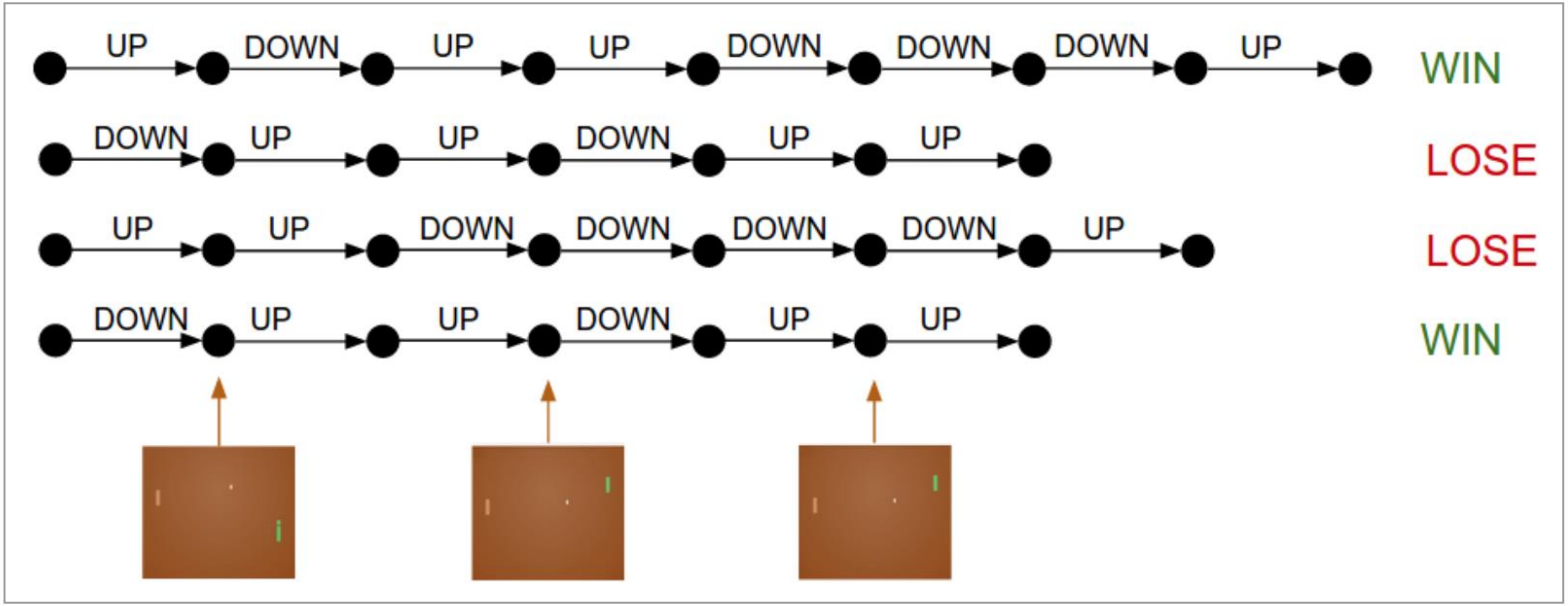
$$\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[Q^{\pi}(s,a)\frac{\partial \log \pi(a|s:\theta)}{\partial \theta}\right]$$

- For deterministic policy $a = \pi(s:\theta)$ when a is continuous and Q is differentiable

$$\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[\frac{\partial Q^{\pi}(s,a)}{\partial a}\frac{\partial a}{\partial \theta}\right]$$

Policy Based Method example: Policy Gradient

> *Policy Gradients: Run a policy for a while. See what actions led to high rewards. Increase their probability.*



http://karpathy.github.io/2016/05/31/rl/

Reinforcement Learning Basics

- Value Based RL
- Policy Based RL
- **Model Based RL**

Model Based RL

- Transition model estimates the future state(and reward)

$$p(s_{t+1}, r_t | s_t, a_t)\text{, or } s_{t+1} = f(s_t, a_t)$$

- Model-based RL
  - Build a transition model of the environment
  - Plan (e.g. by lookahead) using model $p(s_{t+1}, r_t | s_t, a_t)$

Model Based RL Example: Model Predictive Control

every N steps

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

Model Based RL Example: Model Predictive Control

every N steps

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions **How?**
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

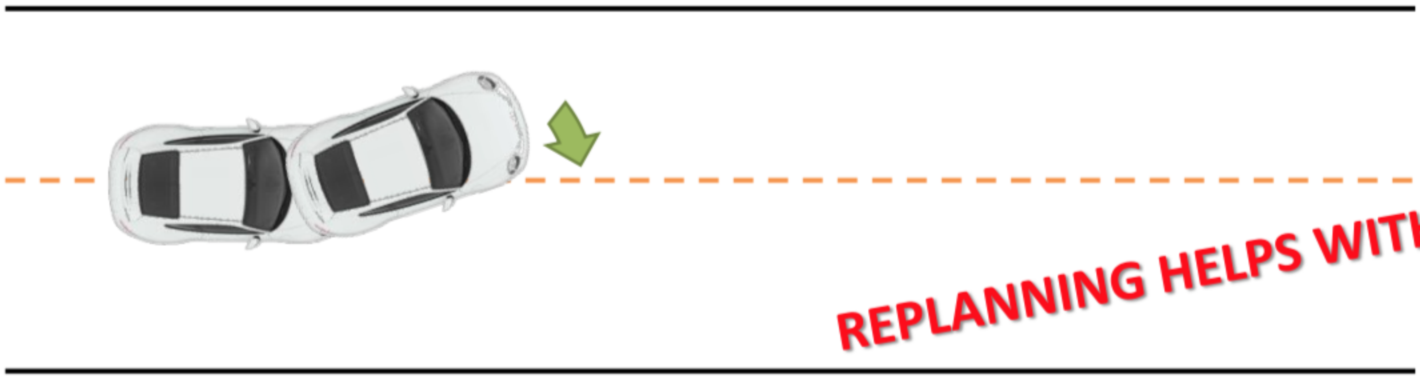
Random, LQR, Backprop, etc. More on this later

Model Based RL Example: Model Predictive Control

every N steps

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

   Why?
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$



REPLANNING HELPS WITH MODEL ERRORS

Model Based RL Example: Model Predictive Control

every N steps

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$ **Why?**

Agent could go to states that π_0 would have never seen

Deal with distribution mismatch problem in $\mathcal{D}$

RL Basics: Recap

- Value-based RL
  - Estimate the optimal value function $Q^*(s, a)$
  - This is the maximum value achievable under any policy
- Policy-based RL
  - Search directly for the optimal policy π^*
  - This is the policy achieving maximum future reward
- Model-based RL
  - Build a transition model of the environment
  - Plan (e.g. by lookahead) using model $p(s_{t+1}, r_t | s_t, a_t)$

- Recap and Concepts
- Reinforcement Learning Basics
- **Advanced Reinforcement Learning**
- Challenges and Approaches

Advanced RL

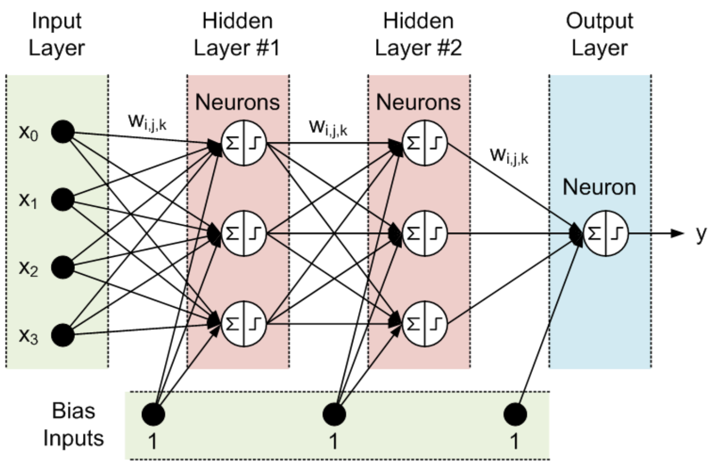
- **DQN and Advanced Value Based RL**
- Advanced Policy Based RL
- Advanced Model Based RL

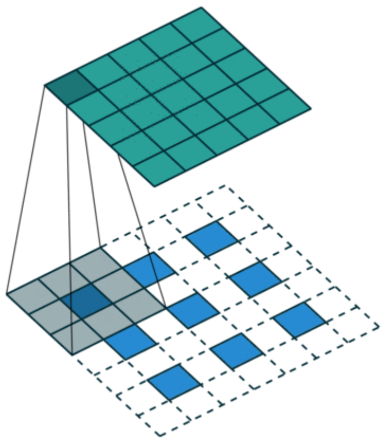
DQN and Advanced Value Based RL

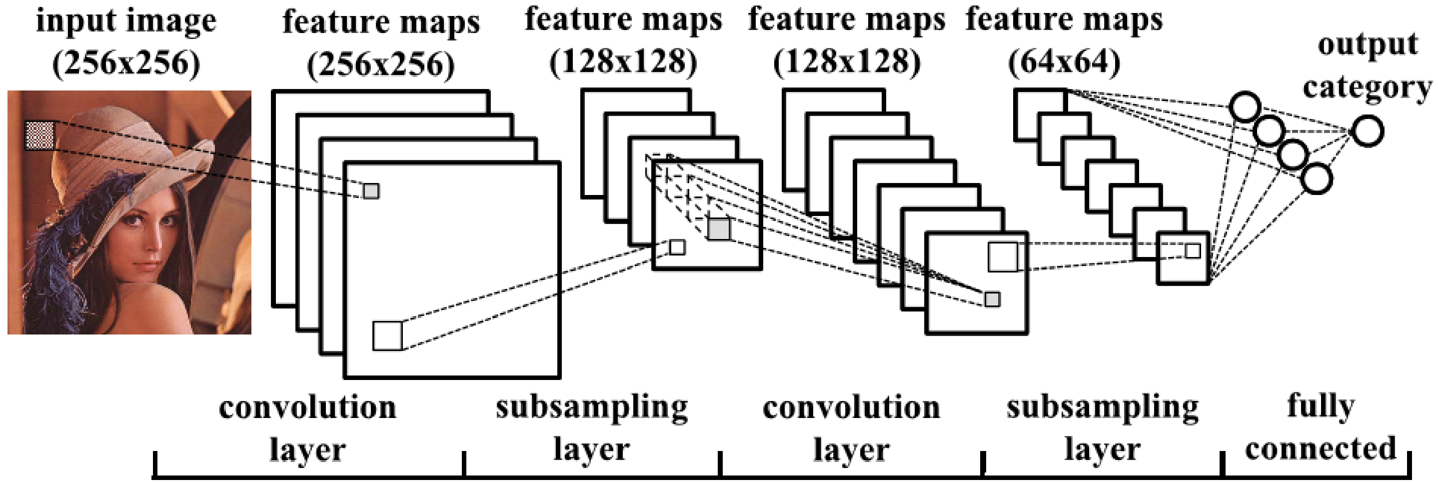
- **DQN**
- Dueling DQN
- Double DQN
- Prioritized Experience Replay
- Optimality Tightening
- Rainbow

Recap: Neural Networks

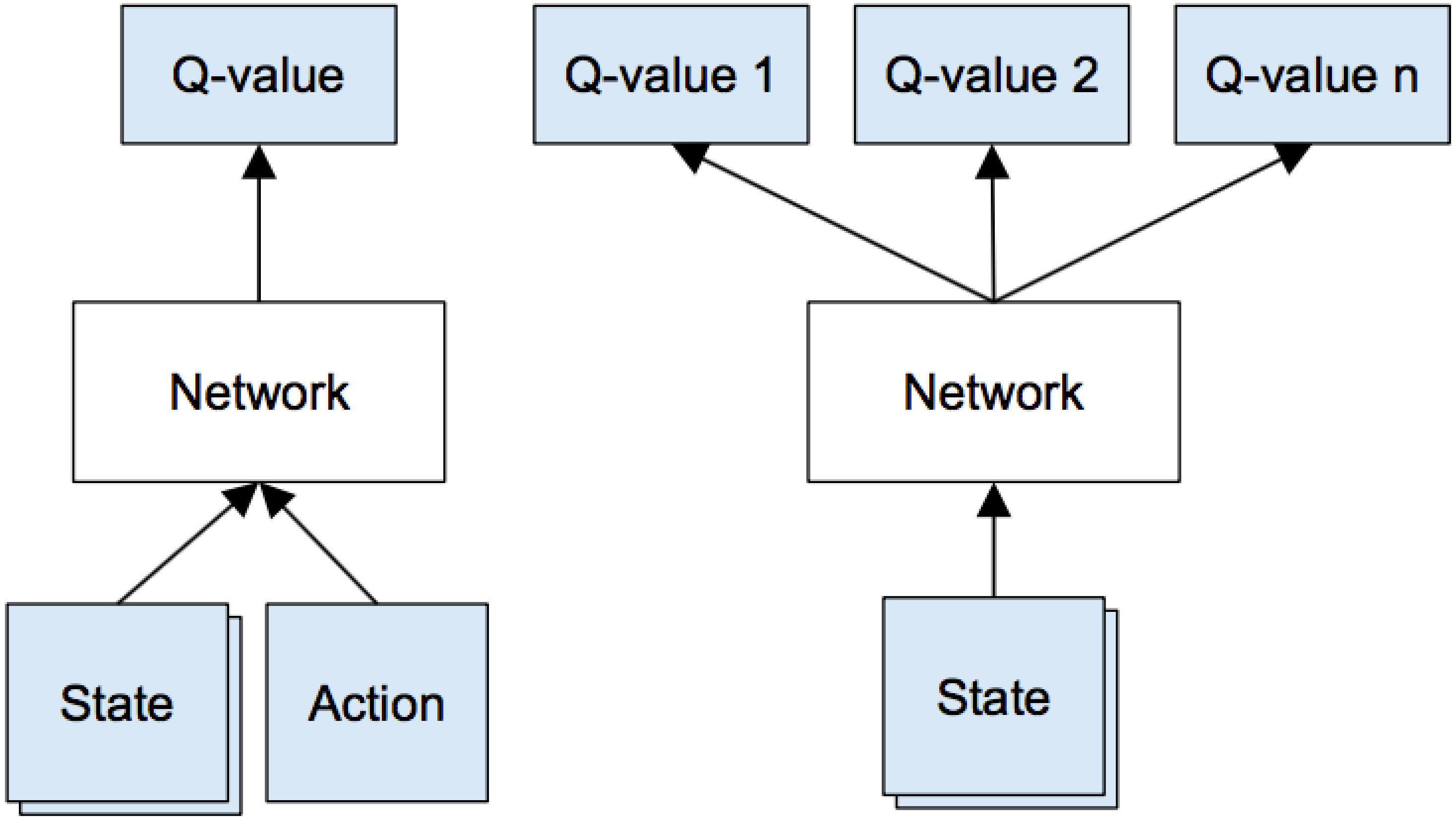
- Universal Function Approximator
- Convolutional Network:
  - Function that accepts images as input







