[Season2] Reinforcement Learning Paper Review

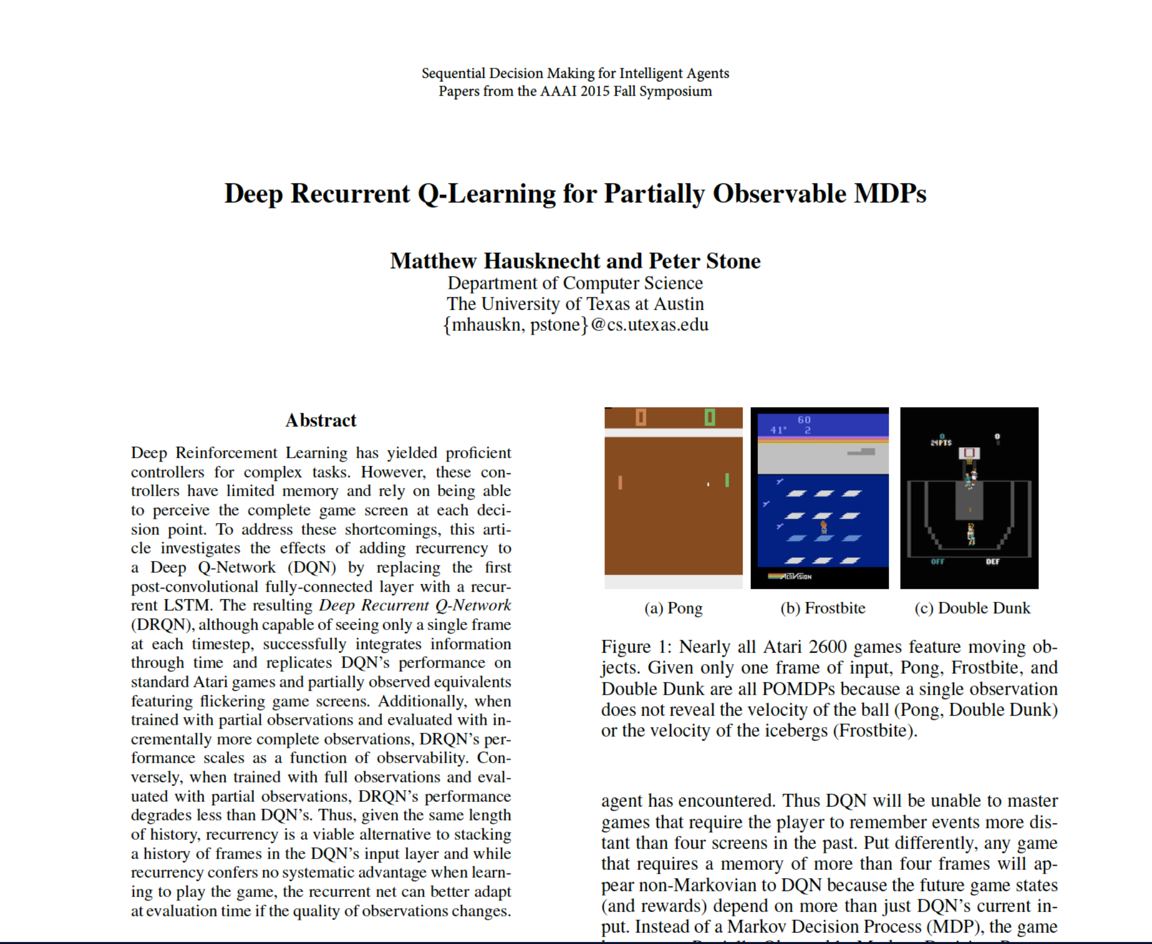
Deep Recurrent Q-Learning for Partially Observable MDPs

M. Hausknecht and P. Stone, *AAAI*, 2015

Department of Artificial Intelligence,
Korea University, Seoul
Minsuk Sung

Minsuk Sung
(minsuksung@korea.ac.kr)

Table of Contents



Sequential Decision Making for Intelligent Agents
Papers from the AAAI 2015 Fall Symposium

Deep Recurrent Q-Learning for Partially Observable MDPs

Matthew Hausknecht and Peter Stone
Department of Computer Science
The University of Texas at Austin
{mhauskn, pstone}@cs.utexas.edu

Abstract

Deep Reinforcement Learning has yielded proficient controllers for complex tasks. However, these controllers have limited memory and rely on being able to perceive the complete game screen at each decision point. To address these shortcomings, this article investigates the effects of adding recurrency to a Deep Q-Network (DQN) by replacing the first post-convolutional fully-connected layer with a recurrent LSTM. The resulting *Deep Recurrent Q-Network* (DRQN), although capable of seeing only a single frame at each timestep, successfully integrates information through time and replicates DQN's performance on standard Atari games and partially observed equivalents featuring flickering game screens. Additionally, when trained with partial observations and evaluated with incrementally more complete observations, DRQN's performance scales as a function of observability. Conversely, when trained with full observations and evaluated with partial observations, DRQN's performance degrades less than DQN's. Thus, given the same length of history, recurrency is a viable alternative to stacking a history of frames in the DQN's input layer and while recurrency confers no systematic advantage when learning to play the game, the recurrent net can better adapt at evaluation time if the quality of observations changes.

