Deep Recurrent Q-Learning for Partially Observable MDPs

Matthew Hausknecht and Peter Stone

University of Texas at Austin

November 13, 2015

Motivation

Intelligent decision making is the heart of AI

Motivation

Intelligent decision making is the heart of AI

Desire agents capable of learning to act intelligently in diverse environments

Motivation

Intelligent decision making is the heart of AI

Desire agents capable of learning to act intelligently in diverse environments

Reinforcement Learning provides a general learning framework

Motivation

Intelligent decision making is the heart of AI

Desire agents capable of learning to act intelligently in diverse environments

Reinforcement Learning provides a general learning framework

RL + deep neural networks yields robust controllers that learn from pixels (DQN)

Motivation

Intelligent decision making is the heart of AI

Desire agents capable of learning to act intelligently in diverse environments

Reinforcement Learning provides a general learning framework

RL + deep neural networks yields robust controllers that learn from pixels (DQN)

DQN lacks mechanisms for handling partial observability

Motivation

Intelligent decision making is the heart of AI

Desire agents capable of learning to act intelligently in diverse environments

Reinforcement Learning provides a general learning framework

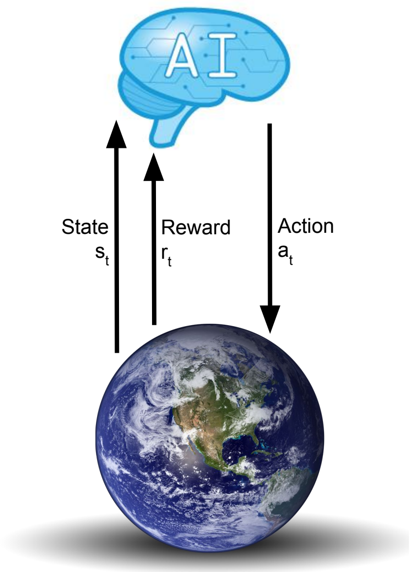
RL + deep neural networks yields robust controllers that learn from pixels (DQN)

DQN lacks mechanisms for handling partial observability

Extend DQN to handle Partially Observable Markov Decision Processes (POMDPs)

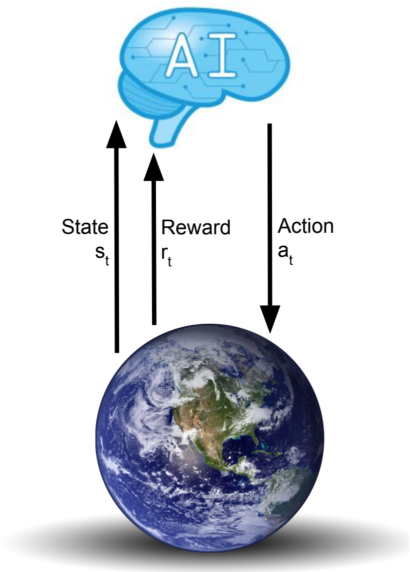
Outline

Markov Decision Process (MDP)



At each timestep Agent performs actions a_t and receives reward r_t and state s_{t+1} from the environment

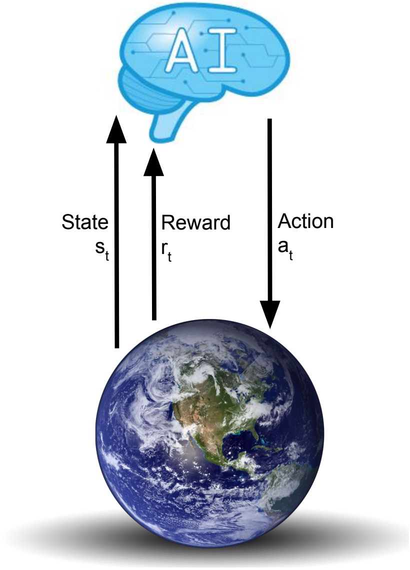
Markov Decision Process (MDP)



