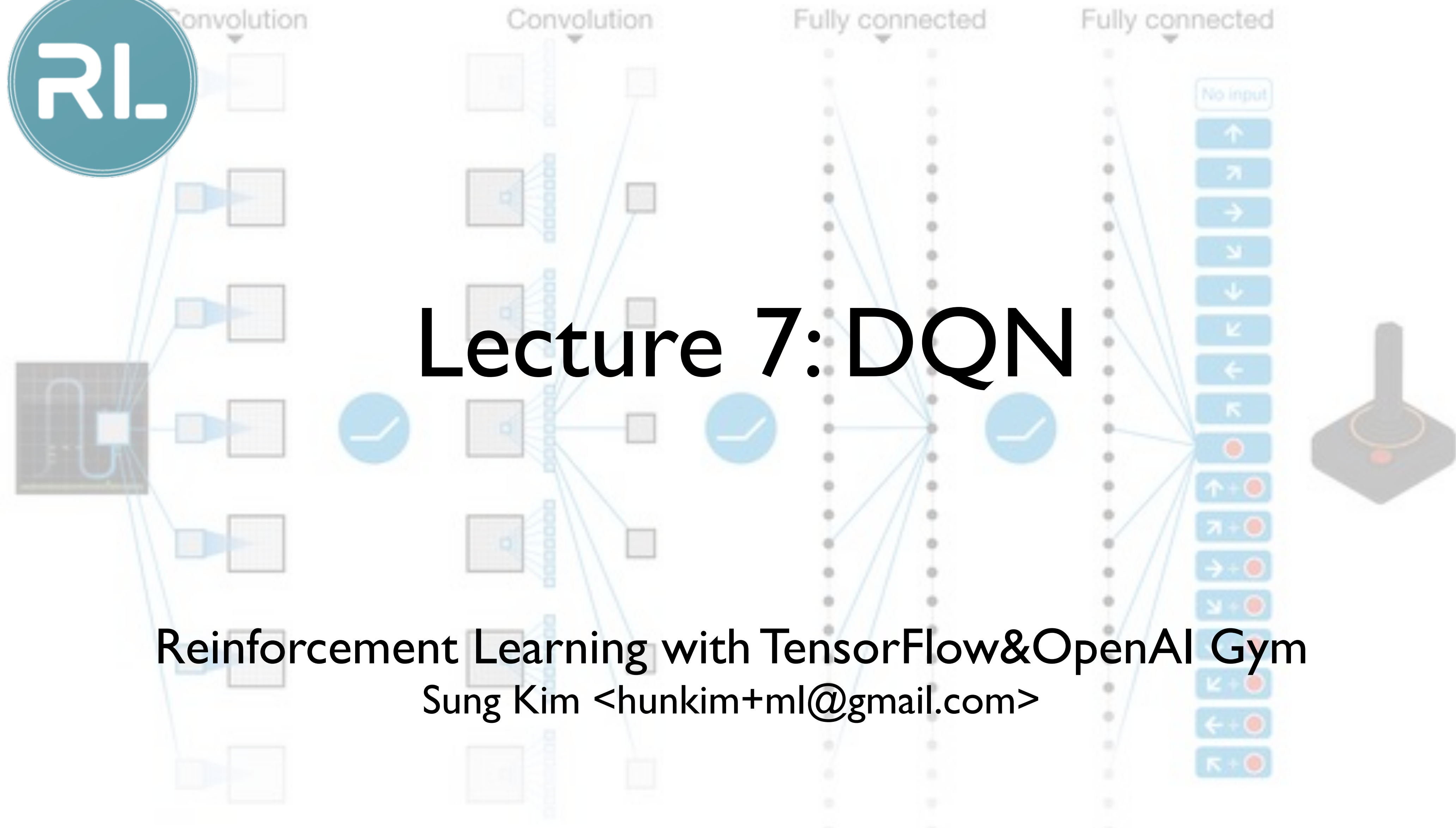
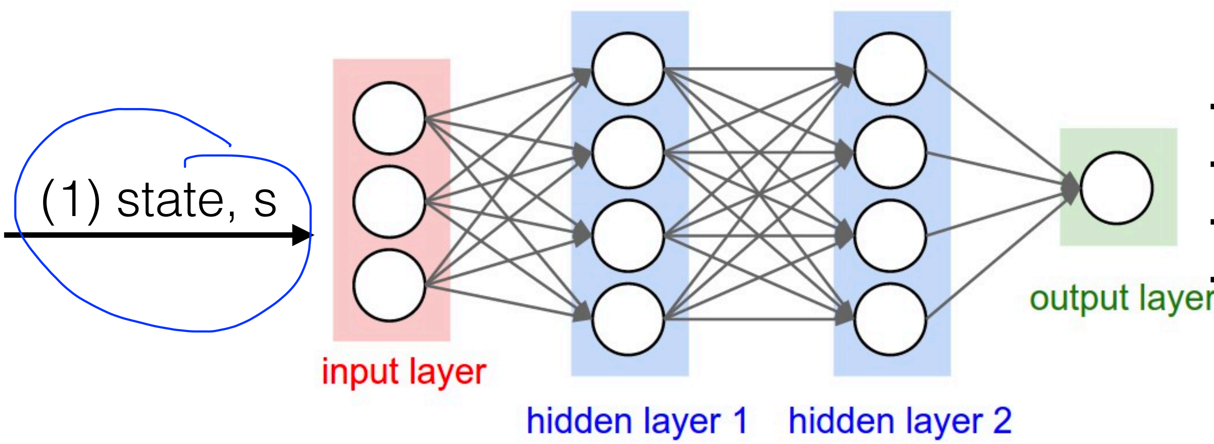
Lecture 7: DQN