(a) Pong (b) Frostbite (c) Double Dunk

Figure 1: Nearly all Atari 2600 games feature moving objects. Given only one frame of input, Pong, Frostbite, and Double Dunk are all POMDPs because a single observation does not reveal the velocity of the ball (Pong, Double Dunk) or the velocity of the icebergs (Frostbite).

agent has encountered. Thus DQN will be unable to master games that require the player to remember events more distant than four screens in the past. Put differently, any game that requires a memory of more than four frames will appear non-Markovian to DQN because the future game states (and rewards) depend on more than just DQN's current input. Instead of a Markov Decision Process (MDP), the game

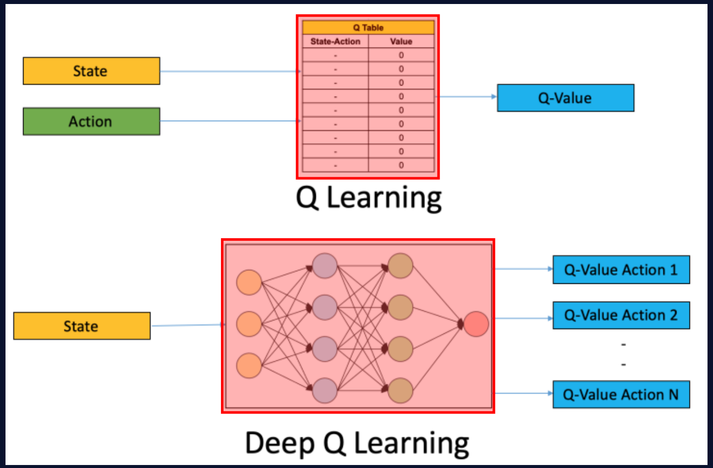
Minsuk Sung
(minsuksung@korea.ac.kr)

INTRODUCTION

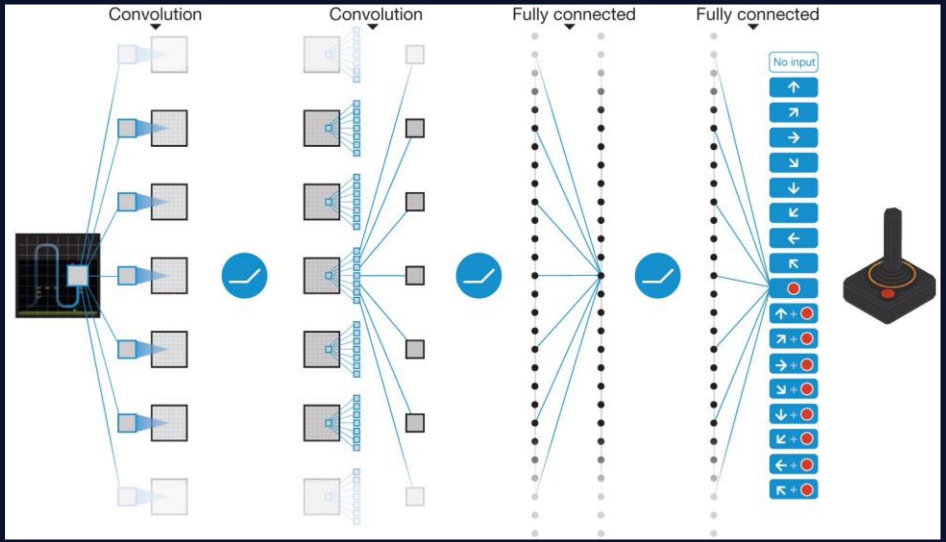
Minsuk Sung
(minsuksung@korea.ac.kr)

Deep Q-Network

- **Q-Learning**
  - Estimating the state-action values (Q-values) of executing an action from a given state
    - Q-values function
      - $Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$
  - Limitations
    - Hard to estimate Q-value when too many unique states exist
- **Deep Q-Network (DQN)** *[Mnih et al., 2015]*
  - Approximating the Q-values using neural network parameterized by weights and biases collectively denoted as θ
    - Loss function
      - $L(s,a|\theta_i) = \left(r + \gamma \max_{a'} Q(s',a'|\theta_i) - Q(s,a|\theta_i)\right)^2$
    - Weight parameter update
      - $\theta_{i+1} = \theta_i + \alpha \nabla_\theta L(\theta_i)$
  - Limitations
    - Hard to approximate Q-value on partially-observable system



