Introduction to Deep Reinforcement Learning

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

DRL Lecture Contents

- A brief review of RL

- Deep RL
  - What happens when RL modules are implemented with DNNs?
  - DQN, TRPO and DDPG

- Multi-agent DRL
  - Key challenges
  - Some featured multi-agent simulators

REVIEW

Machine Learning Categories

- Supervised Learning
  - To perform the desired output given the data and labels

$$p(y|x)$$

- Unsupervised Learning
  - To analyze and make use of the underlying data patterns/structures

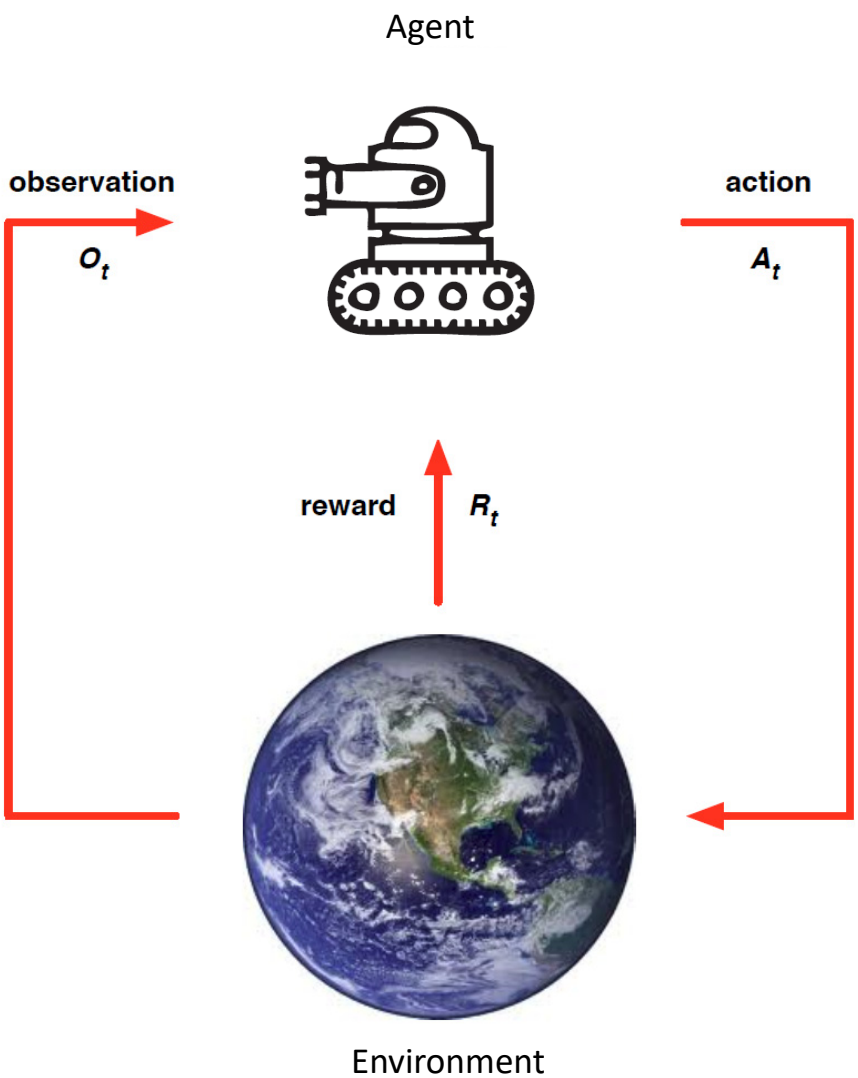
$$p(x)$$

- Reinforcement Learning
  - To learn a policy of taking actions in a dynamic environment and acquire rewards

$$\pi(a|x)$$

Reinforcement Learning

- Learning from interaction with the environment
- The agent
  - senses the observations from environment
  - takes actions to deliver to the environment
  - gets reward signal from the environment
- Normally, the environment is stationary



Elements of RL Systems

- History is the sequence of observations, action, rewards

$$H_t = O_1, R_1, A_1, O_2, R_2, A_2, \ldots, O_{t-1}, R_{t-1}, A_{t-1}, O_t, R_t$$

  - i.e. all observable variables up to time *t*
  - E.g., the sensorimotor stream of a robot or embodied agent

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

$$S_t = f(H_t)$$

Elements of RL Systems

- Policy is the learning agent's way of behaving at a given time
  - It is a map from state to action
  - Deterministic policy

$$a = \pi(s)$$

  - Stochastic policy

$$\pi(a|s) = P(A_t = a|S_t = s)$$

Elements of RL Systems

- Reward
  - A scalar defining the goal in an RL problem
  - For immediate sense of what is good

- Value function
  - State value is a scalar specifying what is a good state in the long run, i.e., the cumulative reward

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]$$

  - Action value is a scalar specifying what is a good action at a specific state in the long run

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a]$$

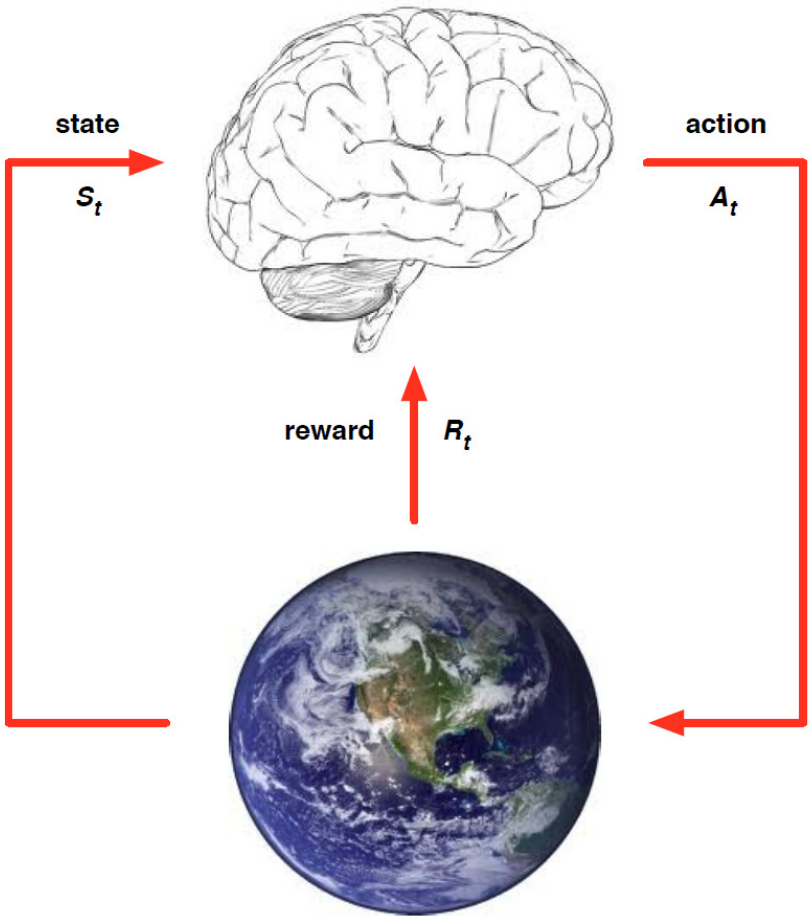
Elements of RL Systems

- A Model of the environment that mimics the behavior of the environment
  - Predict the next state

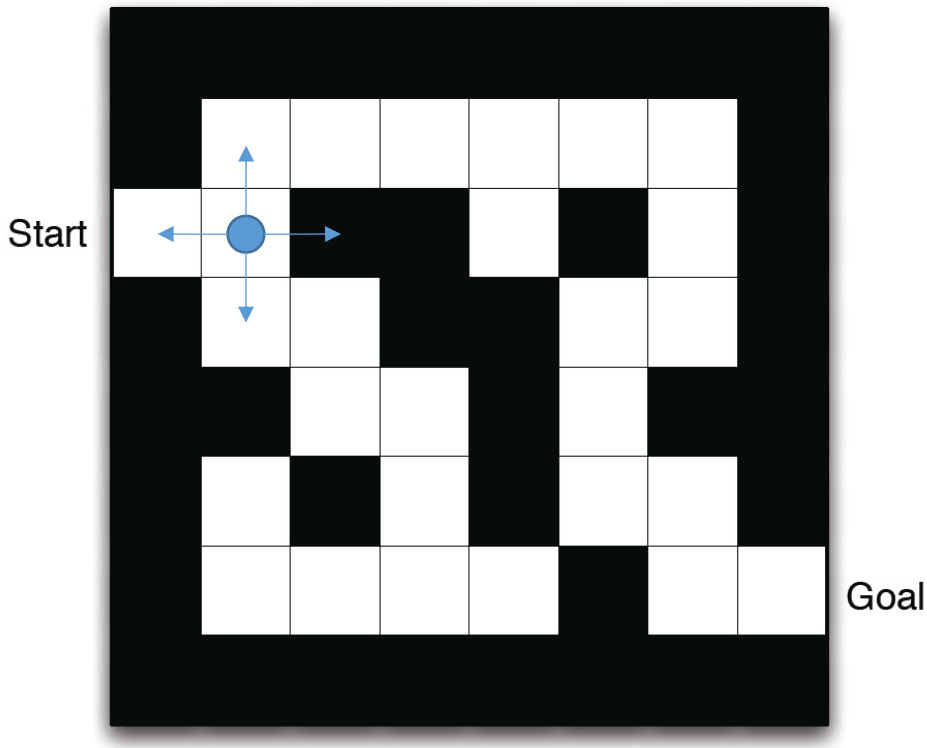
$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

  - Predicts the next (immediate) reward

$$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

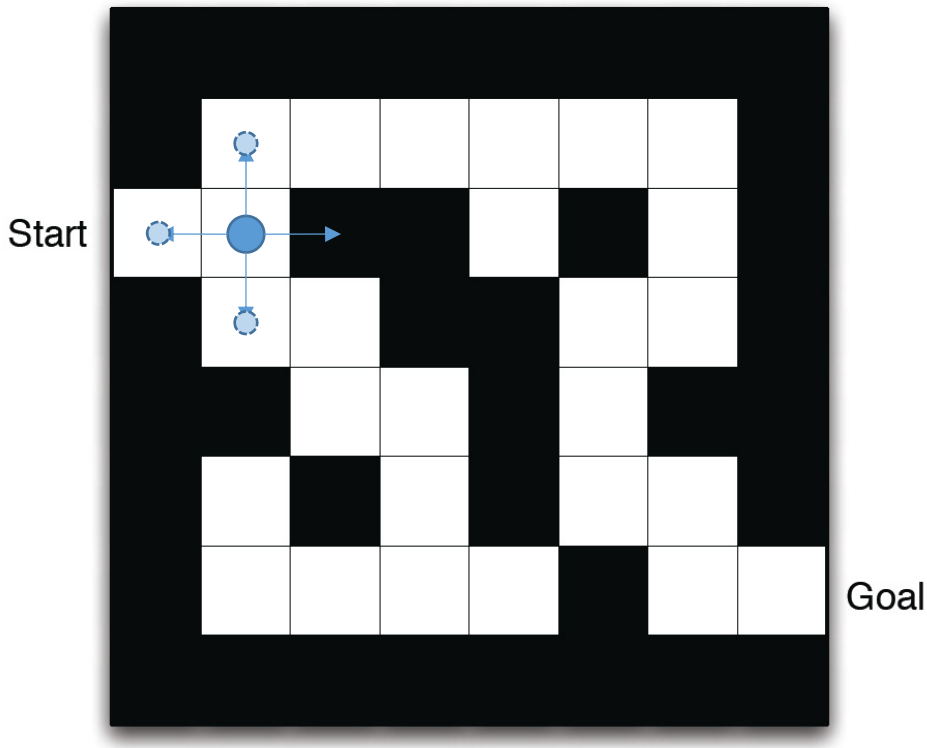


Maze Example



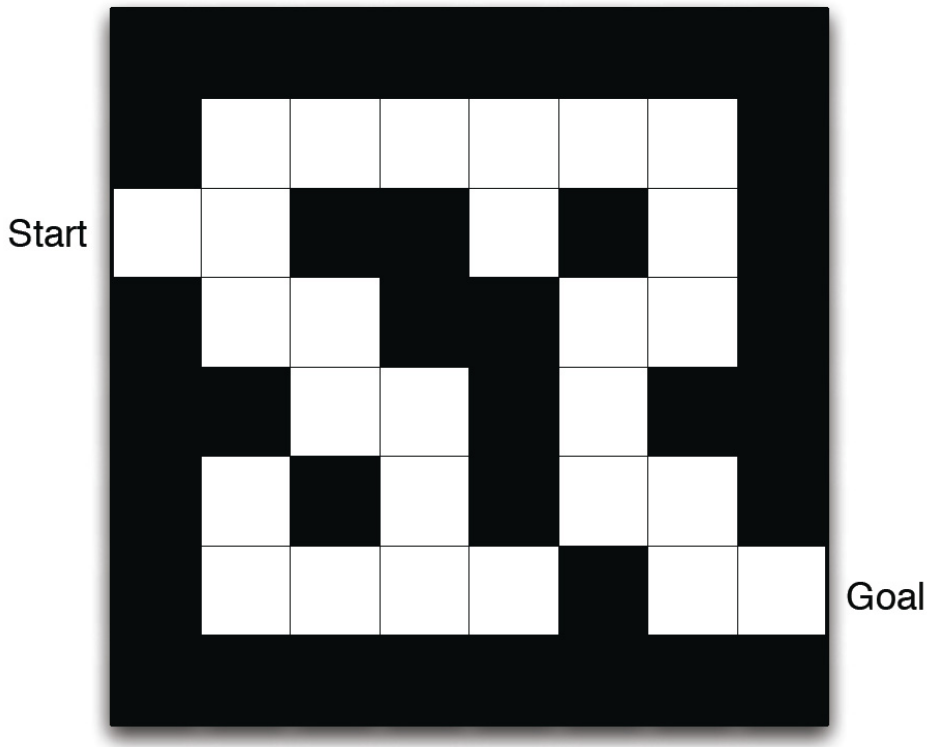
- State: agent's location
- Action: N,E,S,W