Deep Q-Network



Deep Q-Learning

- Represent value function by deep Q-network $Q(s, a{:}\,\theta)$
- Define a MSE loss function for Q-value approximation

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

- Optimize by SGD

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s', a'{:}\,\theta) - Q(s, a{:}\,\theta)\right)\frac{\partial Q(s, a{:}\,\theta)}{\partial \theta}\right]$$

Challenges in Deep Q-Learning

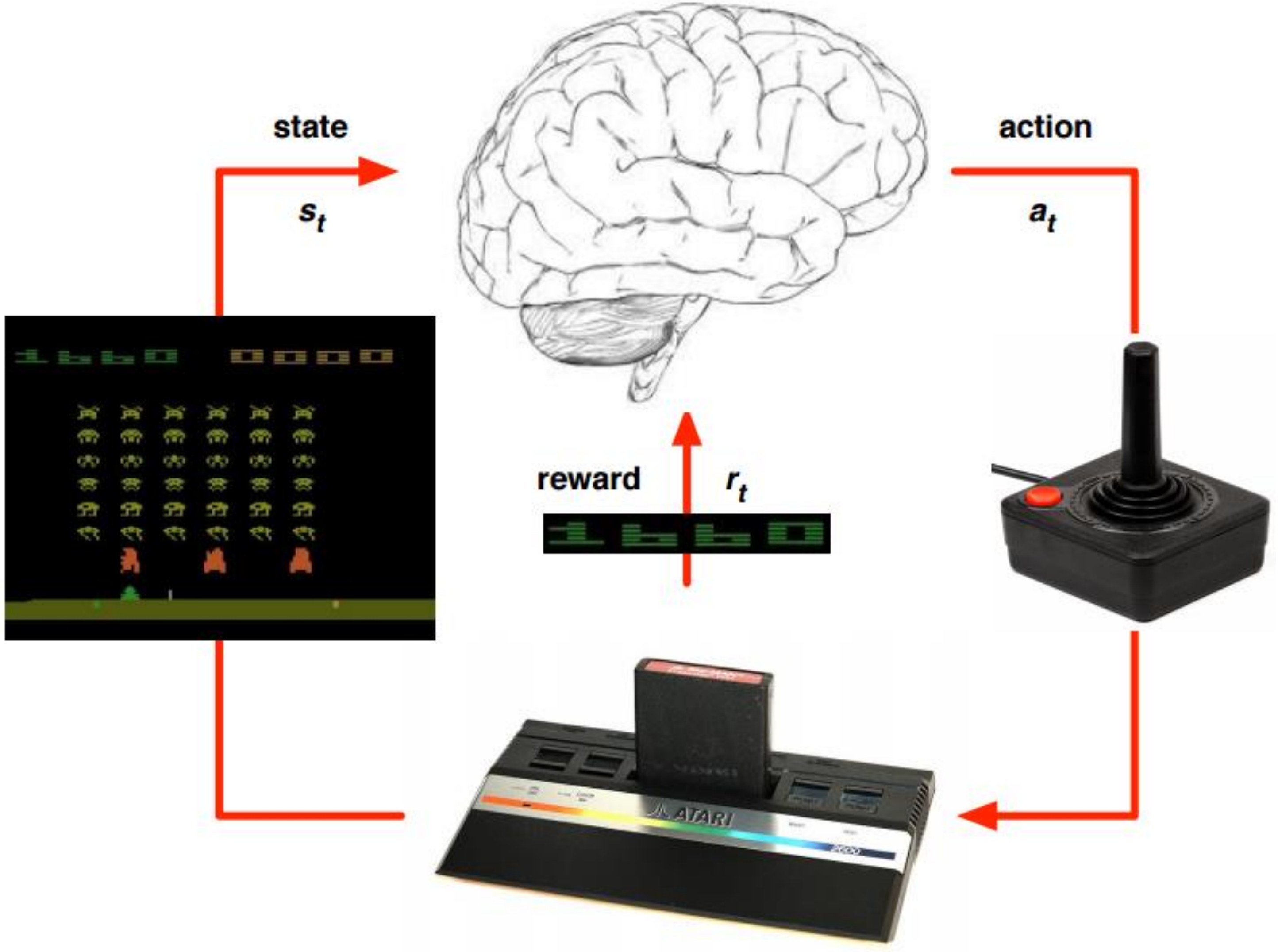
- Exploration-exploitation dilemma
  - Random exploration gradually becomes greedy and crude with exploitation of converging Q functions
- Sequential data are NOT i.i.d
  - Choosing an certain action affects the coming transitions
  - Highly correlated data are harmful for SGD method
- Non-stationary target
  - Target changes while θ is updated, causing oscillation during training

- Successful tricks:
  - ε-greedy exploration & experience replay & target Q-network

Deep Q-Learning with Experience Replay

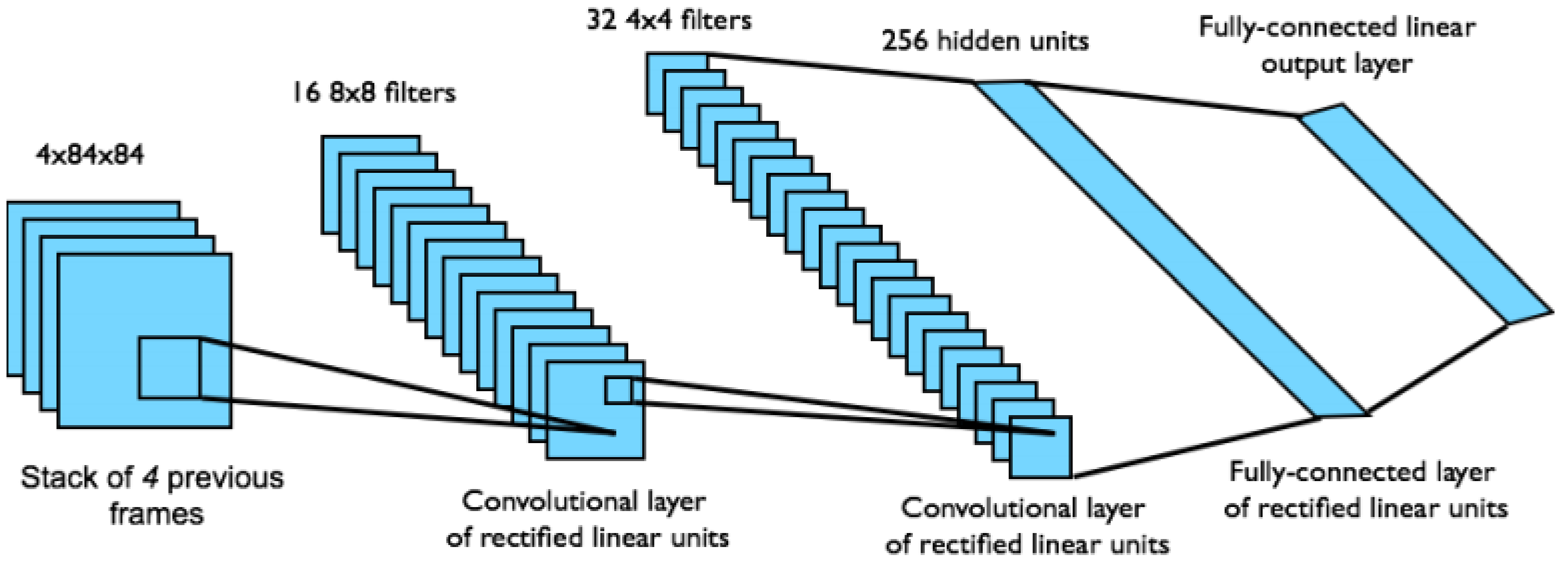
```
initialize replay memory D
initialize action-value function Q with random weights
observe initial state s
repeat
      select an action a
            with probability ε select a random action
            otherwise select a = argmax_a' Q(s,a')
      carry out action a
      observe reward r and new state s'
      store experience <s, a, r, s'> in replay memory D

      sample random transitions <ss, aa, rr, ss'> from replay memory D
      calculate target for each minibatch transition
            if ss' is terminal state then tt = rr
            otherwise tt = rr + γmax_a' Q(ss', aa')
      train the Q network using (tt - Q(ss, aa))^2 as loss

      s = s'
until terminated
```

RL in Atari

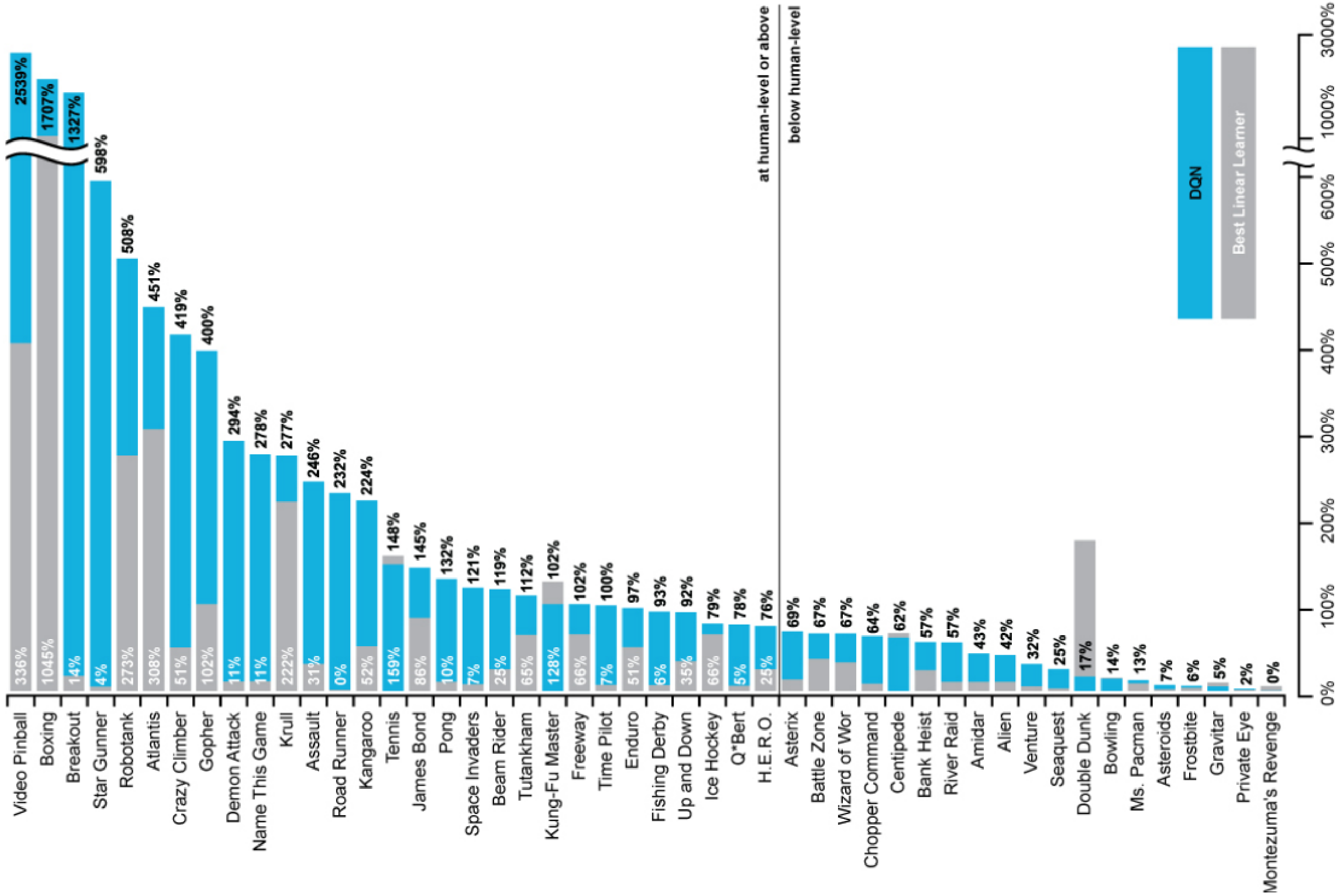


Deep Q-Learning in Atari

- End-to-end learning of values Q(s, a) from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is Q(s, a) for 18 joystick/button positions
- Reward is change in score for that step



