Introduction to
Deep Reinforcement Learning

Yen-Chen Wu

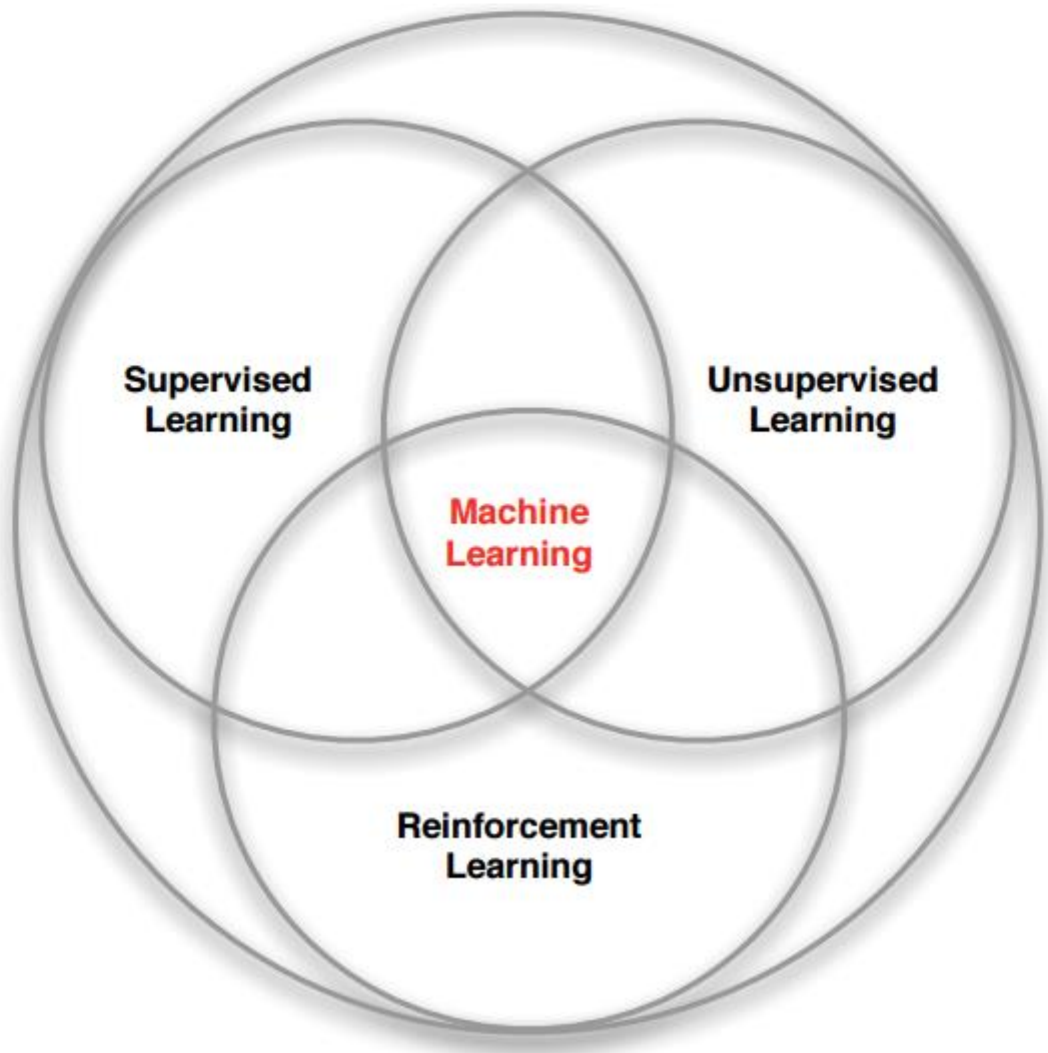
2015/12/11

Outline

- Reinforcement Learning
- Markov Decision Process
- How to Solve MDPs
  - DP
  - MC
  - TD
  - Q-learning (DQN)
- Paper Review

REINFORCEMENT LEARNING

Branches of Machine Learning

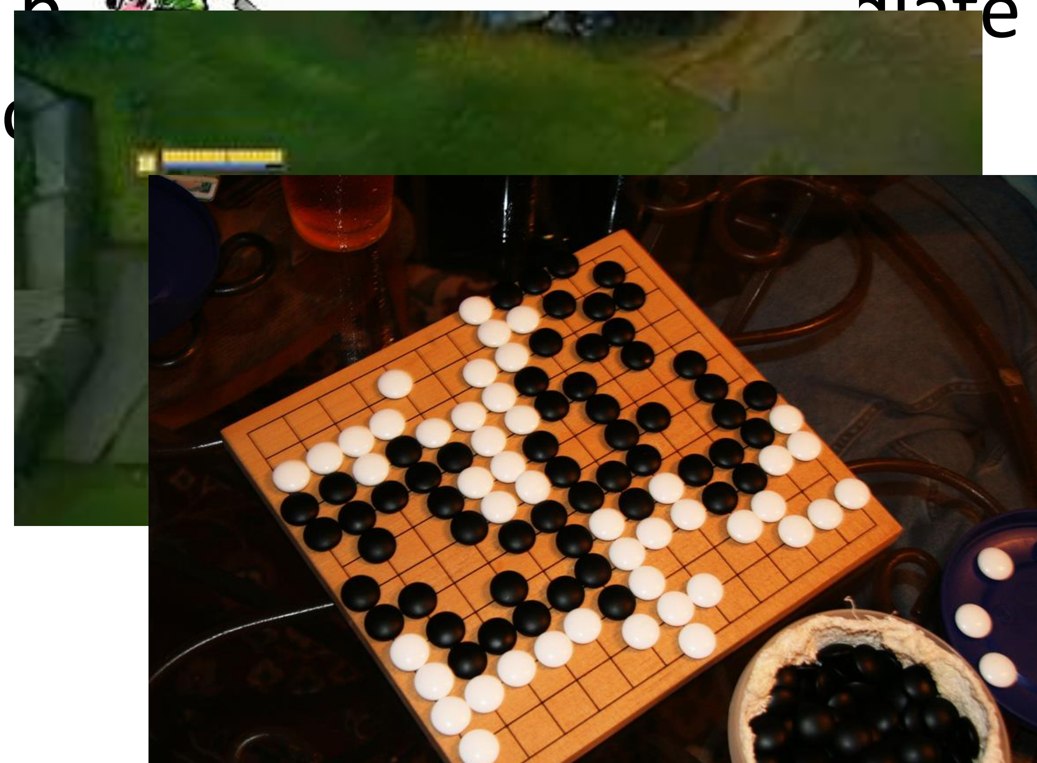


What makes different?

