

LETTER

doi:10.1038/nature14236

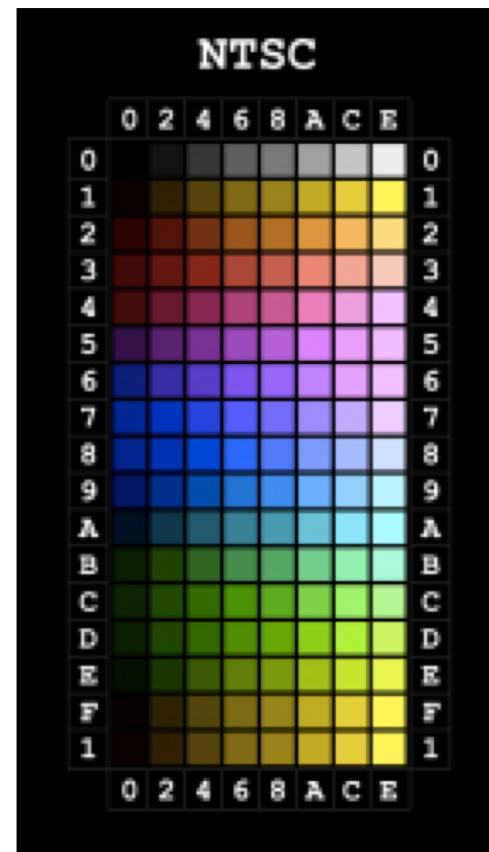
Human-level control through deep reinforcement learning

Volodymyr Mnih^{1*}, Koray Kavukcuoglu^{1*}, David Silver^{1*}, Andrei A. Rusu¹, Joel Veness¹, Marc G. Bellemare¹, Alex Graves¹, Martin Riedmiller¹, Andreas K. Fidjeland¹, Georg Ostrovski¹, Stig Petersen¹, Charles Beattie¹, Amir Sadik¹, Ioannis Antonoglou¹, Helen King¹, Dharshan Kumaran¹, Daan Wierstra¹, Shane Legg¹ & Demis Hassabis¹

Qing Lyu

5/24/2018

Background: Atari 2600



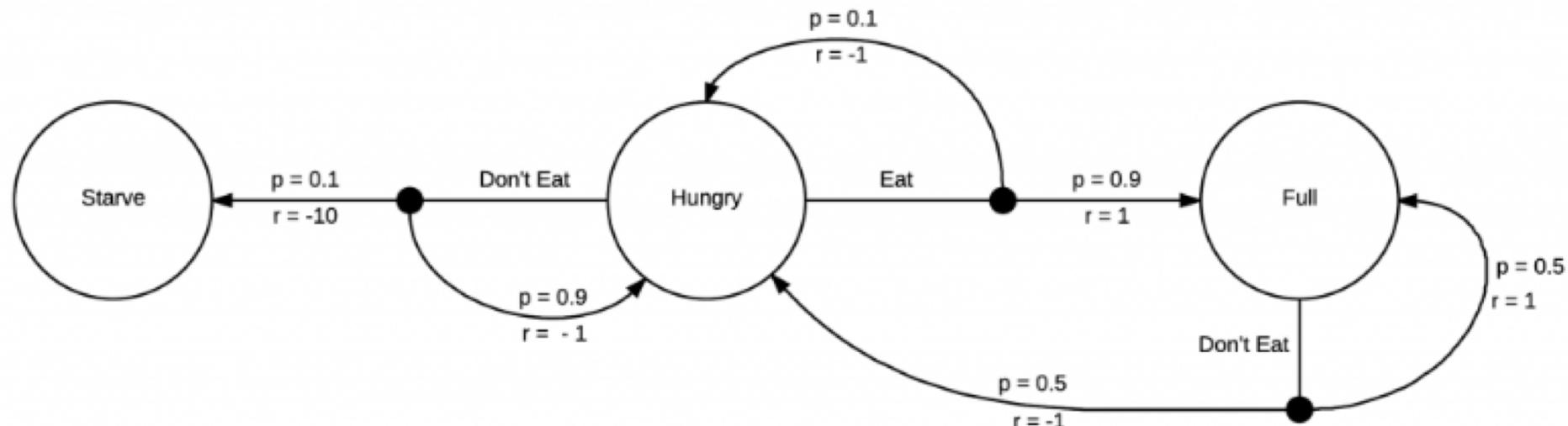
Background: Reinforcement Learning

- Reward
- Policy

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- It is a function that takes in a state and an action and returns the probability of taking that action in that state.