Q-Learning vs DQN

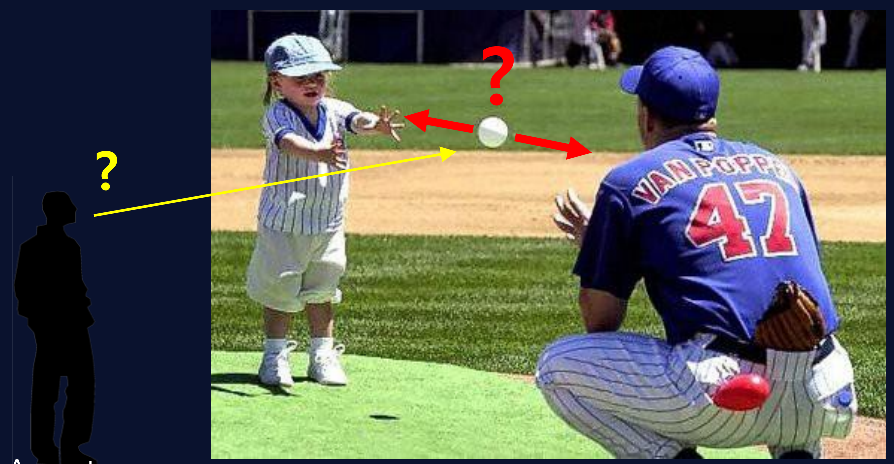


DQN Architecture

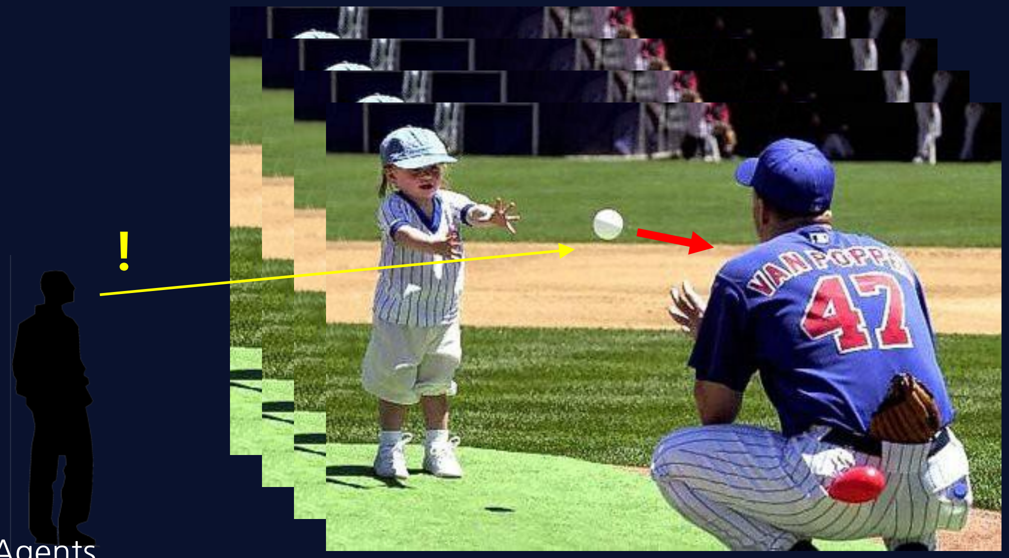
Minsuk Sung
(minsuksung@korea.ac.kr)

Full Observability

- **Markov Decision Process (MDP)**
  - Described by a 4-tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$)
    - At each timestep t, an agent interacting with the MDP observes a state $s_t \in \mathcal{S}$ which determines the reward $r_t \sim \mathcal{R}(s_t, a_t)$ and next state $s_{t+1} \sim \mathcal{P}(s_t, a_t)$
  - Limitations of MDP
    - Rare that the full state of the system can be provided to the agent or even determined



Agents

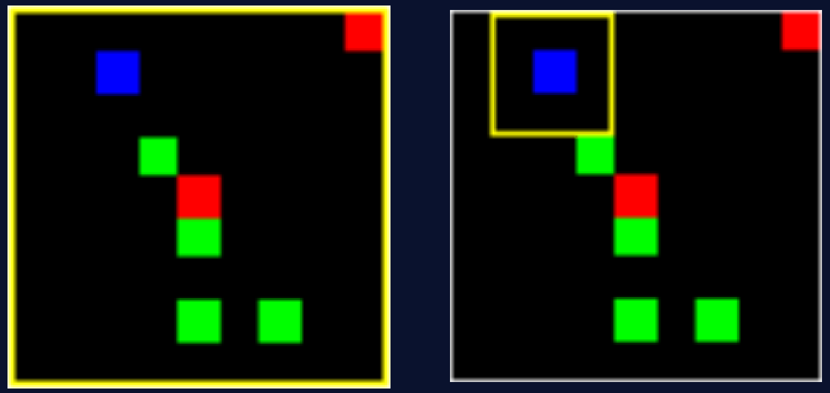


Agents

Minsuk Sung
(minsuksung@korea.ac.kr)

Partial Observability

- **Partially-Observable Markov Decision Process (POMDP)**
  - Described by a 6-tuple ($\mathcal{S},\ \mathcal{A},\ \mathcal{P},\ \mathcal{R},\ \Omega,\ \mathrm{O}$)
    - $\mathcal{S},\ \mathcal{A},\ \mathcal{P},\ \mathcal{R}$ are same with MDP
    - Agent receives an observation $o \in \Omega$ and o generated from the underlying system state according to the probability distribution $o \sim \mathrm{O}$
  - Better captures the dynamics of many real-world environments
    - In the general case, estimating a Q-value from an observation can be different since $Q(o, a|\theta) \neq Q(s, a|\theta)$



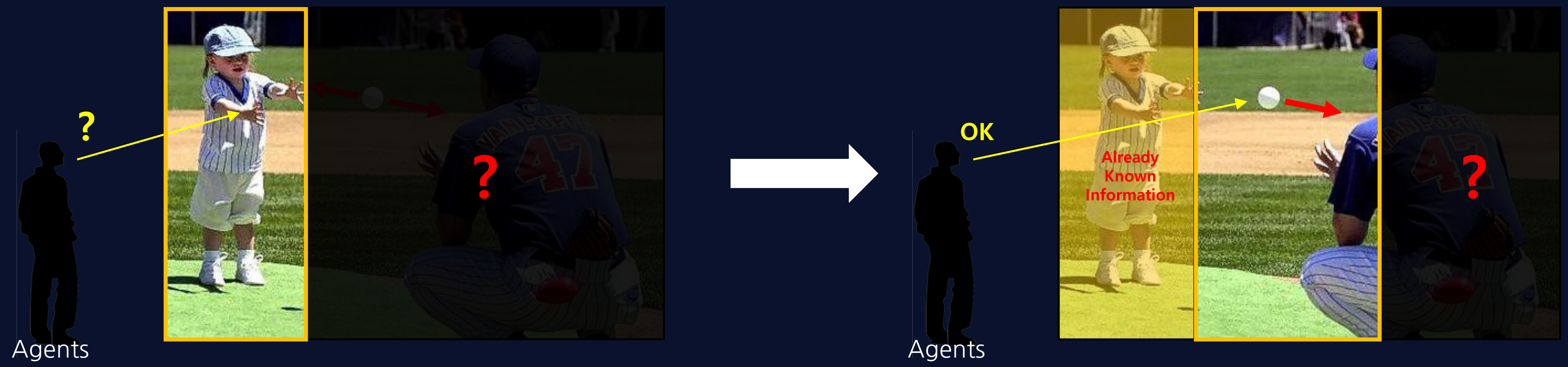
Examples of frozen lake on MDP (Left) and POMDP (Right)