At each timestep Agent performs actions a_t and receives reward r_t and state s_{t+1} from the environment

Markov property ensures that s_{t+1} depends only on s_t, a_t

Markov Decision Process (MDP)

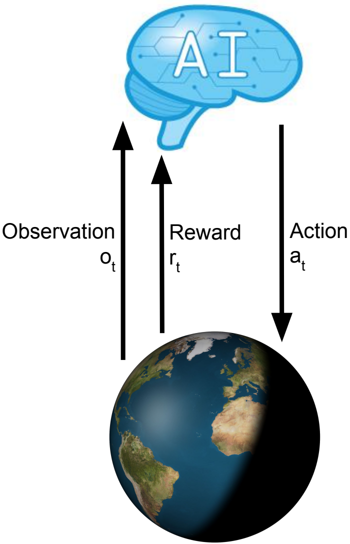


At each timestep Agent performs actions a_t and receives reward r_t and state s_{t+1} from the environment

Markov property ensures that s_{t+1} depends only on s_t, a_t

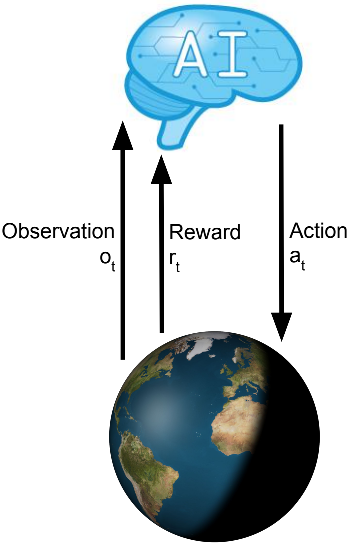
Learning an optimal policy π^* requires no memory of past states

Partially Observable Markov Decision Process (POMDP)



True state of environment is hidden. Observations o_t provide only partial information.

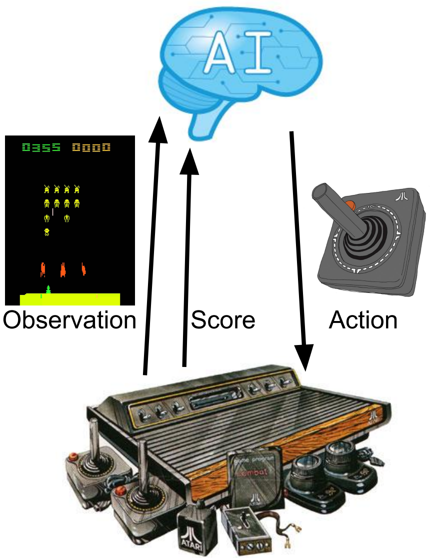
Partially Observable Markov Decision Process (POMDP)



True state of environment is hidden. Observations o_t provide only partial information.

Memory of past observations may help understand true system state, improve the policy

Atari Domain



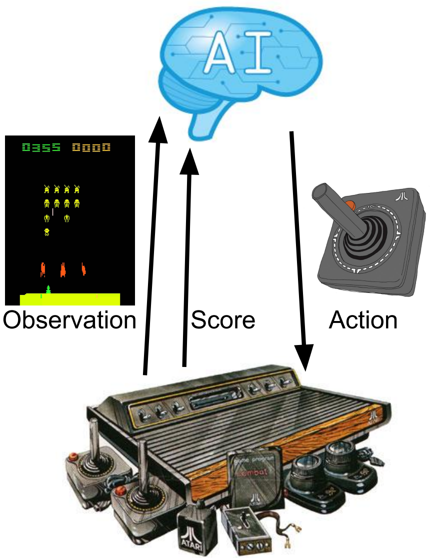
160×210 state space
$\rightarrow 84 \times 84$ grayscale