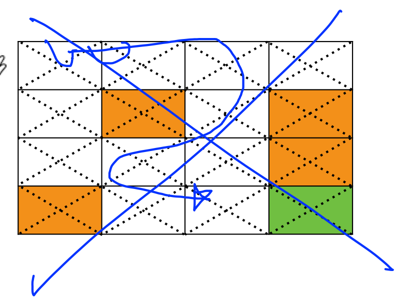
Reinforcement Learning with TensorFlow&OpenAI Gym

Sung Kim <hunkim+ml@gmail.com>

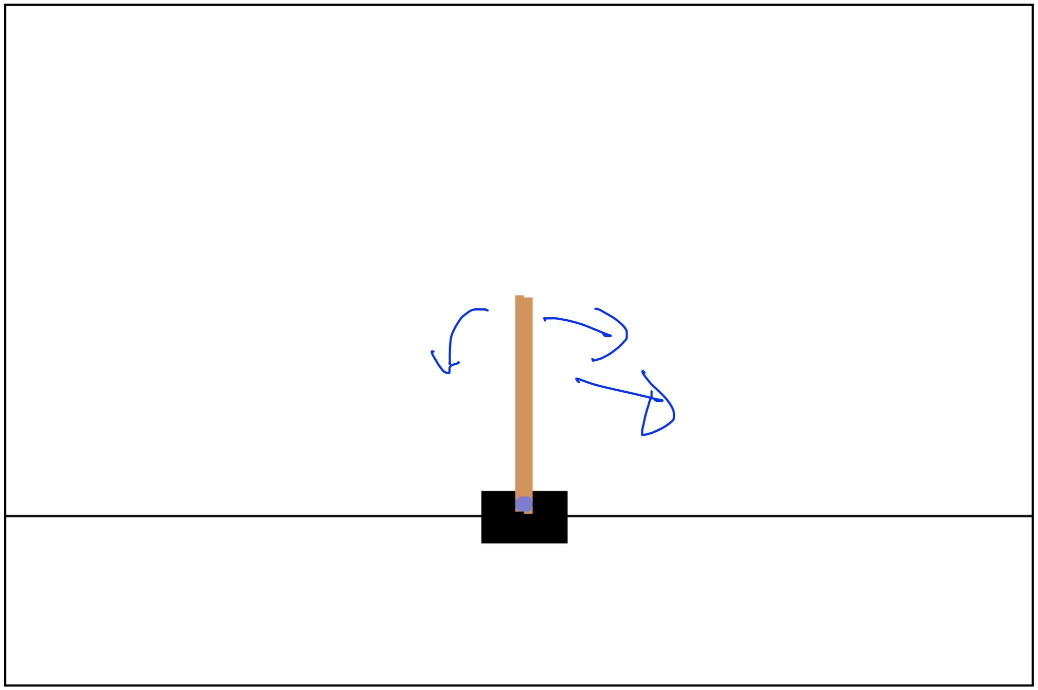
Q-function Approximation: Q-Nets



(2) quality (reward) for all actions
(eg, [0.5, 0.1, 0.0, 0.8]
LEFT: 0.5,
RIGHT 0.1
UP: 0.0,
DOWN: 0.8)



Q-Nets are unstable



```
Episode: 1988   steps: 14
Episode: 1989   steps: 25
Episode: 1990   steps: 15
Episode: 1991   steps: 23
Episode: 1992   steps: 19
Episode: 1993   steps: 17
Episode: 1994   steps: 46
Episode: 1995   steps: 20
Episode: 1996   steps: 17
Episode: 1997   steps: 15
Episode: 1998   steps: 33
Episode: 1999   steps: 22
2017-02-08 16:59:31.216 Pyth
Total score: 15.0
```

Convergence

$\hat{Q}$ denote learner's current approximation to Q.

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2$$

- Converges to Q^* using table lookup representation
- But diverges using neural networks due to:
  - Correlations between samples
  - Non-stationary targets

Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind

Reinforcement + Neural Net

25

There are some research papers on the topic:

- Efficient Reinforcement Learning Through Evolving Neural Network Topologies (2002)
- Reinforcement Learning Using Neural Networks, with Applications to Motor Control
- Reinforcement Learning Neural Network To The Problem Of Autonomous Mobile Robot Obstacle Avoidance

And some code:

- Code examples for neural network reinforcement learning.

Those are just some of the top google search results on the topic. The first couple of papers look like they're pretty good, although I haven't read them personally. I think you'll find even more information on neural networks with reinforcement learning if you do a quick search on Google Scholar.

- But diverges using neural networks due to:
  - Correlations between samples
  - Non-stationary targets

http://stackoverflow.com/questions/10722064/training-a-neural-network-with-reinforcement-learning

DQN paper

www.nature.com/articles/nature14236

DQN source code:

sites.google.com/a/deepmind.com/dqn/

Two big issues

- But diverges using neural networks due to:
  - Correlations between samples
  - Non-stationary targets

Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind

1. Correlations between samples

Algorithm 1 Deep Q-learning

Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

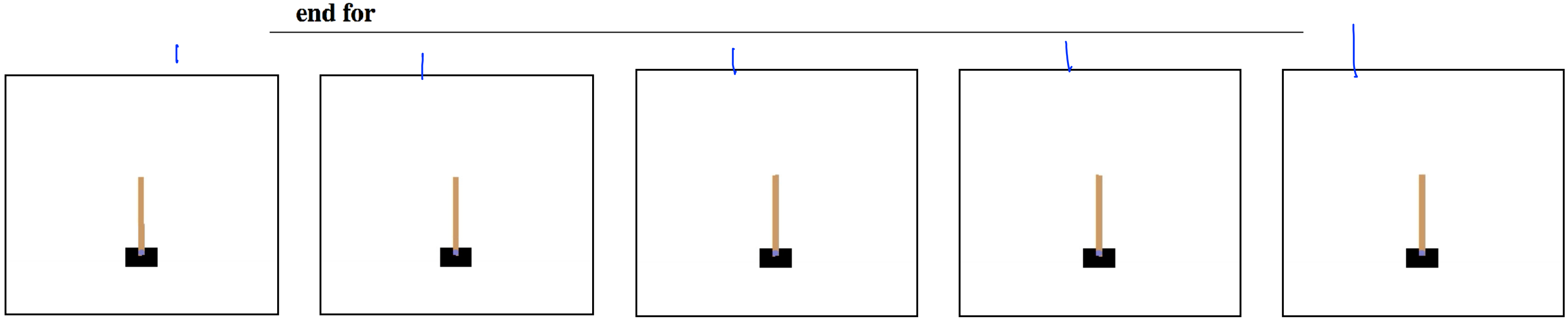
Playing Atari with Deep Reinforcement Learning - University of Toronto by V Mnih et al.

1. Correlations between samples

Algorithm 1 Deep Q-learning

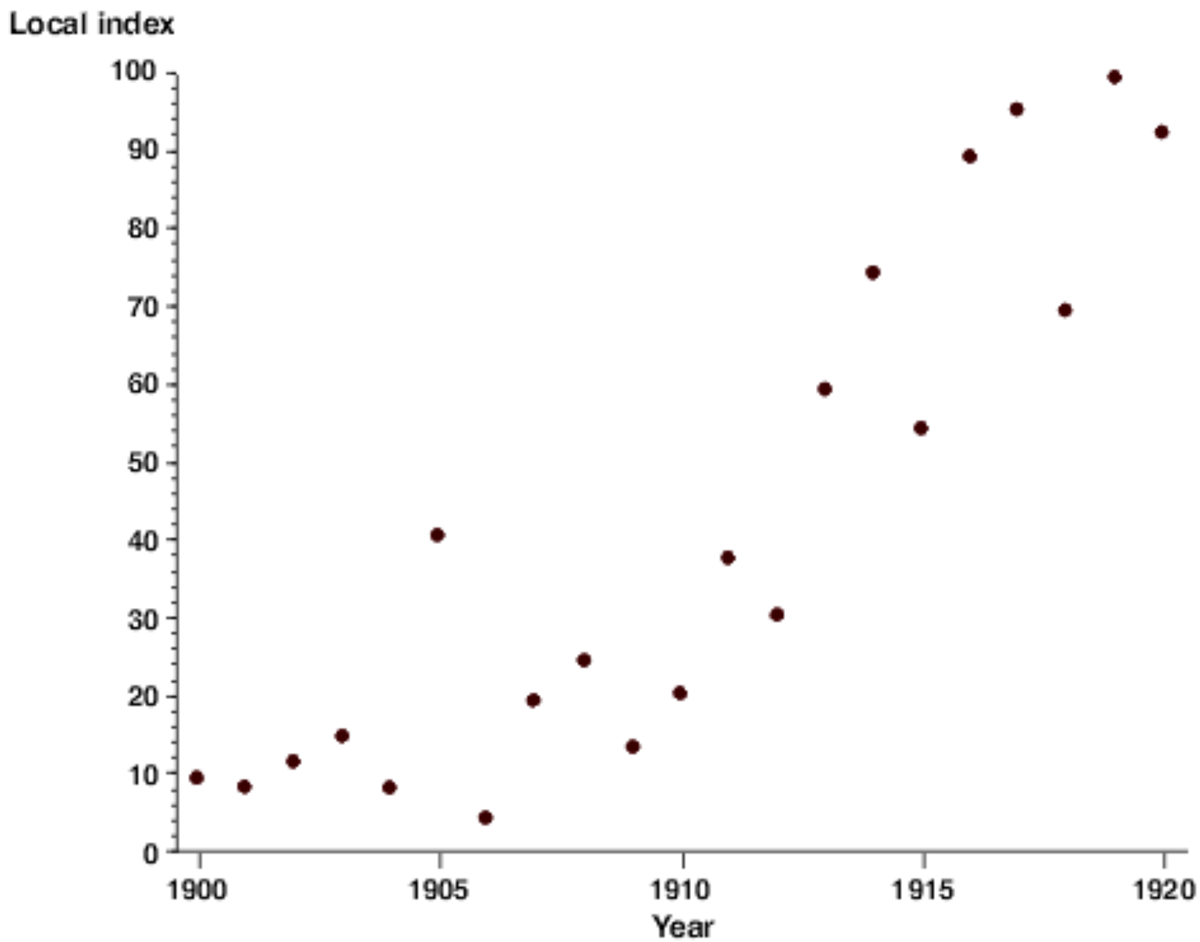
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