[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

Minsuk Sung
(minsuksung@korea.ac.kr)

Goal

To narrow the gap between $Q(o, a|\theta)$ and $Q(s, a|\theta)$
with adding recurrency to Deep Q-Network on POMDP



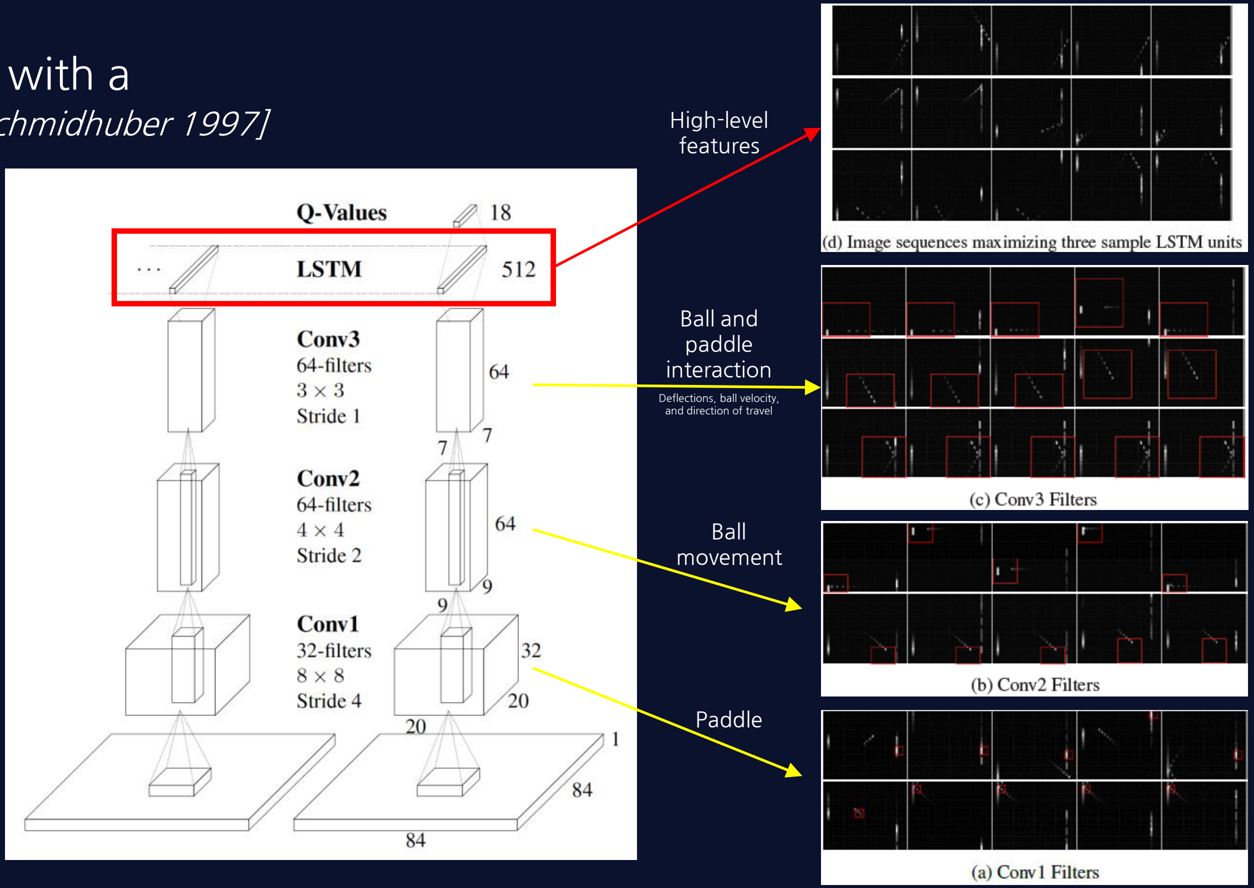
Minsuk Sung
(minsuksung@korea.ac.kr)

DRQN ARCHITECTURE

Minsuk Sung
(minsuksung@korea.ac.kr)

Architecture of DRQN

- **DRQN = DQN + LSTM**
  - DRQN replaces DQN's first fully connected layer with a Long Short-Term Memory (LSTM) *[Hochreiter and Schmidhuber 1997]*
- **Overall Architecture**
  - Input Layer
    - Taking a single 84×84 preprocessed image, instead of the last four images required by DQN
  - Conv1 ~ Conv3 Layer
    - Detecting paddle, ball, and interaction between each objects
  - LSTM Layer
    - Detecting high-level features
      - The agent missing the ball
      - Ball reflections off of paddles
      - Ball reflections off the walls
  - Fully Connected Layer



Q&A
(1 / 3)

Minsuk Sung
(minsuksung@korea.ac.kr)

STABLE
RECURRENT UPDATES

Minsuk Sung
(minsuksung@korea.ac.kr)

Two types of Updates

- **Bootstrapped Sequential Updates**
  - Episodes are selected randomly from replay memory
  - Updates begin at the beginning of the episode and proceed forward through time to conclusion of the episode
  - The targets at each timestep are generated from the target Q-network
  - RNN's hidden state is carried forward throughout the episode
- **Bootstrapped Random Updates**
  - Episodes are selected randomly from replay memory
  - Updates begin at the random points in the episode and proceed for only unroll iterations timesteps
  - The targets at each timestep are generated from the target Q-network
  - RNN's initial state is zeroed at the start of the update

[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

Minsuk Sung
(minsuksung@korea.ac.kr)

Sequential Updates vs Random Updates

- **Sequential Updates**
  - Having advantage of carrying of carrying the LSTM's hidden state forward from the beginning of the episode
  - Violating DQN's random sampling policy by sampling experiences sequentially for a full episode
- **Random Updates**
  - Better adjusting to the policy of randomly sampling experience
  - LSTM hidden state must be zeroed at the start of each update
    - Making it harder for the LSTM to learn functions that span longer time scales
  - Experiments show both types of updates yield convergent policies with similar performance
    - All results in these paper use the randomized update strategy to limit complexity

ATARI GAMES:
MDP or POMDP?