Maze Example



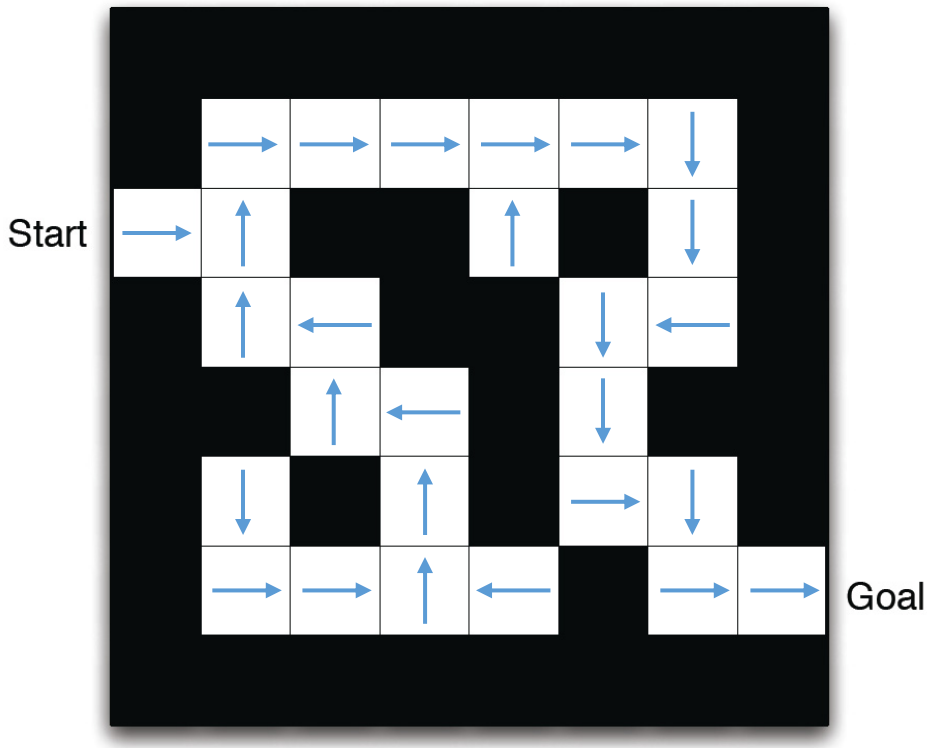
- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
  - No move if the action is to the wall

Maze Example



- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
- Reward: -1 per time step

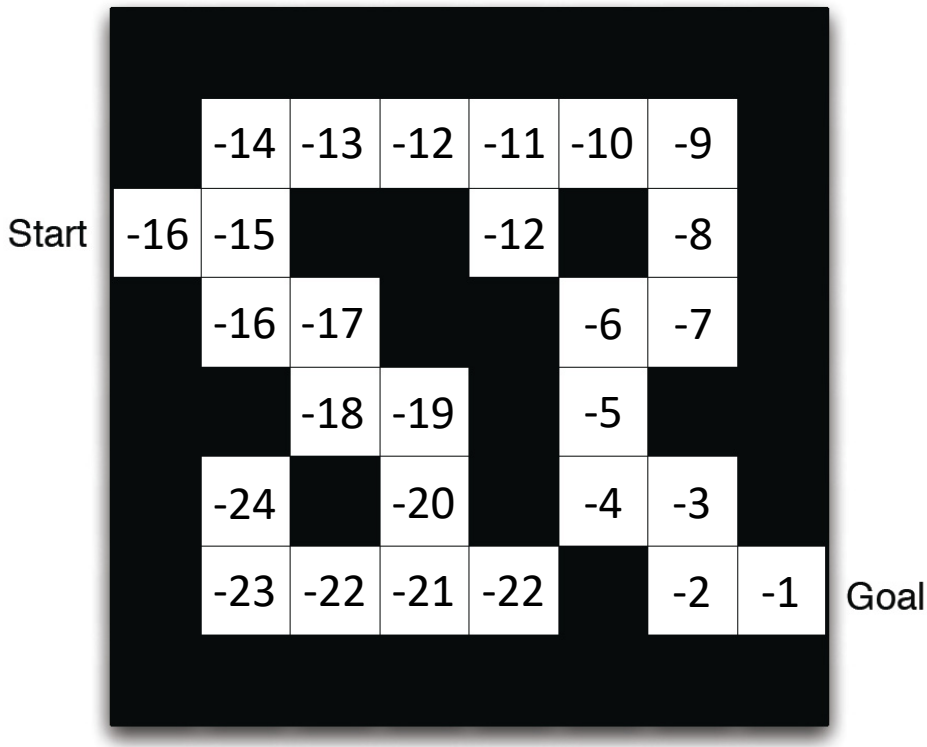
Maze Example



- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
- Reward: -1 per time step

- Given a policy as shown above
  - Arrows represent policy $\pi(s)$ for each state s

Maze Example



- State: agent's location
- Action: N,E,S,W
- State transition: move to the next grid according to the action
- Reward: -1 per time step

- Numbers represent value $v_\pi(s)$ of each state s

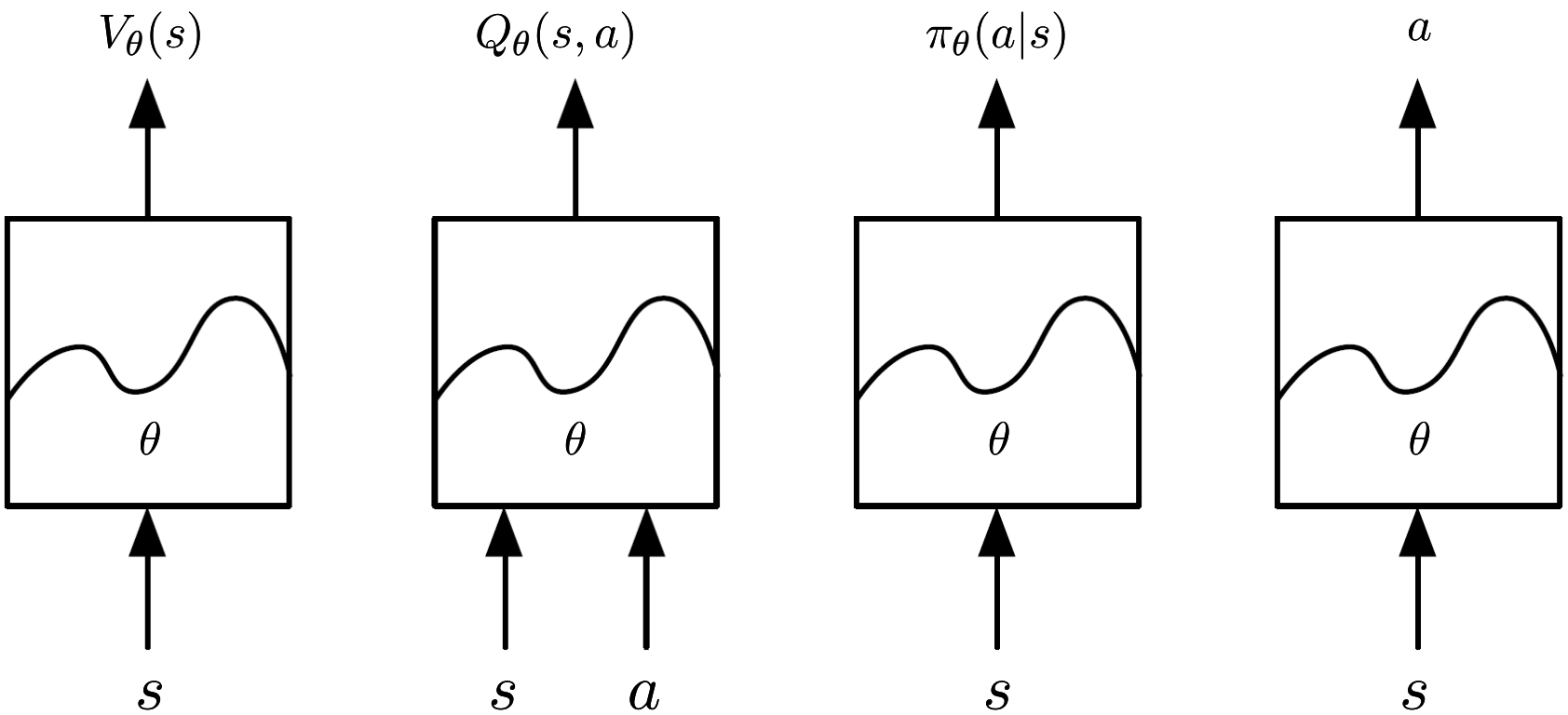
DRL Lecture Contents

- A brief review of RL

- Deep RL
  - What happens when RL modules are implemented with DNNs?
  - DQN, TRPO and DDPG

- Multi-agent DRL
  - Key challenges
  - Some featured multi-agent simulators

Value and Policy Approximation



State-value and action-value approx.

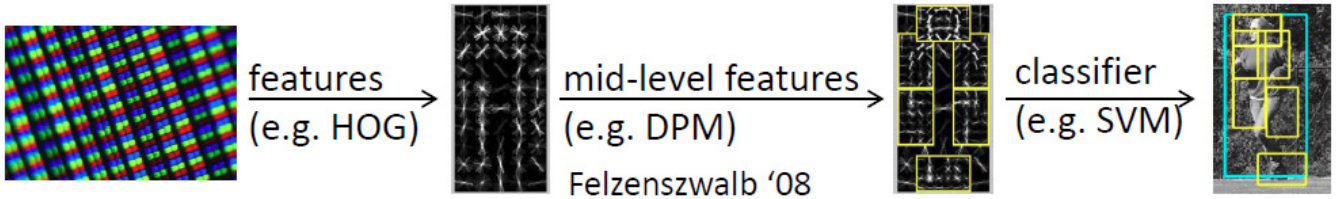
Stochastic policy approx.

Deterministic policy approx.

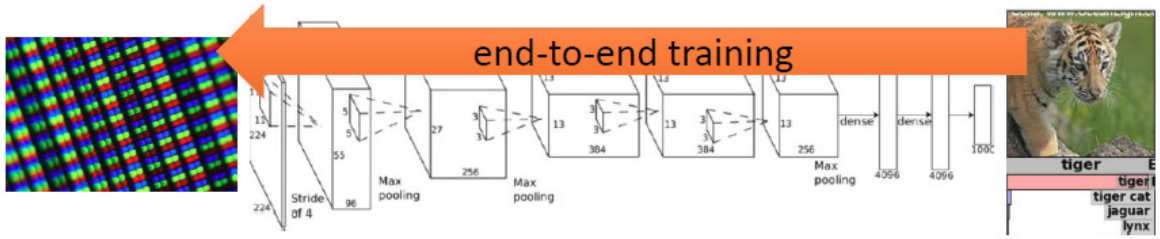
- What if we directly build these approximate function with deep neural networks?