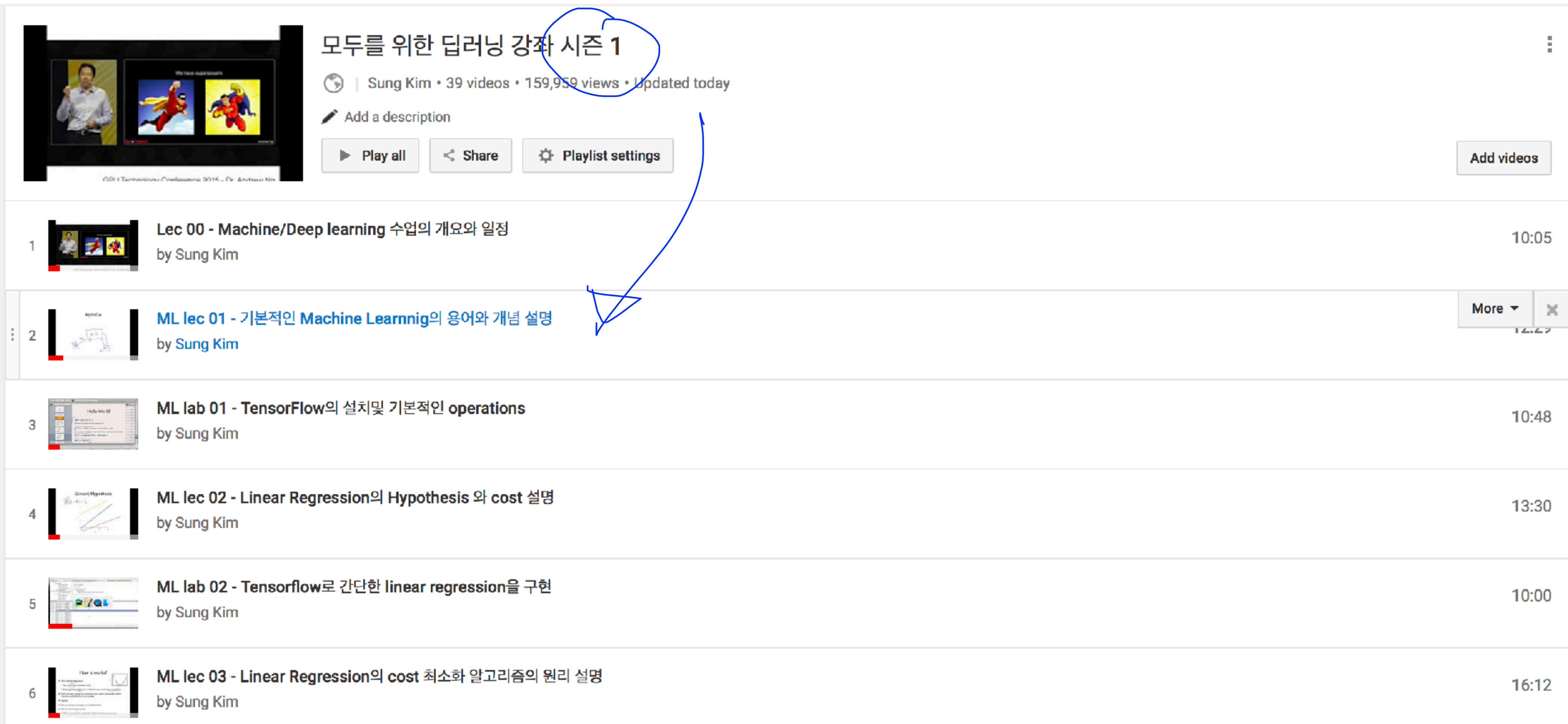


Playing Atari with Deep Reinforcement Learning - University of Toronto by V Mnih et al.