Minsuk Sung
(minsuksung@korea.ac.kr)

Atari Games: MDP or POMDP?

- **Atari 2600 Games**
  - Fully described by 128 bytes of console RAM
  - Observed only the console-generated game screens
- **POMDP → MDP**
  - A single screen during Atari games is insufficient to determine the state of the system
  - DQN infers the full state of an Atari game by expanding the state representation to encompass the last four game screens
- **Modification to the Pong game**
  - To introduce partial observability to Atari games without reducing the number of input frames

[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

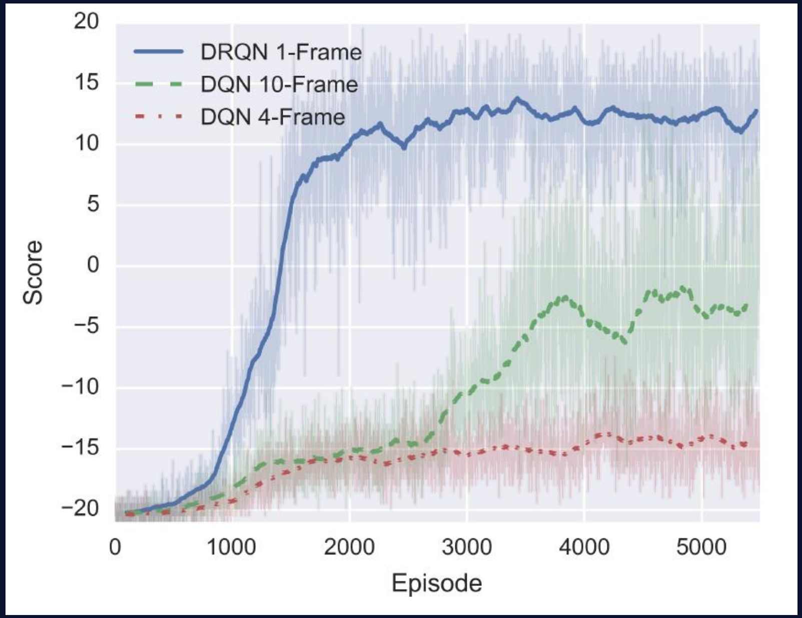
FLICKERING PONG POMDP

Minsuk Sung
(minsuksung@korea.ac.kr)

Flickering Pong POMDP

- **Flickering Pong POMDP**
  - A modification to the classic game of Pong
  - At each time step, the screen is either fully reveal or fully obscured with probability $p = 0.5$
  - Obscuring frames induces an incomplete memory of observations
- **Three types of networks to play Flickering Pong**
  - The recurrent 1-frame DRQN
  - A standard 4-frame DQN *[Mnih et al., 2015]*
  - An augmented 10-frame DQN

Flickering	DRQN $\pm std$	DQN $\pm std$
Asteroids	1032 ($\pm$410)	1010 ($\pm$535)
Beam Rider	618 ($\pm$115)	**1685.6** ($\pm$875)
Bowling	65.5 ($\pm$13)	57.3 ($\pm$8)
Centipede	4319.2 ($\pm$4378)	5268.1 ($\pm$2052)
Chopper Cmd	1330 ($\pm$294)	1450 ($\pm$787.8)
Double Dunk	-14 ($\pm$2.5)	-16.2 ($\pm$2.6)
Frostbite	414 ($\pm$494)	436 ($\pm$462.5)
Ice Hockey	-5.4 ($\pm$2.7)	-4.2 ($\pm$1.5)
Ms. Pacman	1739 ($\pm$942)	1824 ($\pm$490)
Pong	**12.1** ($\pm$2.2)	-9.9 ($\pm$3.3)

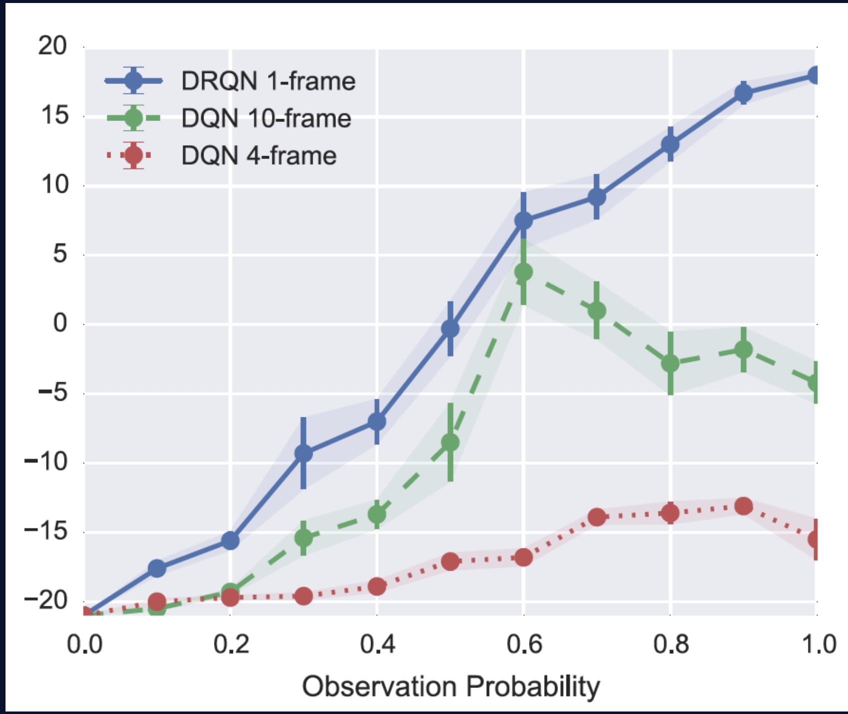


[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

Minsuk Sung
(minsuksung@korea.ac.kr)