End-to-End Reinforcement Learning

Standard computer vision

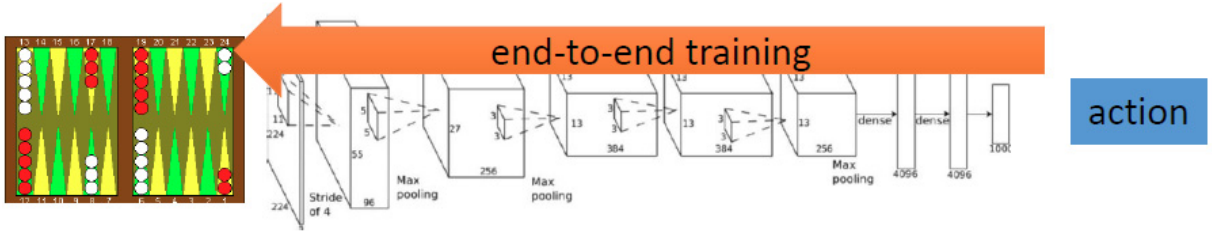


Deep learning



Standard reinforcement learning

Deep reinforcement learning

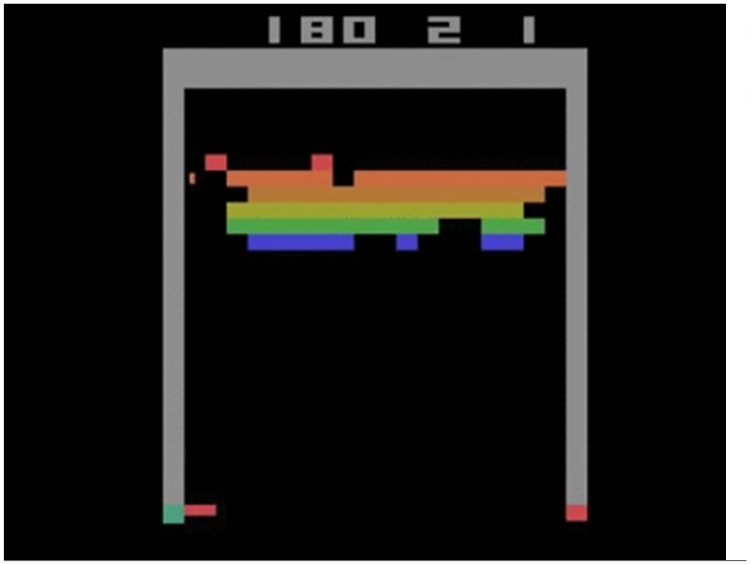


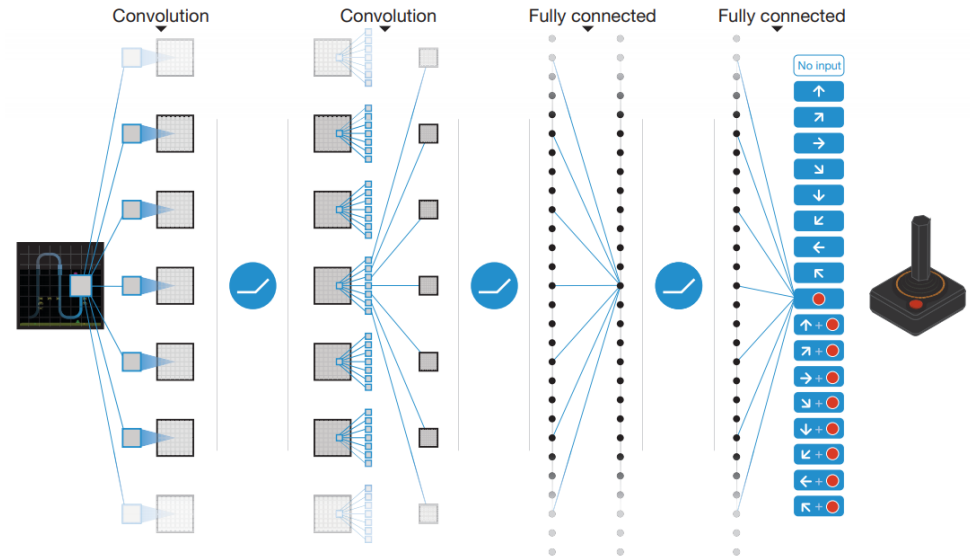
Deep Reinforcement Learning is what allows RL algorithms to solve complex problems in an end-to-end manner.

Slide from Sergey Levine. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf

Deep Reinforcement Learning

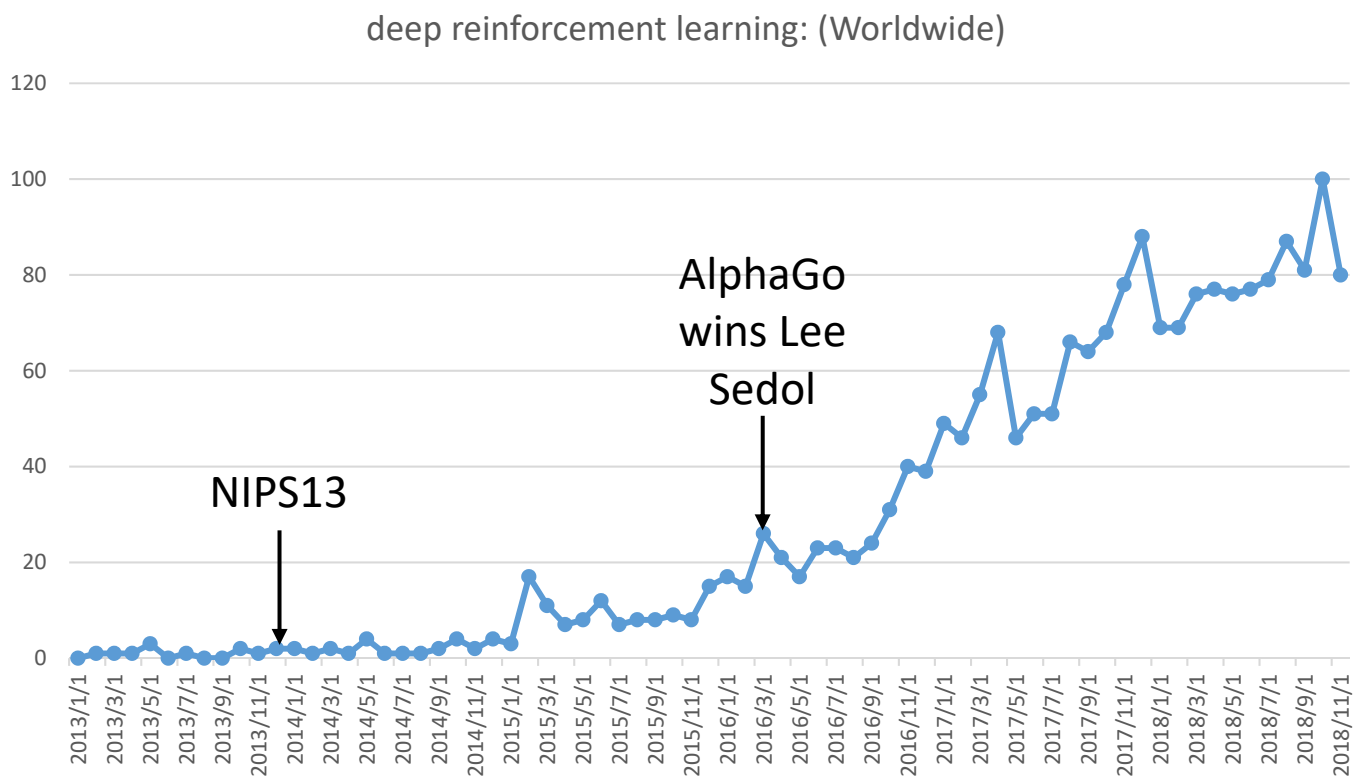
- Deep Reinforcement Learning
  - leverages deep neural networks for value functions and policies approximation
  - so as to allow RL algorithms to solve complex problems in an end-to-end manner.





Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

Deep Reinforcement Learning Trends



- Google search trends of the term ‘deep reinforcement learning’

Key Changes Brought from DRL

- What will happen when combining DL and RL?
  - Value functions and policies are now deep neural nets
  - Very high-dimensional parameter space
  - Hard to train stably
  - Easy to overfit
  - Need a large amount of data
  - Need high performance computing
  - Balance between CPUs (for collecting experience data) and GPUs (for training neural networks)
  - …
- These new problems motivate novel algorithms for deep reinforcement learning

REVIEW

Q-Learning

- For off-policy learning of action-value $Q(s,a)$
- The next action is chosen using behavior policy $a_{t+1} \sim \mu(\cdot|s_t)$
- But we consider alternative successor action $a \sim \pi(\cdot|s_t)$
- And update $Q(s_t,a_t)$ towards value of alternative action

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

action
from π
not μ

REVIEW

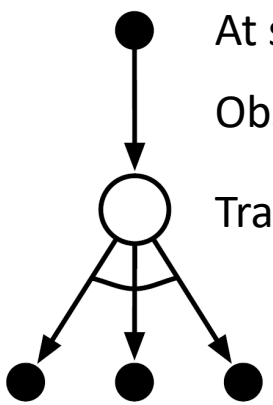
Off-Policy Control with Q-Learning

- Allow both behavior and target policies to improve
- The target policy π is greedy w.r.t. $Q(s,a)$

$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a')$$

- Q-learning update

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



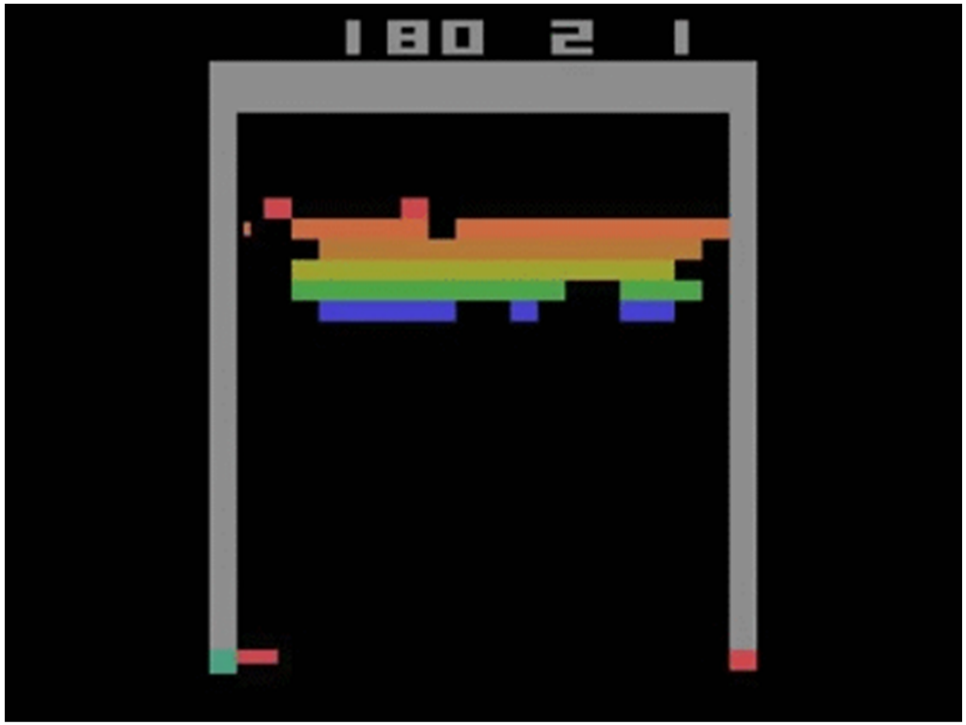
At state *s*, take action *a*

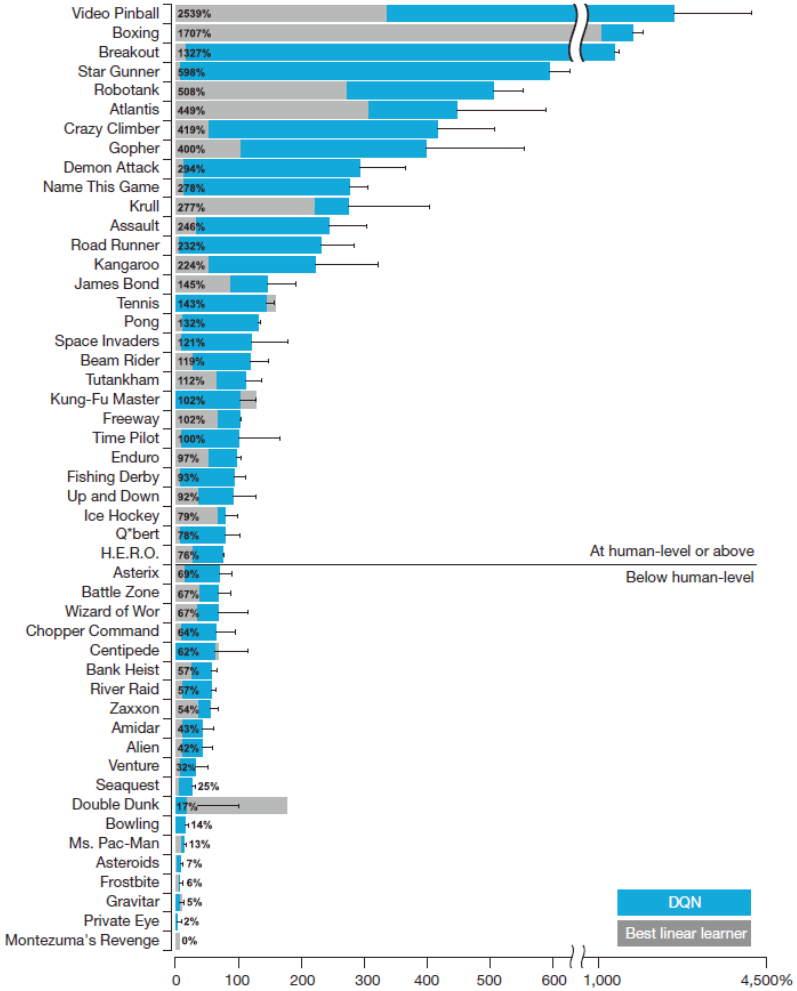
Observe reward *r*

Transit to the next state *s'*

At state *s'*, take action argmax *Q*(s',*a'*)