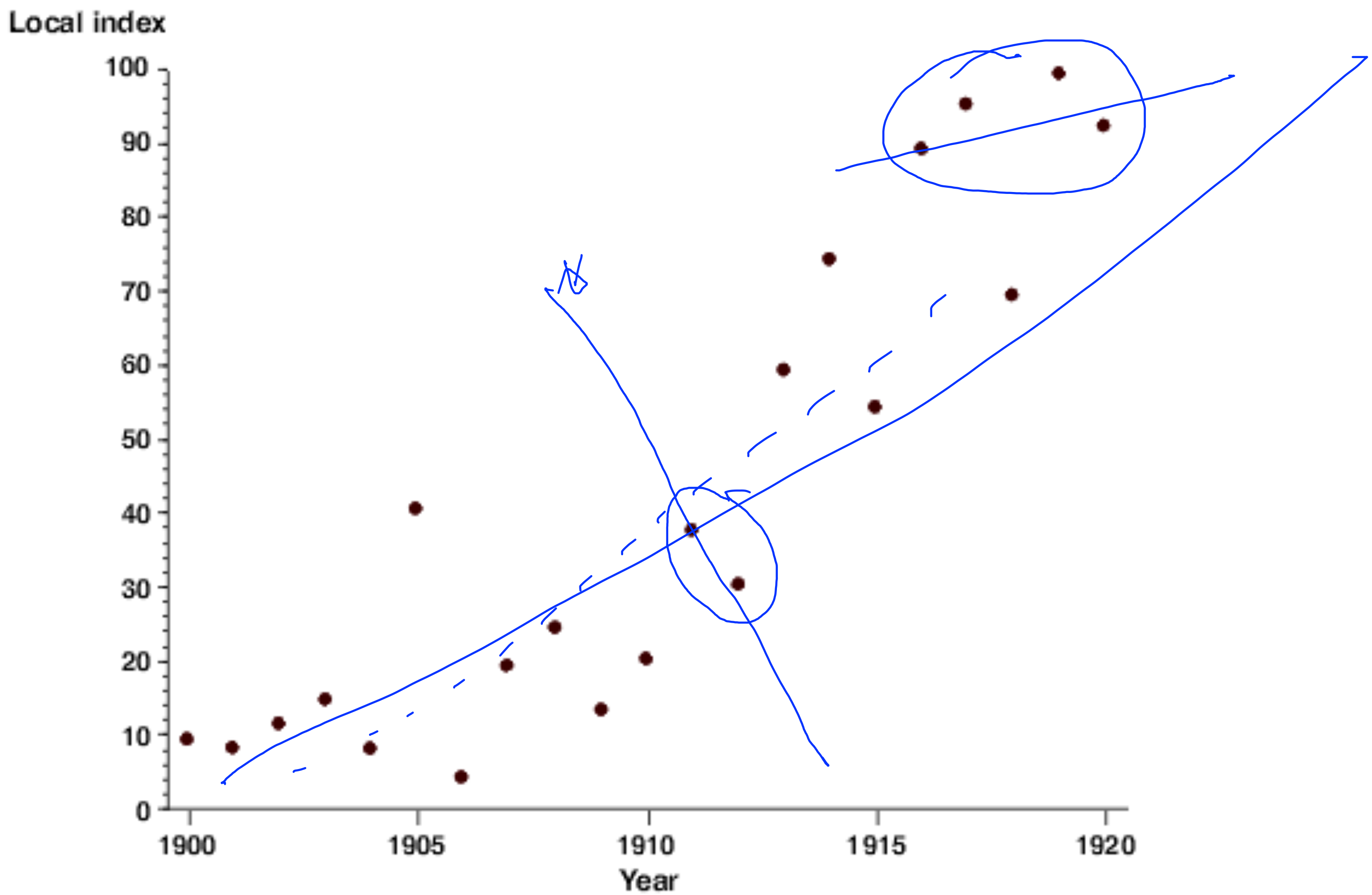
1. Correlations between samples



Prerequisite: http://hunkim.github.io/ml/ or https://www.inflearn.com/course/기본적인-머신러닝-딥러닝-강좌/



1. Correlations between samples



2. Non-stationary targets

$$\min_{\theta} \sum_{t=0}^{T} [\underbrace{\hat{Q}(s_t, a_t|\theta)}_{Q\ \text{pred}} - \underbrace{(r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))}_{Y\ \text{target}}]^2$$

2. Non-stationary targets

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2$$

pred

w

$$\hat{Y} = \hat{Q}(s_t, a_t|\theta)$$

target

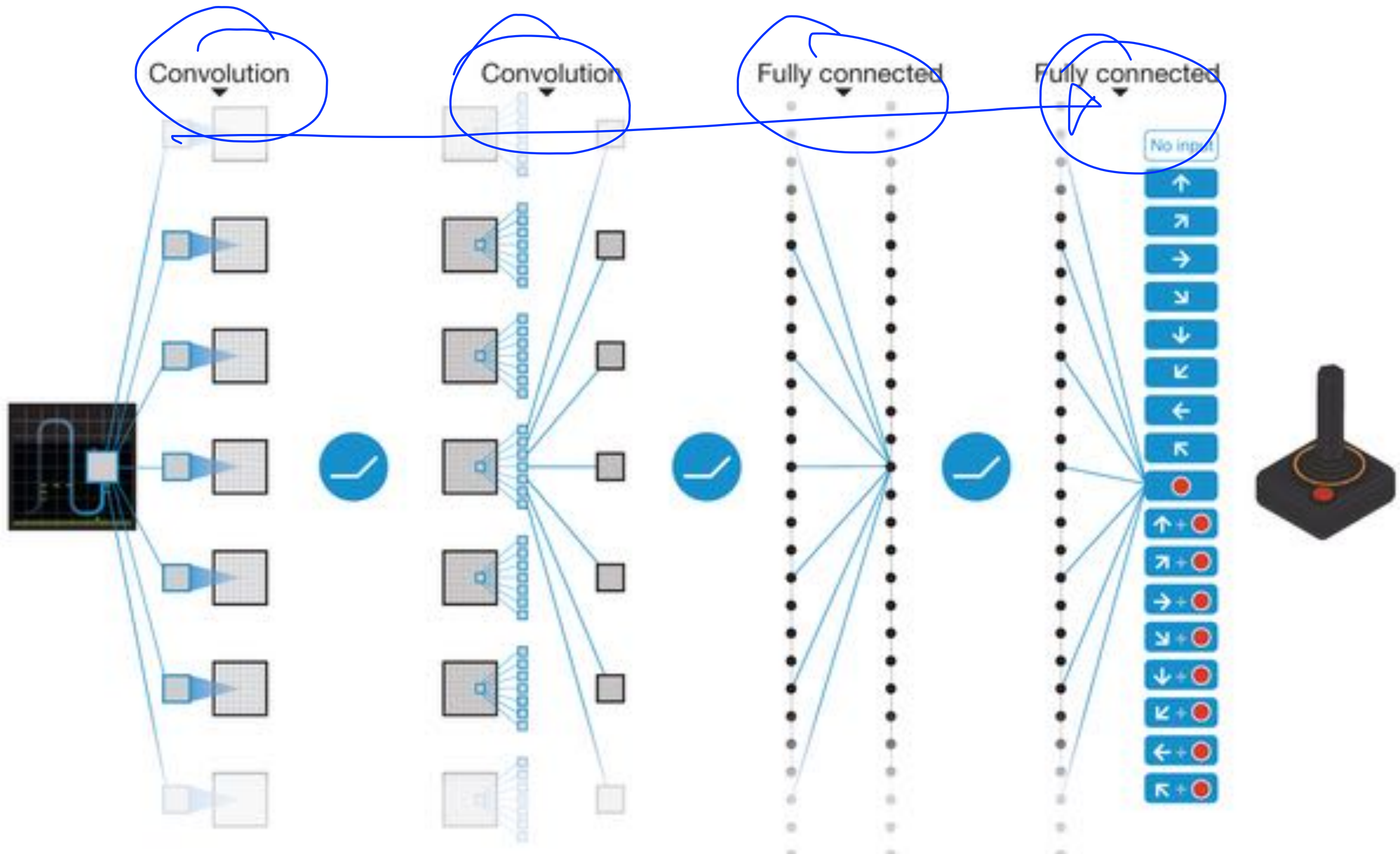
$$Y = r_t + \gamma \max_{a'} \hat{Q}_{\theta}(s_{t+1}, a'|\theta)$$

DQN's three solutions

1. Go deep
2. Capture and replay
   - Correlations between samples
3. Separate networks: create a target network
   - Non-stationary targets