18 discrete actions

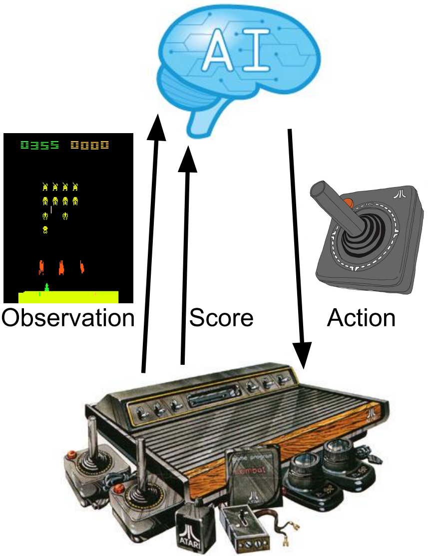
Rewards clipped $\in \{-1, 0, 1\}$

Source: www.arcadelearningenvironment.org

Atari Domain: MDP or POMDP?

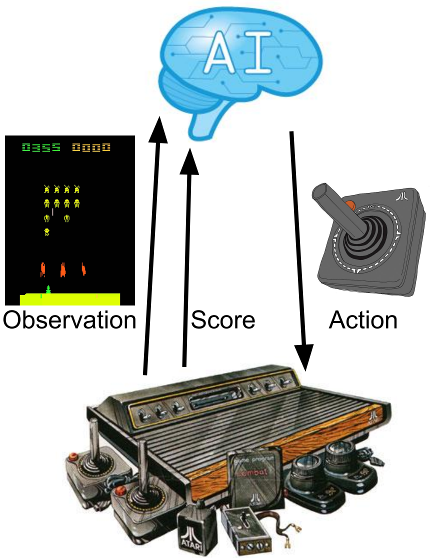


Atari Domain: MDP or POMDP?



Depends on the state representation!

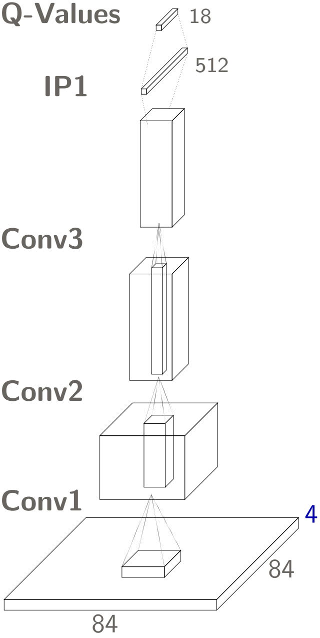
Atari Domain: MDP or POMDP?



Depends on the state representation!

- Single Frame ⇒ POMDP
- Four Frames ⇒ MDP
- Console RAM ⇒ MDP

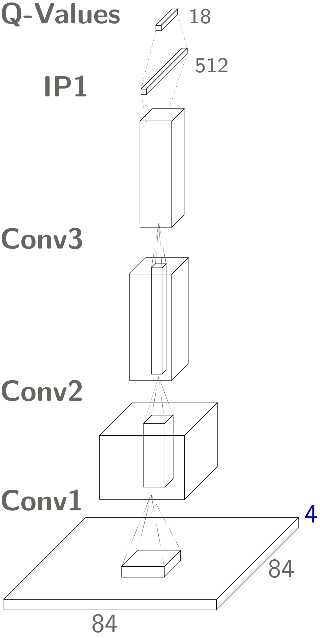
Deep Q-Network (DQN)



Model-free Reinforcement Learning method using deep neural network as Q-Value function approximator Mnih et al. (2015)

Takes the last four game screens as input: enough to make most Atari games Markov

Deep Q-Network (DQN)

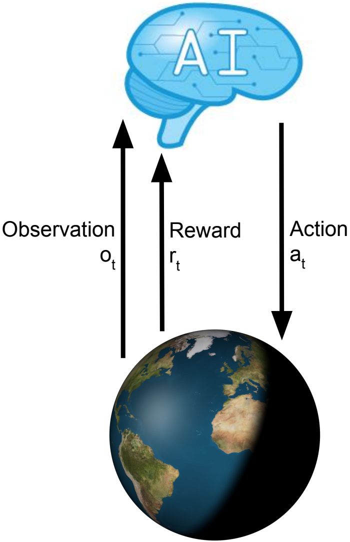


Model-free Reinforcement Learning method using deep neural network as Q-Value function approximator Mnih et al. (2015)