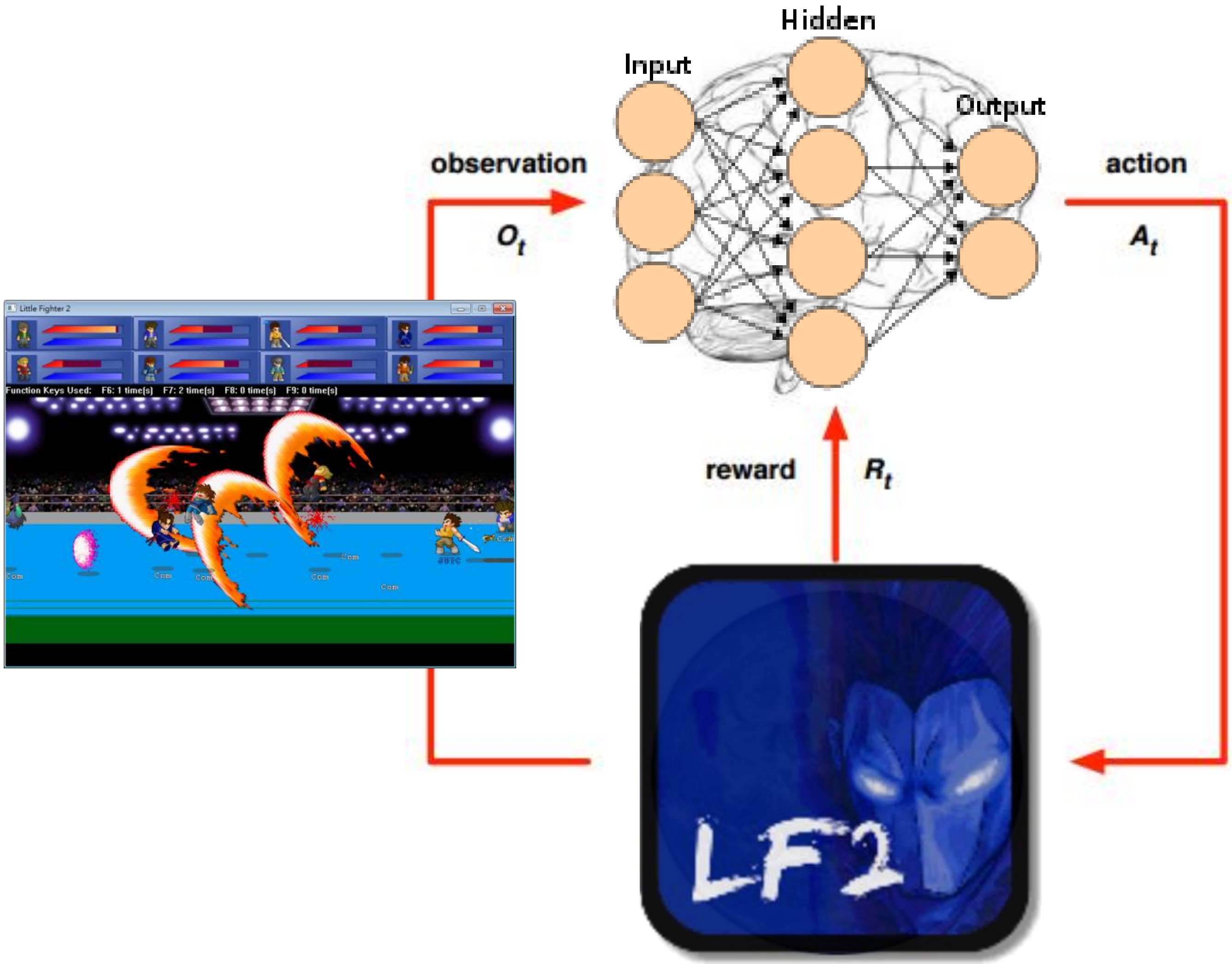
- There is no supervisor, only a reward signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Goal:
Maximize Cumulative Reward

- Actions may have long term consequences
- Reward may be delayed
- It may be b diate reward
  to gain m