Deep Q-Network (DQN)



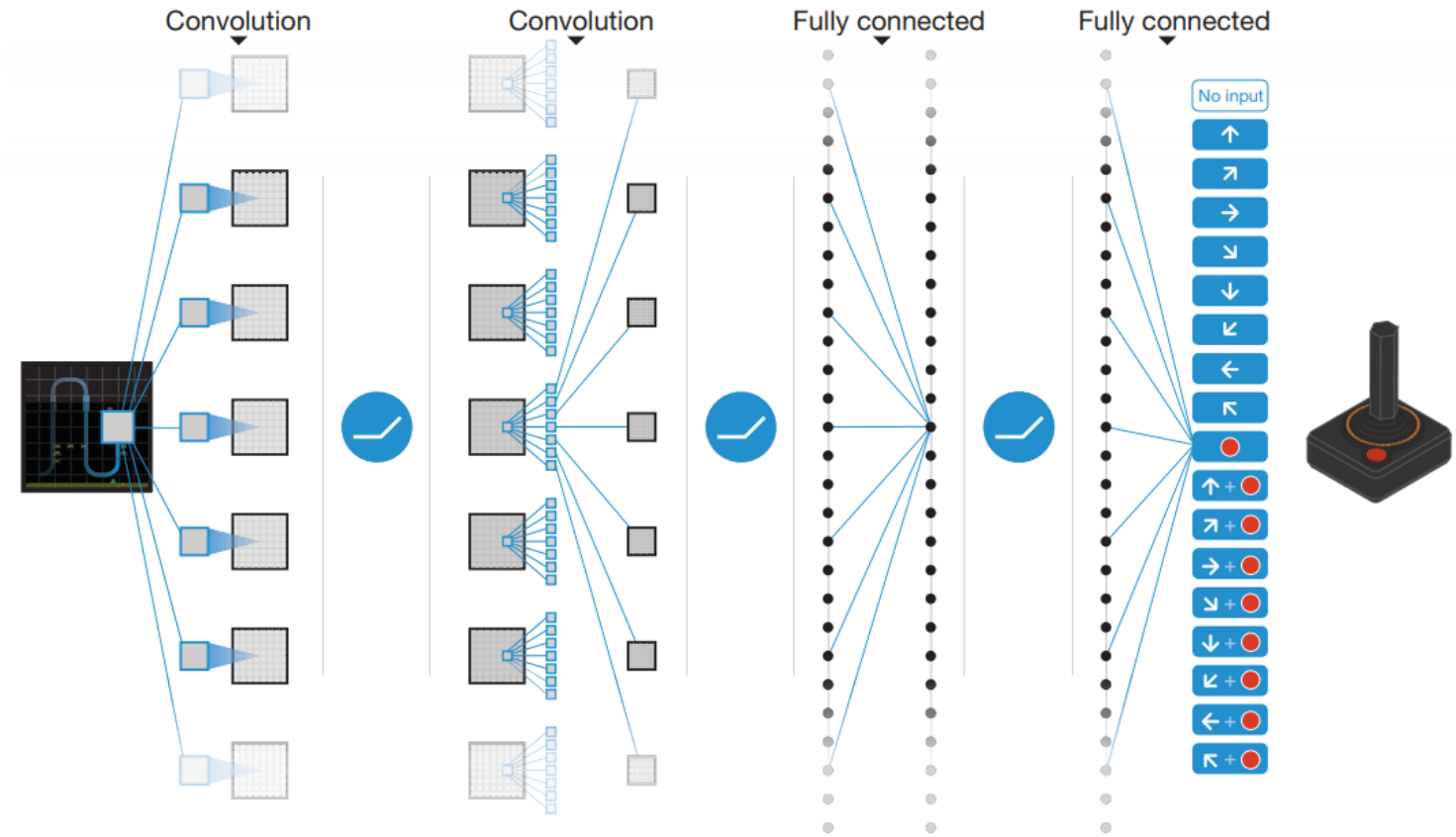


DQN (NIPS 2013) is the beginning of the entire deep reinforcement learning sub-area.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.
Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

Deep Q-Network (DQN)

- Implement Q function with deep neural network
  - Input a state, output Q values for all actions



Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

Deep Q-Network (DQN)

- The loss function of Q-learning update at iteration *i*

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\Big[\big(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{target Q value}} - \underbrace{Q(s, a; \theta_i)}_{\text{estimated Q value}}\big)^2\Big]$$

  - ϑ_i are the network parameters to be updated at iteration *i*
    - Updated with standard back-propagation algorithms
  - ϑ_i^- are the target network parameters
    - Only updated with ϑ_i for every *C* steps
  - $(s,a,r,s')\sim U(D)$: the samples are uniformly drawn from the experience pool *D*
    - Thus to avoid the overfitting to the recent experiences

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

Deep Q-Network (DQN)

- The loss function of Q-learning update at iteration *i*

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\Big[\big(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{target Q value}} - \underbrace{Q(s, a; \theta_i)}_{\text{estimated Q value}}\big)^2\Big]$$

- For each experience (*s*,*a*,*r*,*s'*)~*U*(*D*), the gradient is

$$\theta_{i+1} = \theta_i + \eta\big(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\big)\underbrace{\nabla_\theta Q(s, a; \theta_i)}_{\text{backpropagation}}$$

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

Policy Gradient Theorem

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right]$$

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
  - Optimize the long-term value
- Theorem
  - For any differentiable policy $\pi_\theta(a|s)$, for any of policy objective function $J = J_1, J_{avR}, J_{avV}$, the policy gradient is

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

Please refer to appendix of the slides for detailed proofs

Policy Network Gradients

- For stochastic policy, typically the action probability is defined as a softmax

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

  - where $f_\vartheta(s,a)$ is the score function of a state-action pair parametrized by ϑ, which can be implemented with a neural net

- The gradient of its log-form

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \frac{1}{\sum_{a'} e^{f_\theta(s,a')}} \sum_{a''} e^{f_\theta(s,a'')} \frac{\partial f_\theta(s,a'')}{\partial \theta}$$

$$= \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)} \left[\frac{\partial f_\theta(s,a')}{\partial \theta} \right]$$

Policy Network Gradients

- With the gradient form

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]$$

- The policy network gradient is

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\left(\underbrace{\frac{\partial f_\theta(s,a)}{\partial \theta}}_{\text{backpropagation}} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\underbrace{\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]}_{\text{backpropagation}}\right) Q^{\pi_\theta}(s,a)\right]\end{aligned}$$

Trust-Region Policy Optimization

$$A^{\pi_{\text{old}}}(s,a) = \mathbb{E}_{s' \sim \rho(s'|s,a)}[r(s) + \gamma V^{\pi_{\text{old}}}(s') - V^{\pi_{\text{old}}}(s)]$$

$$L(\pi) = \mathbb{E}_{s \sim \pi_{\text{old}}, a \sim \pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a)\right]$$

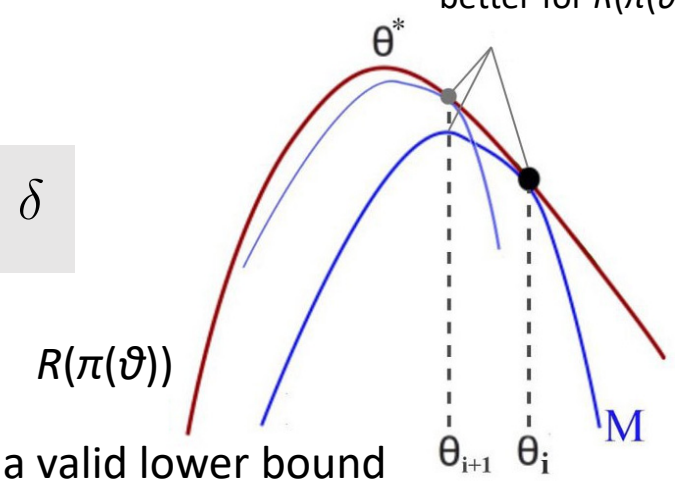
Optimize the approximated policy value with constraints

$$\max_{\pi} \; L_{\pi_{\text{old}}}(\pi)$$

$$\text{s.t.} \; \mathbb{E}_{s \sim \rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s) \| \pi(\cdot|s)))] \leq \delta$$

Trust region

Better for M must be better for $R(\pi(\vartheta))$



To guarantee the following inequality to make *M* a valid lower bound

$$\left|R(\pi) - (R(\pi_{\text{old}}) + L_{\pi_{\text{old}}}(\pi))\right| \leq C\sqrt{\mathbb{E}_{s \sim \rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s) \| \pi(\cdot|s)))]}$$

Schulman, John, et al. "Trust region policy optimization." *International Conference on Machine Learning*. 2015.

Trust-Region Policy Optimization

Line search
(like gradient ascent)

Optimization in Trust Region

https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9

Stochastic vs. Deterministic Policies

- Stochastic policy

for discrete actions $\pi(a|s;\theta) = \dfrac{\exp\{Q_\theta(s,a)\}}{\sum_{a'} \exp\{Q_\theta(s,a')\}}$

for continuous actions $\pi(a|s;\theta) \propto \exp\{(a - \mu_\theta(s))^2\}$

- Deterministic policy

for discrete actions $\pi(s;\theta) = \arg\max_a Q_\theta(s,a)$

(non-differentiable)

for continuous actions $a = \pi_\theta(s)$ (can be differentiable)

Deterministic Policy Gradient

- A critic module for state-action value estimation

$$Q^w(s,a) \simeq Q^\pi(s,a)$$

$$L(w) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[(Q^w(s,a) - Q^\pi(s,a))^2\right]$$

- With the differentiable critic, the deterministic continuous-action actor can be updated as
  - Deterministic policy gradient theorem

$$J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[Q^\pi(s,a)]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)}]$$

On-policy Chain rule

D. Silver et al. Deterministic Policy Gradient Algorithms. ICML 2014.

DDPG: Deep Deterministic Policy Gradient

- For deterministic policy gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)}]$$

- In practice, a naive application of this actor-critic method with neural function approximators is unstable for challenging problems
- DDPG solutions over DPG
  - Experience replay (off-policy)
  - Target network
  - Batch normalization on Q network prior to the action input
  - Add noise on continuous actions

Lillicrap et al. Continuous control with deep reinforcement learning. NIPS 2015.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $\boxed{a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t}$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions $\boxed{(s_i, a_i, r_i, s_{i+1}) \text{ from } R}$
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
    Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Noise on action

Off-policy

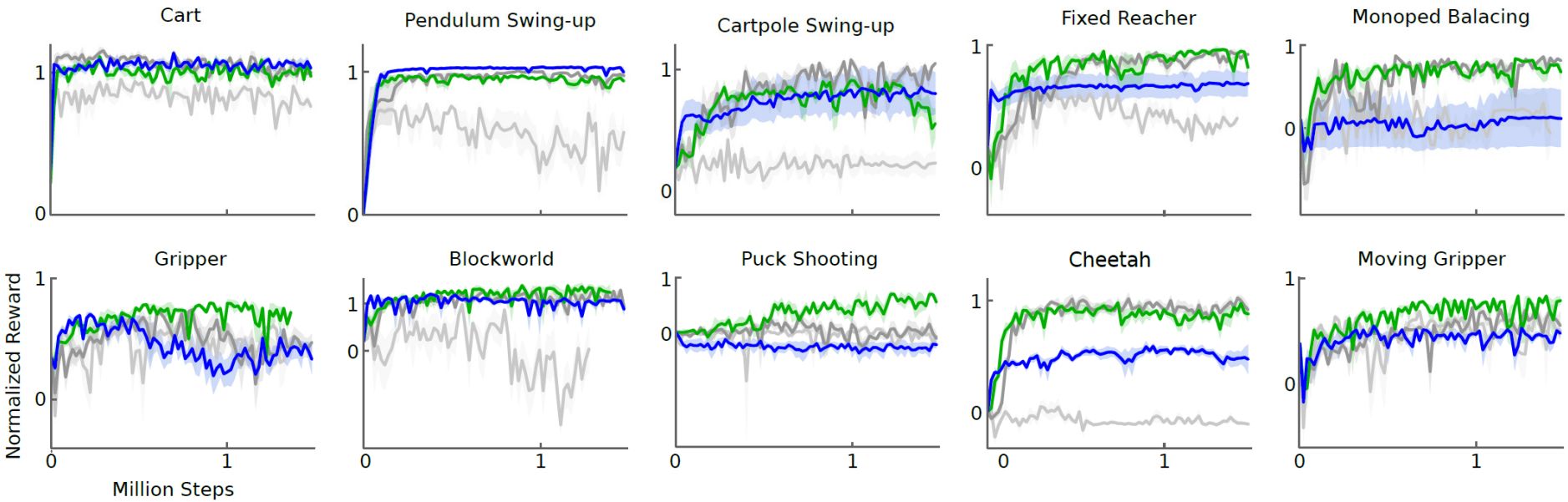
Update critic net

Target critic network

Target actor network

Update actor net

DDPG Experiments



- Performance curves for a selection of domains using variants of DPG
  - Light grey: original DPG algorithm with batch normalization
  - Dark grey: with target network
  - Green: with target networks and batch normalization
  - Blue: with target networks from pixel-only inputs.
- Target networks are crucial.