Takes the last four game screens as input: enough to make most Atari games Markov

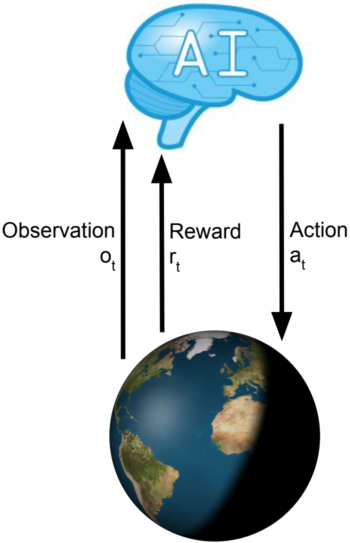
How well does DQN perform in partially observed domains?

Flickering Atari



Induce partial observability by stochastically obscuring the game screen

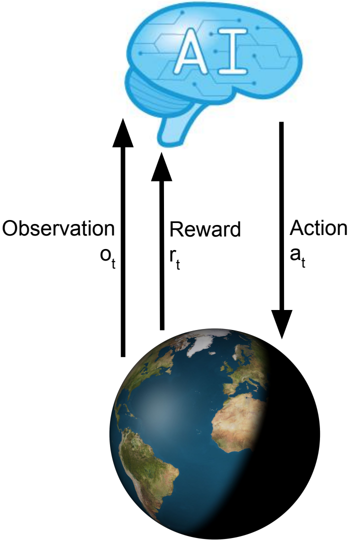
Flickering Atari



Induce partial observability by stochastically obscuring the game screen

$$o_t = \begin{cases} s_t & \text{with } p = \frac{1}{2} \\ < 0, \ldots, 0 > & \text{otherwise} \end{cases}$$

Flickering Atari



Induce partial observability by stochastically obscuring the game screen

$$o_t = \begin{cases} s_t & \text{with } p = \frac{1}{2} \\ < 0, \ldots, 0 > & \text{otherwise} \end{cases}$$

Game state must now be inferred from past observations

DQN Pong

True Game Screen

Perceived Game Screen

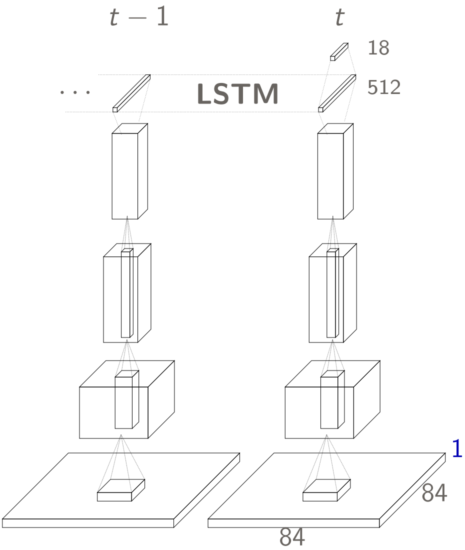
DQN Flickering Pong

True Game Screen

Perceived Game Screen

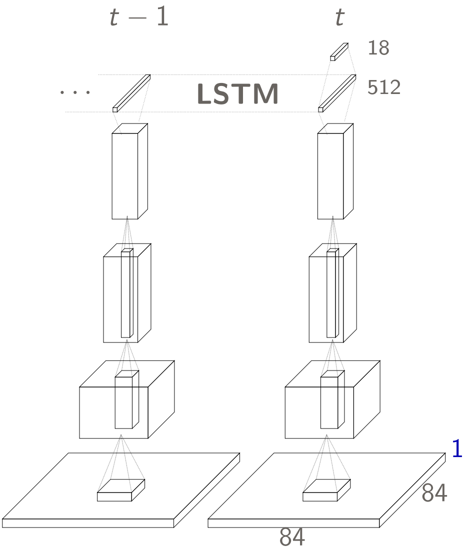
Outline

Deep Recurrent Q-Network



Long Short Term Memory
Hochreiter (1997)