Results



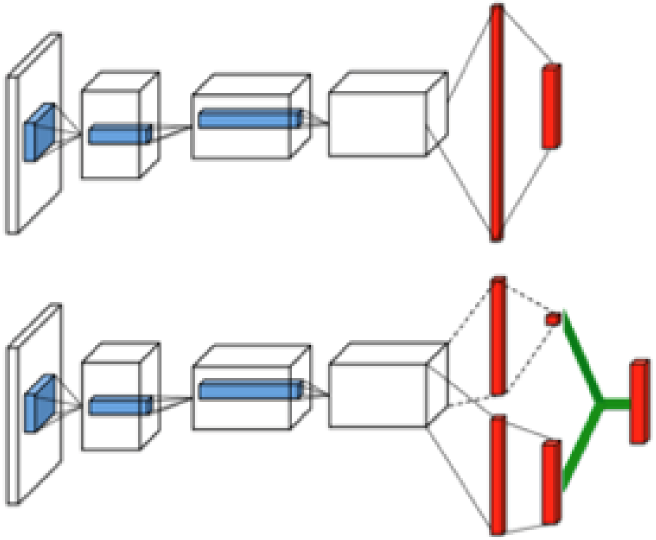
DQN
Best Linear Learner
at human-level or above
below human-level
0%
100%
200%
300%
400%
500%
600%
1000%
3000%
Video Pinball 2539% 336%
Boxing 1707% 1045%
Breakout 1327% 14%
Star Gunner 598% 4%
Robotank 508% 273%
Atlantis 451% 308%
Crazy Climber 419% 51%
Gopher 400% 102%
Demon Attack 294% 11%
Name This Game 278% 11%
Krull 277% 222%
Assault 246% 31%
Road Runner 232% 0%
Kangaroo 224% 52%
Tennis 148% 159%
James Bond 145% 86%
Pong 132% 10%
Space Invaders 121% 7%
Beam Rider 119% 25%
Tutankham 112% 65%
Kung-Fu Master 102% 128%
Freeway 102% 66%
Time Pilot 100% 7%
Enduro 97% 51%
Fishing Derby 93% 6%
Up and Down 92% 35%
Ice Hockey 79% 66%
Q*Bert 78% 5%
H.E.R.O. 76% 25%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pacman 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%

DQN and Advanced Value Based RL

- DQN
- **Dueling DQN**
- Double DQN
- Prioritized Experience Replay
- Optimality Tightening
- Rainbow

Better Model

- Dueling Network Architectures for Deep Reinforcement Learning
- $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha)$



http://arxiv.org/abs/1511.06581

DQN and Advanced Value Based RL

- DQN
- Dueling DQN
- **Double DQN**
- **Prioritized Experience Replay**
- **Optimality Tightening**
- Rainbow

Convergence Problem in DQN

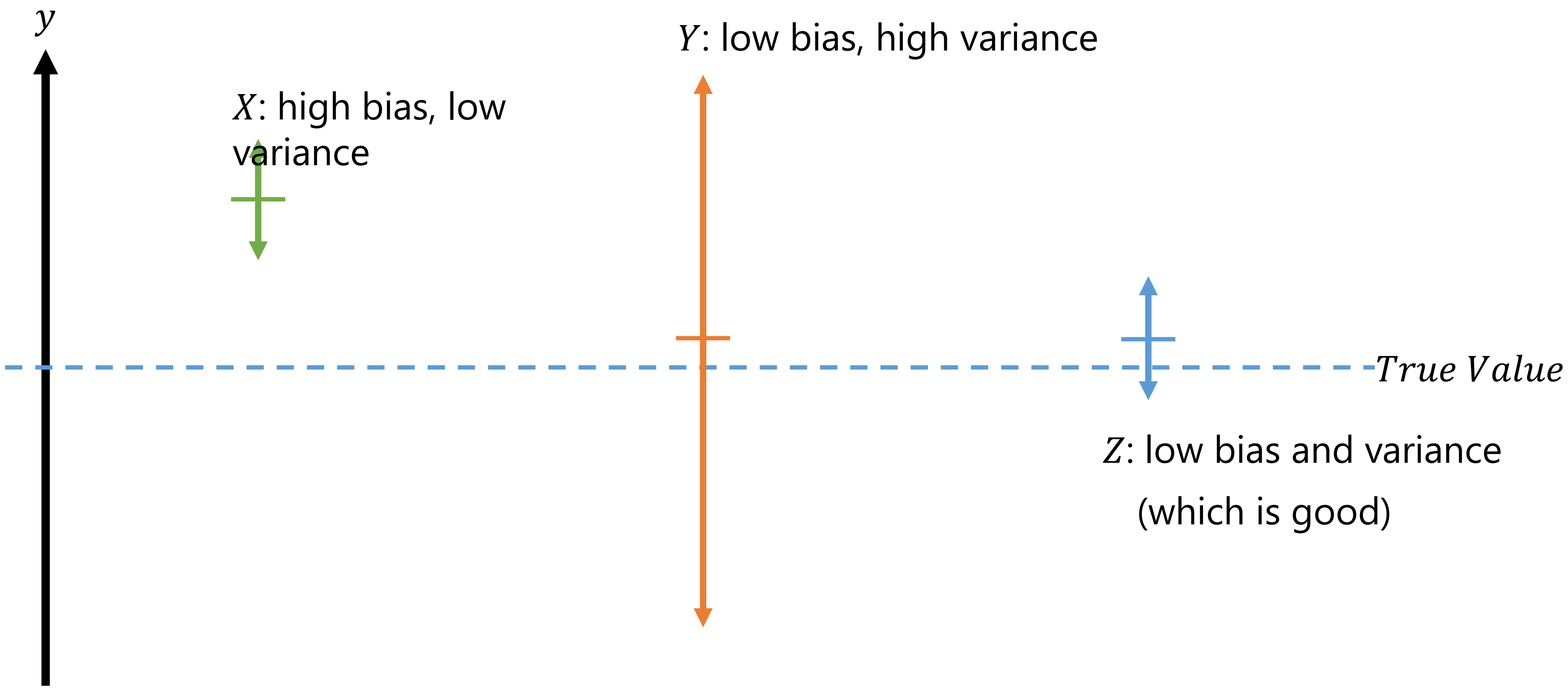
- Supervised Learning:
  - $\mathcal{L}(\theta) = \mathbb{E}_x\left[\left(\boldsymbol{y} - f(x:\theta)\right)^2\right]$
- Reinforcement Learning(DQN):
  - $\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(\boldsymbol{r} + \gamma \max_{a'} Q(s',a':\theta) - Q(s,a:\theta)\right)^2\right]$
  - sparse ground truth
  - Estimation bias in $\max_{a} Q$
  - Only 1-step propagation along trajectory

Convergence Problem: Better use of data

- Prioritized Experience Replay
  - Some transitions are more informative than others
  - TD-error guided
  - for rarely visited valuable transitions
- $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$
- $w(i) = (\frac{1}{N} * \frac{1}{P(i)})^\beta$

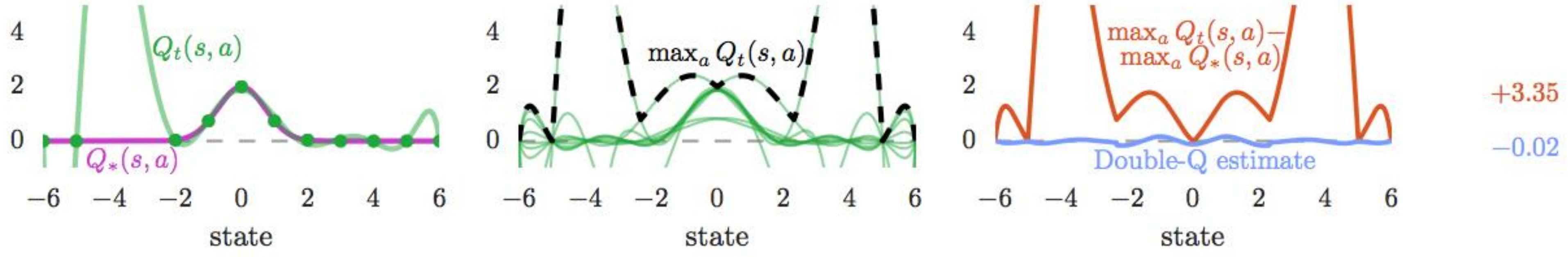
http://arxiv.org/abs/1511.05952

Recap:
Bias and Variance of Value Estimation



Convergence Problem: Double DQN

- $\boldsymbol{Y^{DQN}} = \boldsymbol{r} + \gamma \boxed{\max_{a'} Q(s', a' : \theta^{-})}$ **Upward bias!**
- $\boldsymbol{Y^{DoubleDQN}} = \boldsymbol{r} + \gamma\, Q\left(s', \max_{a'} Q(s', a' : \theta) : \theta^{-}\right)$
- Not too optimistic when estimating Q



https://arxiv.org/pdf/1509.06461.pdf

Convergence Problem: Faster Q propagation

- Optimality Tightening
- $\boldsymbol{Q}^*(\boldsymbol{s_t}, \boldsymbol{a_t}) \; = \; \boldsymbol{r} + \gamma \max_a \boldsymbol{Q}^*(\boldsymbol{s_{t+1}}, a)$
  - $\geq \sum_{\boldsymbol{i=0}}^{\boldsymbol{k}} \gamma^i \boldsymbol{r_{t+i}} + \gamma^{k+1} \max_a \boldsymbol{Q}^*(\boldsymbol{s_{t+k+1}}, a)$
- additional upper/lower bound along trajectory to Q

$$\min_\theta \sum_{(s_j,a_j,r_j,s_{j+1})\in\mathcal{B}} \left[(Q_\theta(s_j,a_j)-y_j)^2+\lambda(L_j^{\max}-Q_\theta(s_j,a_j))_+^2+\lambda(Q_\theta(s_j,a_j)-U_j^{\min})_+^2\right]$$

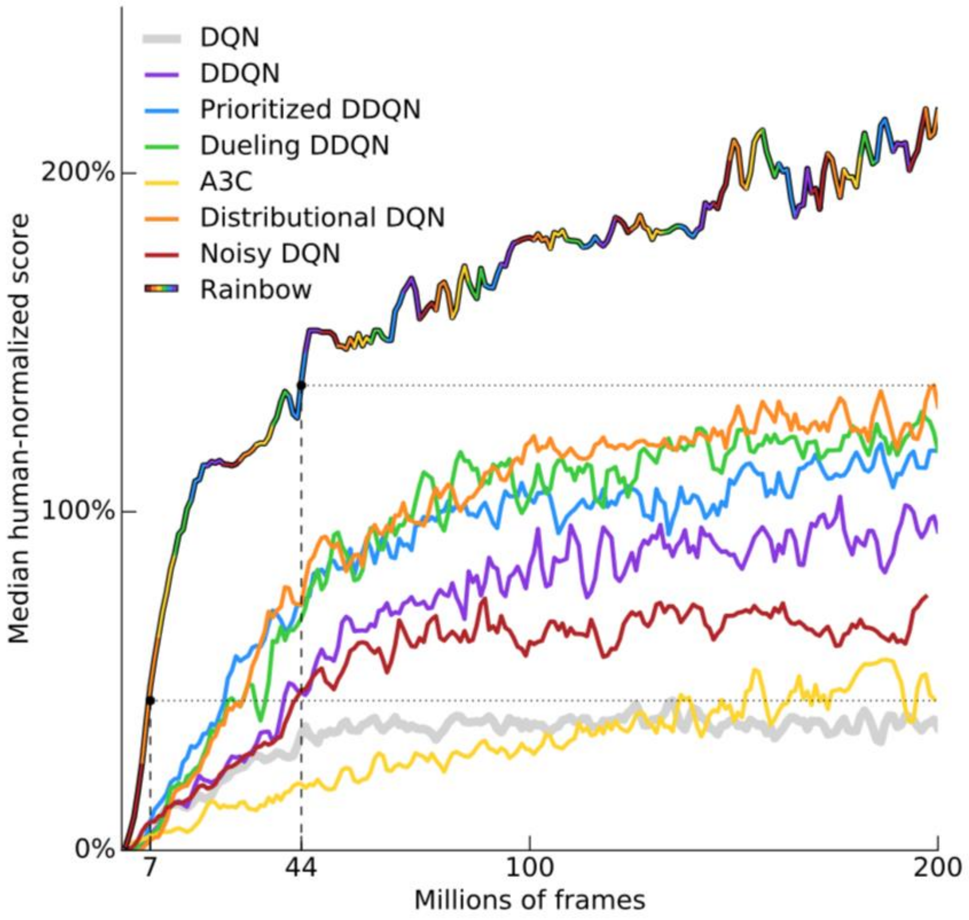
https://www.arxiv.org/abs/1611.01606v1

DQN and Advanced Value Based RL

- DQN
- Dueling DQN
- Double DQN
- Prioritized Experience Replay
- Optimality Tightening
- **Rainbow**