Agent & Enviroment



→←↑↓
Defense
Attack
Jump

Markov Processes

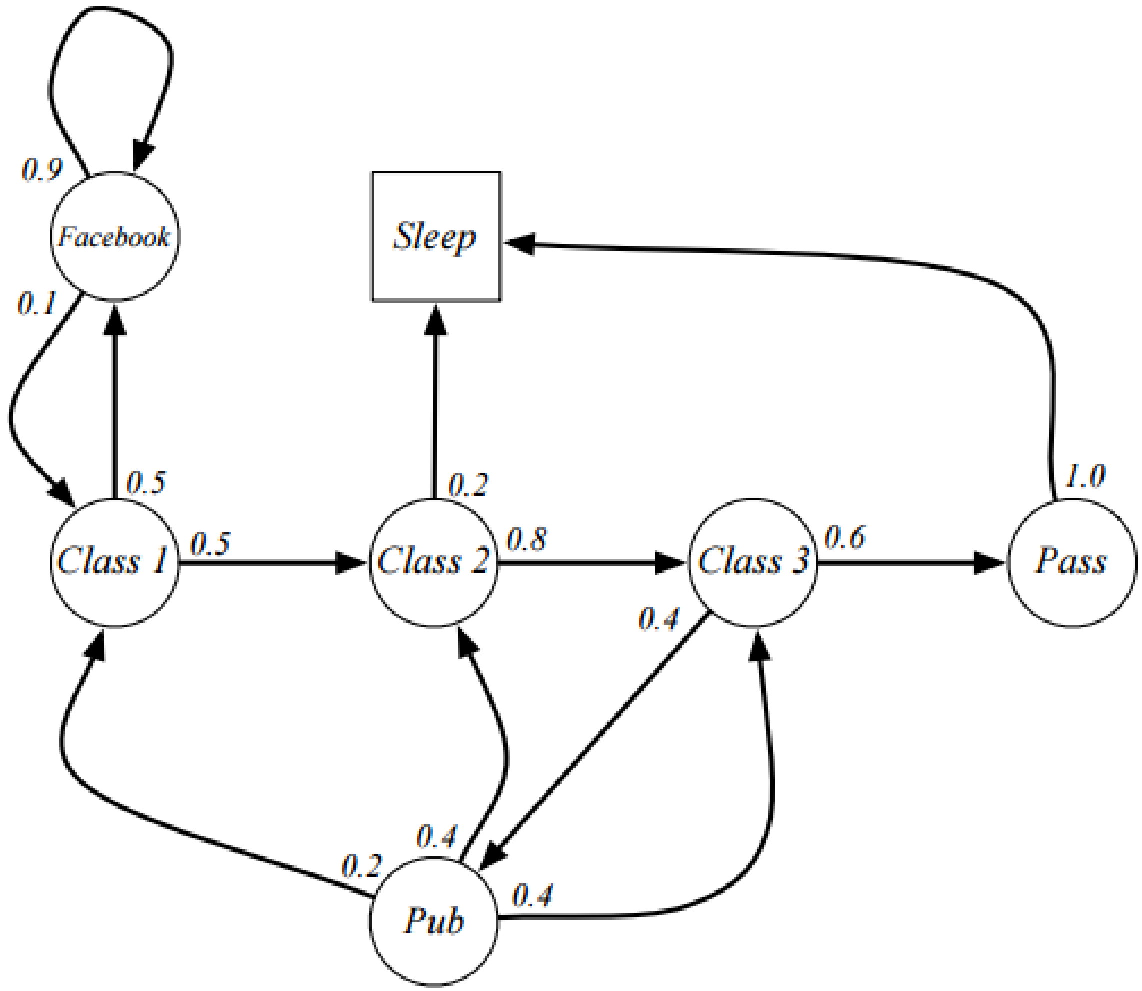
Markov Reward Processes

Markov Decision Processes

MARKOV DECISION PROCESS

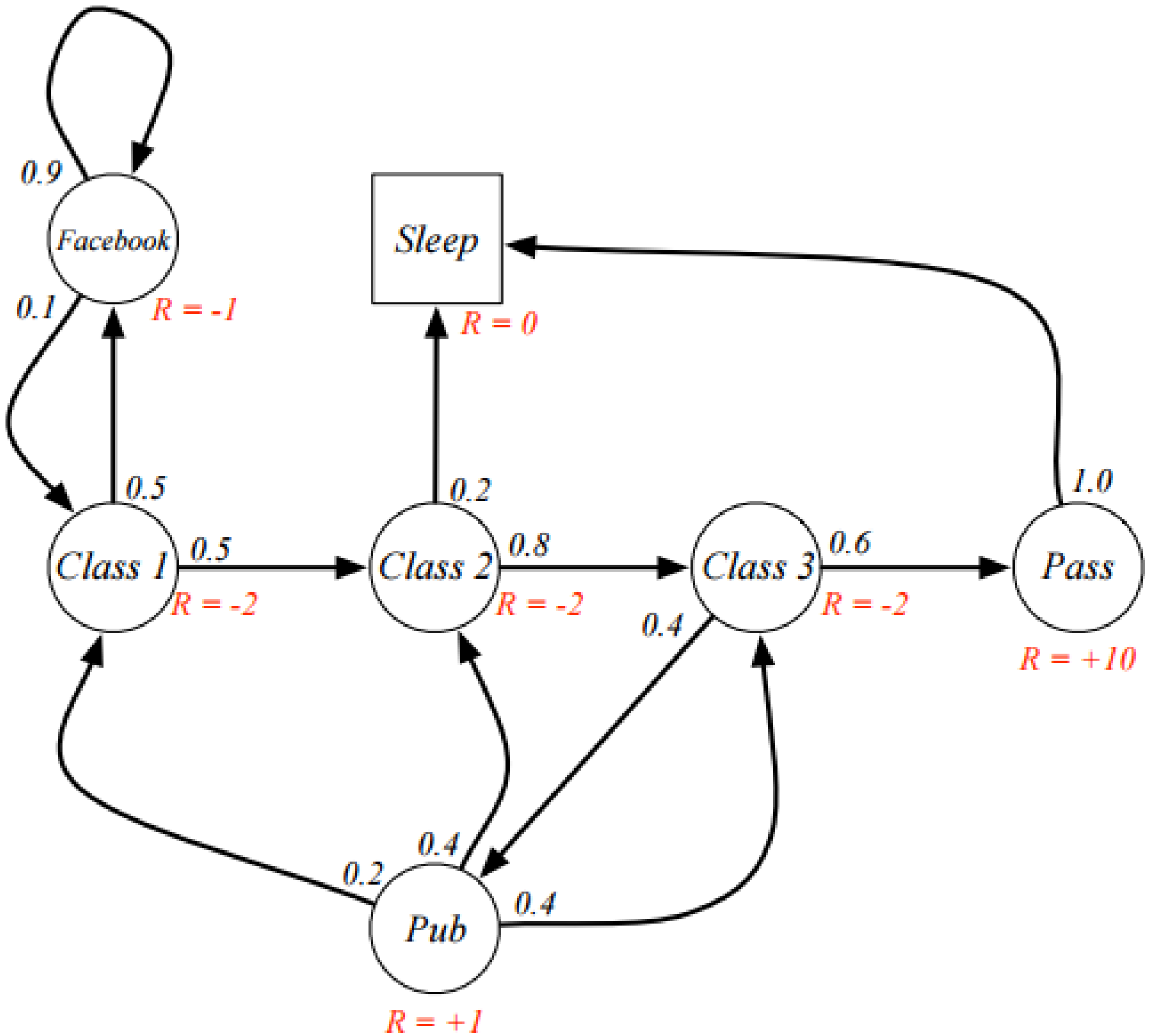
Markov Process

Example: Student Markov Chain



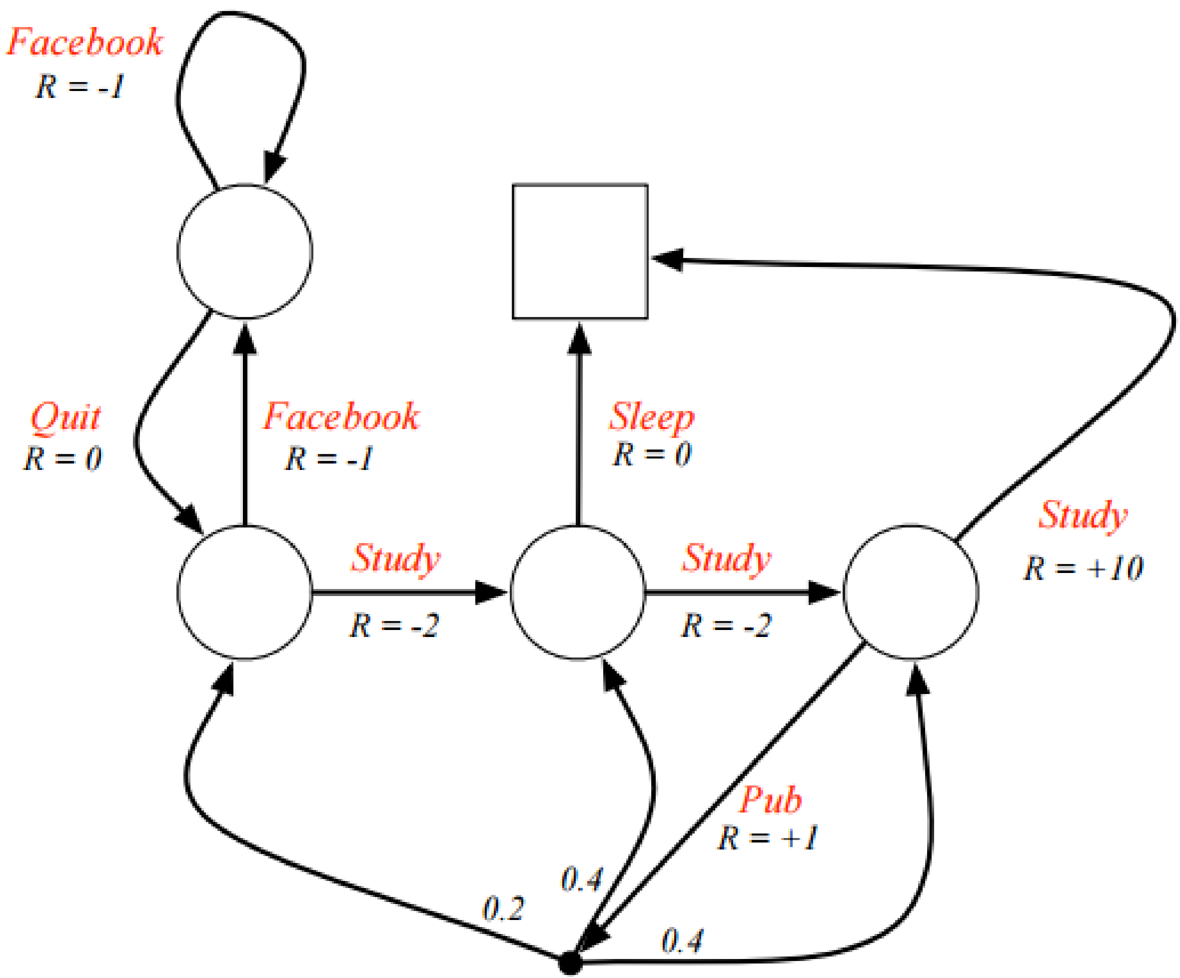
Markov Reward Processes

Example: Student MRP