Generalization Performance

- **Evaluating the Best Policies for DRQN, 10-frame DQN, and 4-frame DQN**
  - Trained on Flickering Pong with $p = 0.5$
  - Evaluated against different p values
- **Observation Quality**
  - DRQN learns a policy which allows performance to scale as a function of observation quality
    - More information, more high score
  - Valuable for domains in which the quality of observations varies through time



Minsuk Sung
(minsuksung@korea.ac.kr)

Q&A

(2 / 3)

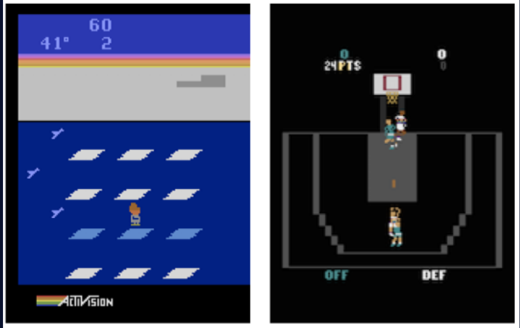
Minsuk Sung
(minsuksung@korea.ac.kr)

EVALUATION
ON STANDARD ATARI GAMES

Minsuk Sung
(minsuksung@korea.ac.kr)

Evaluation on Standard Atari Games

- Selected 9 games for Evaluations for DRQN
  1. *Asteroids*
  2. ***Beam Rider** → Worse than DQN*
  3. *Bowling*
  4. *Centipede*
  5. *Chopper Command*
  6. ***Double Dunk** → Better than DQN*
  7. ***Frostbite** → Better than DQN*
  8. *Ice Hockey*
  9. *Ms. Pacman*



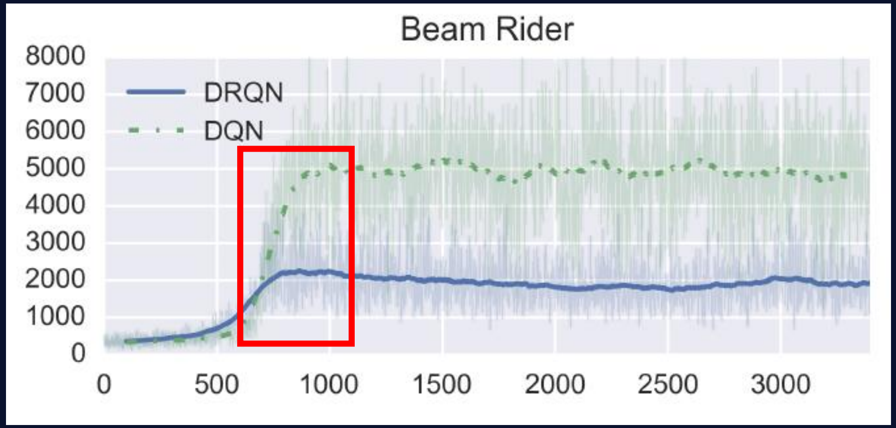
(b) Frostbite (c) Double Dunk

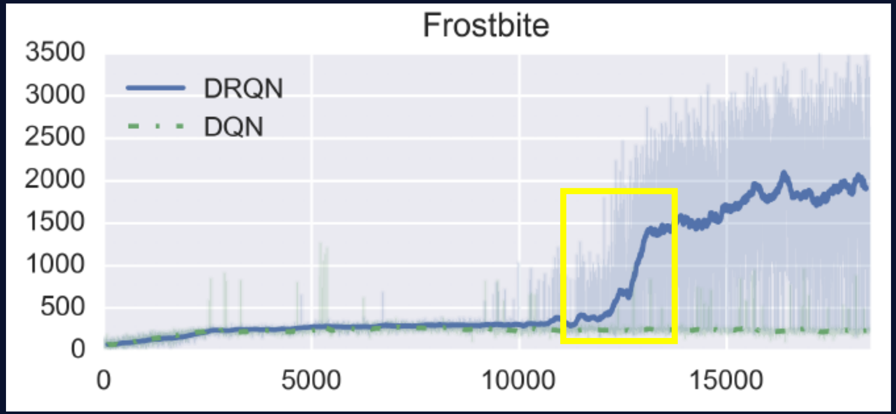
	DRQN $\pm std$	DQN $\pm std$	
Game		Ours	Mnih et al.
Asteroids	1020 ($\pm$312)	1070 ($\pm$345)	1629 ($\pm$542)
Beam Rider	3269 ($\pm$1167)	**6923** ($\pm$1027)	6846 ($\pm$1619)
Bowling	62 ($\pm$5.9)	72 ($\pm$11)	42 ($\pm$88)
Centipede	3534 ($\pm$1601)	3653 ($\pm$1903)	8309 ($\pm$5237)
Chopper Cmd	2070 ($\pm$875)	1460 ($\pm$976)	6687 ($\pm$2916)
Double Dunk	**-2** ($\pm$7.8)	-10 ($\pm$3.5)	-18.1 ($\pm$2.6)
Frostbite	**2875** ($\pm$535)	519 ($\pm$363)	328.3 ($\pm$250.5)
Ice Hockey	-4.4 ($\pm$1.6)	-3.5 ($\pm$3.5)	-1.6 ($\pm$2.5)
Ms. Pacman	2048 ($\pm$653)	2363 ($\pm$735)	2311 ($\pm$525)