Rainbow

Combination of orthogonal improvements over DQN



Advanced RL

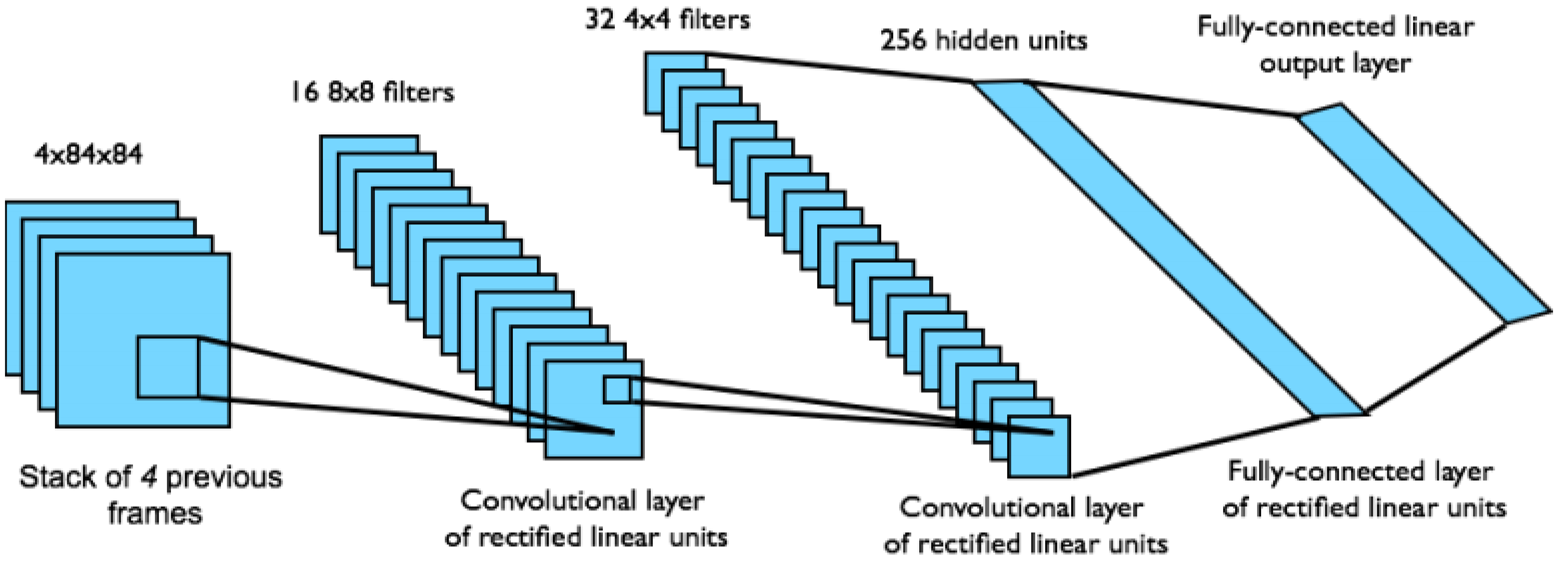
- DQN and Advanced Value Based RL
- **Advanced Policy Based RL**
- Advanced Model Based RL

Advanced Policy Based RL

- **Deep Policies**
- Reducing Variance in Policy Gradient Estimation
- Improved Sample Efficiency

Deep Policies

- Uses deep neural network as policy function approximators



Advanced Policy Based RL

- Deep Policies
- **Reducing Variance in Policy Gradient Estimation**
- Improved Sample Efficiency

Policy Gradient Recap

- Policy is a behavior function choosing actions given states

$$a = \pi(s) \quad \text{or} \quad p^{\pi}(a|s)$$

- Define the loss function of policy $\pi(*:\theta)$ as

$$\mathcal{H}(\theta) = \mathbb{E}[R_t|\pi(*:\theta)] = \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots |\pi(*:\theta)\right]$$

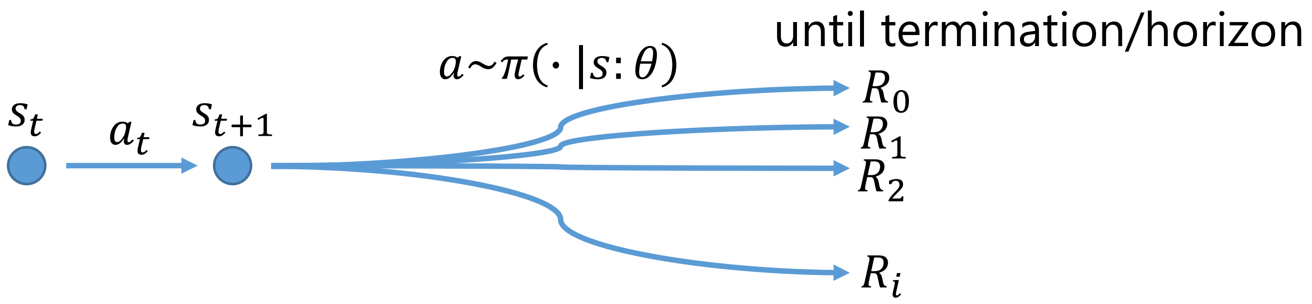
- For stochastic policy $\pi(a|s:\mu)$

$$\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[\boxed{Q^{\pi}(s,a)}\frac{\partial \log \pi(a|s:\theta)}{\partial \theta}\right]$$

How to estimate this?

Policy Gradient Estimation

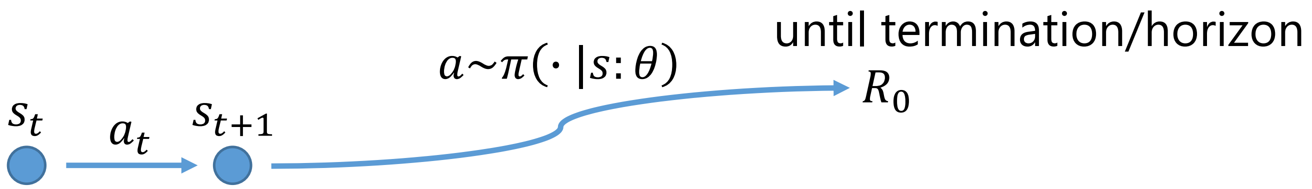
- Estimate $\mathbb{E}[Q^{\pi}(s,a)]$ to get $\mathbb{E}\left[Q^{\pi}(s,a)\frac{\partial \log \pi(a|s{:}\theta)}{\partial \theta}\right]$
- Rollout after executing a!



$\mathbb{E}[R_i]$ is a good estimation, but very costly or not possible

Policy Gradient Estimation

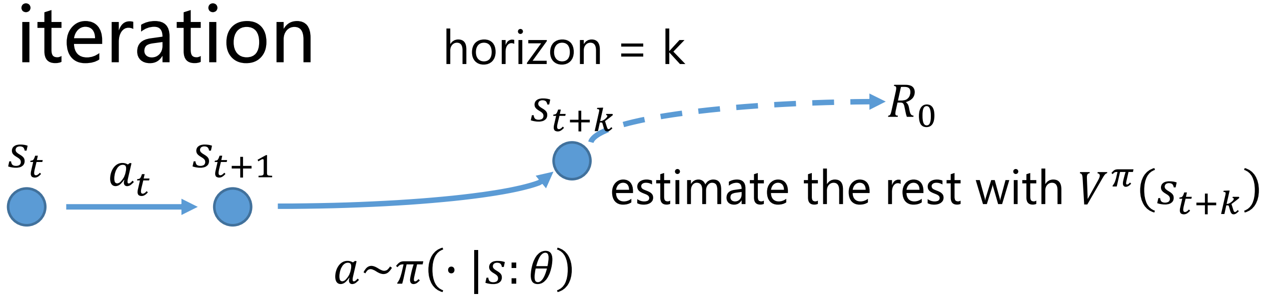
- Estimate $\mathbb{E}[Q^\pi(s,a)]$ to get $\mathbb{E}\left[Q^\pi(s,a)\frac{\partial \log \pi(a|s{:}\theta)}{\partial \theta}\right]$
- Uses a single rollout



R_0 is an unbiased estimation, but with high variance!
Further the horizon, higher the variance!

Actor Critic

- Critic Function $V_{\varphi}^{\pi}(s)$: Average R from state s under policy π
- $Q^{\pi}(s_t, a_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V^{\pi}(s_{t+k})$
- $V_{\varphi}^{\pi}(s)$: Bellman iteration



Shorter horizon(k), lower variance
But introduce bias with $V_{\varphi}^{\pi}(s)$

Advantage v.s. $Q^{\pi}(s,a)$

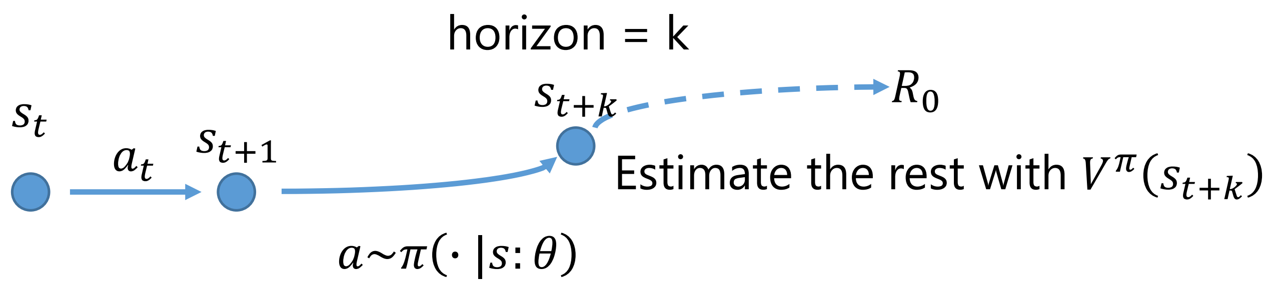
- Policy Gradient: $\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[Q^{\pi}(s,a)\frac{\partial \log \pi(a|s;\theta)}{\partial \theta}\right]$
- $\mathbb{E}\left[(Q^{\pi}(s,a) - b(s))\frac{\partial \log \pi(a|s;\theta)}{\partial \theta}\right]$ is also unbiased estimation
  - But possibly with lower variance
- $b(s)$: baseline function depending on state only
- Intuitively, the purpose of policy gradient is to evaluate relative goodness/badness of actions

Advantage based Actor Critic

- $V_\varphi^\pi(s)$ is a good $b(s)$
- Advantage: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$
- $\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[(Q^\pi(s,a) - b(s))\frac{\partial \log \pi(a|s{:}\theta)}{\partial \theta}\right]$
  - $= \mathbb{E}\left[(\sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V^\pi(s_{t+k}) \ - V^\pi(s_t))\frac{\partial \log \pi(a|s{:}\theta)}{\partial \theta}\right]$

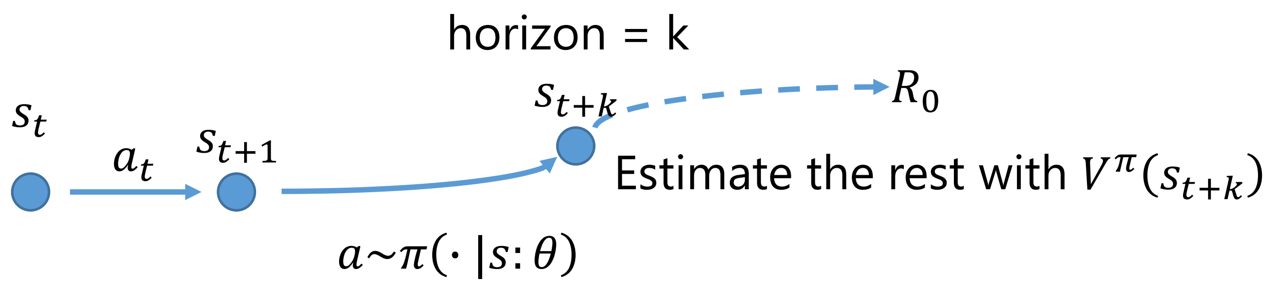
Recap: Bias and Variance in Actor Critic

- $Q^{\pi}(s_t, a_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V^{\pi}(s_{t+k})$
- Longer k, higher variance (from $\sum_{i=0}^{k-1} \gamma^i r_{t+i}$)
- Shorter k, higher bias (from $V^{\pi}(s_{t+k})$)



Generalized Advantage Estimation

- $Q^\pi(s_t, a_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V^\pi(s_{t+k})$
- GAE: Weighted average of Q^π of different k
- A good balance between variance and bias