Deep Recurrent Q-Network

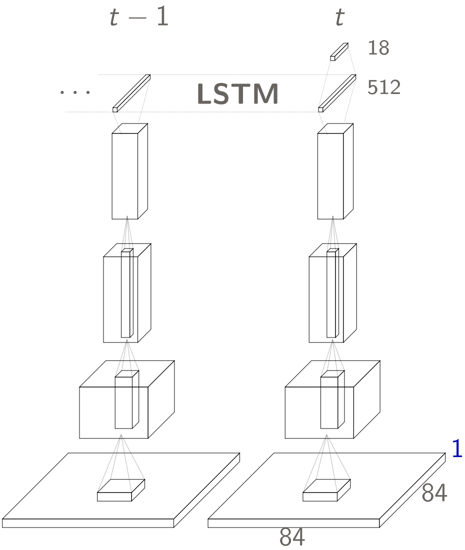


Long Short Term Memory
Hochreiter (1997)

Identical to DQN Except:

- Replaces DQN's **IP1** with recurrent **LSTM** layer of same dimension
- Each timestep takes a single frame as input

Deep Recurrent Q-Network



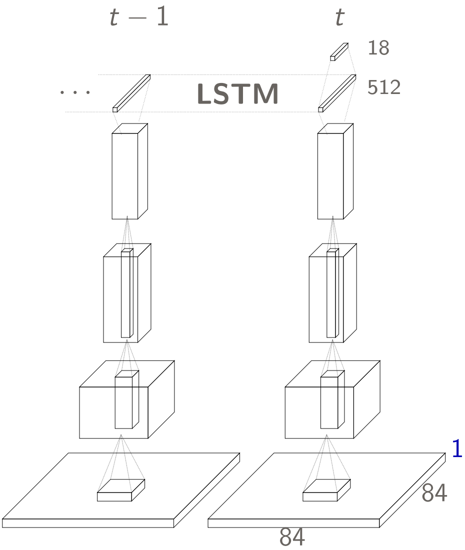
Long Short Term Memory
Hochreiter (1997)

Identical to DQN Except:

- Replaces DQN's **IP1** with recurrent **LSTM** layer of same dimension
- Each timestep takes a single frame as input

LSTM provides a selective memory of past game states

Deep Recurrent Q-Network



Long Short Term Memory
Hochreiter (1997)

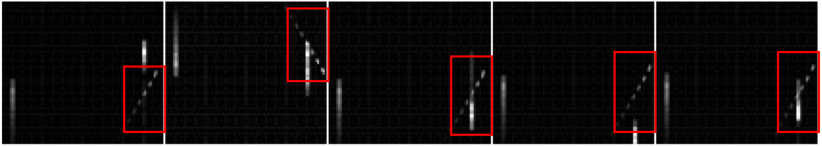
Identical to DQN Except:

- Replaces DQN's **IP1** with recurrent **LSTM** layer of same dimension
- Each timestep takes a single frame as input

LSTM provides a selective memory of past game states

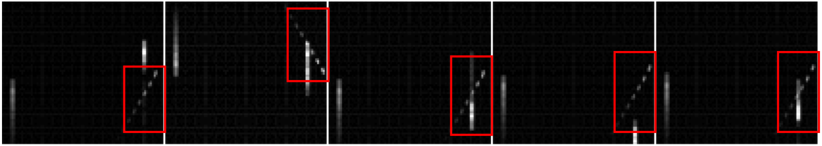
Trained end-to-end using BPTT: unrolled for last 10 timesteps