Minsuk Sung
(minsuksung@korea.ac.kr)

Evaluation on Standard Atari Games

- Selected 9 games for Evaluations for DRQN
  1. *Asteroids*
  2. ***Beam Rider** → Worse than DQN*
  3. *Bowling*
  4. *Centipede*
  5. *Chopper Command*
  6. ***Double Dunk** → Better than DQN*
  7. ***Frostbite** → Better than DQN*
  8. *Ice Hockey*
  9. *Ms. Pacman*





[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

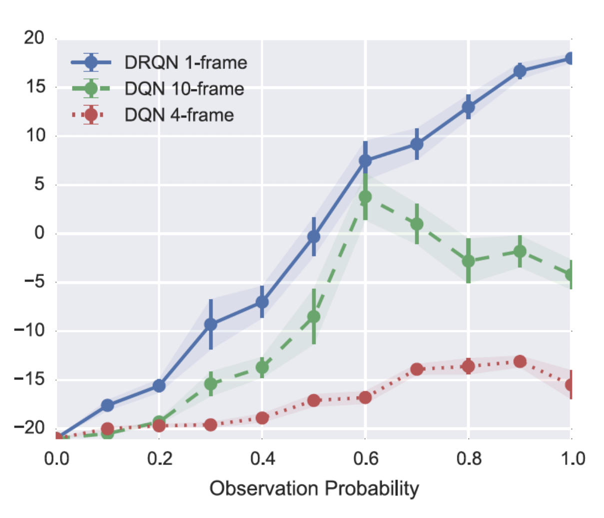
Minsuk Sung
(minsuksung@korea.ac.kr)

MDP to POMDP GENERALIZATION

Minsuk Sung
(minsuksung@korea.ac.kr)

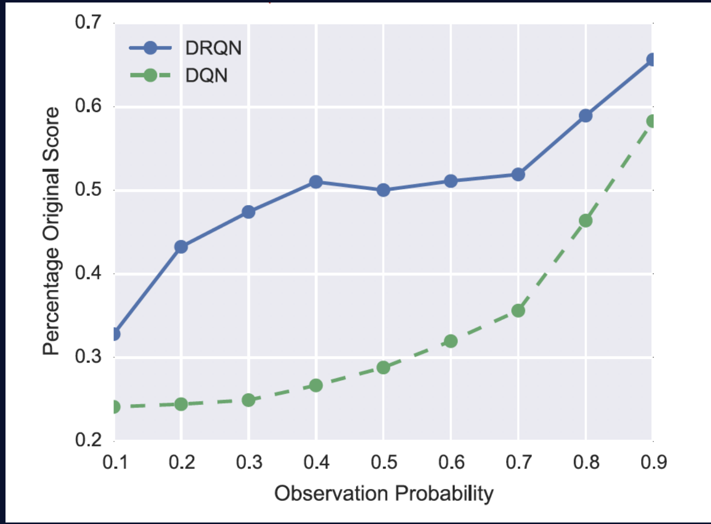
MDP to POMDP Generalization

- **Comparison Performance between DQN and DRQN**
  - Train: POMDP / Test: MDP
  - Train: MDP / Test: POMDP
- **Recurrent Controller**
  - Robustness against missing information, even trained with full state information



Game	DRQN $\pm std$	DQN $\pm std$ Ours	DQN $\pm std$ Mnih et al.
Asteroids	1020 (±312)	1070 (±345)	1629 (±542)
Beam Rider	3269 (±1167)	**6923** (±1027)	6846 (±1619)
Bowling	62 (±5.9)	72 (±11)	42 (±88)
Centipede	3534 (±1601)	3653 (±1903)	8309 (±5237)
Chopper Cmd	2070 (±875)	1460 (±976)	6687 (±2916)
Double Dunk	**-2** (±7.8)	-10 (±3.5)	-18.1 (±2.6)
Frostbite	**2875** (±535)	519 (±363)	328.3 (±250.5)
Ice Hockey	-4.4 (±1.6)	-3.5 (±3.5)	-1.6 (±2.5)
Ms. Pacman	2048 (±653)	2363 (±735)	2311 (±525)

Table 1: On standard Atari games, DRQN performance parallels DQN, excelling in the games of Frostbite and Double Dunk, but struggling on Beam Rider. Bolded font indicates statistical significance between DRQN and our DQN.[5]



[RLPR-S2] Matthew Hausknecht and Peter Stone, **"Deep Recurrent Q-Learning for Partially Observable MDPs"**, *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

DISCUSSION AND CONCLUSION

Minsuk Sung
(minsuksung@korea.ac.kr)

Discussion and Conclusion

- **Better Performance than DQN on POMDP**
  - DRQN handling the noisy and incomplete characteristic of POMDPs by combining a LSTM with DQN
  - Only a single frame at each step, DRQN still integrating information across frames to detect relevant information
- **Generalization Performance**
  - Trained with partial observations
    - DRQN learns policies that are both robust enough to handle to missing game screens, and scalable enough to improve performance as more data becomes available
  - Trained with fully observations
    - DRQN performs better than DQN's at all levels of partial information
- **Adding Recurrency**
  - Experiments show that LSTM is viable method for handling multiple state observations