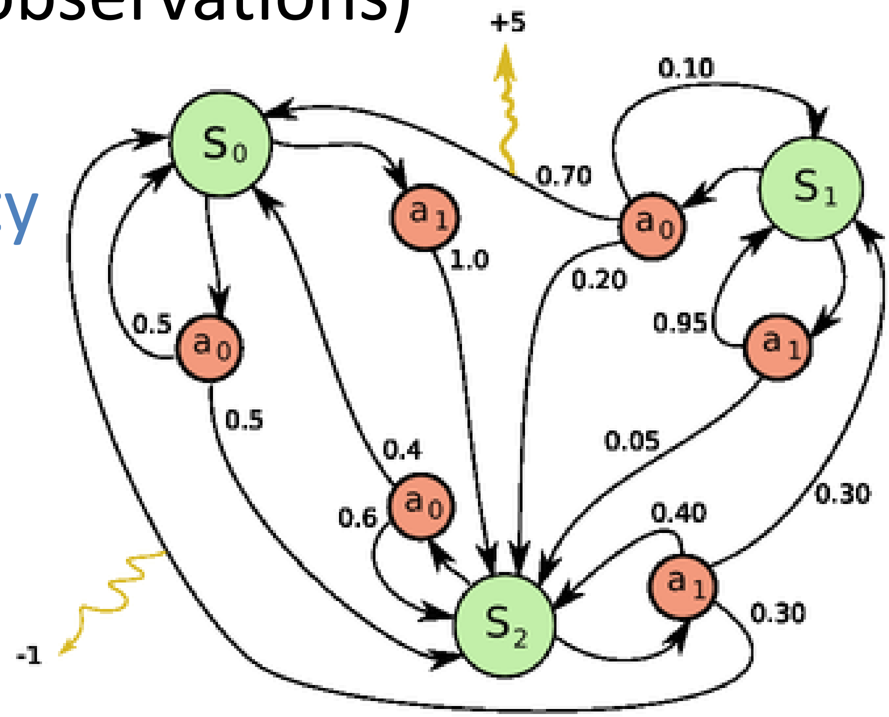
Markov Decision Process

Example: Student MDP



Markov Decision Process(MDP)

- S : finite set of states (observations)
- A : finite set of actions
- P : transition probability
- R : immediate reward
- γ : discount factor



- Goal :
  - Choose policy π
  - Maximize expected return : $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$