$$\sum_a \pi(s, a) = 1$$



Background: Reinforcement Learning

- Value Functions
 - **state value function:** the value of a state when following a policy
$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s]$$
 - **action value function:** the value of taking an action in some state when following a certain policy
$$Q^\pi(s, a) = \mathbb{E}_\pi[R_t | s_t = s, a_t = a]$$
- Bellman Equations

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s' | s, \pi(s)) V^\pi(s')$$

Background: Reinforcement Learning

- Q-learning algorithm
 - Before learning begins, Q is initialized to a possibly arbitrary fixed value (chosen by the programmer). Then, at each time t the agent selects an action a_t , observes a reward r_t , enters a new state s_{t+1} (that may depend on both the previous state s_t and the selected action), and Q is updated. The core of the algorithm is a simple value iteration update, using the weighted average of the old value and the new information:

$$Q(s_t, a_t) \leftarrow \underbrace{(1 - \alpha) \cdot Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

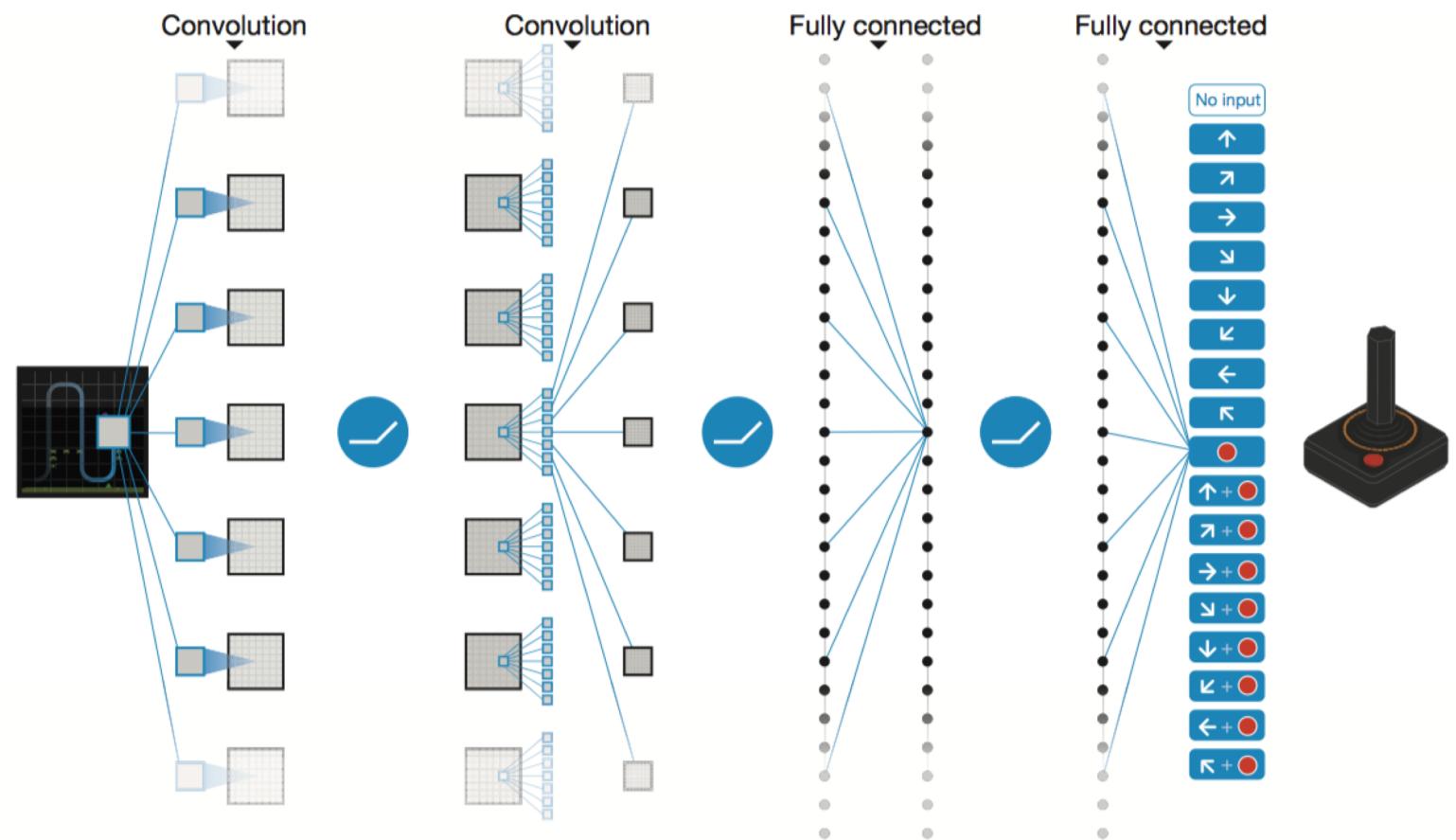
Method

- Preprocessing
 - Take the maximum value for each pixel colour value over the frame being encoded and the previous frame
 - extract the Y channel, also known as luminance, from the RGB frame and rescale it to 84×84