[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

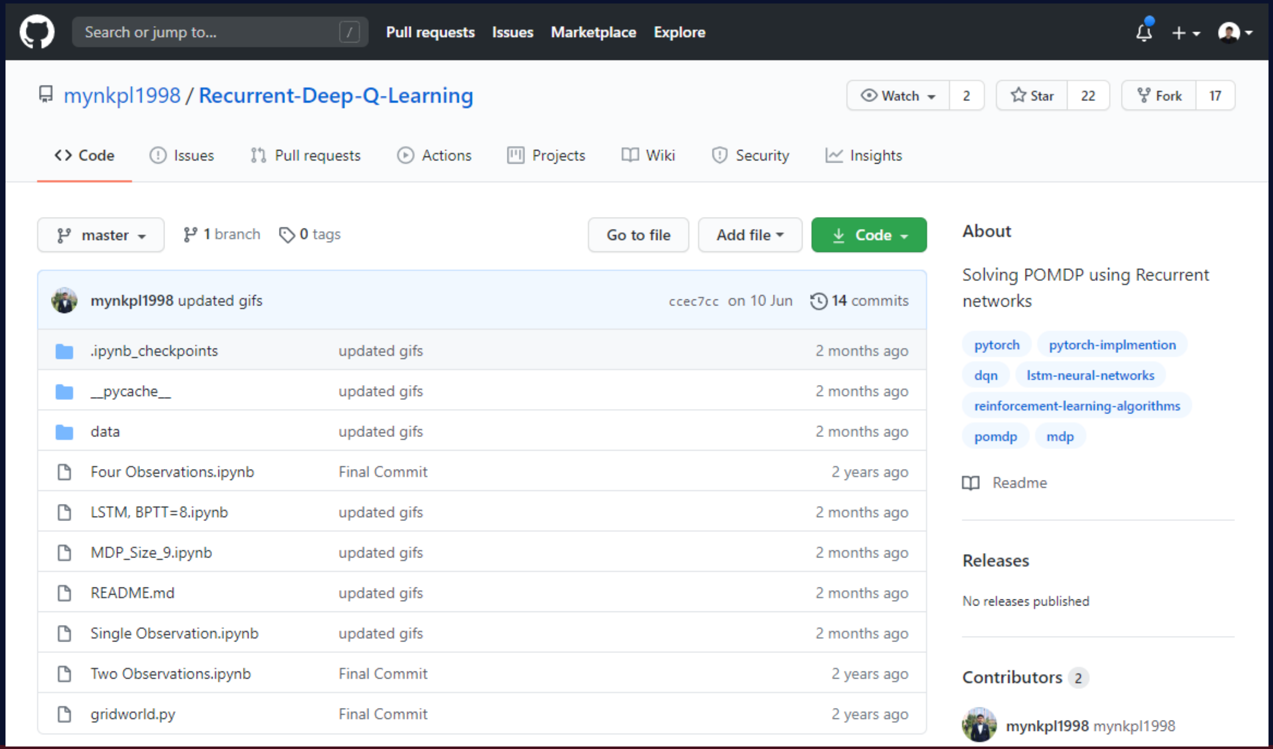
Minsuk Sung
(minsuksung@korea.ac.kr)

IMPLEMENTATION

Minsuk Sung
(minsuksung@korea.ac.kr)

DQRN

- GitHub
  - Caffe : https://github.com/mhauskn/dqn/tree/recurrent
  - PyTorch : https://github.com/mynkpl1998/Recurrent-Deep-Q-Learning



Minsuk Sung
(minsuksung@korea.ac.kr)

DQRN *(PyTorch ver)*

How to Run ?

I ran the experiment for the following cases. The corresponding code/jupyter files are linked to each experiment.

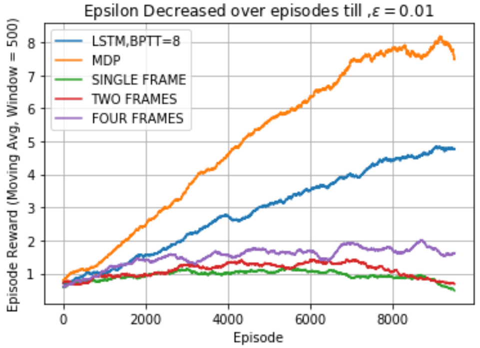
- MDP Case - The underlying state was fully visible. The whole grid was given as the input to the agent.
- Single Observation - In this case, the most recent observation was used as the input to agent.
- Last Two Observations - In this case, the last two most recent observation was used as the input to agent to encode the temporal information among observations.
- LSTM Case - In this case, an LSTM layer is used to pass the temporal information among observations.

Learned Policies

Fully Observable	Single Observation	LSTM
		

Results

The figure given below compares the performance of different cases. MDP case is the best we can do as the underlying state is fully visible to the agent. However, the challenge is to perform better given an observation. The graph clearly shows the LSTM consistently performed better as the total reward per episode was much higher than using some last k-frames.



[RLPR-S2] Matthew Hausknecht and Peter Stone, "**Deep Recurrent Q-Learning for Partially Observable MDPs**", *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

Minsuk Sung
(minsuksung@korea.ac.kr)

Q&A
(3 / 3)

Minsuk Sung
(minsuksung@korea.ac.kr)

THANK YOU
FOR LISTENING

Minsuk Sung
(minsuksung@korea.ac.kr)

References

- Paper
  - https://www.aaai.org/ocs/index.php/FSS/FSS15/paper/viewPaper/11673
- Blog
  - https://sumniya.tistory.com/22
  - https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-6-partial-observability-and-deep-recurrent-q-68463e9aeefc
  - https://jay.tech.blog/2017/02/06/deep-recurrent-q-learning/
  - https://whereisend.tistory.com/108
  - https://ddanggle.github.io/demystifyingDL
- Code
  - https://github.com/mhauskn/dqn/tree/recurrent
  - https://github.com/mynkpl1998/Recurrent-Deep-Q-Learning
  - https://github.com/qfettes/DeepRL-Tutorials/blob/master/11.DRQN.ipynb
  - https://github.com/awjuliani/DeepRL-Agents/blob/master/Deep-Recurrent-Q-Network.ipynb