Dynamic Programming

Monte-Carlo

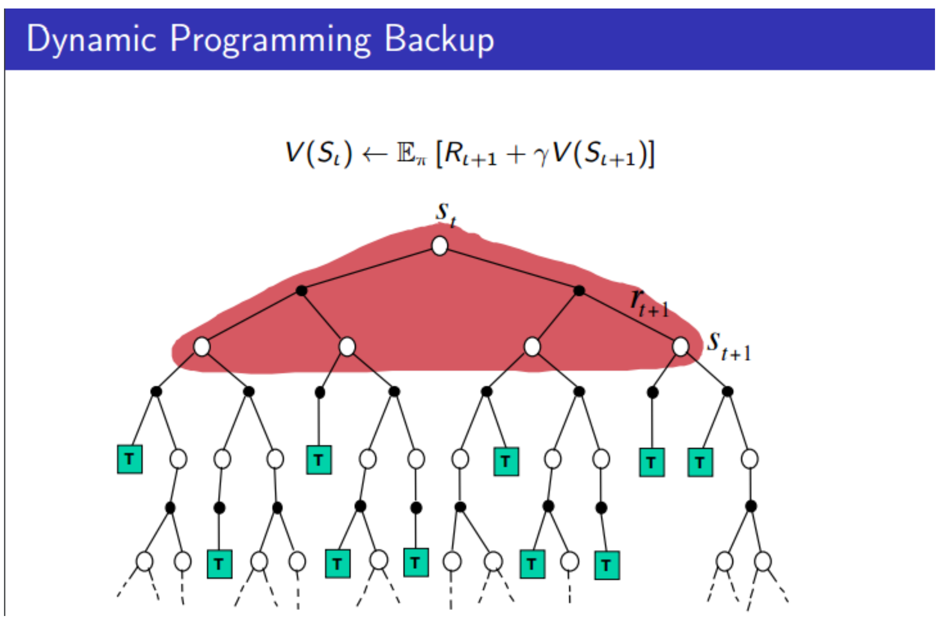
Temporal-Difference

Q-Learning

HOW TO SOLVE MDP

Model-based

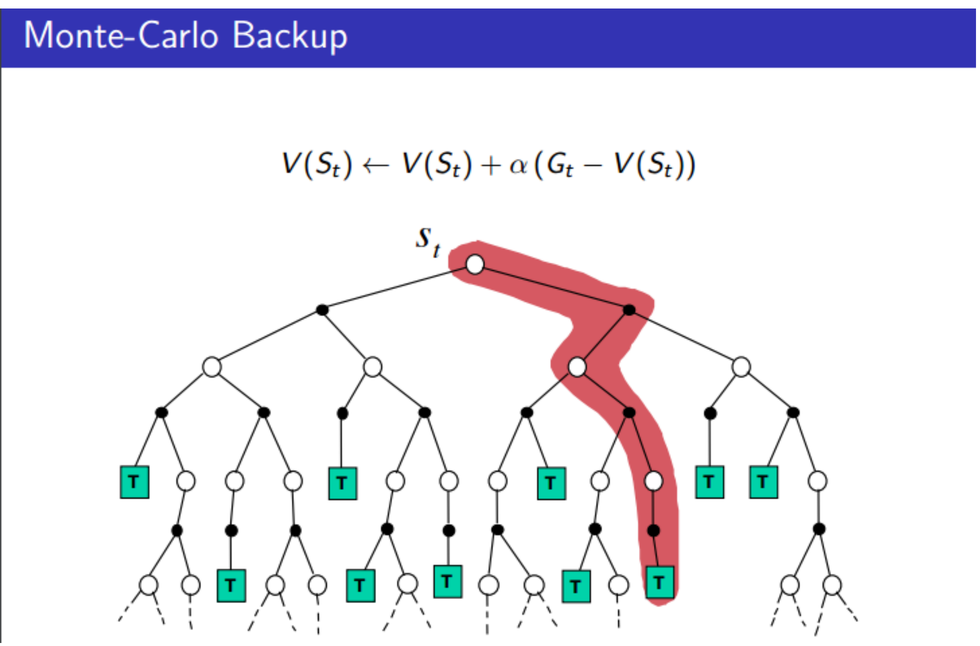
- Dynamic Programming
  - Evaluate policy
  - Update policy

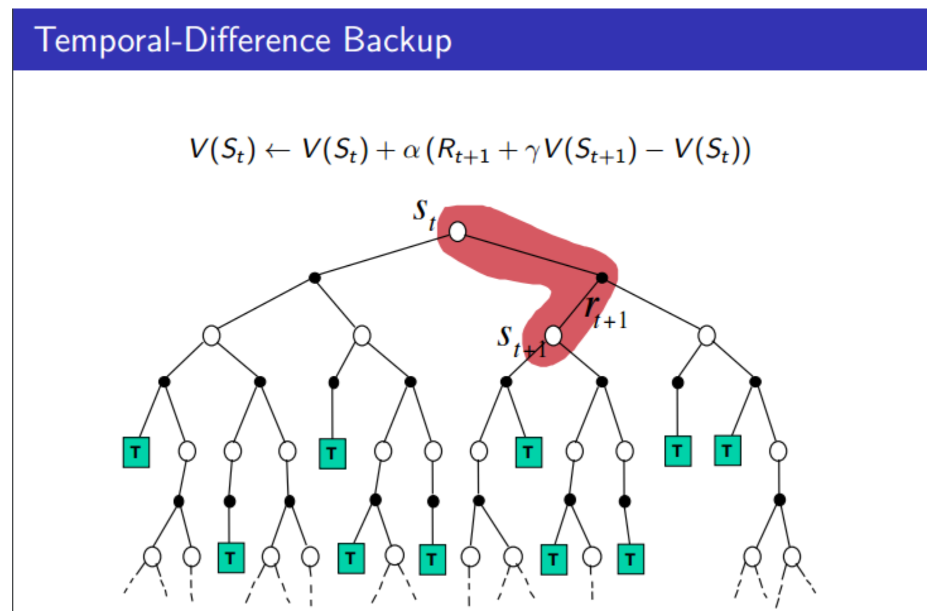


Right: A simple Gridworld solved with a Dynamic Programming. Very exciting. Head over to the GridWorld: DP demo to play with the GridWorld environment and policy iteration.

Model Free

- Unknown Transition Probability & Reward
- MC vs TD





Model Free: Q-learning

- Instead of tabular
- optimal action-value function (Q-learning)
  - $Q^*(s,a) = \max_\pi \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$
- Bellman equation

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

- Basic idea : iterative update (lack of generalization)
- In practical : function approximator $Q(s,a;\theta) \approx Q^*(s,a)$
  - Linear ?
  - Using DNN !


LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

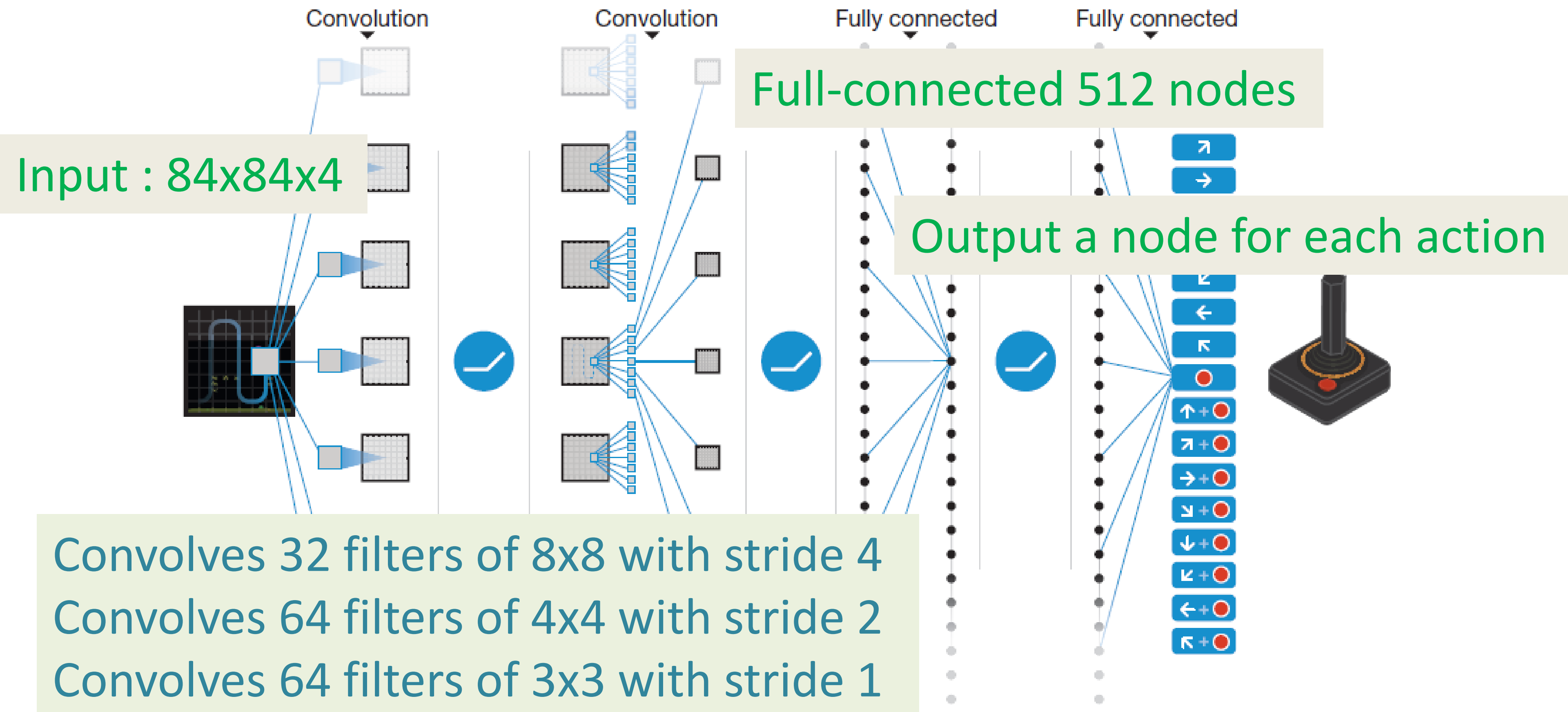
DEEP Q-NETWORK (DQN)

Video

- https://www.youtube.com/watch?v=LJ4oCb6u7kk

Deep Q-Network

- compute Q-values for all actions



Update DQN

- Loss function

$$L_i(\theta_i) = \mathbb{E}_{s,a,r}\left[\left(\mathbb{E}_{s'}\left[y|s,a\right] - Q(s,a;\theta_i)\right)^2\right]$$
$$= \mathbb{E}_{s,a,r,s'}\left[\left(y - Q(s,a;\theta_i)\right)^2\right] + \mathbb{E}_{s,a,r}\left[\mathbb{V}_{s'}\left[y\right]\right]$$

- Gradient

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

Two Technique

- Experience Replay
  - Experience
  - Pooled Mer $e_t = (s_t, a_t, r_t, s_{t+1})$
    - Data efficiency ($D_t = \{e_1, \ldots, e_t\}$
    - Avoid correlation between samples (variance between batches)
    - Off –policy is suitable for Q-learning
  - Random sampled mini-batch

	Example	Learn the value of...	Pros & Cons
On-policy	SARSA	policy being carried out by the agent	Fast but weak
Off-policy	DQN	optimal policy independently of the agent's actions	Slow but robust

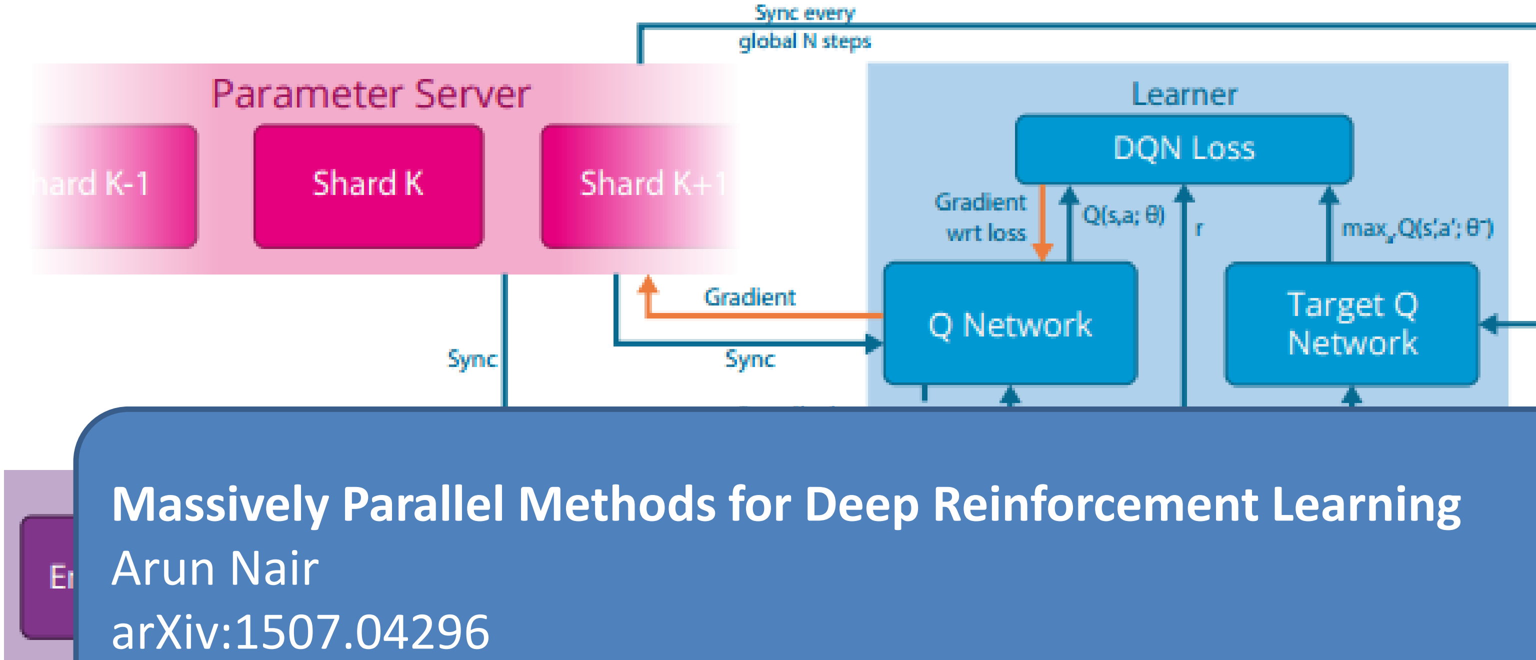
DEMO

PAPER REVIEW

Paper list

- **Massively Parallel Methods for Deep Reinforcement Learning**
- **Continuous control with deep reinforcement learning**
- **Deep Reinforcement Learning with Double Q-learning**
- **Policy Distillation**
- **Dueling Network Architectures for Deep Reinforcement Learning**
- **Multiagent Cooperation and Competition with Deep Reinforcement Learning**