DRQN Maximal Activations

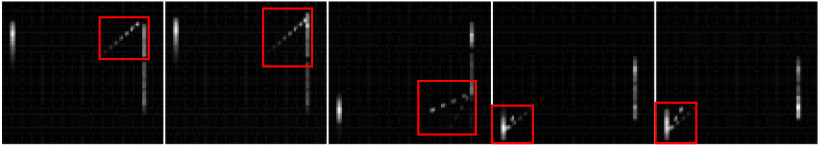


Unit detects the agent missing the ball

DRQN Maximal Activations

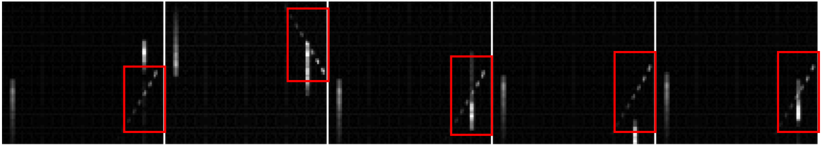


Unit detects the agent missing the ball

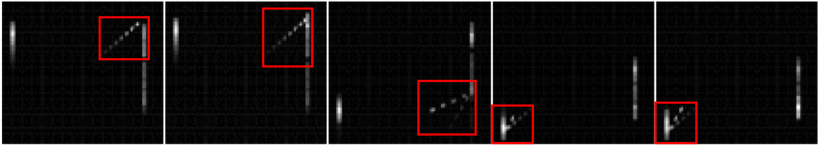


Unit detects ball reflection on paddle

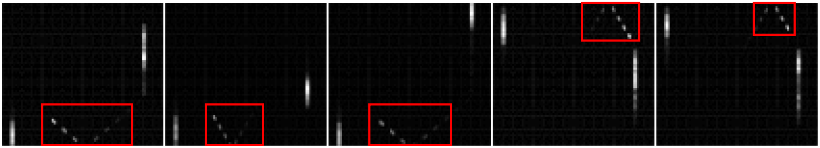
DRQN Maximal Activations



Unit detects the agent missing the ball



Unit detects ball reflection on paddle



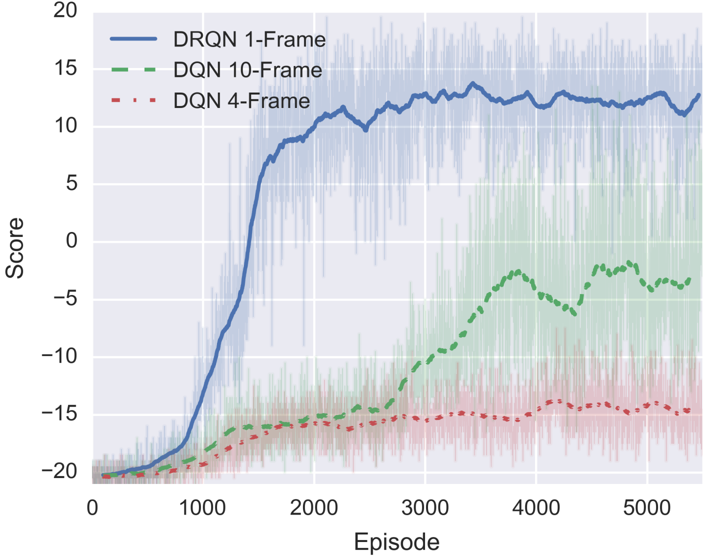
Unit detects ball reflection on wall

Outline

DRQN Flickering Pong

True Game Screen Perceived Game Screen

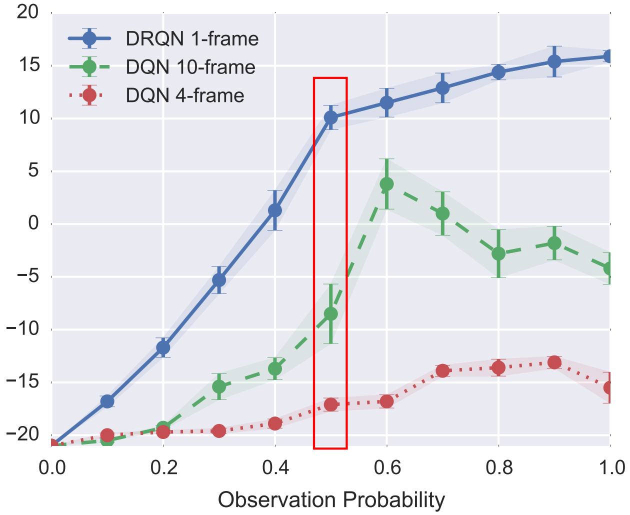
Flickering Pong



Pong Generalization: POMDP $\Rightarrow$ MDP

How does DRQN generalize when trained on Flickering Pong and evaluated on standard Pong?

Pong Generalization: POMDP ⇒ MDP



Performance on Flickering Atari Games

Game	10-frame DRQN $\pm std$	10-frame DQN $\pm std$
Pong	**12.1** ($\pm$2.2)	-9.9 ($\pm$3.3)

Performance on Flickering Atari Games

Game	10-frame DRQN $\pm std$	10-frame DQN $\pm std$
Pong	**12.1** ($\pm$2.2)	-9.9 ($\pm$3.3)
Beam Rider	618 ($\pm$115)	**1685.6** ($\pm$875)

Performance on Flickering Atari Games