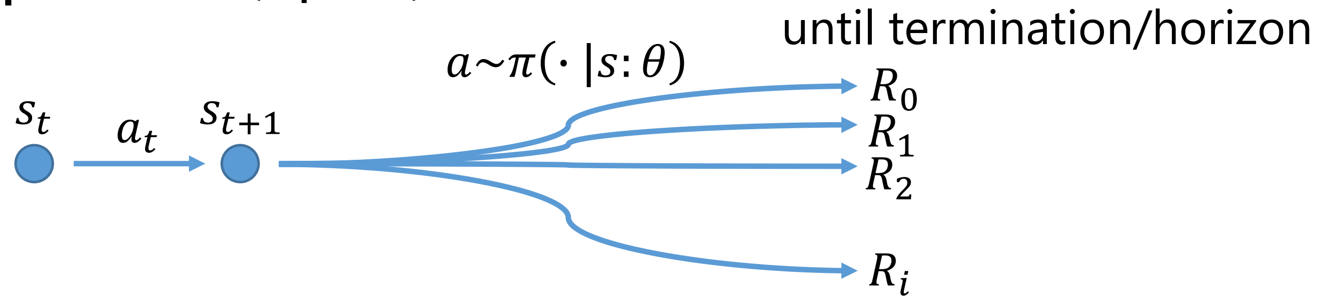
https://arxiv.org/abs/1506.02438

Advanced Policy Based RL

- Deep Policies
- Reducing Variance in Policy Gradient Estimation
- **Improved Sample Efficiency**

Policy Gradient: On-Policy Methods

- Policy Gradient: $\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[Q^{\pi}(s,a)\frac{\partial \log \pi(a|s{:}\theta)}{\partial \theta}\right]$
  - On-Policy: Trajectory generated by π can only be used to estimate $Q^{\pi}(s,a)$ and update $\pi(\cdot\,|s{:}\,\theta)$

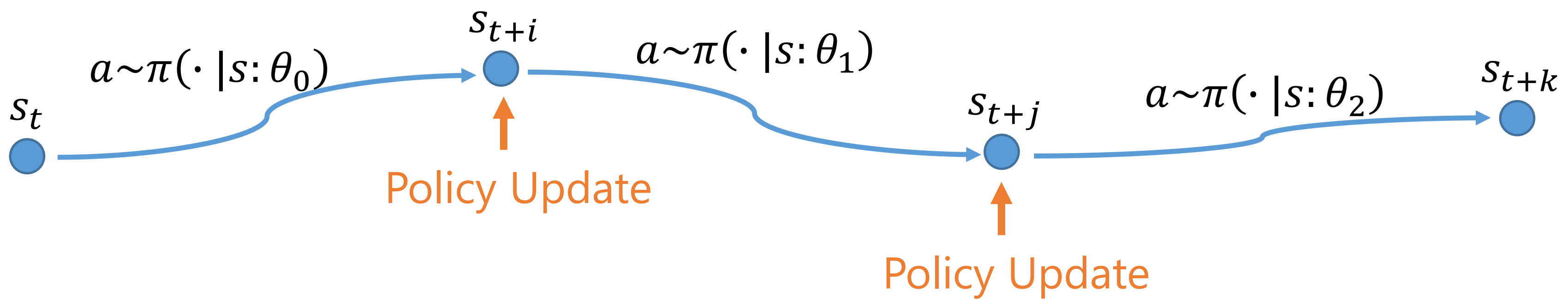


$\mathbb{E}[R_i]$ is a good estimation, but very costly!

Because every transition is used only once

Policy Gradient: On-Policy Methods

- Policy Gradient: $\frac{\partial \mathcal{H}(\theta)}{\partial \theta} = \mathbb{E}\left[Q^{\pi}(s,a)\frac{\partial \log \pi(a|s:\theta)}{\partial \theta}\right]$
  - On-Policy: Trajectory generated by π can only be used to estimate $Q^{\pi}(s,a)$ and update $\pi(\cdot\,|s:\theta)$



Every transition is used only once

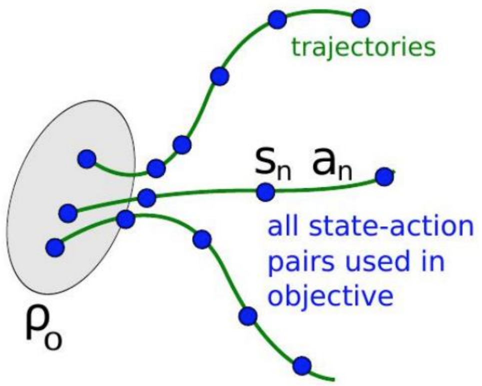
Policy Gradient: On-Policy Methods

- Can we do multiple policy update step based on a batch of transitions?
- Can we improve $\pi(\cdot\,|s{:}\,\theta_i)$ using transitions collected by $\pi(\cdot\,|s{:}\,\theta_0)$?

Trust Region Policy Optimization

- Monotonic policy improvement
  - Advantage between policies
  - Minorization-maximizaion
- Slightly off-policy
  - Trust Region: KL-divergence bounded
- Second order optimization

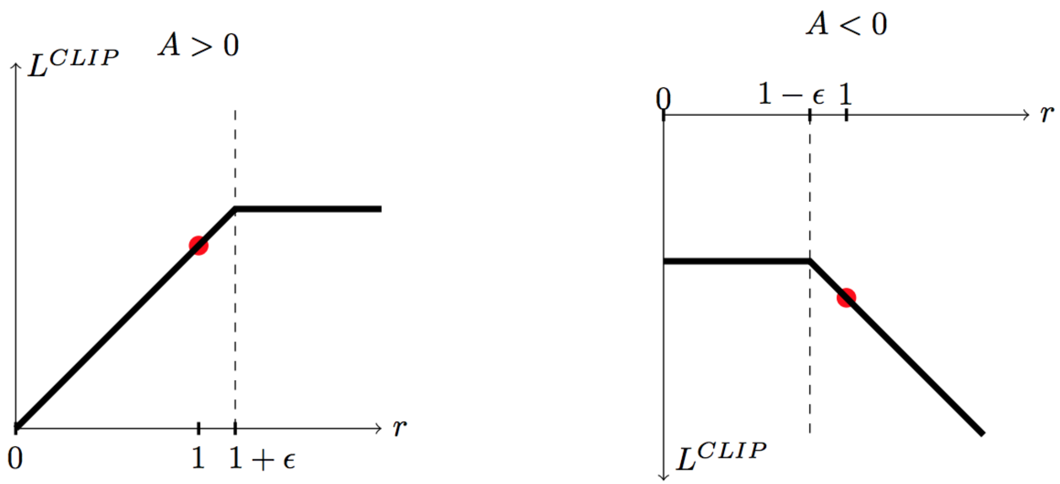
$$\underset{\theta}{\text{maximize}}\ \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{subject to}\ \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)\ \|\ \pi_\theta(\cdot|s))\right] \le \delta.$$



http://proceedings.mlr.press/v37/schulman15.pdf

Proximal Policy Optimization

- First order approximation to TRPO
  - KL divergence constraint in the form of loss term, instead of constrained optimization problem
- Easier to scale



https://arxiv.org/abs/1707.06347

Advanced RL

- DQN and Advanced Value Based RL
- Advanced Policy Based RL
- **Advanced Model Based RL**

Advanced Model Based RL

- **Planning Methods**
- Model Based with Model Free

Recap: Model Based RL

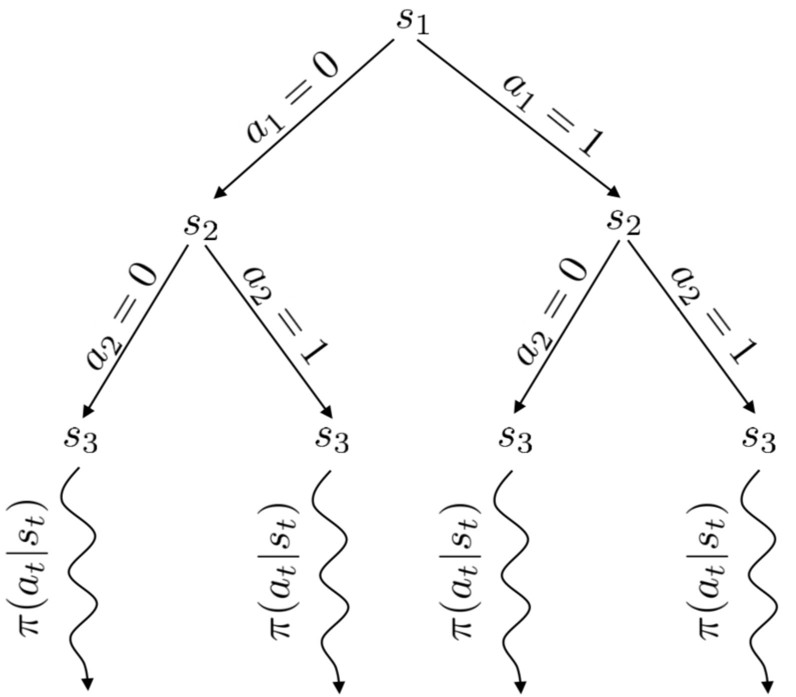
every N steps

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions **How?**
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

Random, LQR, Backprop, etc. More on this later

Tree Search Based Methods

- Monte Carlo tree search
  - Iteratively expand nodes into subtrees
- UCT Tree Search
  - Balance between promising nodes and underexplored nodes

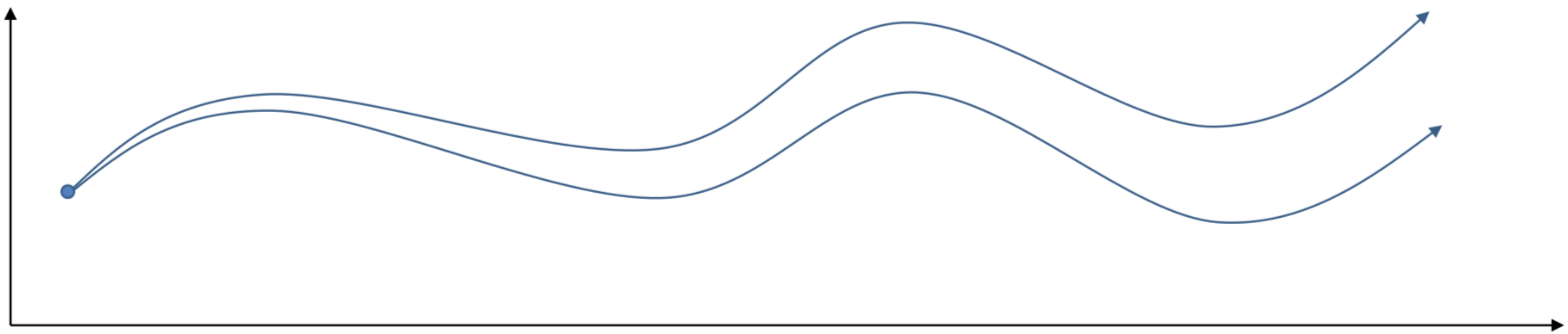


LQR: Linear Quadratic Regulator

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c(\mathbf{x}_1,\mathbf{u}_1) + c(f(\mathbf{x}_1,\mathbf{u}_1),\mathbf{u}_2) + \cdots + c(f(f(\ldots)\ldots),\mathbf{u}_T)$$

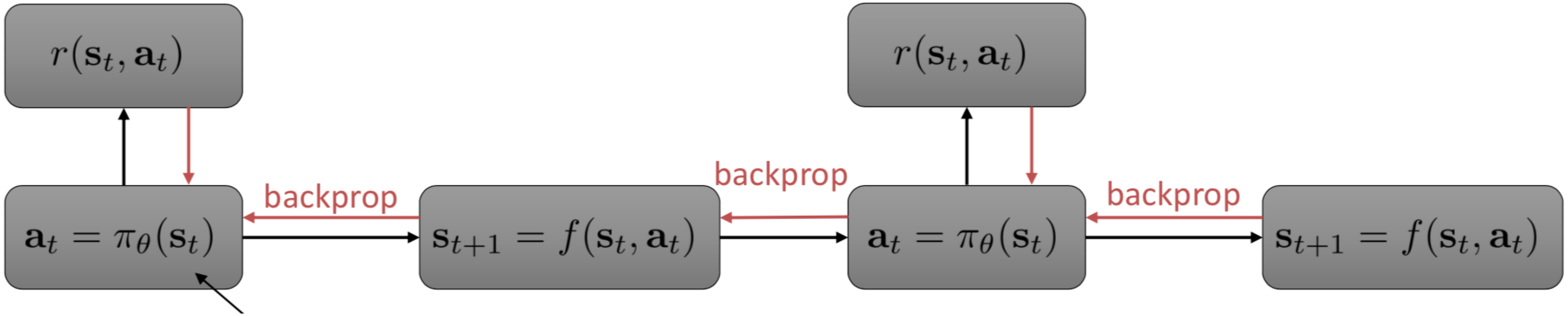
$$c(\mathbf{x}_t,\mathbf{u}_t) = \frac{1}{2}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix}^T \mathbf{C}_t \begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix}^T \mathbf{c}_t$$

$$f(\mathbf{x}_t,\mathbf{u}_t) = \mathbf{F}_t\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} + \mathbf{f}_t$$



Back-Propagation into Policy

- Backpropagate through model into the policy
- Choose action according to policy