Tutorial: Deep Reinforcement Learning, David Silver, Google DeepMind

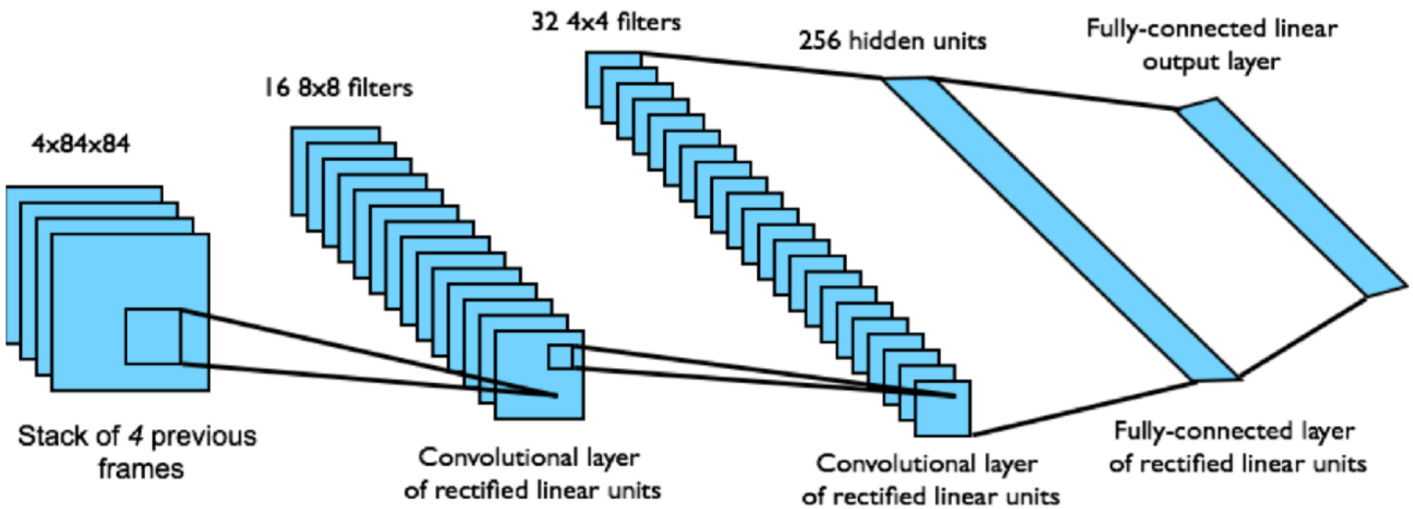
Solution 1: go deep



Human-level control through deep reinforcement learning, Nature
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