Gorila (GOogle ReInforcement Learning Architecture)

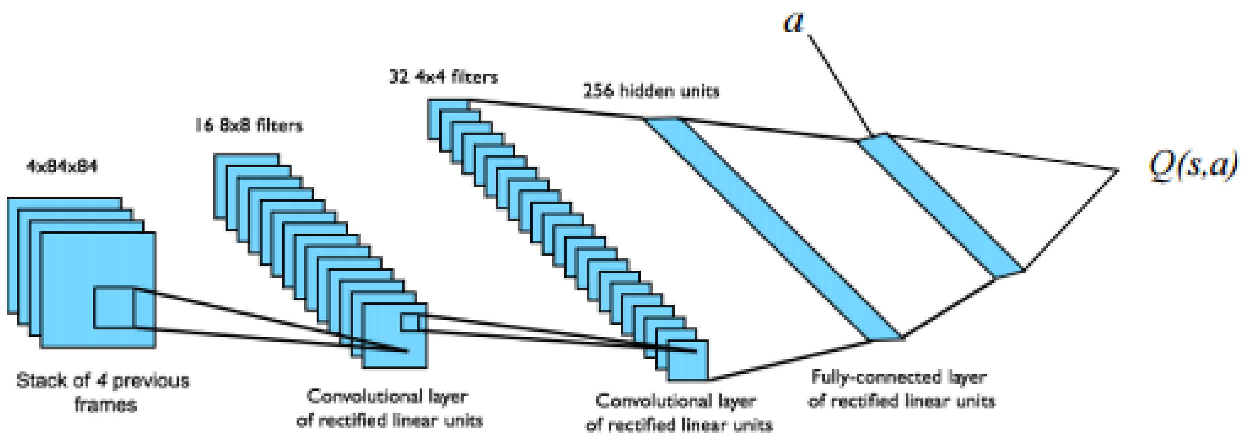


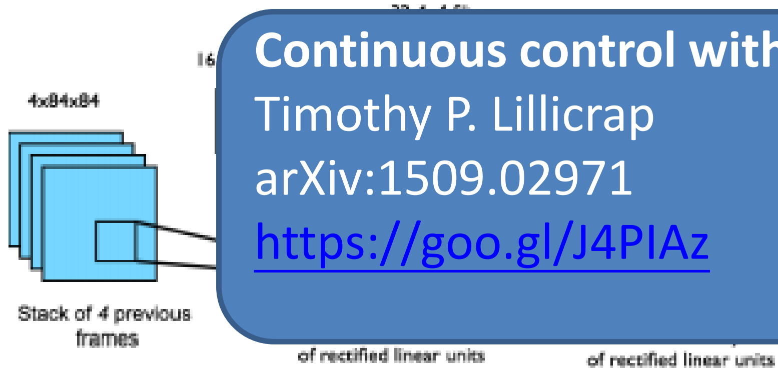
Massively Parallel Methods for Deep Reinforcement Learning
Arun Nair
arXiv:1507.04296

- Parallel acting: generate new interactions
- Distributed replay memory: save interactions
- Parallel learning: compute gradients from replayed interactions
- Distributed neural network: update network from gradients

DDPG (Deterministic Policy Gradient)

- DDAC (Deep Deterministic Actor-Critic)





Continuous control with deep reinforcement learning
Timothy P. Lillicrap
arXiv:1509.02971
https://goo.gl/J4PIAz

Double Q-learning

$$Y_t^{\text{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-) \, .$$

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t) \, .$$

Policy Distillation

- Soft target

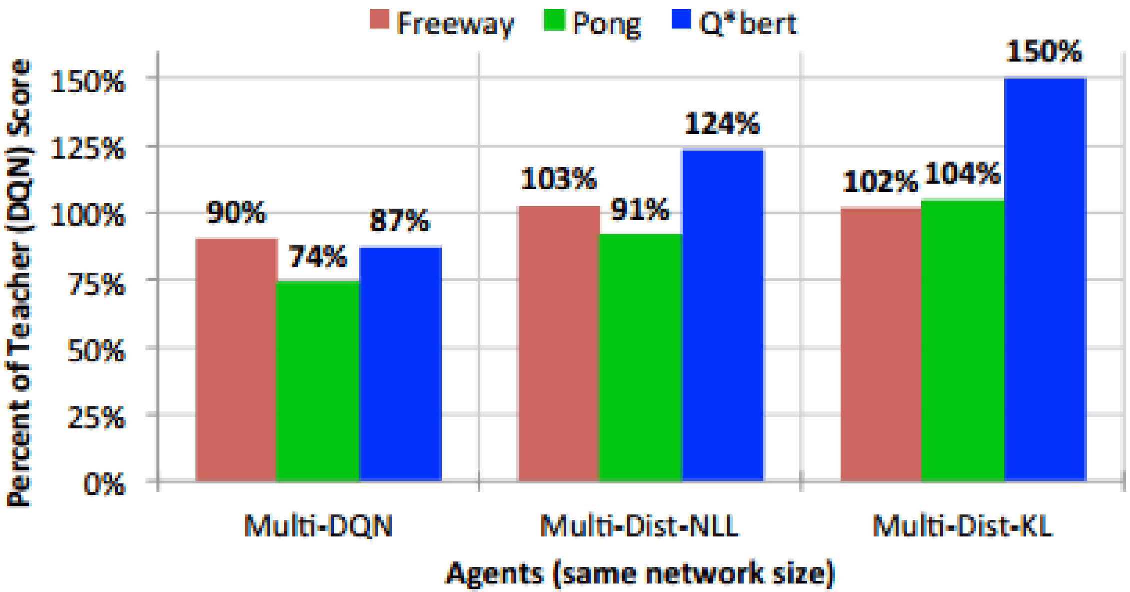


Figure 3: Performance of multi-task agents with identical network architecture and size, relative to respective single-task DQN teachers. A detailed results table is given in Appendix B