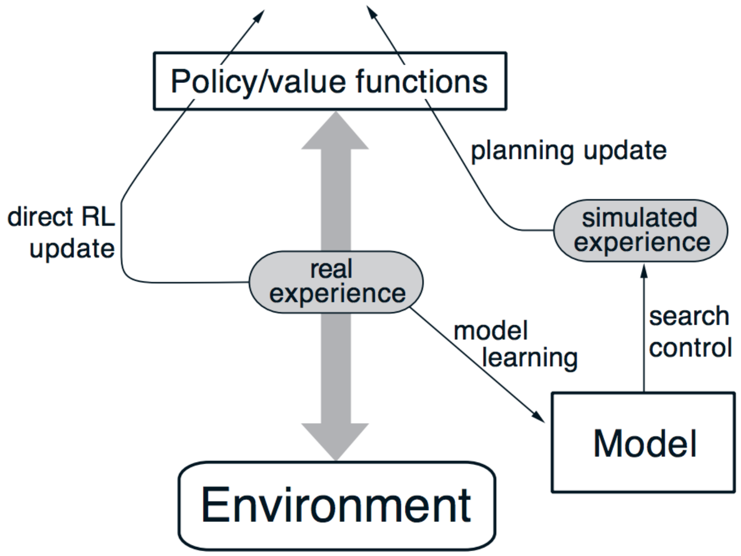


Advanced Model Based RL

- Planning Methods
- **Model Based with Model Free**

Dyna-Q: Model Based with Value Based

- Learn Q function and transition model
- Experience generate by transition model is also used to train Q function



Richard S. Sutton, 1990

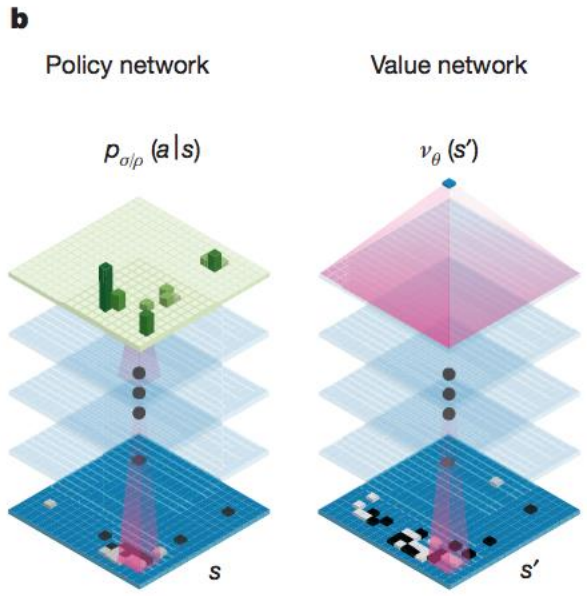
Model Based with Policy Based

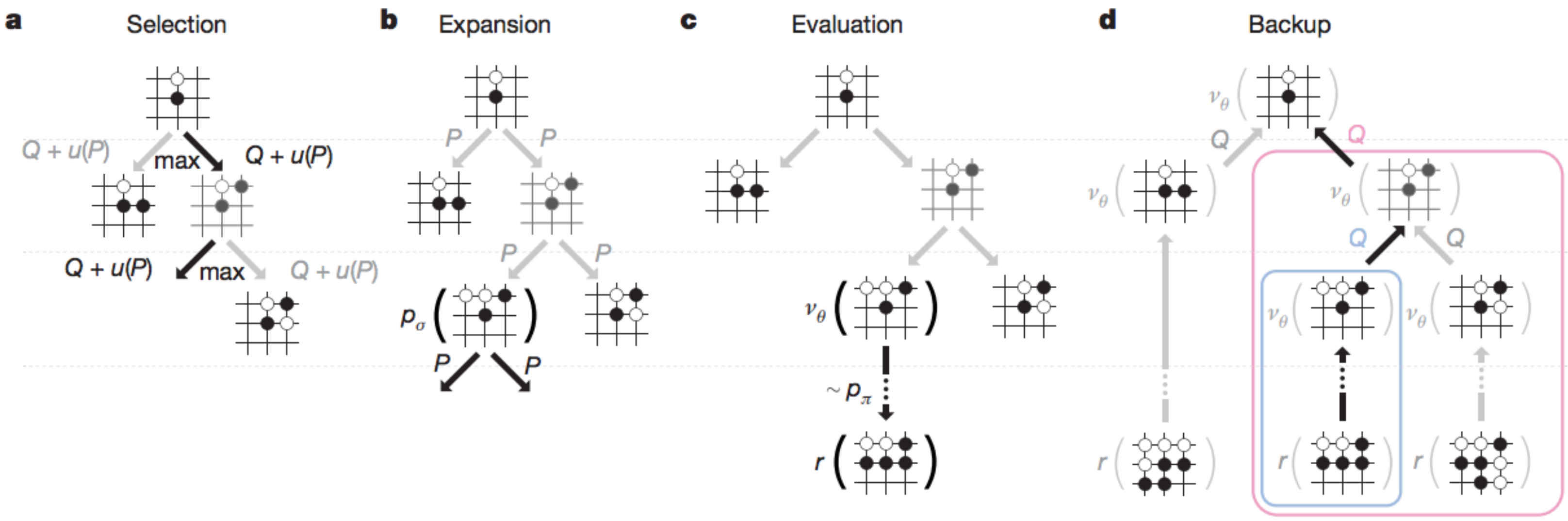
- Model Based DRL with Model-Free Fine-Tuning
  - Train model first
  - Use model(MPC) as expert to initialize Policy function
  - Policy Based RL afterwards

https://arxiv.org/abs/1708.02596

AlphaGo

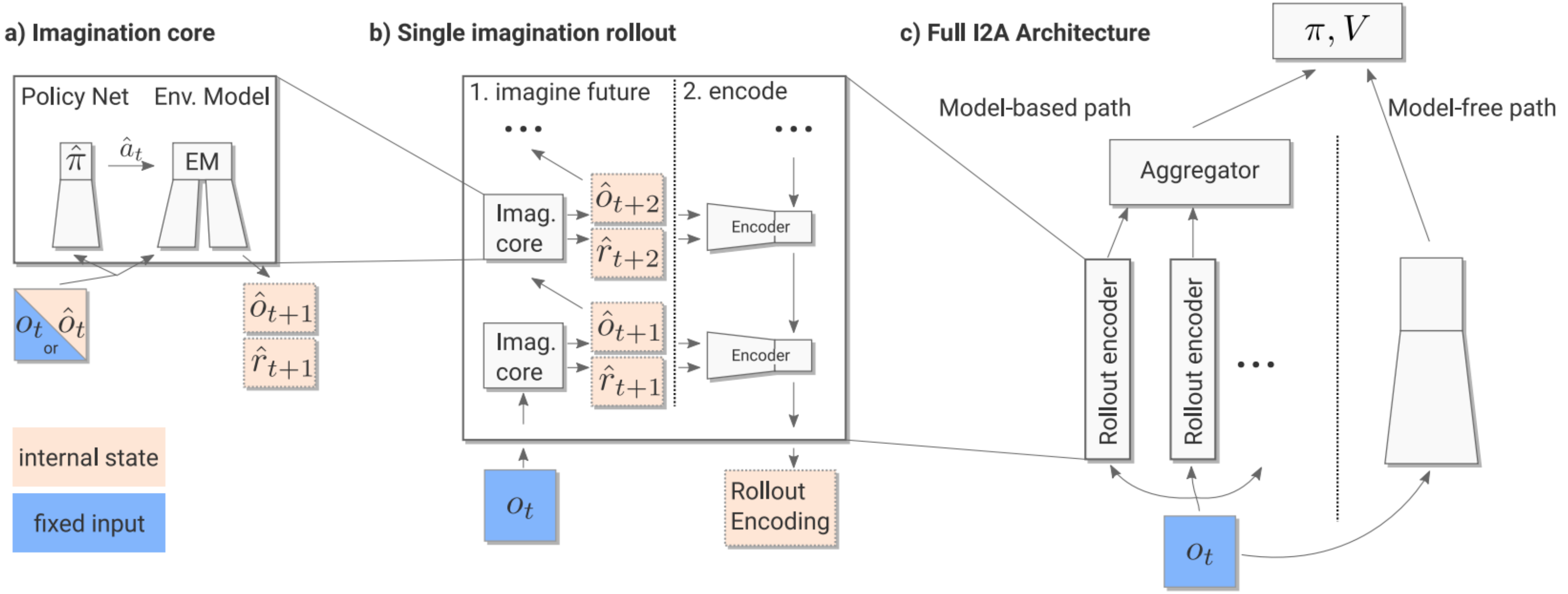
- Tree search into the future
- Value function to evaluate leaf node
- Policy function to choose moves





Imagination Augmented Agents

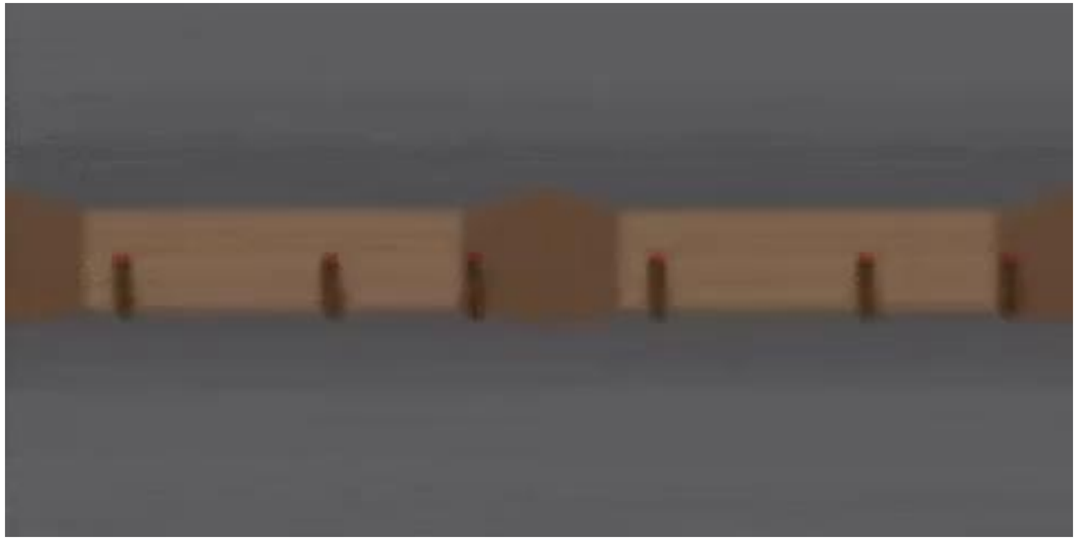
- Explicitly incorporates model prediction operation into policy network architecture



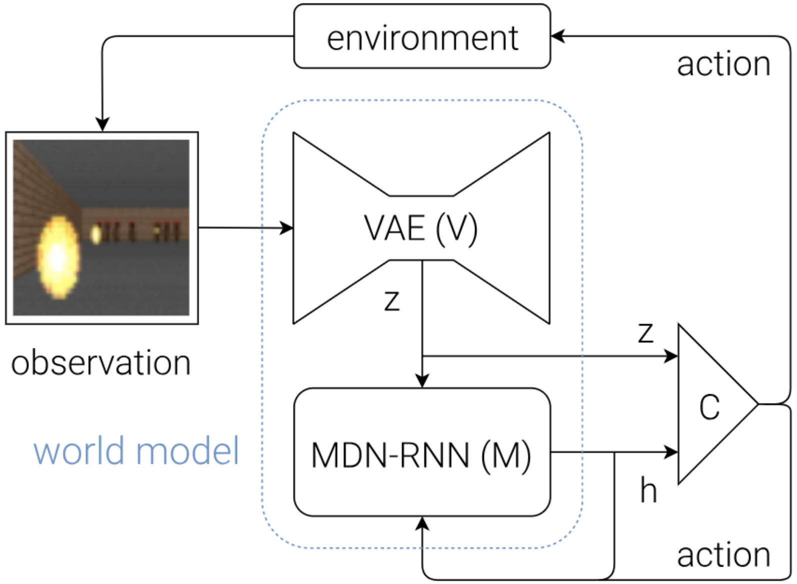
https://arxiv.org/abs/1707.06203

World Models

- Learn good latent state representation(with VAE)
- Train transition model on this latent state space
- Define & train simple controller(even with trajectory generated by transition model)



https://arxiv.org/pdf/1803.10122



- Recap and Concepts
- Reinforcement Learning Basics
- Advanced Reinforcement Learning
- **Challenges and Approaches**

Challenges and Approaches

- **Learning from Experts**
- Better Exploration
- Distributed Learning
- Hierarchical Methods
- Meta Learning

Learning From Experts

- Behavior cloning
  - Supervised Learning with data collected from experts
  - Cannot deal with states never seen before
- DAGGER
  - Query human experts for states unseen in previous dataset