Method

Input: $84 \times 84 \times 4$ image

Output: a fully-connected linear layer with a single output for each valid action



$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

Method

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 $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$

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

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

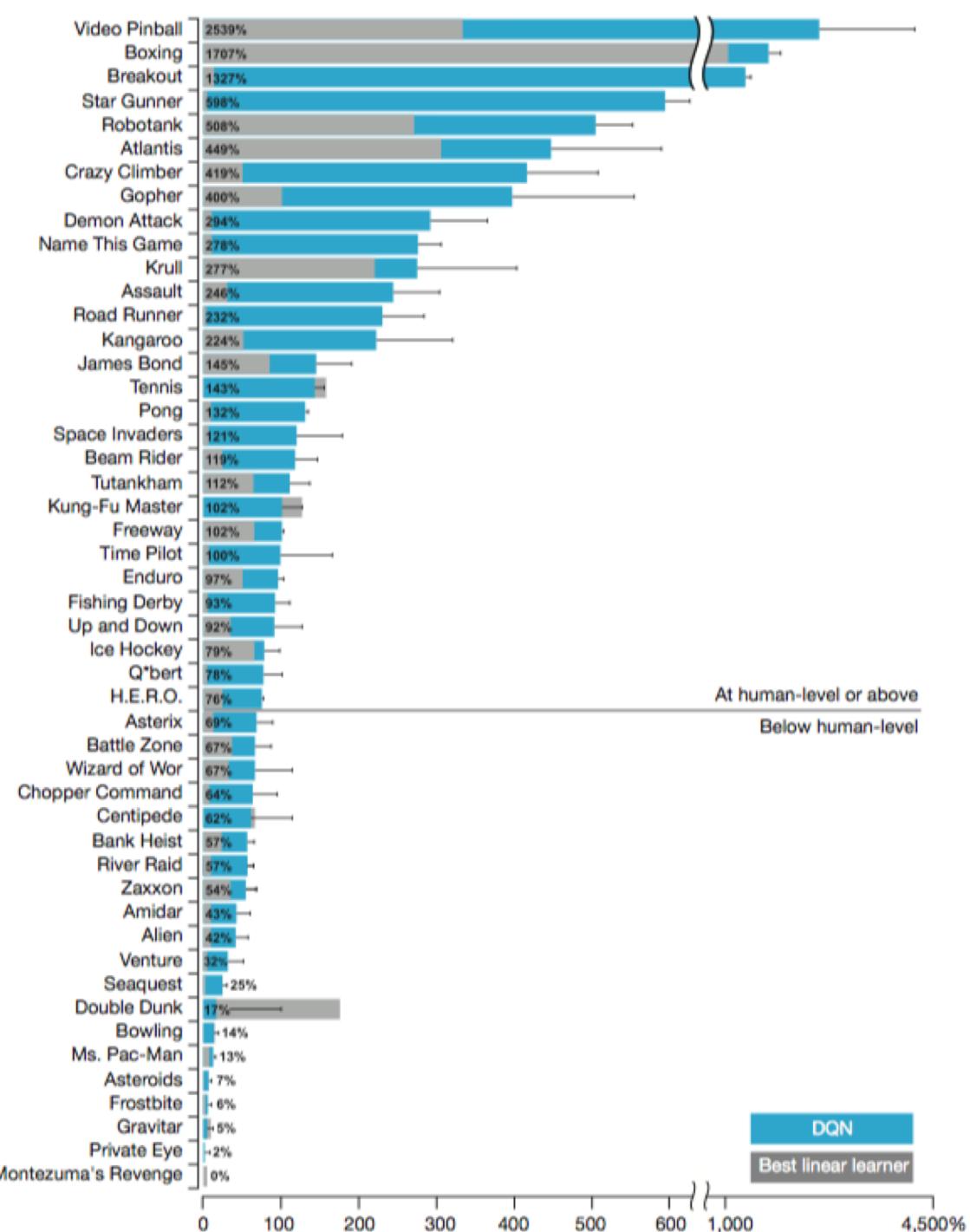
 Every C steps reset $\hat{Q} = Q$

End For

End For

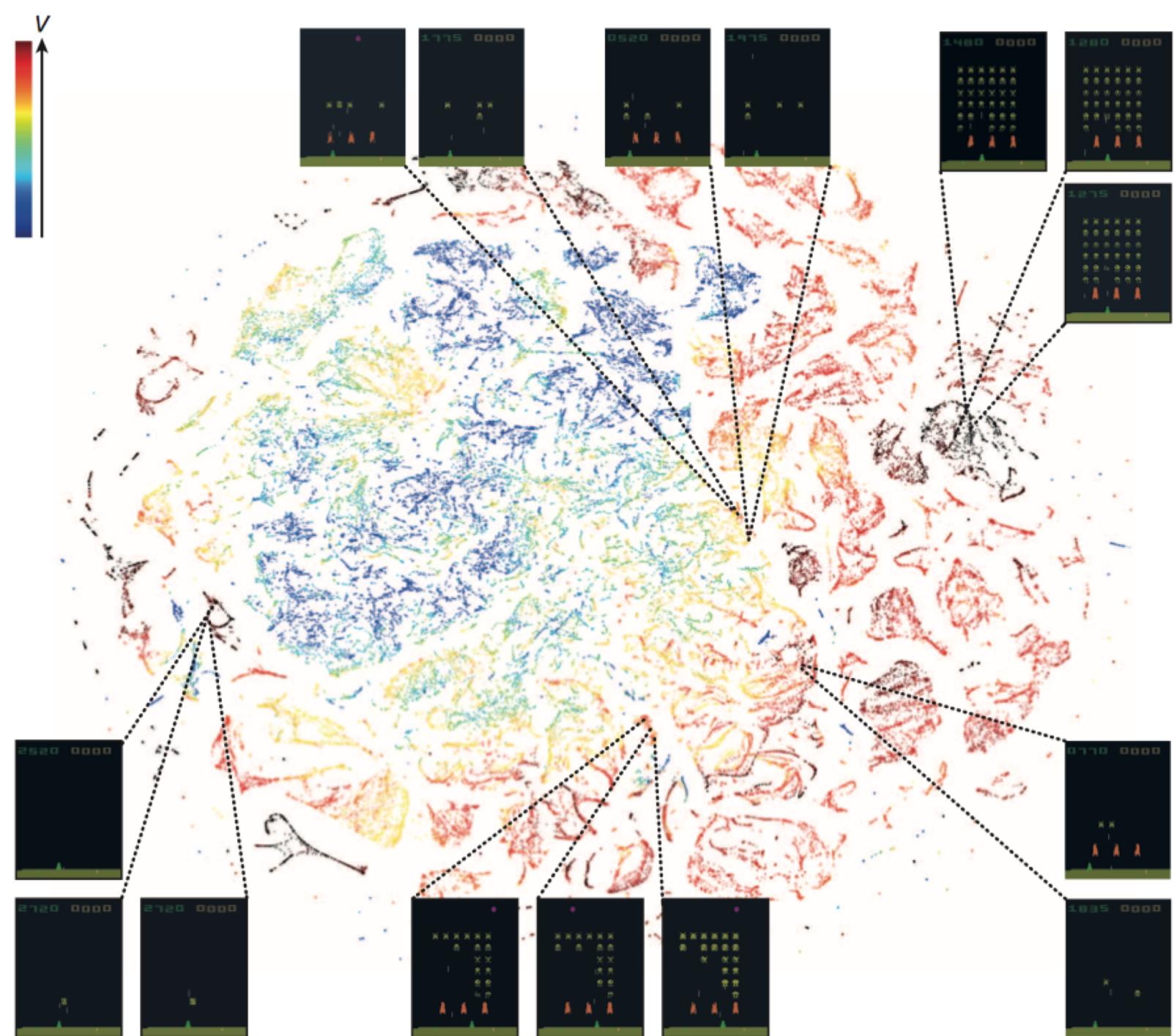
Result

the normalized performance of DQN,
expressed as a percentage, is calculated as:
 $100 \times (\text{DQN score} - \text{random play score}) / (\text{human score} - \text{random play score})$



Result

Two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders.



Result

- Videos
 - <https://www.nature.com/articles/nature14236>

Reference

- <https://joshgreaves.com/reinforcement-learning/introduction-to-reinforcement-learning/>
- <https://lvdmaaten.github.io/tsne/>
- http://artint.info/html/ArtInt_265.html
- <https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287>