Lillicrap et al. Continuous control with deep reinforcement learning. NIPS 2015.

Deep Reinforcement Learning Categories

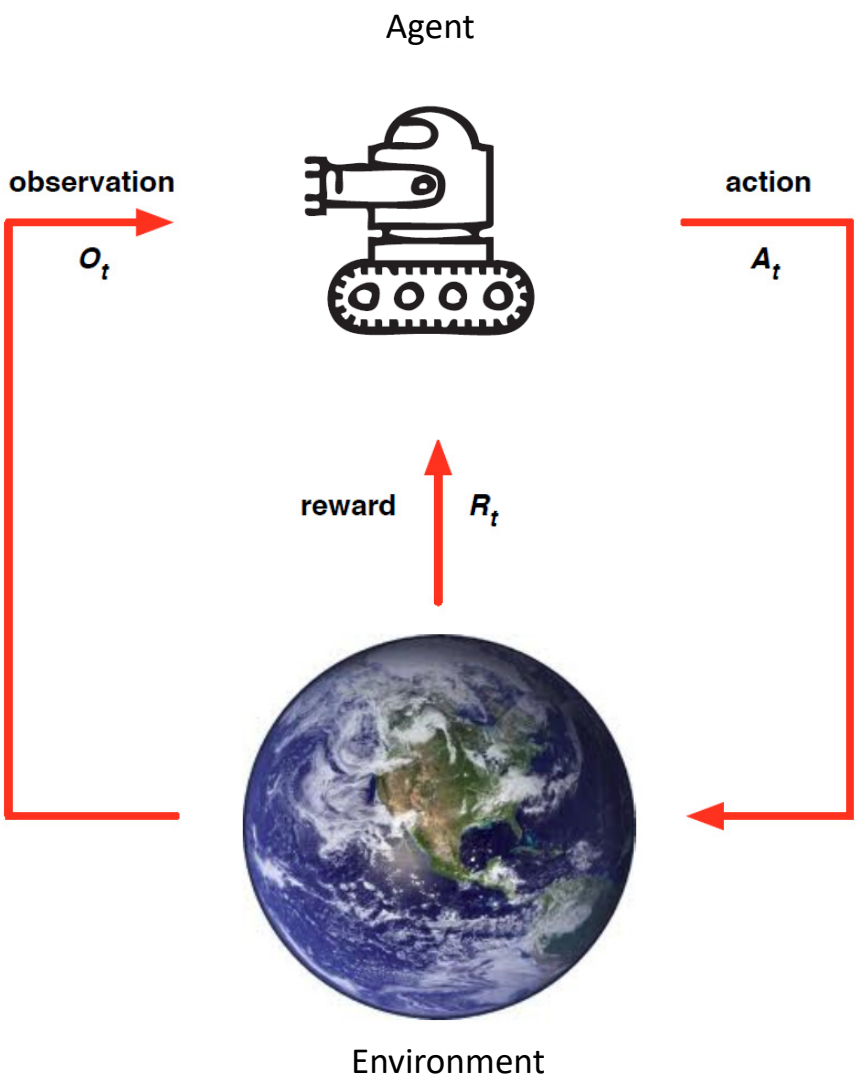
- DRL = RL + DL
  - One of the most challenging problems in machine learning with very fast develop during the recent 5 years
- Value-based methods
  - Deep Q-network and its extensions
- Stochastic policy-based methods
  - Policy gradients with NNs, natural policy gradient, trust-region policy optimization, proximal policy optimization, A3C
- Deterministic policy-based methods
  - Deterministic policy gradient, DDPG

DRL Lecture Contents

- A brief review of RL
- Deep RL
  - What happens when RL modules are implemented with DNNs?
  - DQN, TRPO and DDPG
- Multi-agent DRL
  - Key challenges
  - Some featured multi-agent simulators

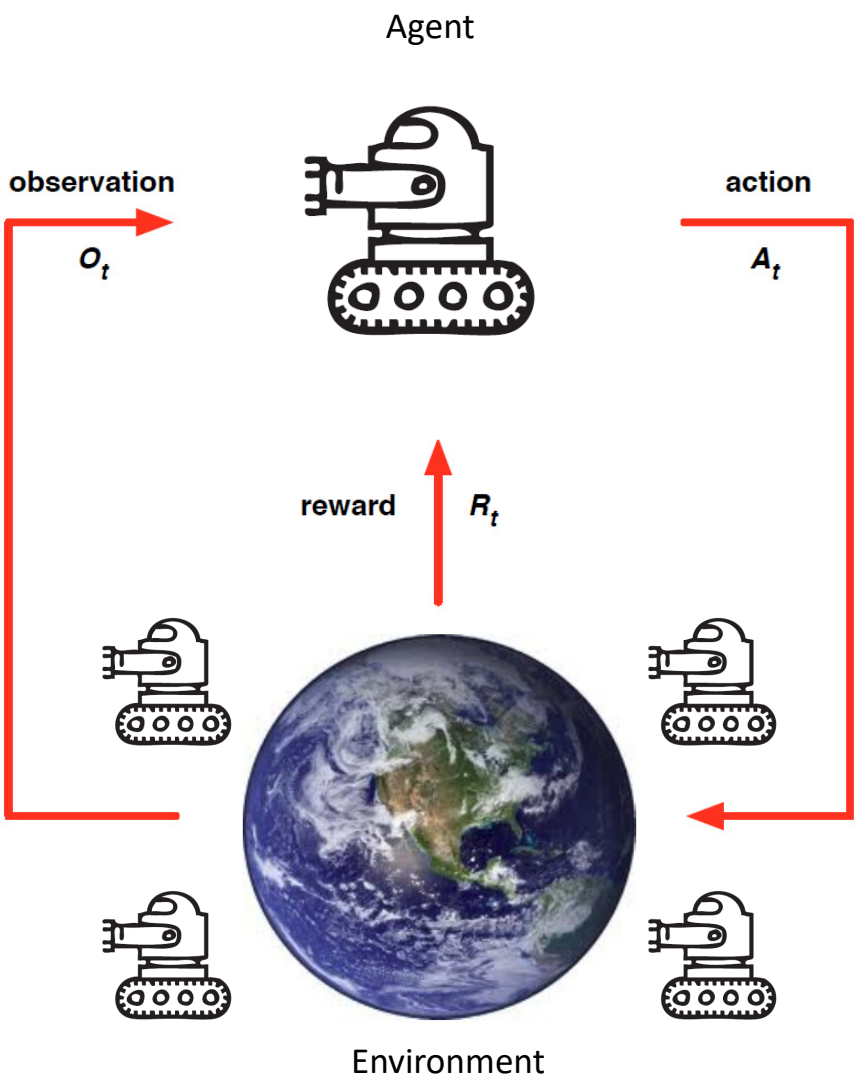
Reinforcement Learning

- Learning from interaction with the environment
- The agent
  - senses the observations from environment
  - takes actions to deliver to the environment
  - gets reward signal from the environment
- Normally, the environment is stationary



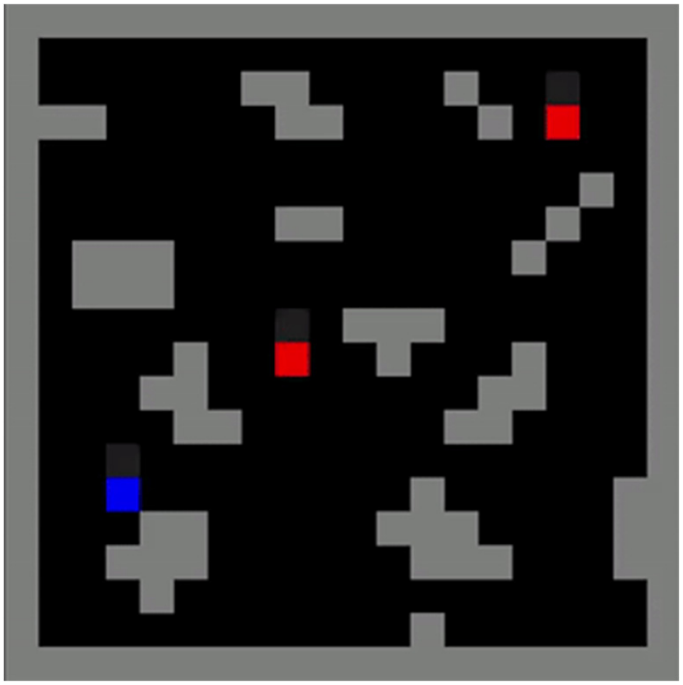
Multi-Agent Reinforcement Learning

- Learning from interaction with the environment
- The environment contains other agents that are learning and updating
- Non-stationary environment



Use Case: Multi-Agent Game Playing

- Multi-agent game playing
  - Learning to cooperate and compete



Wolfpack game

- Red agents are the predators
- Blue agent is the prey
- Red agent gets close to blue agent to make a capture, then the whole team gets a reward

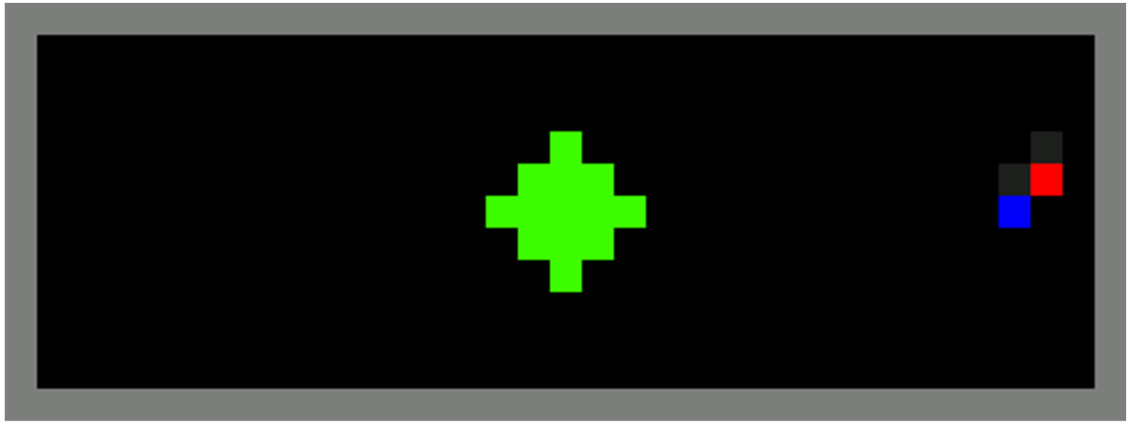
Results

- Red agents learn to cooperate.

Leibo, Joel Z., et al. "Multi-agent Reinforcement Learning in Sequential Social Dilemmas." *AAMAS 2017.*

Use Case: Multi-Agent Game Playing

- Multi-agent game playing
  - Learning to cooperate and compete



Gathering game

- Red and blue agents are compete for food
- Each agent can either move to eat or attack the other to make it paused

Results

- Red agents learn to compete when food resource is insufficient.

Leibo, Joel Z., et al. "Multi-agent Reinforcement Learning in Sequential Social Dilemmas." *AAMAS 2017.*

Use Case: Multi-Agent Game Playing

- Multi-agent game playing
  - Learning to cooperate and compete

Peng Peng, Jun Wang et al. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. NIPS workshop 2017.