Game	10-frame DRQN $\pm std$	10-frame DQN $\pm std$
Pong	**12.1** ($\pm$2.2)	-9.9 ($\pm$3.3)
Beam Rider	618 ($\pm$115)	**1685.6** ($\pm$875)
Asteroids	1032 ($\pm$410)	1010 ($\pm$535)
Bowling	65.5 ($\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)

Performance on Standard Atari Games

Game	10-frame DRQN $\pm$*std*	10-frame DQN $\pm$*std*
Double Dunk	**-2** ($\pm$7.8)	-10 ($\pm$3.5)
Frostbite	**2875** ($\pm$535)	519 ($\pm$363)

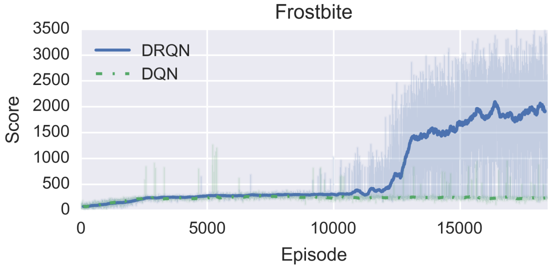
Performance on Standard Atari Games

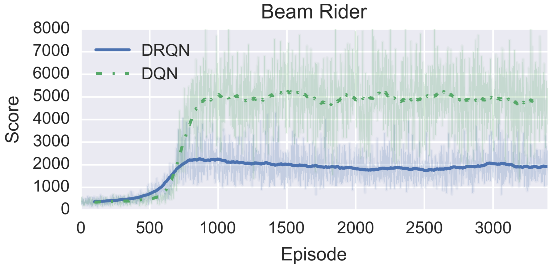
Game	10-frame DRQN $\pm std$	10-frame DQN $\pm std$
Double Dunk	**-2** ($\pm$7.8)	-10 ($\pm$3.5)
Frostbite	**2875** ($\pm$535)	519 ($\pm$363)
Beam Rider	3269 ($\pm$1167)	**6923** ($\pm$1027)

Performance on Standard Atari Games

Game	10-frame DRQN $\pm std$	10-frame DQN $\pm std$
Double Dunk	**-2** ($\pm$7.8)	-10 ($\pm$3.5)
Frostbite	**2875** ($\pm$535)	519 ($\pm$363)
Beam Rider	3269 ($\pm$1167)	**6923** ($\pm$1027)
Asteroids	1020 ($\pm$312)	1070 ($\pm$345)
Bowling	62 ($\pm$5.9)	72 ($\pm$11)
Centipede	3534 ($\pm$1601)	3653 ($\pm$1903)
Chopper Cmd	2070 ($\pm$875)	1460 ($\pm$976)
Ice Hockey	-4.4 ($\pm$1.6)	-3.5 ($\pm$3.5)
Ms. Pacman	2048 ($\pm$653)	2363 ($\pm$735)