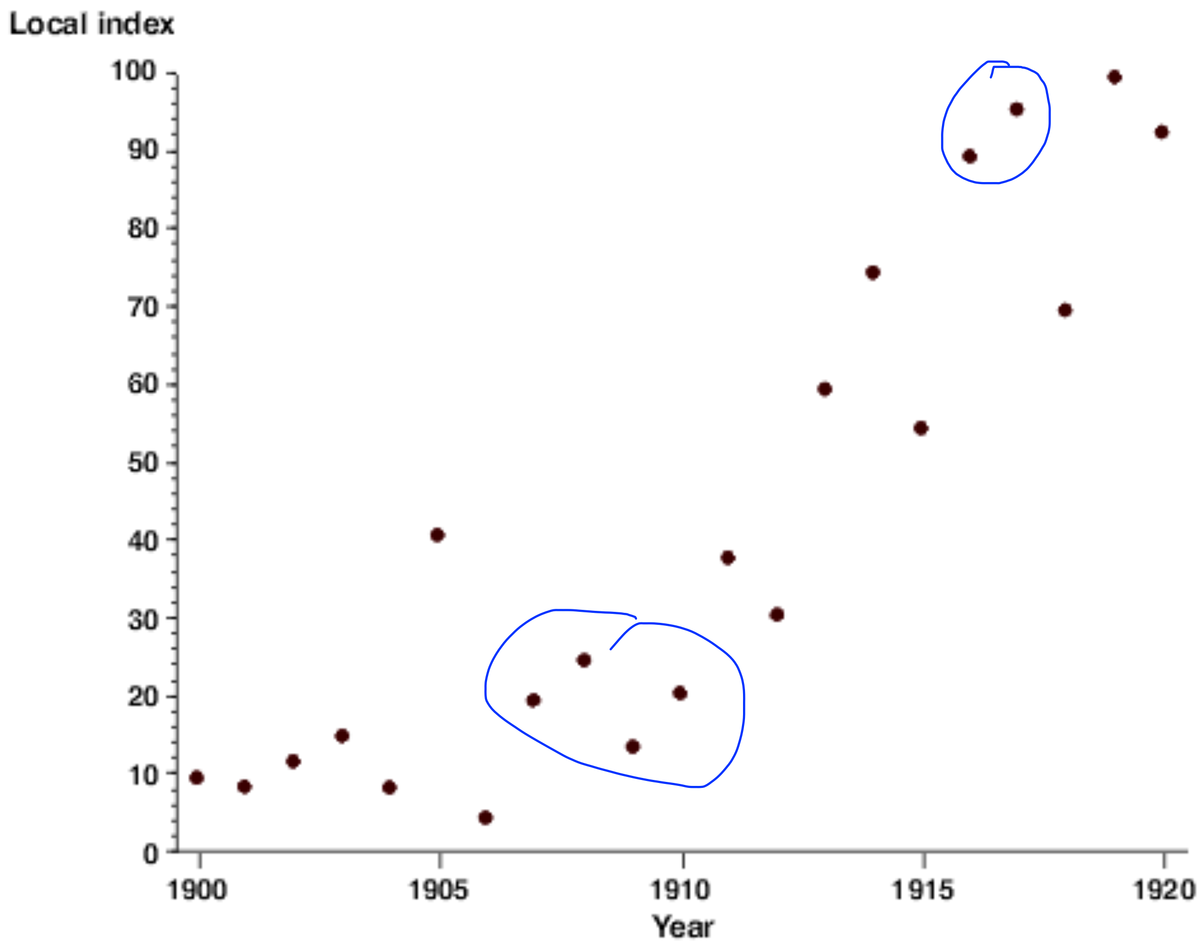
Solution 1: go deep

- 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

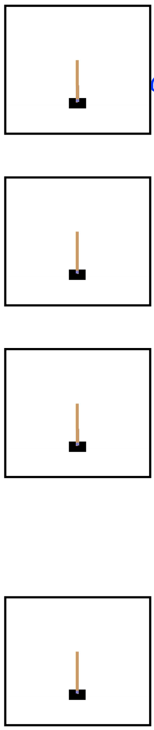


Network architecture and hyperparameters fixed across all games

Problem 2: correlations between samples



Solution 2: experience replay



Capture

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
...
$s_t, a_t, r_{t+1}, s_{t+1}$

random sample & Replay

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2$$

Solution 2: experience replay

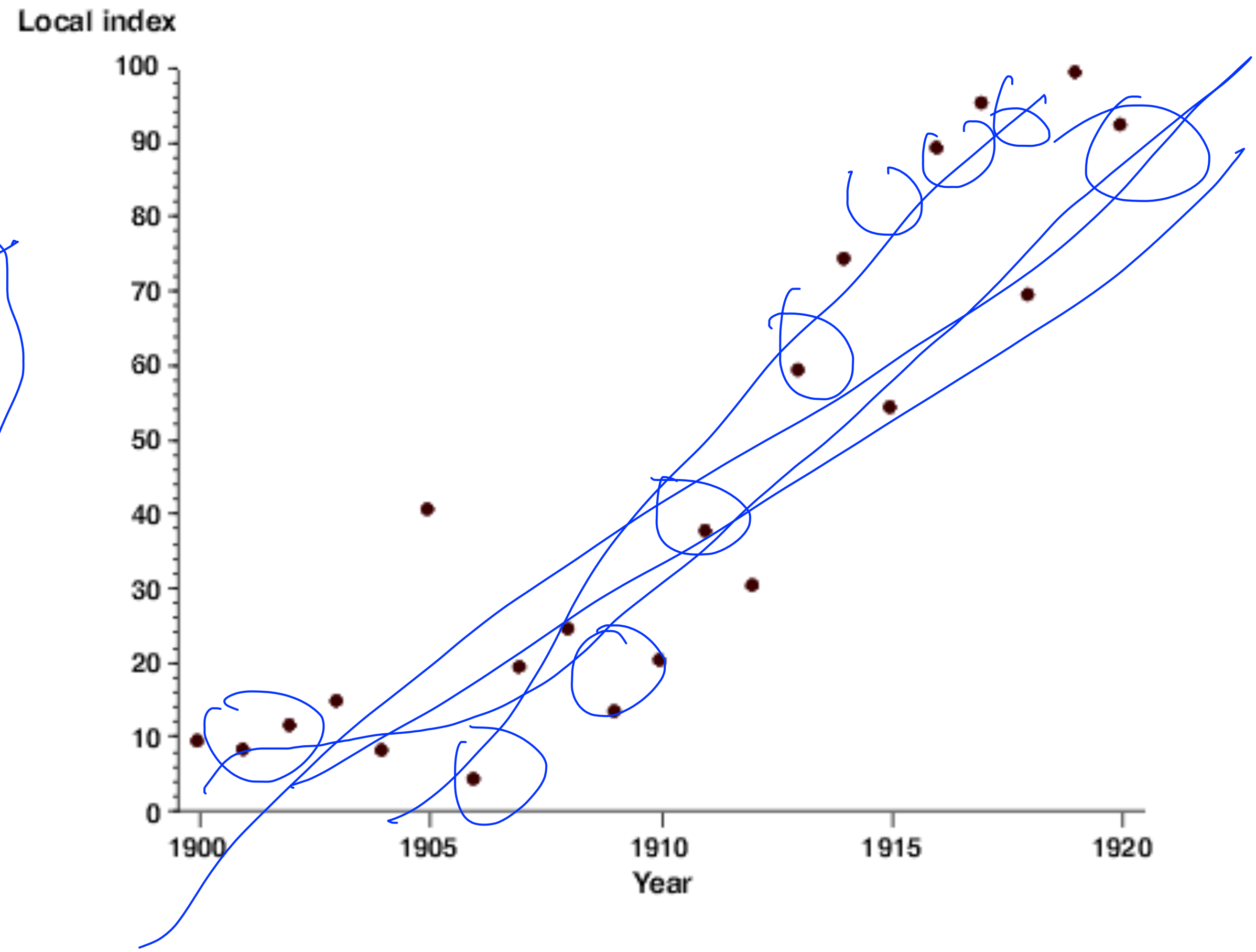
Algorithm 1 Deep Q-learning with Experience Replay

Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    $\boxed{\text{Store transition } (\phi_t, a_t, r_t, \phi_{t+1}) \text{ in } \mathcal{D} \quad \text{Sample random minibatch of transitions } (\phi_j, a_j, r_j, \phi_{j+1}) \text{ from } \mathcal{D}}$
    buffer
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Playing Atari with Deep Reinforcement Learning - University of Toronto by V Mnih et al.