Difficulty in Multi-Agent Learning

- MAL is fundamentally more difficult
  - since agents not only interact with the environment but also with each other

- If use single-agent Q learning by considering other agents as a part of the environment
  - Such a setting breaks the theoretical convergence guarantees and makes the learning unstable
  - i.e., the changes in strategy of one agent would affect the strategies of other agents and vice versa

From Multi- to Many-Agent RL

- What will happen when agent number grows?
  - Reward function of agent $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$
  - Transition probability $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Omega(\mathcal{S})$
- Both reward function and state transition probability get exponentially larger
  - More difficult to model
  - The environment is more dynamic and sensitive
  - Need more exploration data
  - More computational resources

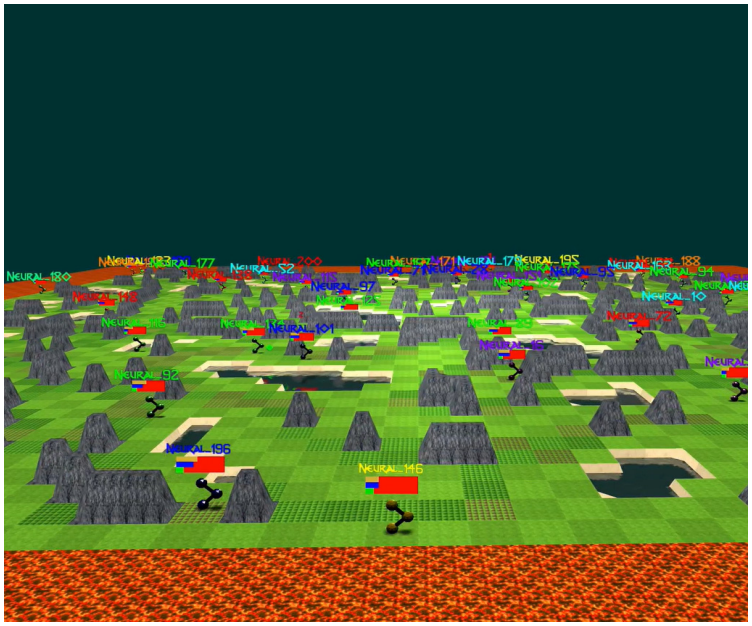
Key Factors for Successful MARL

- Computation: High computational resource for reinforcement learning
- Data: a huge amount of data for training the models
- Environment: a low-cost environment for RL agents to interact with

- Solution: an effective simulator

Simulator

- Training RL algorithm heavily relies on simulator



OpenAI's Neural MMO

CARLA

Why is Simulator a Necessity

- Large number of training samples required
- High trial and error cost

- Not possible in real world environment
- Even more problematic in real-life scenarios

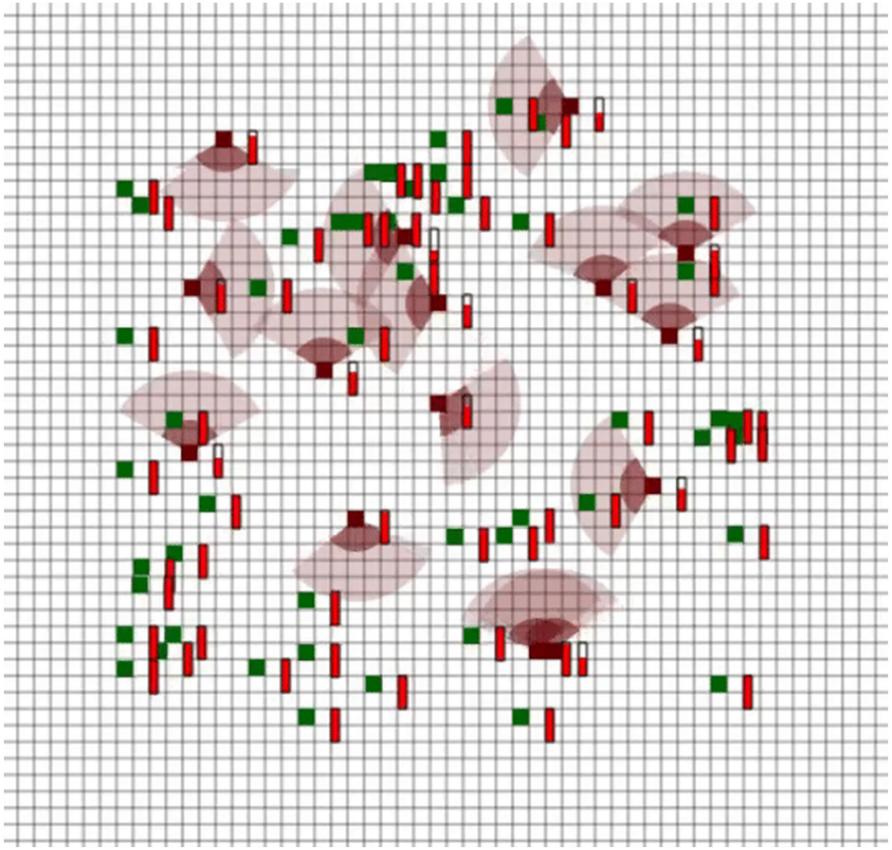
Featured Simulators

- Discrete world: MAgent
  - https://github.com/geek-ai/MAgent

- Continuous world: Cityflow
  - https://github.com/cityflow-project/CityFlow/

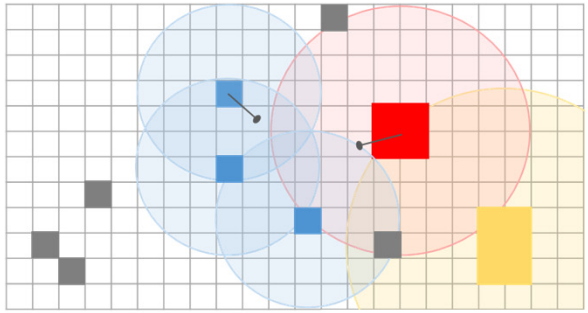
The Challenges

- High demand of computation
  - Large scale computation in training, inference and simulation
- Scalable and dynamic solution
  - The number of agents is highly dynamic. Agents can enter and exit
- Complicated interaction
  - It is hard to exactly model the interaction among agents
- Visualization

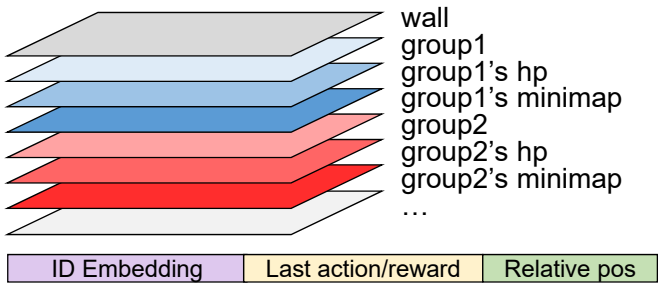


The Challenges

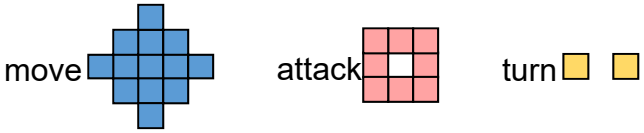
Grid World

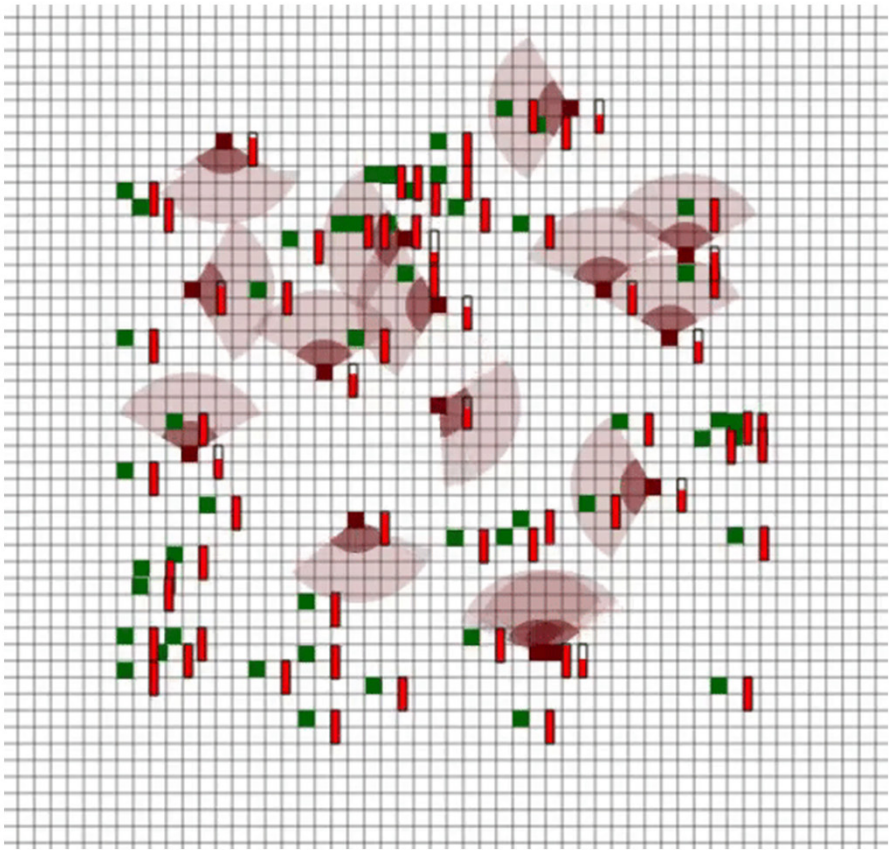


Observation Space



Action Space



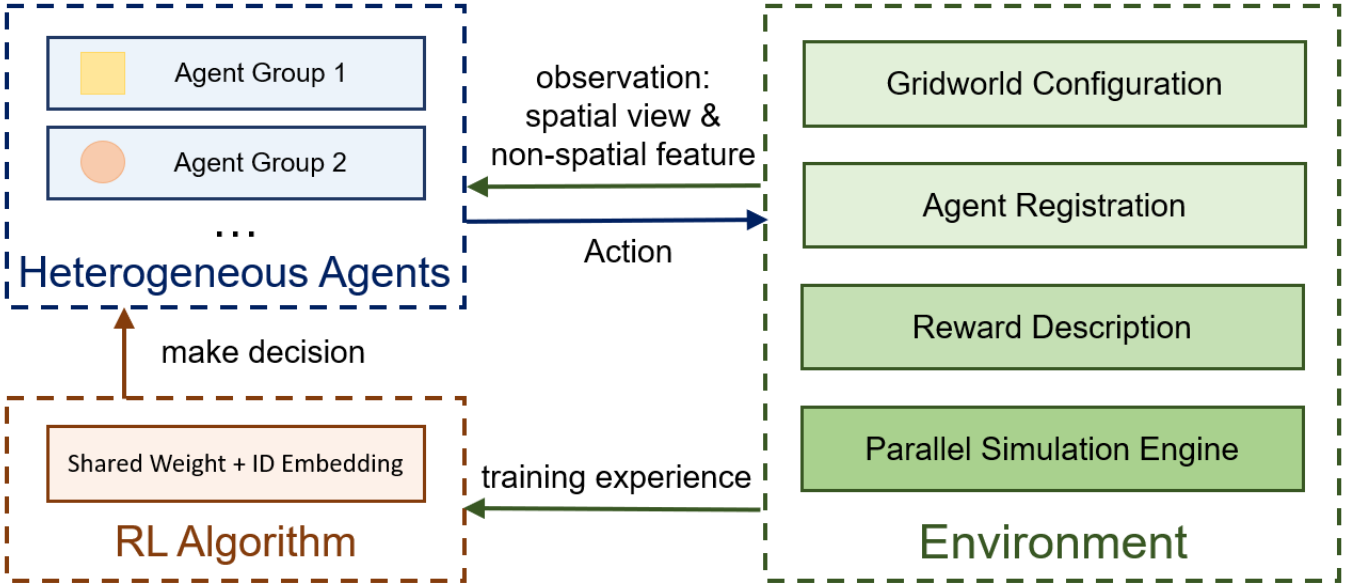


Other MARL Platforms

Platform	Number of agents	Learning Interface
OpenAI Gym	< 10	√
Malmo	< 1000	√
Starcraft Learning Environment	< 2000	√
Arcade Learning Environment	< 10	√
NetLogo	~ 1000,000	×
MAgent	**~ 1,000,000**	√

Decentralized MARL

- When agent number is too large to maintain a centralized meta-agent for controlling
  - Sharing Q-network for scalability
  - Agent ID for personalization



$$Q(s_t^i, a_t^i) \leftarrow Q(s_t^i, a_t^i) + \alpha[r_t^i + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}^i, a') - Q(s_t^i, a_t^i)]$$