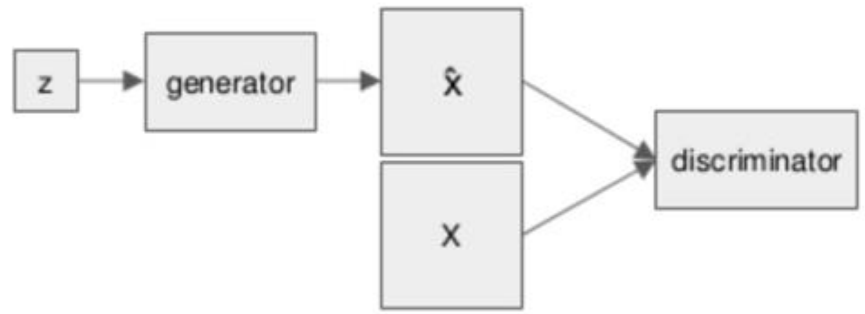
Learning From Experts

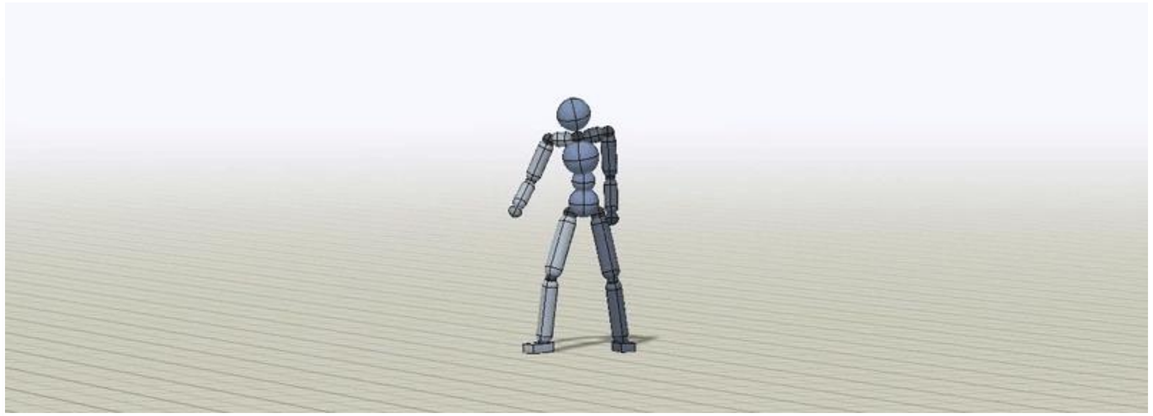
- Inverse Reinforcement Learning
  - Recover reward function under which the experts trajectories are optimal
  - Iteratively optimize reward function and policy function

Learning From Experts

- Generative based Imitation Learning
  - Generative Adversarial Imitation Learning
  - DeepMimic

$$D_w : \mathcal{S} \times \mathcal{A} \rightarrow (0, 1)$$





Challenges and Approaches

- Learning from Experts
- **Better Exploration**
- Distributed Learning
- Hierarchical Methods
- Meta Learning

Better Exploration

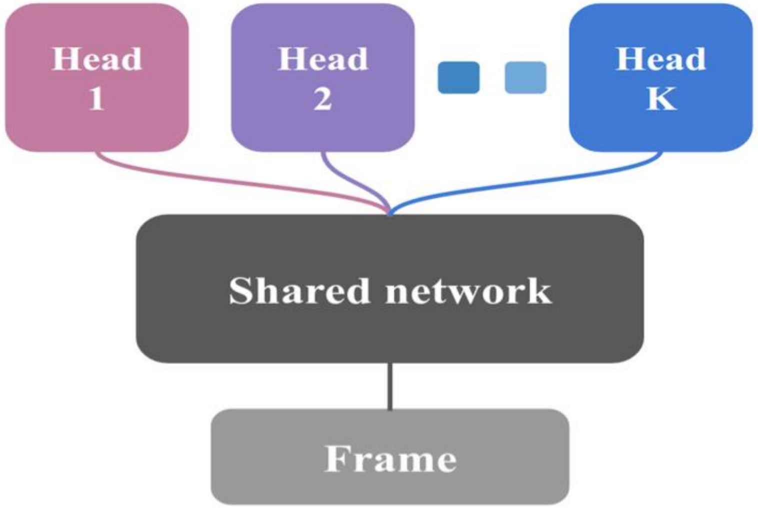
- Exploration/Exploitation Dilemma
  - Exploit knowledge learnt to achieve higher reward
  - Explore new strategy to avoid local maximum

Better Exploration

- Noise in the Action Space
  - Q learning: Epsilon-Greedy
  - Policy: Entropy
    - $\lambda H(\pi : \theta)$
  - Deterministic Continuous Action: additional noise
    - $A = U + \mathcal{N}$
    - $\mathcal{N}$ could be Gaussian or Ornstein-Uhlenbeck Noise

Better Exploration

- Noise in Policy Space
  - Bootstrapped DQN
    - Switching between heads for deep exploration
    - Stick to one head during entire episode
  - Noisy networks
    - Introduce noise in parameter space



Bootstrapped DQN

https://arxiv.org/pdf/1602.04621.pdf

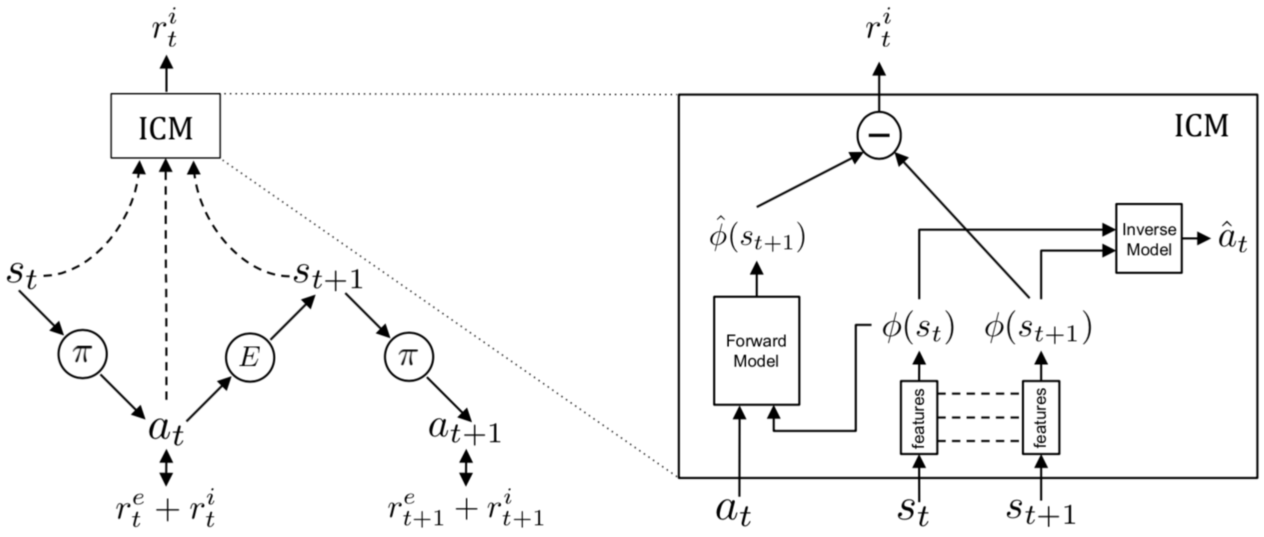
https://arxiv.org/pdf/1706.10295.pdf

Better Exploration

- Intrinsic Reward

$$\mathcal{R}_{\text{Bonus}}(s,a) = \mathcal{R}(s,a) + \beta\mathcal{N}(s,a)$$

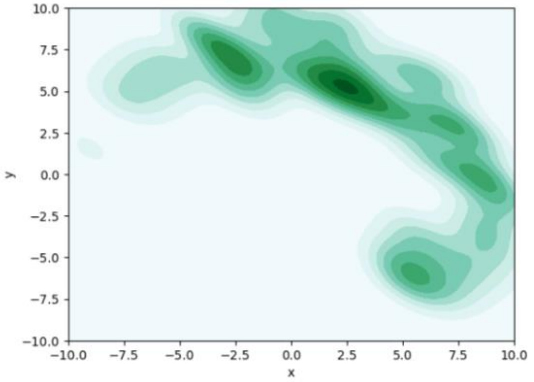
  - Curiosity driven exploration: $\mathcal{N}$ defined as loss of transition model



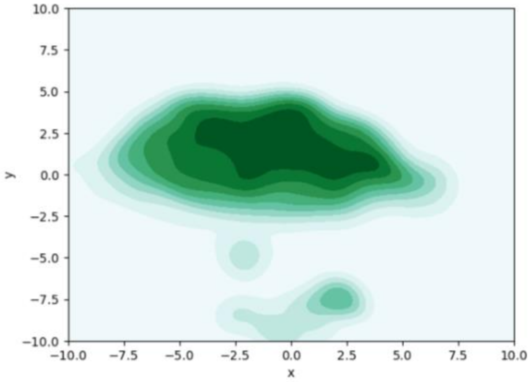
https://arxiv.org/abs/1705.05363

Better Exploration

- Learning to explore with meta-policy gradient
  - the policy gathering trajectories can be independent from the policy being optimized for the task
  - Meta reward: $\hat{\mathcal{R}}(D_0) = \hat{R}_{\pi'} - \hat{R}_{\pi}$



(d) Meta-Teacher (late)



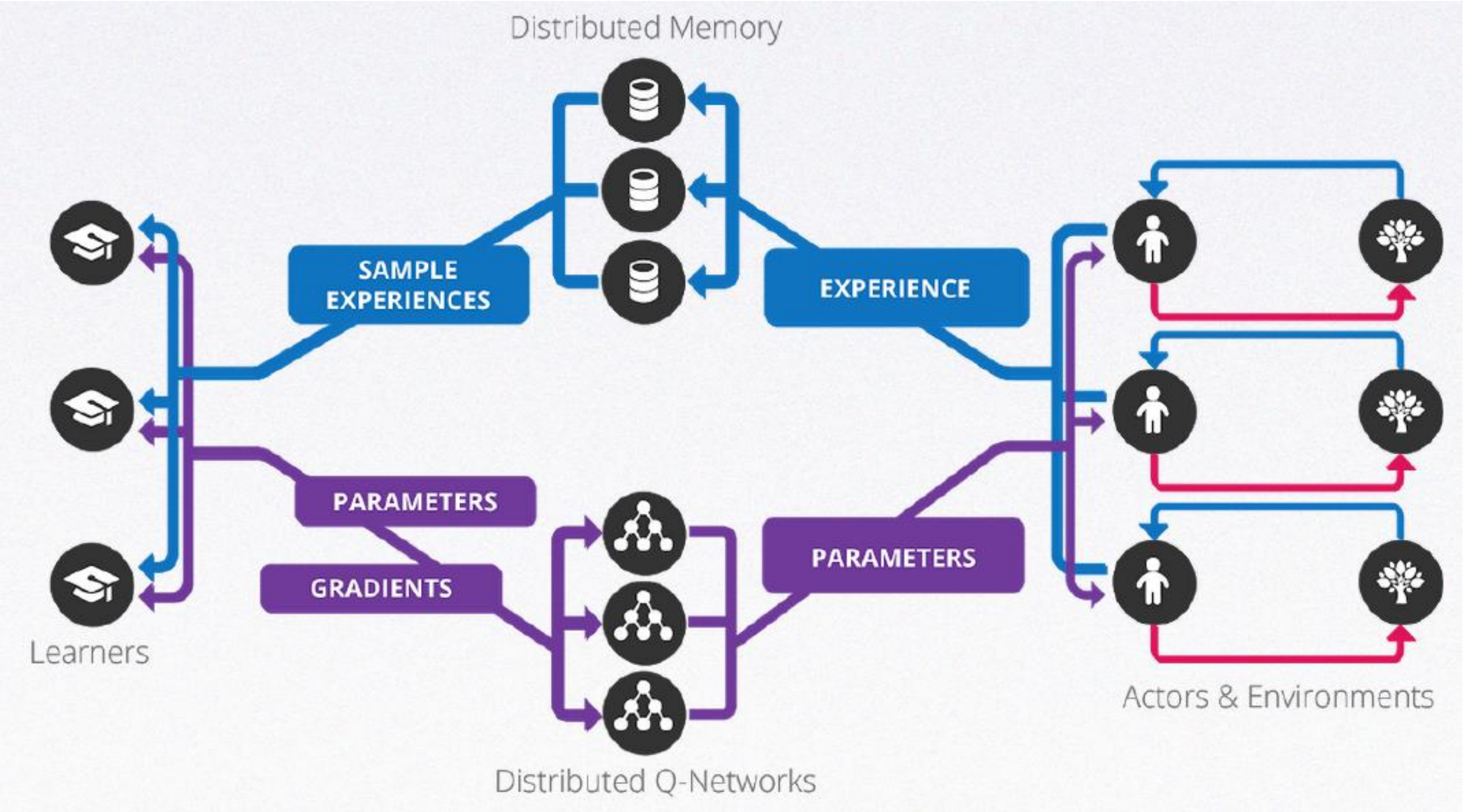
(e) Meta-Student (late)

Challenges and Approaches

- Learning from Experts
- Better Exploration
- **Distributed Learning**
- Hierarchical Methods
- Meta Learning