Problem 2: correlations between samples

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\ldots$
$s_t, a_t, r_{t+1}, s_{t+1}$



Problem 3: non-stationary targets

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2$$

$$\hat{Y} = \hat{Q}(s_t, a_t|\theta) \qquad Y = r_t + \gamma \max_{a'} \hat{Q}_{\theta}(s_{t+1}, a'|\theta)$$

Solution 3: separate target network

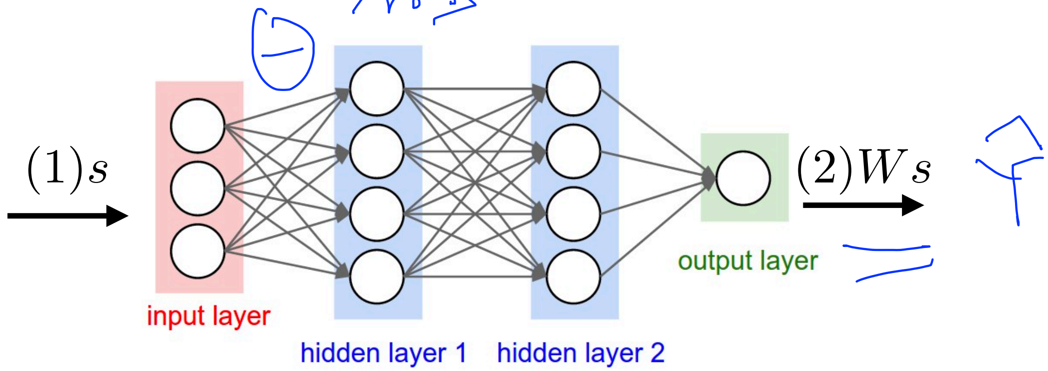
$$\min_\theta \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\bar{\theta}))]^2$$

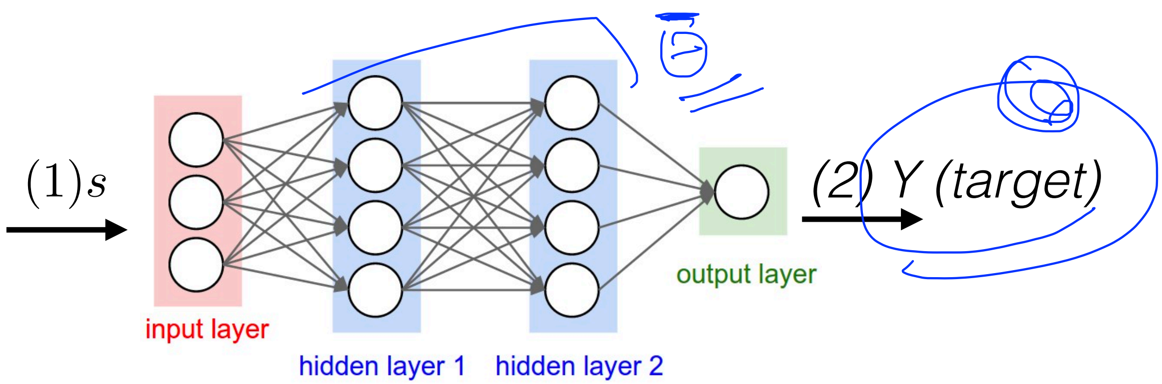
$$\cancel{\min_\theta \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\theta))]^2}$$

Human-level control through deep reinforcement learning, Nature
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

Solution 3: separate target network

$$\min_{\theta} \sum_{t=0}^{T} [\hat{Q}(s_t, a_t|\theta) - (r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'|\bar{\theta}))]^2$$





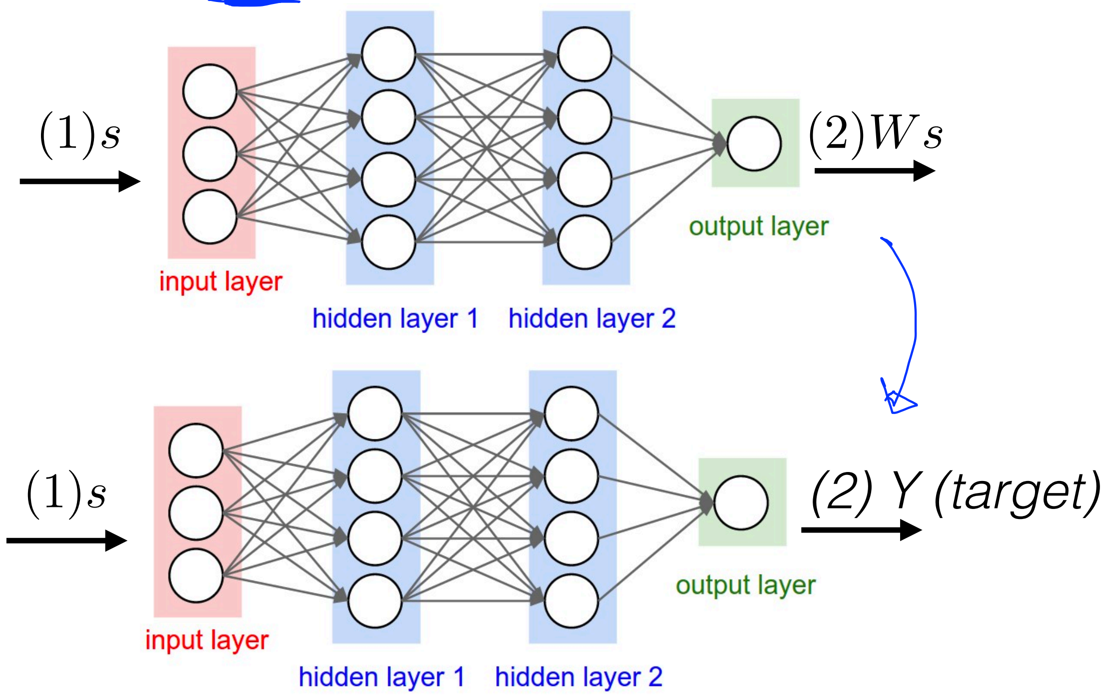
Solution 3: copy network

Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

Every C steps reset $\hat{Q} = Q$



Understanding Nature Paper (2015)

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    target
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Human-level control through deep reinforcement learning, Nature
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

Next

Lab: DQN