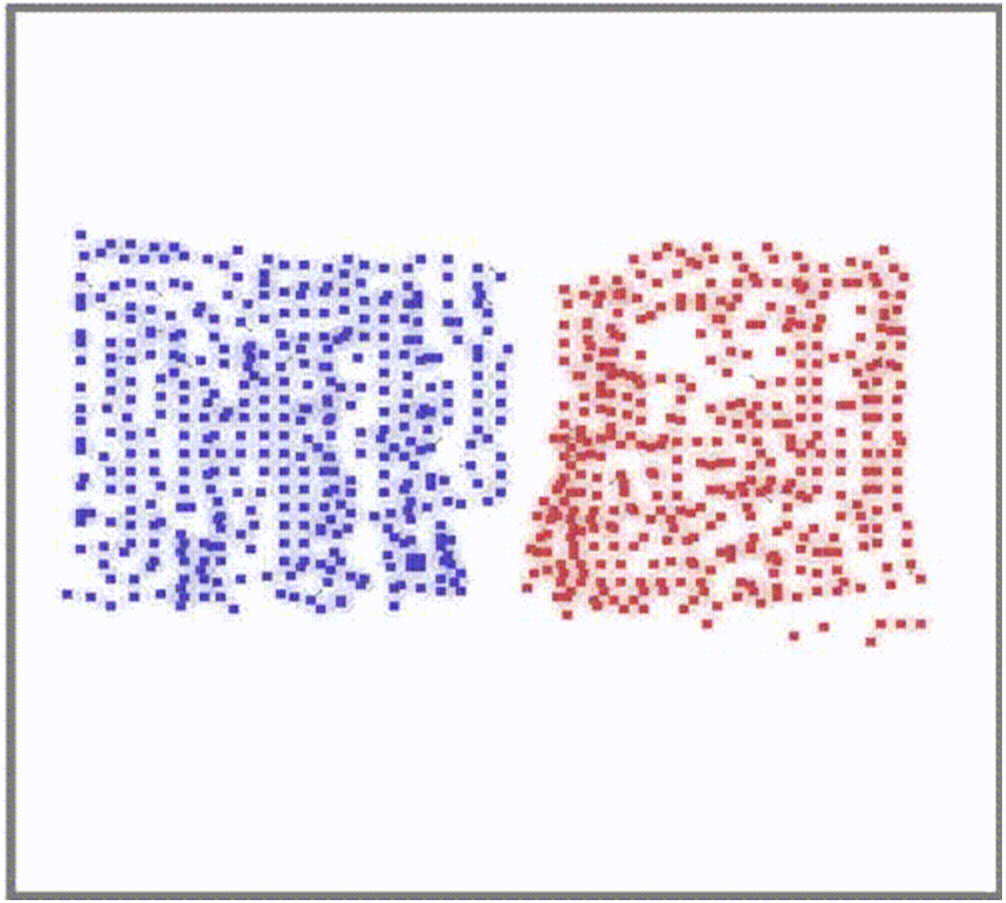
Use Case: Many-Agent Interactions

- MAgent game: aligning

Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

Use Case: Many-Agent Interactions

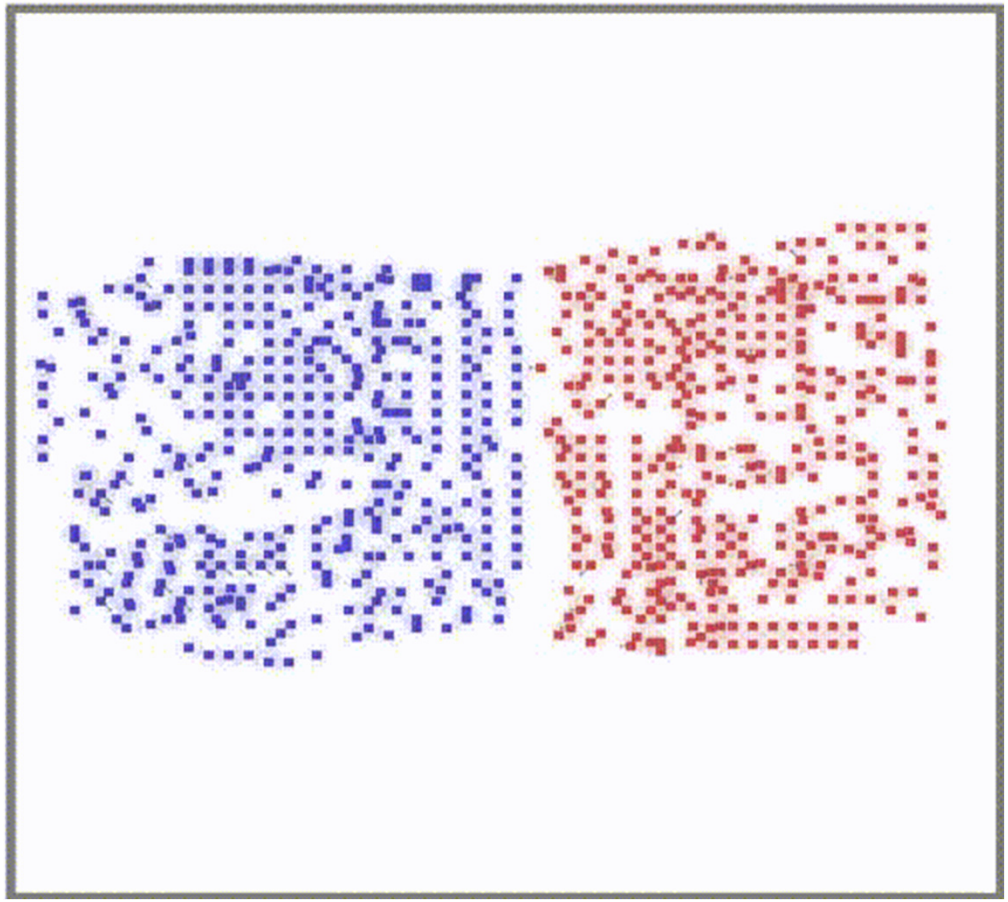
- MAgent game: battle



Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

Use Case: Many-Agent Interactions

- MAgent game: battle



Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

Use Case: Many-Agent Interactions

- MAgent game: city simulation

Lianmin Zheng, Jiacheng Yang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. NIPS 2017 & AAAI 2018 Demos.

Featured Simulators

- Discrete world: MAgent
  - https://github.com/geek-ai/MAgent

- Continuous world: Cityflow
  - https://github.com/cityflow-project/CityFlow/

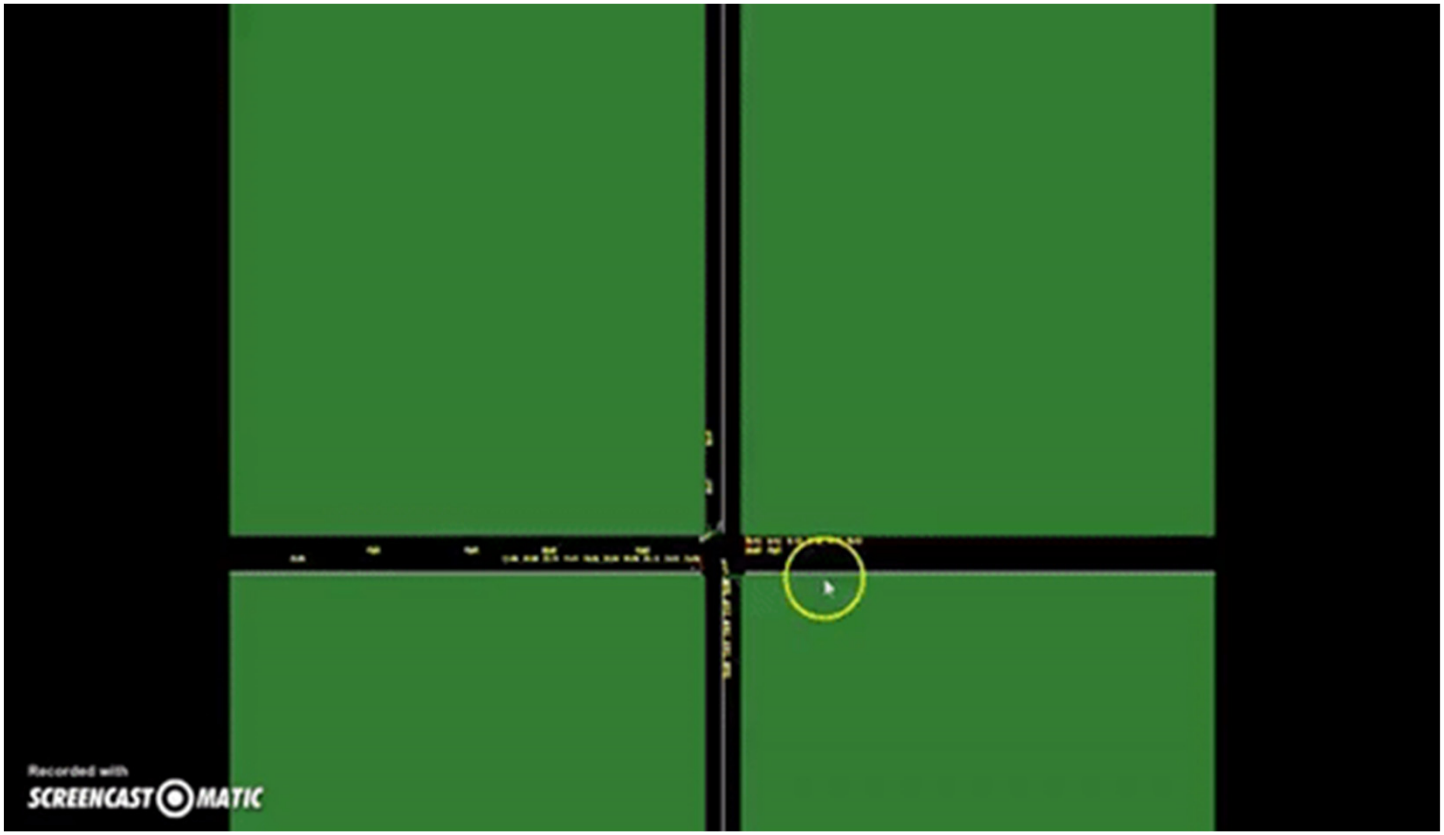
Current City Simulator

- Focus on traffic scenario
  - Commercial:
    - VISSIM PTV VISSIM
    - AIMSUN.NEXT aimsun.next
  - Open source:
    - Most Popular: SUMO SUMO
    - Ours: CityFlow

- Microscopic Simulator
  - Simulate the movement of each single vehicle or object

SUMO

- **S**imulation of **U**rban **Mo**bility
- Institute of Transportation Systems @ German Aerospace Center
- Project starts from 2000



https://gfycat.com/hopefulsolidamericancreamdraft

What accounts for an effective simulator?

High Efficiency

- Interact with multiple agents in a high speed
- Multi-thread and multi-machine deployment

High Reality

- The simulation results should be as close to reality as possible
- Match in both micro and macro levels

Extendibility

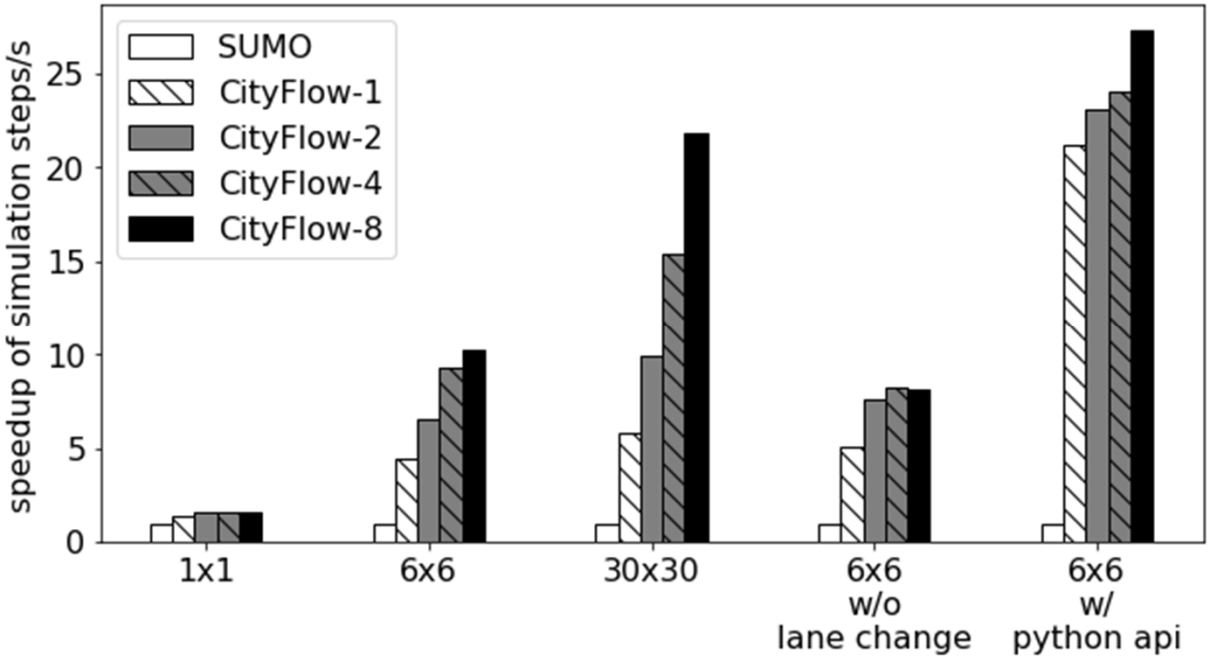
- Flexible to adapt to new tasks with little effort