Distributed Learning

Gorilla

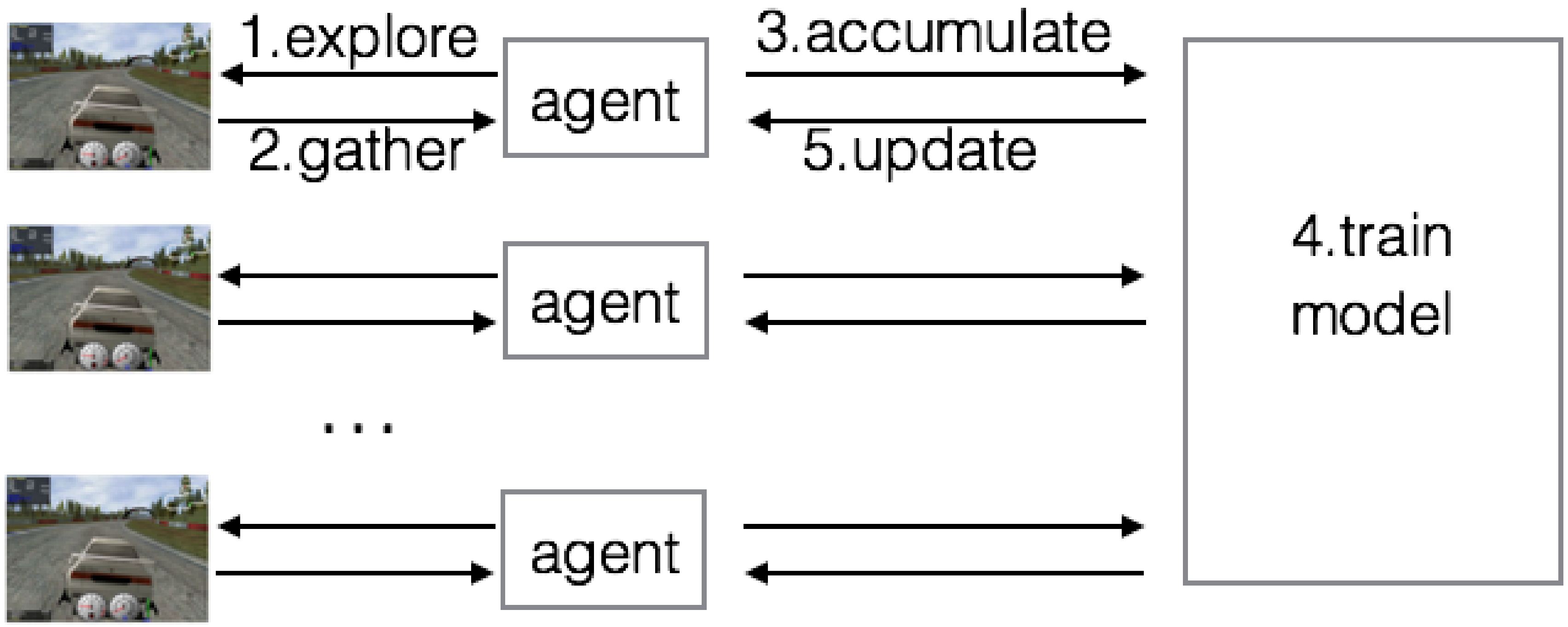


Every component can be distributed

https://arxiv.org/abs/1602.01783

Distributed Learning

- Asynchronous methods for Deep RL



https://arxiv.org/abs/1602.01783

Distributed Learning

- Asynchronous methods for Deep RL
  - Superlinear speedup!

	Number of threads				
Method	1	2	4	8	16
1-step Q	1.0	**3.0**	**6.3**	**13.3**	**24.1**
1-step SARSA	1.0	**2.8**	**5.9**	**13.1**	**22.1**
n-step Q	1.0	**2.7**	**5.9**	**10.7**	**17.2**
A3C	1.0	2.1	3.7	6.9	12.5

https://arxiv.org/abs/1602.01783

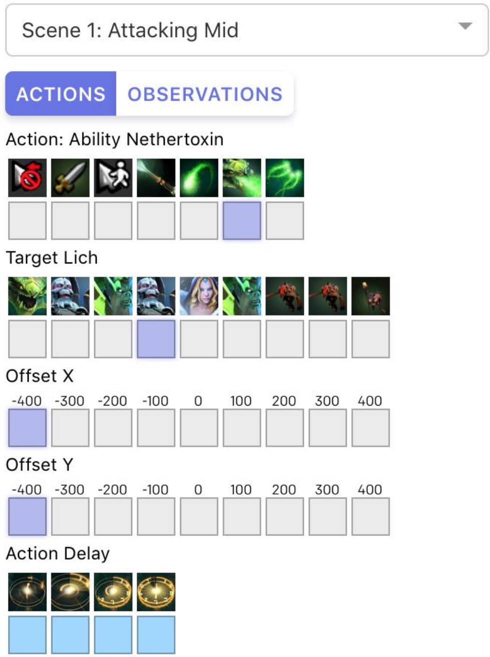
Distributed Learning

- AlphaGo Zero
  - 64 GPUs for optimization
  - 19 CPUs for parameter server

https://deepmind.com/blog/alphago-zero-learning-scratch/

Distributed Learning

- OpenAI Five
  - PPO
  - 12800 CPUs
  - 256 GPUs
  - OpenAI Rapid



https://blog.openai.com/openai-five/

Challenges and Approaches

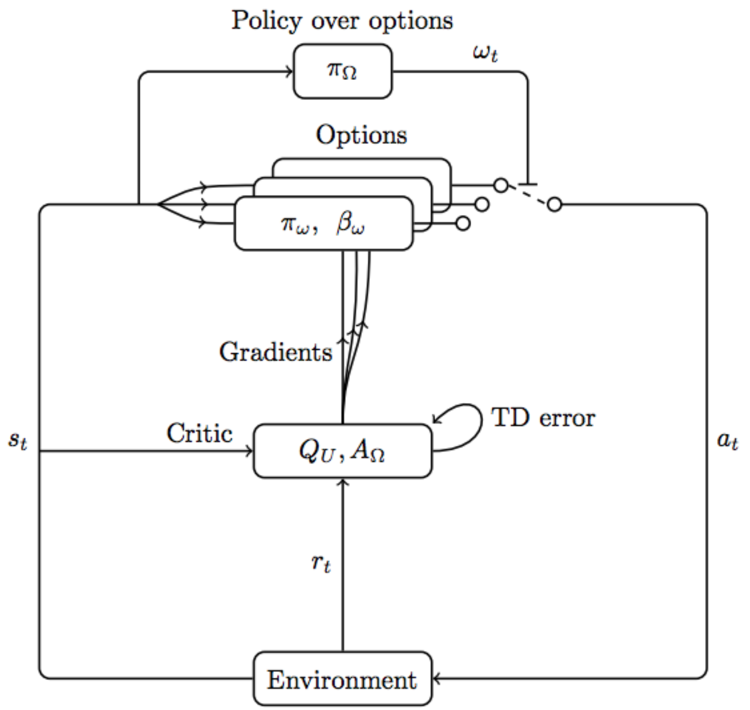
- Learning from Experts
- Better Exploration
- Distributed Learning
- **Hierarchical Methods**
- Meta Learning

Hierarchical Methods

- Complex tasks require decisions to be made
  - At different time scale
  - At different abstraction level
- …and can often be divided into simpler tasks handled by simpler policies

Hierarchical Methods

- Option Critic
  - Options defined as sub-policies
  - β_ω: termination condition
  - π_Ω: choose between options to complete task



https://arxiv.org/pdf/1609.05140.pdf

Hierarchical Methods

- Universal Function Approximators
  - Universal Value Function: $Q^{\pi}(s, a, g)$
  - Universal Policy Function: $\pi(s||g)$
  - Hindsight Experience Replay
    - If the agent is given goal g but end up in g', the agent still learns about how to achieve g'
  - Scheduled Auxiliary Control
    - Make sure the agent receive very different sensory observation at multiple dimension

Challenges and Approaches

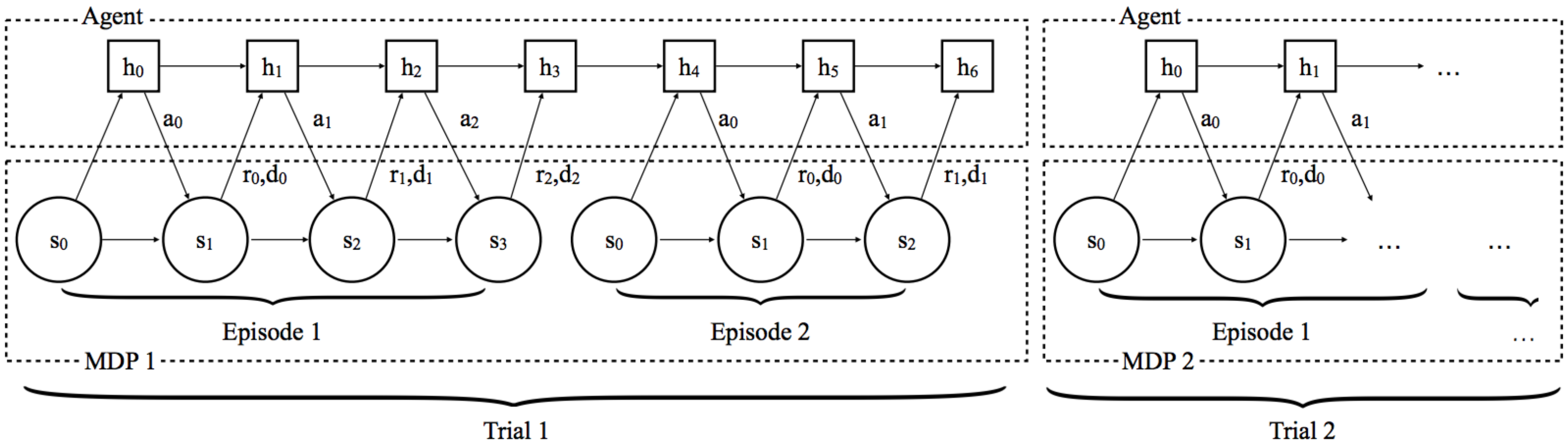
- Learning from Experts
- Better Exploration
- Distributed Learning
- Hierarchical Methods
- **Meta Learning**

Meta Learning

- RL is slow(at least for now)
- What learnt from previous tasks would hopefully accelerate learning of future tasks

Meta Learning

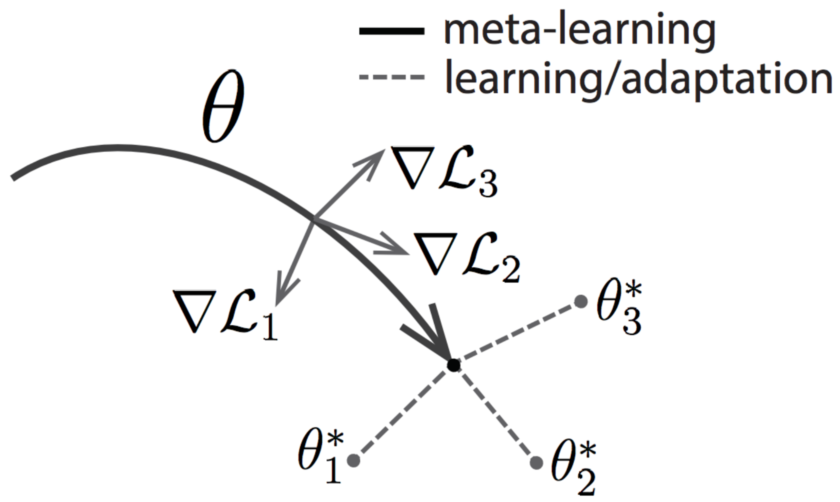
- Learning to Learn
  - RL^2: Fast RL via Slow RL
  - The agent learns about how to solve new tasks as fast as possible



https://arxiv.org/abs/1611.02779

Meta Learning

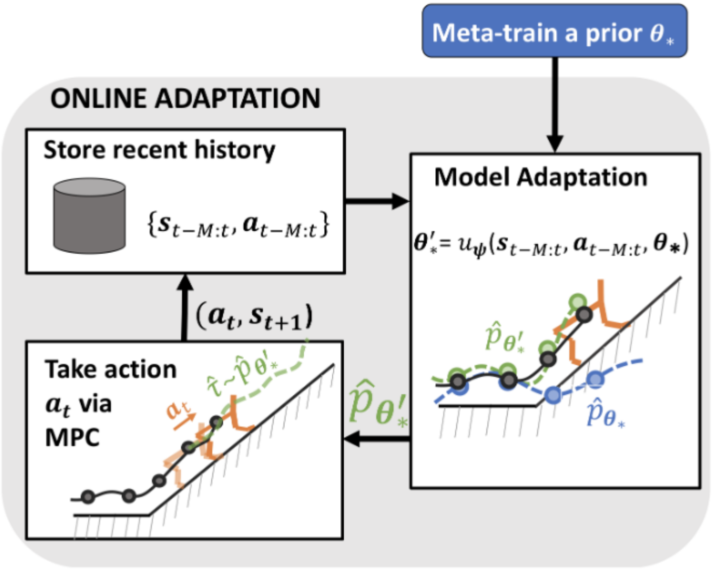
- Learning an initialization
  - Model-Agnostic Meta-Learning
    - Learns network parameter initialization for faster learning for a set of tasks



https://arxiv.org/abs/1703.03400

Meta Learning

- Learning an initialization
  - Learning to Adapt: Meta Learning for model based control
    - MAML for Model based RL
    - Learns about transition model parameter from recent interaction to the new environment



https://arxiv.org/abs/1803.11347

- Recap and Concepts
- Reinforcement Learning Basics
- Advanced Reinforcement Learning
- Challenges and Approaches

One More Thing

Practical Advices

Engineering is Very Important

Understand Your Task

Verify in Toy Tasks First

Look Closely into Data

Good Luck!

- Engineering is very important
- Understand your task
- Verify in toy tasks first
- Look closely into data

Q&A

Thank you.

And welcome:

dream@hobot.cc