Dueling Network

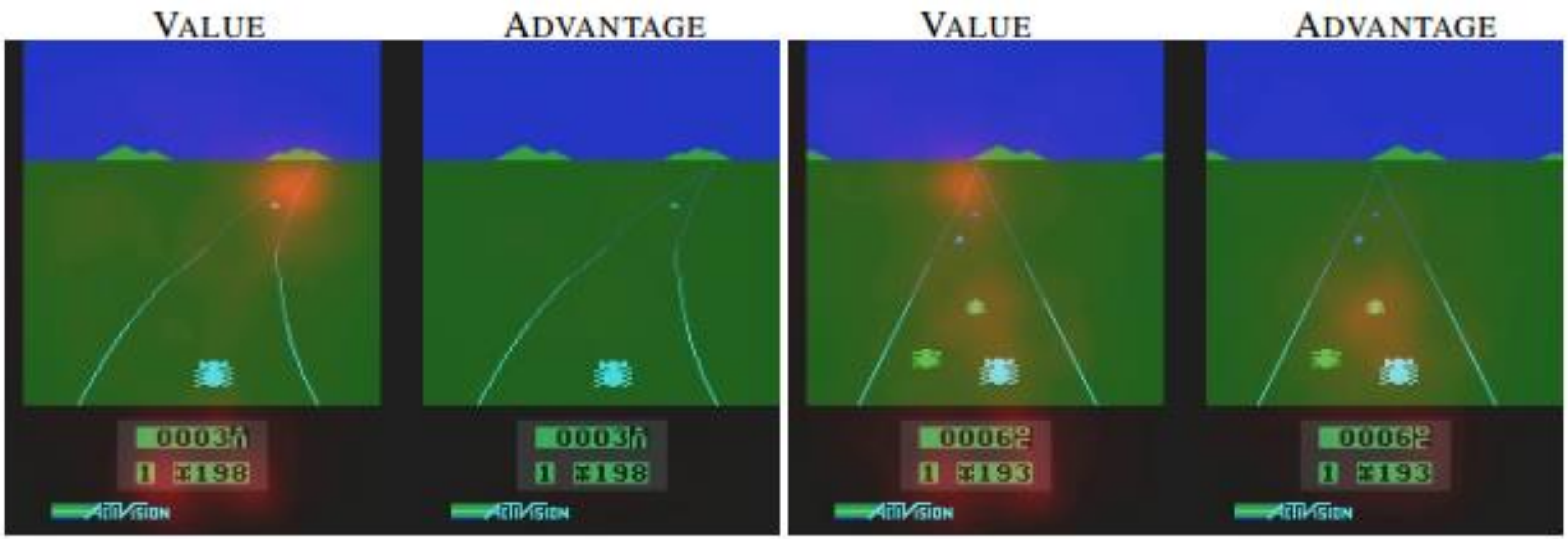
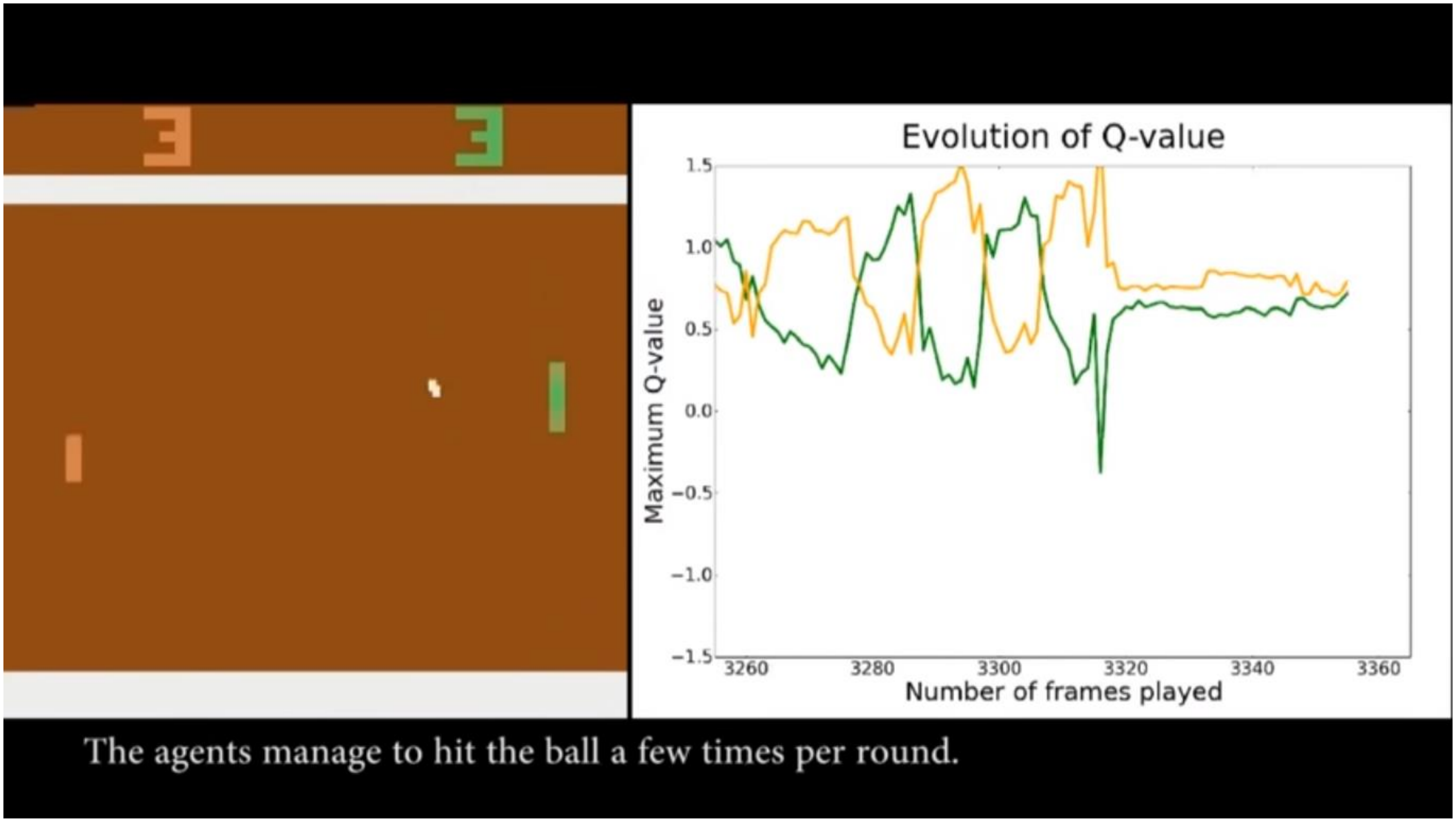


Figure 2: See, attend and drive: Value and advantage saliency maps on the Enduro game for a trained dueling architecture. The value stream learns to pay attention to the road. The advantage stream learns to pay attention only when there are cars immediately in front, so as to avoid collisions.

Multiagent


3
3
Evolution of Q-value
Maximum Q-value
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
3260
3280
3300
3320
3340
3360
Number of frames played
The agents manage to hit the ball a few times per round.