Interaction

- Easy to visualize and friendly for human interaction

CityFlow

- Focus on speed
  - Data structure design
  - Simulation algorithm design
  - Multithread
  - Faster python api (compared to SUMO)



CityFlow

Status Panel

当前车辆数	96
模拟总步数	1000
当前模拟步数	113
当前进度	11.20%

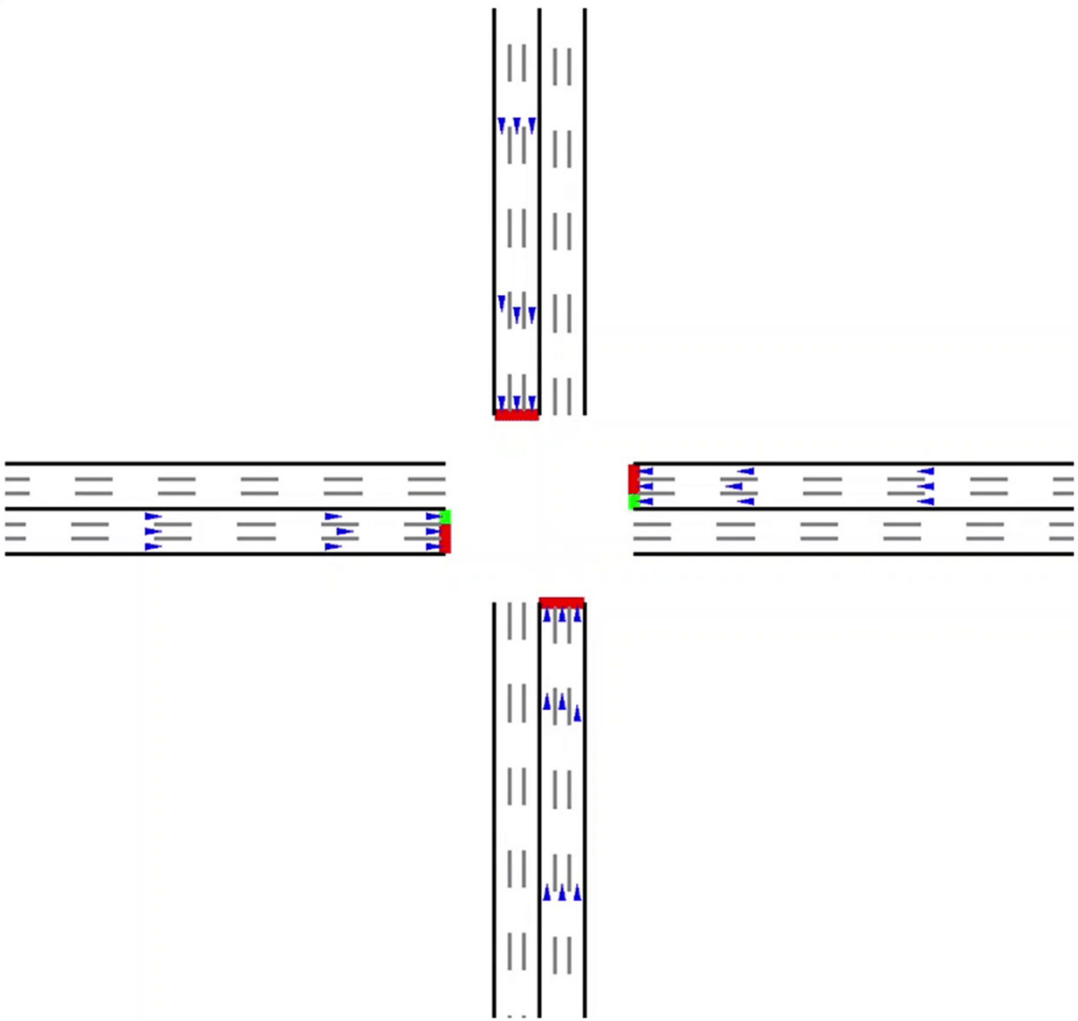
Navigation Keys

left right up down - =
p(pause) 1(slowdown)
2(speedup)

Info Box

roadnet file loaded.
simulation start!

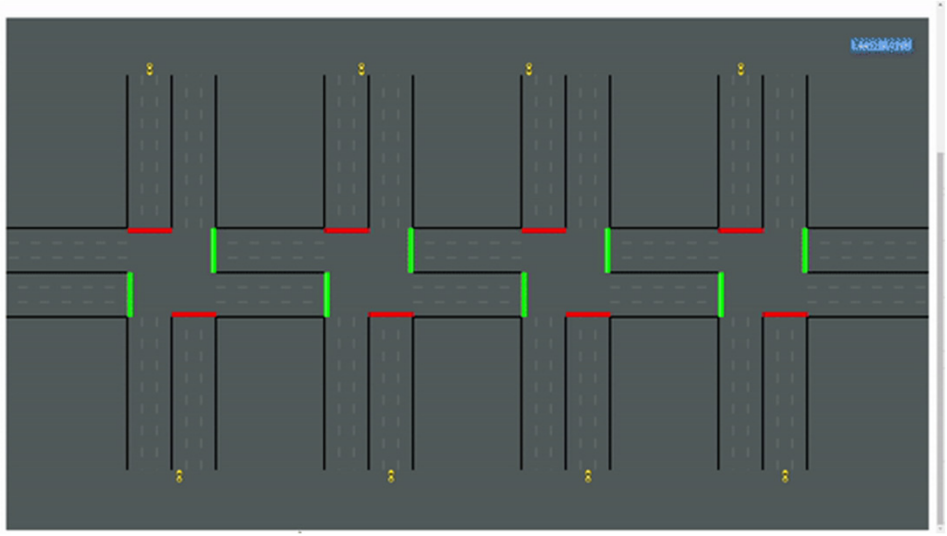
City Simulator



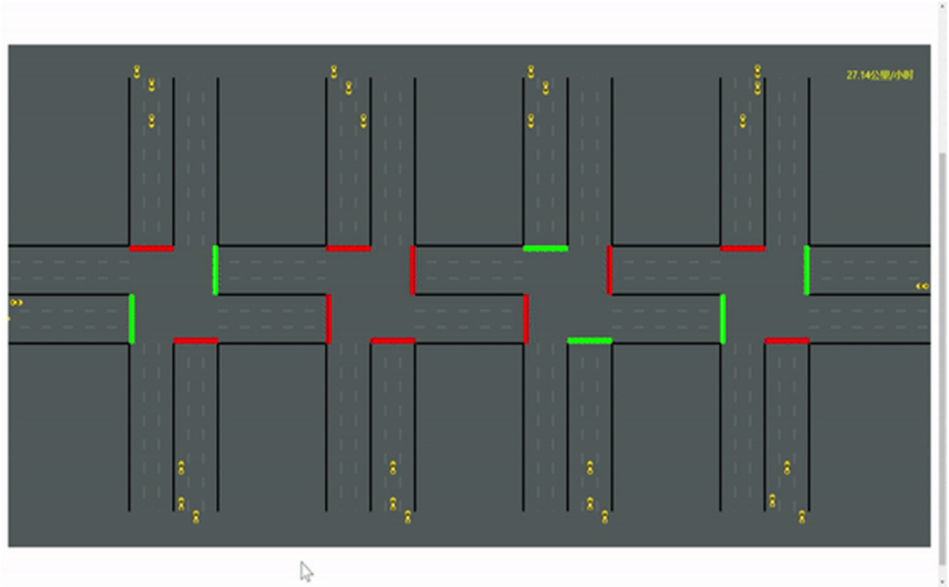
CityFlow

- A special version of UI
- Same flow
- Difference between

Bad Traffic Signal Plan



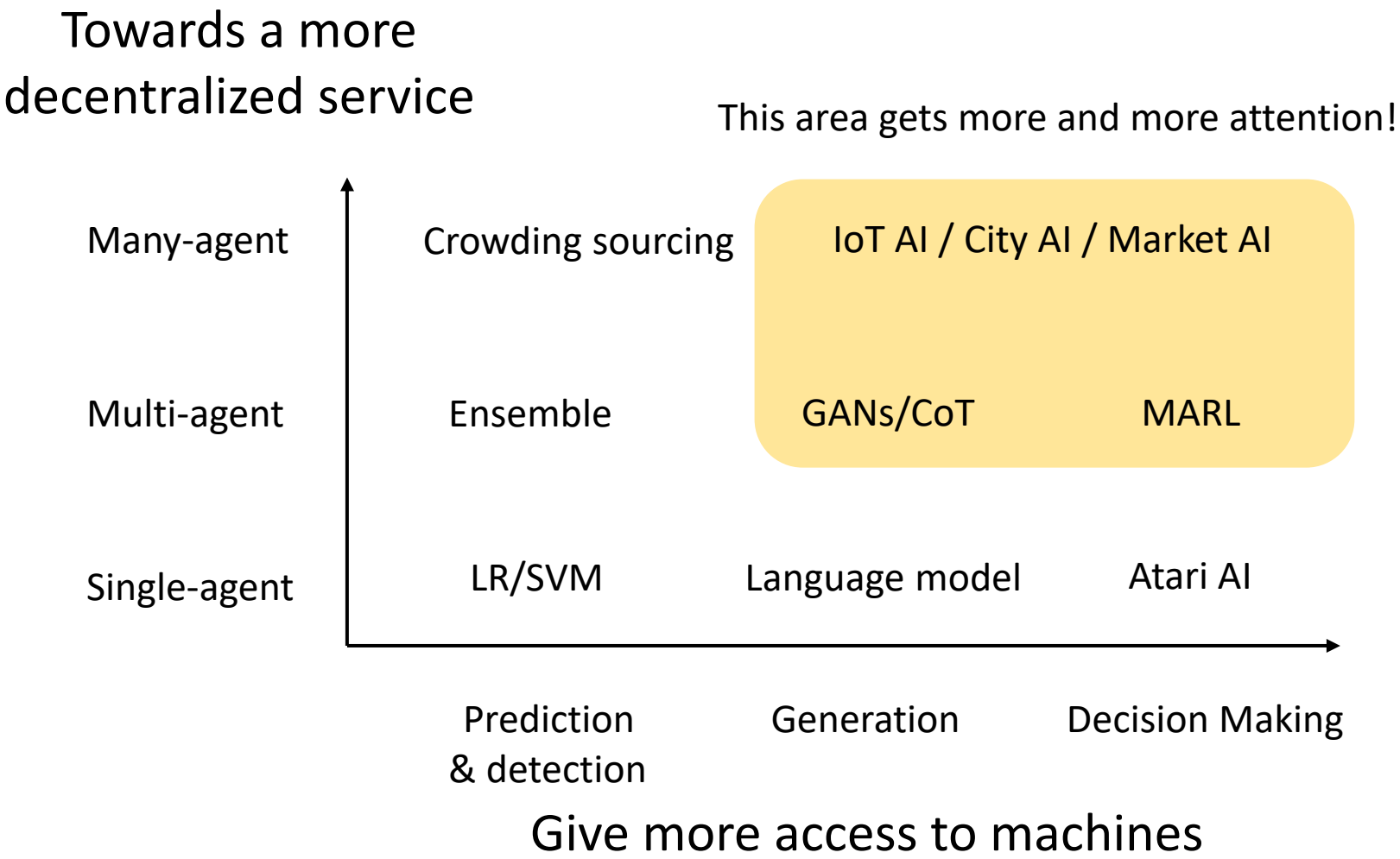
Good Traffic Plan



TSCC2050

- Use a customized version of CityFlow as backend

Summary Machine Learning Paradigm Extension



Thank You!
Questions?

Weinan Zhang

Assistant Professor at Shanghai Jiao Tong University

http://wnzhang.net

Key references

1. Huichu Zhang, Weinan Zhang et al. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. WWW 2019.
2. Lianmin Zheng, Weinan Zhang et al. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. AAAI 2018.
3. Yaodong Yang, Weinan Zhang et al. Mean Field Multi-Agent Reinforcement Learning. ICML 2018.