Performance on Standard Atari Games





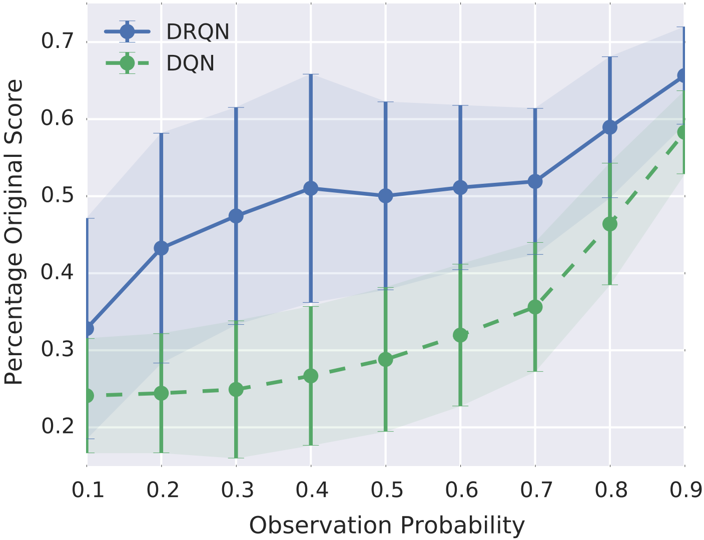
DRQN Frostbite

True Game Screen   Perceived Game Screen

Generalization: MDP $\Rightarrow$ POMDP

How does DRQN generalize when trained on standard Atari and evaluated on flickering Atari?

Generalization: MDP ⇒ POMDP



Outline

Related Work

Deep Recurrent Q-Network

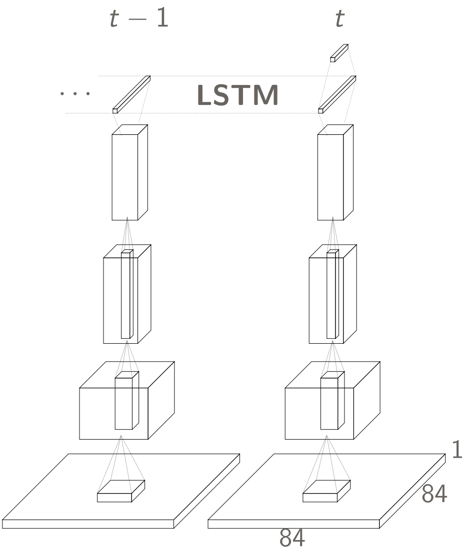
Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. *CoRR*, abs/1506.08941.

Wierstra, D., Foerster, A., Peters, J., and Schmidthuber, J. (2007). Solving deep memory POMDPs with recurrent policy gradients.

Thanks!



LSTM can help deal with partial observability

Largest gains in generalization between MDP $\Leftrightarrow$ POMDP

Future work understanding why DRQN does better/worse on certain games

Source: `https://github.com/mhauskn/dqn/tree/recurrent`

Matthew Hausknecht and Peter Stone

Outline

Computational Efficiency

	Backwards (ms)			Forwards (ms)		
Frames	1	4	10	1	4	10
Baseline	8.82	13.6	26.7	2.0	4.0	9.0
Unroll 1	18.2	22.3	33.7	2.4	4.4	9.4
Unroll 10	77.3	111.3	180.5	2.5	4.4	8.3
Unroll 30	204.5	263.4	491.1	2.5	3.8	9.4

Table : Average milliseconds per backwards/forwards pass. Frames refers to the number of channels in the input image. Baseline is a non recurrent network (e.g. DQN). Unroll refers to an LSTM network backpropagated through time 1/10/30 steps.